Cross-Domain Graph Learning for Multivariate Time Series Forecasting

Difan Zou
knowzou@ucla.edu
Department of Computer Science,
University of California, Los Angeles
Los Angeles, CA, USA

Ni Ma
nma55@bloomberg.net
Bloomberg L.P.
New York City, New York, USA

Saher Esmeir
sesmeir2@bloomberg.net
Bloomberg L.P.
London, UK

ABSTRACT

In multivariate time series forecasting, people have observed that the correlations in the spatial domain (inter-series) and temporal domain (intra-series) are both crucial to modeling the time series. Most existing works, however, deal with these two correlations separately, which poses potential issues in precisely characterizing the cross-domain correlation. In this paper, we propose a cross-domain graph learning method, aiming to learn such cross-domain correlation more effectively. In particular, we combine the spatial and (local) temporal domain to build a larger graph, where each node is denoted by a time-location tuple. We then develop an attention based mechanism to learn the correlation between each pair of nodes. Temporal correlations will be characterized by standard GRU modules. The resulting graph is significantly larger. To reduce the computation and memory costs, we introduce efficient sampling based methods for training and inference. Extensive experiments demonstrate the superior performance of our proposed model, and validate the effectiveness of the proposed sampling-based training and inference procedures.

KEYWORDS

time series, graph convolution, temporal-spatial correlation, attention, graph sampling, road traffic

ACM Reference Format:
Difan Zou, Ni Ma, and Saher Esmeir. 2022. Cross-Domain Graph Learning for Multivariate Time Series Forecasting. In *Proceedings of 8th SIGKDD Workshop on Mining and Learning from Time Series (MiLeTS '22).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Multivariate time series forecasting has been widely applied in real-world scenarios such as traffic forecasting, financial investment, and pandemic projections. This problem is generally more complicated than the univariate one since it involves complicated correlations over two domains: temporal domain and spatial domain. Characterizing the correlations across these two domains is rather challenging, especially in the scenarios where these two correlations are also mutually responsive.

Recent advances in deep learning have shed some light on building effective models for representing the multivariate time series data. In particular, in the temporal domain (or for one single time series), standard temporal units, such as GRU, GLU, or LSTM, have been widely applied to capture intra-series correlations. In the spatial domain, graph neural networks (GNNs), such as graph convolutional networks (GCNs) [7] and graph attention networks (GAT) [13], have become a prevalent choice in modeling the inter-series dependencies. Motivated by this, a series of recent works [1, 6, 10, 11, 17, 18] propose to model the multivariate time series data by stacking these two different deep learning modules. Specifically, Li et al. [10] proposes DCRNN, a model built by modifying the linear operations in the update formula of the GRU cell using graph convolution operations. Shang and Chen [11] take the randomness and uncertainty of the graph into consideration and develop a probabilistic graph model that is parameterized by a neural network, in contrast to using a fixed graph. Besides, in order to enable learnable correlation in both temporal and spatial domains, the attention mechanism has been applied to replace the GRU or GCN modules in DCRNN. Specifically, Bai et al. [1] introduced AGCRN, a model that applies GATs in the spatial domain and maintains the GRU structure for characterizing the correlation in the temporal domain. Guo et al. [6] proposed ASTGCN, a model that applies transformers both in the temporal and the spatial domains to enable more flexible learning of data correlations.

These works, however, still deal with the temporal and spatial domains separately or tackle them in different orders. In particular, they will either first apply graph convolution over the nodes at the same time and then feed them into the temporal model [10, 12], or first apply temporal model for the data in one single node and then perform the graph convolution over the processed signals [6]. This would pose potential issues in learning cross-domain correlations since different treatments on the temporal and spatial domain may be over-complicated or likely to involve unnecessary noise that overwhelms the discovery of useful signals. In the traffic data, for instance, multiple sensors are located at different positions of a city and collect the traffic speeds with a certain frequency. The speed collected by sensor i at time t may be highly correlated to that collected by the neighbor sensor $i+1$ (e.g., the sensor deployed in the same street) but at a later time, e.g., $t+1$. This implies that, to some extent, developing different modules for characterizing the spatial and temporal correlations will no longer be a good choice for multivariate time series forecasting problems.

MiLeTS '22, August 15, 2022, Washington DC

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

In this paper, we aim to capture such cross-domain correlations in a more effective way. In particular, motivated by the example of traffic data, we propose to combine the (local) temporal and spatial domains, which allows the calculation and learning of the correlation between any pair of temporal-spatial points. This results in a much larger graph where each node represents the time-spatial node tuple and each edge captures both in-domain and cross-domain correlations, which can be characterized more effectively. Then, based on the construction of the cross-domain graph, we further design an attention based graph learning framework that outputs the adjacency matrix of the cross-domain graph based on the input recent historical data, which is able to characterize the dynamic nature of the cross-domain correlations. Finally, combining the cross-domain graph learning with the standard GRU module, we develop the **C**ross-**D**omain **G**raph **R**ecurrent **N**etwork (CDGRN) model for multivariate time series forecasting.

Despite the power of characterizing the cross-domain correlations, this approach comes at a price. The larger graph, whose size is the multiplication of the spatial graph size and the number of (local) time points, will incur high computation and memory costs during both training and inference. In order to reduce the memory and computation costs, we borrow the idea of applying sampling mechanism in training large-scale graph network [3–5, 7, 8, 16, 19] and develop an efficient training and inference algorithm for the proposed CDGRN model. In particular, in each training or inference iteration, a subset of the graph will be randomly sampled from the entire graph according to the prior knowledge of the nodes and their relations. We then build a smaller computation graph of the CDGRN model according to the sampled subgraph. Consequentially, the training and inference can run with significantly less memory and computation resources.

We highlight the contributions of our work as follows:

- We propose a spatial-temporal model to perform multivariate time series forecasting, where the model can directly learn the cross-domain correlations, in contrast to prior works that capture temporal and spatial correlations separately.
- We develop a sampling method in the proposed model to improve the computation and memory costs, which is built based on an importance sampling scheme according to the prior knowledge (e.g., geographic information, category) of the nodes and their relations. Compared to the prior works that mainly use the full-batch graph for multivariate time series forecasting, our proposed sampling based algorithm is more computationally efficient and can be applied to train from much larger datasets.
- Extensive experiments support the effectiveness of the proposed cross-domain graph learning framework and demonstrate the efficiency (in terms of the memory and time costs) of the developed sampling-based training algorithms.

2 PROBLEM SETUP AND PRELIMINARIES

Data Structure. As illustrated previously, the multivariate time series data has two domains: time domain and spatial domain. Therefore, we use $\mathcal{X} = \{\mathbf{X}_t\}_{t=1,\ldots,T}$ to denote all data features, where $\mathbf{X}_t = [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{N\times d}$ denotes the data collected at time t, N is the number of nodes, and d is the number of data features.

Spatial graph. In addition to the historical data, we also have the side information of each node (e.g., locations for traffic data, category for financial data, mobility information for the covid data, etc.). Accordingly, we follow a similar idea to Li et al. [10] and build the adjacency matrix of the spatial graph using the thresholded Gaussian kernel, i.e., let $\mathbf{H}$ be its adjacency matrix and $\mathbf{v}_i$ and $\mathbf{v}_j$ be the side features of the nodes i and j respectively, we have

$$\mathbf{G}_{i,j} = \begin{cases} \exp\left(-\frac{\text{dist}(\mathbf{v}_i,\mathbf{v}_j)^2}{\sigma^2}\right) & \text{dist}(\mathbf{v}_i,\mathbf{v}_j) \leqslant \kappa; \\ 0 & \text{otherwise,} \end{cases}$$

where $\text{dist}(\mathbf{v}_i, \mathbf{v}_j)$ denotes the distance between $\mathbf{v}_i$ and $\mathbf{v}_j$ (which can be either euclidean distance or other structural distance), σ is the standard deviation of all distances, and κ denotes the threshold.

Sequence-to-Sequence Forecasting Problem The goal of our model is to generate accurate forecasts using historical data. However, we do not need to make use of the entire historical data since (1) the model training and inference will be extremely inefficient if we have a long history, i.e., the training sample size is prohibitively large; and (2) observations typically depend on the most recent data, which implies that we can use the recent data only with a fixed window size (i.e., L_I) to generate forecasts. Therefore, we can leverage the standard sequence-to-sequence (seq2seq) model that consists of an encoder and a decoder. The encoder takes a fixed window of recent historical data as input and passes the processed message to the decoder. Then the decoder generates the predictions in a sequential manner, giving an output sequence of length L_O.

3 CROSS-DOMAIN GRAPH ATTENTION GRU MODULE

In this section, we introduce CDGRN, our framework for multivariate time series forecasting, capable of capturing cross-domain correlations effectively.

3.1 Cross-domain Graph

As mentioned, the key to capturing cross-domain correlation is to combine the spatial domain and (local) temporal domain to build a large graph, where each node represents a tuple (t, i) that consists of both time and spatial indices. In particular, let L_{local} be the number of time steps that are combined with the spatial domain, the input sequence $\{\mathbf{X}_t\}_{t=1,\ldots,L_I}$ can be reshaped as $\{\mathbf{X}'_t\}_{t=1,\ldots,L_I/L_{\text{local}}}$, where

$$\mathbf{X}'_t = \mathbf{X}_{(t-1)L_{\text{local}}} || \mathbf{X}_{(t-1)L_{\text{local}}+1} \cdots || \mathbf{X}_{tL_{\text{local}}}, \tag{3.1}$$

and the concatenation $||$ is conducted across the temporal domain, i.e., $\mathbf{X}'_t \in \mathbb{R}^{NL_{\text{local}}\times d}$. Then in the following, we will consider this larger graph with size $NL_{\text{local}} \times NL_{\text{local}}$ instead of the original graph of size $N \times N$. Correspondingly, the input and output sequence lengths have been reduced to L_I/L_{local} and L_O/L_{local} respectively. The concept of designing the cross-domain graph is displayed in Figure 1, where $\{\mathbf{X}'_t\}_{t=1,\ldots,L_I/L_{\text{local}}}$ denotes the input historical data and $\{\hat{\mathbf{X}}'_\tau\}_{\tau=1,\ldots,L_O/L_{\text{local}}}$ denotes the predictions. We note that such design of the cross-domain gives a larger graph and a shorter sequence, which could potentially lead to faster model training and inference since the graph convolution can be performed in parallel

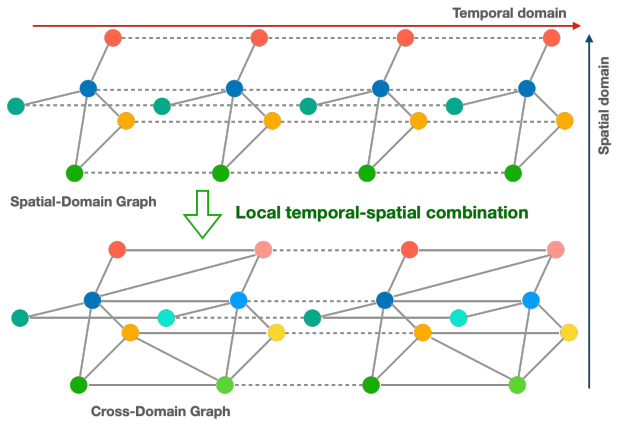

Figure 1: Illustration of the design of cross-domain graph. Here the cross-domain graph is obtained via concatenating the two spatial-domain graphs in two adjacent time steps. Besides, the same nodes have the same color at different times, solid line denotes the correlations between nodes in the spatial-domain graph or cross-domain graph, dashed line denotes the temporal correlation for one single node at different time steps.

while the seq2seq processing can only be performed in a sequential manner.

3.2 Graph Convolutional GRU Module

Graph convolution. Unlike the standard GRU module that operates on each time series separately, the graph convolution component plays an important role in our model by explicitly correlating the representations of different nodes via graph-level convolution, i.e., applying linear transformation in the node-level dimension. In particular, we apply the same idea in Li et al. [10] that uses high-order graph diffusion to aggregate the representations of different nodes. Let $\mathbf{Z} \in \mathbb{R}^{M\times d}$ be the input tensor where $M = NL_{\text{local}}$ is the size of the graph and d is the dimension of the representations, then the graph convolution, given the graph $\mathcal{G}$ with adjacency $\mathbf{A} \in \mathbb{R}^{M\times M}$ and learnable parameter Θ, operated on $\mathbf{Z}$ is defined by

$$\Theta_{\star\mathcal{G}} \cdot \mathbf{Z} = \sum_{k=1}^{K} \left[\theta_{1,k} \cdot (\mathbf{D}_O^{-1}\mathbf{A})^k + \theta_{2,k} \cdot (\mathbf{D}_I^{-1}\mathbf{A}^\top)^k\right] \cdot \mathbf{Z}\mathbf{W}_k, \tag{3.2}$$

where $\mathbf{D}_O = \text{diag}(\mathbf{A}\mathbf{1})$ and $\mathbf{D}_I = \text{diag}(\mathbf{1}\mathbf{A})$ denotes the in-degree and out-degree diagonal matrices respectively (here $\mathbf{1} \in \mathbb{R}^M$ denotes an all-1 vector), and K is the diffusion order.

Node-aware MLP. Motivated by Bai et al. [1] that points out the importance to capture node-specific patterns, we design a node-aware MLP $F(\cdot)$ to map the input data $\mathbf{Z} \in \mathbb{R}^{M\times d}$ to a higher dimensional space:

$$F(\mathbf{Z}) = \text{ELU}(\mathbf{Z}\mathbf{W}_f + \mathbf{b}_f)$$

where $\mathbf{W}_f \in \mathbb{R}^{M\times d\times h}$ and $\mathbf{b}_f \in \mathbb{R}^{M\times h}$ are learnble parameters, and $\text{ELU}(x) = x\,\mathbb{1}(x > 0) + (e^x - 1)\,\mathbb{1}(x \leqslant 0)$ denotes the ELU activation. Here the MLP mapping functions are different for the input features at different nodes [1] to enable the characterization of the node-specific patterns.

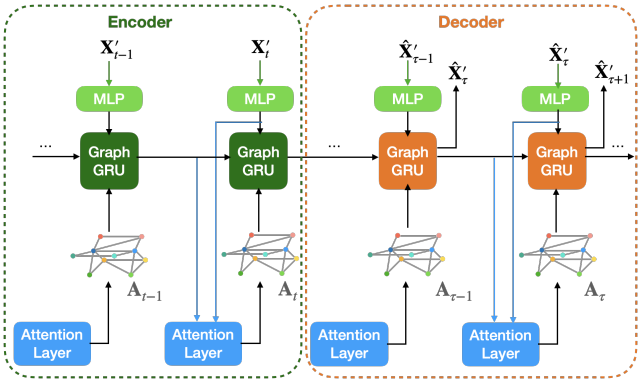

Figure 2: Diagram of the CDGRN model for multivariate time series forecasting. The green arrow line denotes the input of MLP, black arrow line denotes the input/output of Graph-convolution based GRU, and the blue arrow lines denotes the input of the attention layer.

Graph convolutional GRU. Then given the concatenated input $\mathbf{X}'_t$ (see (3.1)), the update of the graph-convolutional GRU module is formulated as

$$\begin{aligned}
\mathbf{r}_t &= \text{sigmoid}(\Theta_{r\star\mathcal{G}} \cdot [F(\mathbf{X}'_t)||\mathbf{H}_{t-1}] + \mathbf{b}_r)\\
\mathbf{u}_t &= \text{sigmoid}(\Theta_{u\star\mathcal{G}} \cdot [F(\mathbf{X}'_t)||\mathbf{H}_{t-1}] + \mathbf{b}_u)\\
\mathbf{c}_t &= \tanh(\Theta_{c\star\mathcal{G}} \cdot [F(\mathbf{X}'_t)||(\mathbf{r}_t \odot \mathbf{H}_{t-1})] + \mathbf{b}_c)\\
\mathbf{H}_t &= \mathbf{u}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{u}_t) \cdot \mathbf{c}_t,
\end{aligned}$$

where $\Theta_{r\star\mathcal{G}}, \Theta_{u\star\mathcal{G}}, \Theta_{c\star\mathcal{G}}$ denotes the graph convolutions in (3.2), $\mathbf{H}_t \in \mathbb{R}^{M\times h}$ denotes the hidden states, and the concatenation $||$ of $F(\mathbf{X}'_t)$ and $\mathbf{H}_{t-1}$ is performed in the feature dimension, i.e., enlarging the dimension from h to $2h$.

3.3 Attention Layer for Graph Learning

The next question is how to build the adjacency matrix on this cross-domain graph. [12] considers directly stacking the spatial graph and applying it in all graph convolution operations. That, however, ignores two important aspects of the multivariate time series data: (1) the graph should be dynamic since the correlation between two time series may change over time; and (2) the spatial graph cannot fully describe the correlation between different nodes, especially when considering the cross-domain graph. In this work, we address this by generating the graph from the input sequence via the attention mechanism, which implicitly performs dynamic graph learning since the input sequence changes over time. In particular, we incorporate attention layers to the aforementioned seq2seq model, which takes the hidden states and historical data as input and outputs the attention map as the graph adjacency. Mathematically, in the update of the t-th time point of the sequence

[1] We will force the MLP function to be the same for the nodes in the cross-domain graph if they were the same in the original spatial graph

($t > 1$), the graph adjacency is updated as follows:

$$\mathbf{Q}_t = [F(\mathbf{X}'_t)||\mathbf{H}_{t-1}] \cdot \mathbf{W}_Q, \tag{3.3}$$

$$\mathbf{K}_t = [F(\mathbf{X}'_t)||\mathbf{H}_{t-1}] \cdot \mathbf{W}_K, \tag{3.4}$$

$$\mathbf{A}_t = \text{softmax}\left(\frac{\mathbf{Q}_t \cdot \mathbf{K}_t^\top}{\sqrt{2h}}\right), \tag{3.5}$$

where $\mathbf{W}_Q$ and $\mathbf{W}_K$ are learnable parameters, the key and query tensors $\mathbf{Q}_t$ and $\mathbf{K}_t$ are calculated according to the hidden state at the previous time $(t-1)$ and the input at the current time (t). Additionally, since the calculation of attention map is unavailable when $t = 1$, we simply set $\mathbf{A}_1$ as the duplication of the spatial graph $\mathbf{G}$ (which will be further normalized by their in-/out- degrees), i.e., duplicating $\mathbf{G}$ by $L_{\text{local}} \times L_{\text{local}}$ times to build an $NL_{\text{local}} \times NL_{\text{local}}$ graph. Deployed with the proposed dynamic attention mechanism, we display the diagram of the CDGRN model in Figure 2.

4 EFFICIENT TRAINING VIA SAMPLING METHODS

One potential issue in the graph-based model is that the size of the spatial domain can be very large, bringing huge computation and memory costs during training and inference. It becomes more significant when applying the cross-domain graph, since we will need to deal with an even larger graph. To overcome this computation burden, we borrow the idea of applying sampling mechanism in training large-scale graph network [3–5, 7, 8, 16, 19] and develop an efficient algorithm for model training and inference.

In particular, in each training iteration, we will sample a subset of nodes from the entire graph, and then perform backward propagation according to this subgraph to update the model parameter. Therefore, in order to guarantee that the sampled subgraph contains sufficient information (i.e., the links between the pair of time-spatial tuples), the key aspect we aim to explore is how to design an effective and efficient sampling approach that can (1) guarantee dense connections between nodes; and (2) maintain sufficient cross-domain correlations.

4.1 Diffusion-based Importance Sampling

To address the aforementioned first point, we propose a diffusion based importance sampling approach. In particular, the sampling process is conducted in multiple stages. In the first stage, a subset of nodes will be uniformly sampled from the entire graph, denoted by $\mathcal{S}^{(0)}$, which can partially reveal the global information of the spatial domain. In the following stages in order to guarantee the dense connections of the sampled graph, we will apply the diffusion based sampling approach according to the spatial graph that is generated using side information. More formally, for stage k, we sample the nodes from the union of neighbors of the nodes sampled in the previous stage, denoted by $\cup_{v\in\mathcal{S}^{(k-1)}} N(v)$. Moreover, instead of uniformly sampling nodes from $\cup_{v\in\mathcal{S}^{(k-1)}} N(v)$, we will apply importance sampling according to their local connections to the nodes in $\mathcal{S}^{(k-1)}$, as the nodes having denser connections to other nodes can be more informative comparing to those with sparser connections. Particularly, let $\mathbf{G} \in \mathbb{R}^{N\times N}$ denote the spatial graph, let $\mathbf{P}^{(k-1)} \in \mathbb{R}^{|\mathcal{S}^{(k-1)}|\times N}$ denotes the row selection matrix according to the nodes sampled in the previous stage (i.e., $k-1$),

Algorithm 1 Diffusion based Importance Sampling

input: Diffusion number: K, Initial sample size: B, Sample size per stage: s, spatial graph adjacency $\mathbf{G}$
Randomly sample B nodes from $\mathcal{G}$, denoted as $\mathcal{S}^{(0)}$
for $k = 1, \ldots, K$ **do**
  Get row selection matrix $\mathbf{P}^{(k-1)}$, calculate the importance probability $p_i^{(k)} = \frac{\|\mathbf{P}^{(k-1)}\mathbf{G}_{*,i}\|_2^2}{\|\mathbf{P}^{(k-1)}\mathbf{G}\|_F^2}$.
  Sample s nodes according to $p_i^{(k)}$, denoted by $\mathcal{S}^{(k)}$
end for
Collect all sample nodes $\mathcal{S}_V = \cup_{k=0}^{K} \mathcal{S}^{(k)}$
output: cross domain sampled nodes: $\mathcal{S}_{V,T}^{(0)} = \{(i,t)\}_{i\in\mathcal{S}^{(0)},t\in\mathcal{T}}$ and $\mathcal{S}_{V,T} = \{(i,t)\}_{i\in\mathcal{S}_V,t\in\mathcal{T}}$.

the importance probabilities are defined as:

$$p_i^{(k)} := \frac{\|\mathbf{P}^{(k-1)}\mathbf{G}_{*,i}\|_2^2}{\|\mathbf{P}^{(k-1)}\mathbf{G}\|_F^2}.$$

It is clear that if the node i has no connections to the nodes in $\mathcal{S}^{(k-1)}$, then we have $\|\mathbf{P}^{(k-1)}\mathbf{G}_{*,i}\|_2 = 0$, implying that this node will never be sampled. Finally, after performing the importance sampling method for K stages, we will take their union, i.e., $\cup_{k=0}^{K}\mathcal{S}^{(k)}$, to build the subgraph that will be used in the training of the CDGRN model.

To address the second point: maintaining sufficient cross-domain correlations, we simply duplicate the sampled nodes in the temporal domain. In particular, let $\mathcal{T}$ of size L_{local} be the set of local time steps and $\mathcal{V}$ of size N be the set of all spatial nodes. We will first apply diffusion based importance sampling over $\mathcal{V}$ and get the samples $\mathcal{S}_V$. Then these nodes will be duplicated in the temporal domain, leading to the samples of nodes in the cross-domain $\mathcal{V}\times\mathcal{T}$, i.e., $\mathcal{S}_{V,T} = \{(i,t)\}_{i\in\mathcal{S}_V,t\in\mathcal{T}}$. We summarize the entire sampling process in Algorithm 1.

4.2 Sampling-based Training and Inference

As we have mentioned previously, the sampled cross-domain graph will be leveraged in each training iteration, i.e., the model parameters will be updated based on the sampled cross-domain subgraph $\mathcal{S}_{V,T}$ in each iteration. However, it remains unclear how to approach this during the inference, since each node could appear in different sampled subgraphs. To address this, we propose to use the entire sampled subgraph $\mathcal{S}_{V,T}$ to perform the graph convolution, while we only focus on the predictions of the nodes sampled in the initial stage, i.e., $\{(i,t)\}_{i\in\mathcal{S}^{(0)},t\in\mathcal{T}}$. Then the predictions for all nodes can be obtained via a standard mini-batching technique.

Memory and time costs. We first remark that two aspects will affect the time and memory complexities of the training and inference process: (1) the size of the (sampled) cross-domain graph; (2) the input/output sequence length. In particular, the memory and time costs will increases as the graph size increases since it leads to a larger computation graph; while the time cost will decrease as the sequence length decreases since the seq2seq model is performed in a sequential manner. This suggests that if the sampling size is fixed, the training/inference will take less time if L_{local} is larger.

Table 1: Basic information of the dataset

dataset	# nodes	train size	val size	test size
METR-LA	207	23974	3425	6850
PEMS-BAY	325	36481	5209	10419
PEMS04	307	10171	3374	3374
PEMS08	170	10689	3547	3547

This suggests that the proposed CDGRN model can enjoy a *faster training speed* when using a *larger cross-domain graph* and a *smaller sampled subgraph*.

5 EXPERIMENTS

In this section, we carry out experiments on real-world datasets to evaluate the proposed model and training algorithms.

5.1 Dataset and Experiment Setup

We consider 4 traffic speed datasets: METR-LA [9], PEMS-BAY, PEMS04, and PEMS08.[2] These datasets contain the traffic speeds/flow recorded by the sensors located on different roads in Los Angeles and San Francisco Bay Area. The raw data is recorded every 30 seconds, which has been further aggregated and processed to enlarge the temporal granularity to 5 minutes. For all dataset, we first formulate them into multiple sequence-to-sequence data, where the sequence length is 12. Besides, for METR-LA and PEMS-BAY dataset, the input data feature consists of: historical traffic speeds, relative time steps, and node locations (longitude and latitude). For PEMS04 and PEMS08 dataset, we only use historical traffic speeds and relative time steps due to the missing location information. Then, we apply standard train-validation-test splitting to generate the training, validation, and test sets. We summarize the detailed information of these datasets in Table 1.

All experiments are conducted on the NVIDIA RTX 2080ti GPU with 11 GB memory. The batch size is set as 32 for all experiments and the hidden dimension is 64. We train all experiments for 100 epochs (each epoch performs one pass of the seq2seq data) using Adam with exponential learning rate decay at the 35th, 60th, and 75th epochs. Moreover, we apply the commonly used teacher forcing method to enable stable and faster convergence, where the teacher forcing ratio will decrease as the training proceeds. Besides, we remark that different from Li et al. [10] that applies the same teacher forcing choice (i.e., using either the previous ground truth or predictions) for all nodes, we conduct this in a node-wise manner, i.e., the choice of using either the previous ground truth or predictions is made independently for different nodes (then some of nodes may use ground truth while others may use predictions).

5.2 Experiments on the Application of Cross-domain Graph

We first demonstrate the effectiveness and benefit of the proposed cross-domain graph learning methods, where we train the CDGRN model with the cross-domain component of different sizes (i.e., built using different numbers of local time steps). In this part, we use **full-batch training**, i.e., conducting the training/inference using the entire graph. Therefore, we only use a subset of all dataset (we only extract the data of 100 nodes from the original dataset according to the indices of the nodes in the original spatial graph) to guarantee that the memory cost will not exceed the GPU memory limit for the largest cross-domain graph. We then consider four different cross-domain graphs, that leverage $L_{\text{local}} = 1$, $L_{\text{local}} = 2$, $L_{\text{local}} = 3$, and $L_{\text{local}} = 4$ local time steps. Accordingly, the sequence lengths of the input data are 12, 6, 4, 3, respectively. Note that the model with $L_{\text{local}} = 1$ can be viewed as the baseline model since it directly uses the original spatial graph. We first evaluate the masked mean absolute error (MAE) [10] for four traffic speed dataset: METR-LA, PEMS-BAY, PEMS04, PEMS08 and report the results in Table 3. It can be seen that the proposed cross-domain graph learning provides clear performance improvements comparing to performing graph learning solely in the spatial domain. This demonstrates the effectiveness of the proposed CDGRN model.

Besides, we also make comparison to baseline models on the entire METR-LA and PEMS-BAY dataset (i.e., using the entire graph), including ARIMA [10], FC-LSTM [10], WaveNet [15] , DCRNN [10], GMAN [18], MTGNN [14], StemGNN [2], and GTS [11], and report the results in Table 2. In particular, due to the GPU memory limit we will only consider $L_{\text{local}} = 1$ and $L_{\text{local}} = 2$, denoted by CDGRN-1 and CDGRN-2 respectively. Experimental results show that the proposed CDGRN models performs better than all baseline models on the METR-LA dataset, and also outperforms most baseline models on the PEMS-BAY dataset. This again demonstrates the effectiveness of the proposed CDGRN model.

5.3 Experiments on the Sampling-based Model Training and Inference

We next evaluate the performance of the sampling based training and inference algorithms. We consider different sample sizes per stage and diffusion numbers. We report the masked MAE for METR-LA and PEMS-BAY dataset in Table 4, where we consider three cases: $L_{\text{local}} = 1$, $L_{\text{local}} = 2$, and $L_{\text{local}} = 3$. We do not include the full-batch training results for $L_{\text{local}} = 3$ since it exceeds the GPU memory. It can be seen that: (1) the test performance becomes better when using a larger sample size, which can be either achieved by increasing sample size per stage or diffusion number; (2) using a properly small sampled subgraph during the training will not lead to a significant performance drop, while can improve the memory and time costs. In particular, with sampling-based training approach, the proposed CDGRN model strictly outperforms a number of baseline models (e.g., ARIMA, FC-LSTM, and DCRNN).

6 CONCLUSION

We propose CDGRN, a spatial-temporal model for multivariate time series forecasting. At the core of the CDGRN is a novel cross-domain graph learning framework, which is built based on a combination of the spatial domain and (local) temporal domain with an attention based graph learning mechanism. To improve the model's scalability and allow efficient training for data with a large spatial graph, we develop sampling based training and inference algorithms. Experimental results on the publicly available traffic sensor data demonstrate a clear performance improvement brought by the utilization of the proposed cross-domain graph learning

[2]PEMS dataset can be found in https://pems.dot.ca.gov/

Table 2: Test MAE of the 1-hr predictions generated for different models, all models are trained using the entire graph. Results show that the proposed cross-domain graph learning framework can outperform most of existing models.

Model	ARIMA	FC-LSTM	DCRNN	WaveNet	STGCN	GMAN	MTGNN	StemGNN	CDGRN-1	CDGRN-2
METR-LA	6.90	4.37	3.60	3.53	4.59	3.44	3.49	3.43	3.39	**3.36**
PEMS-BAY	3.38	2.37	2.07	1.95	2.49	**1.86**	1.94	-	1.95	1.89

Table 3: Test MAE of the 1-hr predictions generated by the CDGRN model using different numbers of local time steps, where we only extract a subset (100 nodes) from the original dataset to guarantee the full-batch training for all models.

L_{local}	1	2	3	4
METR-LA	3.40	**3.34**	3.35	3.40
PEMS-BAY	2.06	**2.02**	2.03	2.03
PEMS04	2.74	2.74	2.76	**2.72**
PEMS08	1.90	1.86	1.88	**1.83**

Table 4: Test error (masked MAE), time cost per training epoch (s), and training memory cost (MB), for the CDGRN model trained with different sample sizes (sample size per stage S, diffusion K).

Dataset	L_{local}	sample size	test MAE	mem	time
METR-LA	1	$s = 16, K = 3$	3.72	**2202**	148
		$s = 32, K = 3$	3.51	2863	152
		full batch	3.39	5230	164
	2	$s = 16, K = 3$	3.58	2286	84
		$s = 32, K = 3$	3.52	3140	104
		full batch	**3.36**	5711	189
	3	$s = 16, K = 3$	3.60	2376	**65**
		$s = 32, K = 3$	3.49	3773	71
PEMS-BAY	1	$s = 16, K = 3$	2.03	**2204**	189
		$s = 32, K = 3$	2.02	3100	250
		full batch	1.95	7921	452
	2	$s = 16, K = 3$	2.07	2286	130
		$s = 32, K = 3$	1.96	3186	142
		full batch	**1.89**	8793	538
	3	$s = 16, K = 3$	2.11	2384	**101**
		$s = 32, K = 3$	2.02	4010	120

method. Besides, from the experimental results, we can also observe the efficiency and effectiveness of the proposed sampling based approaches.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their helpful comments. This work is supported by Bloomberg Data Science Ph.D. Fellowship.

REFERENCES

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems* 33 (2020), 17804–17815.

[2] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. *Advances in Neural Information Processing Systems* 33 (2020), 17766–17778.

[3] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *International Conference on Learning Representations*.

[4] Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *International Conference on Machine Learning*. PMLR, 942–950.

[5] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 257–266.

[6] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 922–929.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[8] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems* 31 (2018).

[9] Hosagrahar V Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. 2014. Big data and its technical challenges. *Commun. ACM* 57, 7 (2014), 86–94.

[10] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International Conference on Learning Representations*.

[11] Chao Shang and Jie Chen. 2021. Discrete Graph Structure Learning for Forecasting Multiple Time Series. In *Proceedings of International Conference on Learning Representations*.

[12] Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. 2020. 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. 914–921.

[13] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

[14] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 753–763.

[15] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling.. In *IJCAI*.

[16] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *International Conference on Learning Representations*.

[17] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2020. Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1177–1185.

[18] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1234–1241.

[19] Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. 2019. Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems* 32 (2019).