Make Bricks with a Little Straw: Large-Scale Spatio-Temporal Graph Learning with Restricted GPU-Memory Capacity

Binwu Wang¹, Pengkun Wang¹,²,*, Zhengyang Zhou¹,², Zhe Zhao¹, Wei Xu¹, Yang Wang¹,², *

¹University of Science and Technology of China (USTC), Hefei, China
²Suzhou Institute for Advanced Research, USTC, Suzhou, China

Abstract

Traffic prediction plays a key role in various smart city applications, which can help traffic managers make traffic plans in advance, assist online ride-hailing companies in deploying vehicles reasonably, and provide early warning of congestion for safety authorities. While increasingly complex models achieve impressive prediction performance, there are concerns about the effectiveness of these models in handling large-scale traffic networks. Especially for researchers who don’t have access to powerful GPU devices, the expensive memory burden limits the usefulness of these models. In this paper, we take a step of learning on the large-scale spatio-temporal graph and propose a divide-and-conquer training strategy for Large Spatio-Temporal Graph Learning, namely LarSTL. The core idea behind this strategy is to divide the large graph into multiple subgraphs, which are treated as task streams to sequentially train the model to conquer each subgraph one by one. We introduce a novel perspective based on the continuous learning paradigm to achieve this goal. In order to overcome forgetting the knowledge learned from previous subgraphs, an experience-replay strategy consolidates the learned knowledge by replaying nodes sampled from previous subgraphs. Moreover, we configure specific feature adaptors for each subgraph to extract personalized features, and it is also beneficial to consolidate the learned knowledge from the perspective of parameters. We conduct experiments using multiple large-scale traffic network datasets on a V100 GPU with only 16GB memory, and the results demonstrate that our LarSTL can achieve competitive performance and high efficiency.

1 Introduction

Traffic forecasting plays a vital role in transportation planning, management, and decision-making processes [Miao et al., 2022a; Liu et al., 2024; Liu et al., 2021a; Zhang et al., 2023b]. Accurate predictions enable efficient resource allocation, optimal route planning, and effective traffic management strategies [Wang et al., 2020; Wang et al., 2020]. In recent years, the advancement of data collection and processing capabilities has resulted in a substantial amount of high-quality traffic data. This has sparked significant interest in the research community to develop state-of-the-art spatio-temporal learning models that can effectively extract spatio-temporal features. Currently, the prevailing approach in this field is based on spatio-temporal graph convolutional networks (STGNN). These models typically consist of spatial and temporal components, which are used to model spatial and temporal correlations separately. For example, DCRNN [Li et al., 2017] combines diffusion graph convolutional networks to represent spatial dependencies between nodes and utilizes recurrent neural networks (RNN) to capture temporal dynamics. The generated node representations are then fed into a predictor, often implemented as fully connected layers or a specialized decoder, to forecast future traffic flows.

The accuracy of traffic prediction has shown improvement thanks to the utilization of complex model structures and large parameter volumes. However, it is important to note that these models have primarily been developed and evaluated on small-scale datasets that possess limited road network sizes. For example, the widely used PeMSX datasets (denoted by X as 03, 04, 07, or 08) have a maximum of 883 nodes. In reality, real-world traffic networks can be much larger, often consisting of thousands of nodes. Consequently, the scalability of these models in effectively handling such large networks remains a significant concern [Liu et al., 2023b].

Existing spatio-temporal graph learning models typically take a complete spatio-temporal graph across multiple time segments as input. These models rely on graph convolution operators in GCN to perform spatio-temporal learning. However, training these models presents a significant challenge due to the dependency of loss items on a large number of neighboring nodes [Chiang et al., 2019; Hamilton et al., 2017]. Unlike other neural networks that can decompose training losses into individual items for each sample, STGNNs require storing the embeddings of all nodes within the computed graph from each STGNN layer during the backpropagation process. This storage requirement applies to various architectures, including MLP-based mod-
els [Shao et al., 2022a; Zhang et al., 2023] and Transformer-based models [Xu et al., 2020]. Consequently, when dealing with large-scale graphs, these models demand significant memory allocation that increases with the complexity of model parameters, posing a notable memory burden. While some research efforts aim to improve the training efficiency of STGNNs for large-scale graphs [Wang et al., 2023c; Wang et al., 2023e], we argue that the memory constraint is a decisive factor that directly affects the availability of these advanced models. The memory limitation not only hampers researchers but also poses challenges for Chinese researchers who may face restrictions imposed by manufacturers, limiting their access to high-end GPU devices with substantial memory capacities.

In this paper, we take a step to address the challenge of large-scale spatio-temporal graph learning with limited GPU memory. An instinct is that we can divide a large spatio-temporal graph into multiple small-scale subgraphs, conquer each subgraph individually, and then merge them incrementally. In this way, GPUs only need to store the representation of the subgraph for saving memory. In response, multiple subgraphs are used to train the model sequentially in a task streaming manner. This involves a continual learning paradigm, enabling graph learning from a static large graph to streaming subgraphs. However, the special challenge posed by continual learning is known as the tradeoff between the plasticity of new knowledge and the stability of old knowledge. In such a setting, achieving the balance between shared information among subgraphs and maintaining personalized information for each subgraph is an open question.

To address challenges, we propose a novel framework for large spatio-temporal graph learning, namely LarSTL. Specifically, we first employ the METIS algorithm as a solution to partition a large spatio-temporal graph into multiple subgraphs. Subsequently, we adopt a task streaming approach, treating each subgraph as an independent task, to sequentially train the spatio-temporal graph convolution model. To mitigate the issue of catastrophic forgetting during sequential training, we introduce a continuous learning strategy based on experience replay. This strategy allows us to consolidate the knowledge gained from previous subgraphs by saving a few nodes. Additionally, we utilize feature adaptors to extract personalized features for each subgraph. Through extensive experiments conducted on various large-scale traffic network datasets containing thousands of nodes, we demonstrate that our proposed method achieves competitive predictive performance while maintaining high training efficiency. Our contributions are as follows:

- We take a step to explore large-scale spatio-temporal graph representation learning and propose a divide-and-conquer strategy based on the continuous learning paradigm, namely LarSTL.
- LarSTL first divides a large spatio-temporal graph into small-scale subgraphs and trains a model sequentially in the task streaming manner. In order to effectively consolidate the previously learned knowledge of subgraphs and their individual features, we propose an experiment-replay strategy and feature adaptors from the data and parametric perspectives.

- We evaluate the proposed LarSTL on three large-scale traffic datasets with thousands of nodes, and the experimental results show that our model can achieve competitive performance and high efficiency.

2 Related Work

Traffic forecasting. Recently, the accuracy of spatio-temporal graph prediction has significantly improved with the emergence of deep learning [Zhou et al., 2023; Wang et al., 2023b; Xia et al., 2023; Huang et al., 2024b; Zhou et al., 2020b]. Spatio-Temporal Graph Neural Networks(STGNNs) are the most representative approaches to capture spatio-temporal dependencies in this field [Miao et al., 2022b; Miao et al., 2024]. For example, D2STGNN [Shao et al., 2022b] employs diffusion graph convolutional networks with RNNs to capture temporal patterns. STAEnet [Liu et al., 2023a] proposes Spatio-Temporal Adaptive Embedding transformer. ST-GAT [Zhang et al., 2019] employs graph attention networks with LSTM for efficient capture of time dependencies to improve the prediction performance of the Transformer. SSTBAN [Guo et al., 2023] follows a multitask framework by incorporating a self-supervised learner to produce robust latent representations for historical traffic data. Some works move to solving the problem of low training efficiency, for example, STHMLP [Qin et al., 2023] and ST-MLP [Wang et al., 2023f] are committed to introducing an all-MLP architecture. PFNet [Wang et al., 2023c] captures temporal correlations using deep multi-view sequence encoders and spatial correlations using graph embedding technologies. CANet [Wang et al., 2023e] effectively captures spatial dependencies by mining the implicit spatial centers of variables. Overall, when dealing with large road networks, these models are difficult to scale to large network structures due to GPU memory requirements.

Large-scale graph learning. Large-scale graph learning has garnered significant interest in various domains [Zhou et al., 2020a; Wu et al., 2020], such as recommended tasks and graph classification tasks. For example, GraphSage [Hamilton et al., 2017] employs a sampling strategy where a tree is sampled with each node as the root, progressively expanding the neighborhood by K steps with a fixed sample size. This method computes the hidden representation of the root node by hierarchically aggregating hidden representations of nodes from the bottom to the top of the sampled tree. PinSage [Ying et al., 2018] proposes short random walks to generate subgraphs for convolution operation to generate graph representation for web-scale recommendation systems. Additionally, Cluster-GCN [Chiang et al., 2019] leverages a graph pooling algorithm to sample a few nodes and subsequently performs graph convolutions on this node. SSE [Dai et al., 2018] updates node hidden states recurrently in a stochastic and asynchronous fashion. It alternatively samples a batch of nodes for state update and a batch of nodes for gradient computation. However, these models specifically designed for recommendation and node classification tasks cannot model spatio-temporal dependencies and cannot be directly applied to spatio-temporal graph learning.
3 Problem Definitions

We use a graph \( G = (V, E, A) \) to represent a large-scale traffic network, where \( V \) denotes the set of nodes, and \( E \) represents the weight between the nodes. \( A \in \mathbb{R}^{N \times N} \) represents the adjacency matrix of \( G \), where \( N \) means the number of nodes and is very large for a large-scale graph. Additionally, let \( X_i = \{x_1, x_2, ..., x_N\} \) represent the historical graph data at \( t \)-th time step, and \( x_i \in \mathbb{R}^d \) represents the state of \( i \)-th node, where \( d \) means the number of features. Given \( G \) and observations of past \( T \) time steps \( X \in \mathbb{R}^{T \times N \times d} = \{X_1, ..., X_T\} \), the spatio-temporal graph prediction task aims to learn a spatio-temporal prediction model with parameter \( \omega \), which can effectively predict graph signals next \( T_p \) time steps \( Y \in \mathbb{R}^{T_p \times N \times d} = \{X_{T+1}, ..., X_{T+T_p}\} \). The parameters \( \omega \) can be decoupled into two parts: spatio-temporal feature extractor \( \phi \) and a predictor \( \varphi \) which can be optimized in an end-to-end manner:

\[
\omega (\phi, \varphi) = \arg\min_{\phi, \varphi} \| Y - \hat{Y} \|^2 \tag{1}
\]

where \( \hat{Y} \) and \( Y \) represent the prediction and ground-truth values, respectively.

4 Method

In this section, for clarity, we first introduce a simple spatio-temporal graph model as the backbone, namely BasSTG. Subsequently, we describe the proposed division-and-conquest strategy, which is based on the continuous learning paradigm to facilitate large-scale spatio-temporal graph learning with BasSTG under limited GPU memory. The details and pseudocode of our proposed method LarSTL are shown in Figure 1 and Algorithm 1.

4.1 Spatio-Temporal Graph Model

GCNs have demonstrated their effectiveness for graph structures across various tasks [Huang et al., 2024a; Wang et al., 2024; Du et al., 2023]. We introduce a simple spatio-temporal graph convolution network as a backbone, BasSTG, which includes two spatio-temporal (ST) layers. Each ST layer includes a diffusion graph convolution module and a TCN module. Specifically, given an adjacency matrix \( A_t \in \mathbb{R}^{N \times N} \) and the representation of the \( l \)-th layer \( \mathbf{H}_l \in \mathbb{R}^{N \times T \times C_l} \) as input, where \( N \) means the number of nodes in the input graph, \( T \) means the length of time steps, and \( C_l \) means the number of channels, a diffusion graph convolution layer is as
After dividing graph $G$ into multiple overlapping subgraphs \( \{g_1, \cdots, g_K\} \), where $K$ means the number of subgraphs. This series of subgraphs is treated as a task streaming to train the model sequentially. The model can continuously absorb new knowledge from the task streaming to adapt to different subgraphs. After such training, the model is expected to extract shared features among subgraphs, while maintaining the personalized features of each subgraph, so that it can effectively make predictions for the entire large graph.

However, when we fine-tune the model with subgraph $G_{\tau}$, the model may forget previously learned knowledge from the previous subgraph $G_{\tau-1}$, i.e., catastrophic forgetting. This notorious phenomenon causes the decline of the model’s prediction performance for previous tasks. To mitigate catastrophic forgetting, continual graph learning has been actively investigated in various fields [Wang et al., 2023a; Liu et al., 2021b]. In this paper, we propose two strategies based on experience replay and parameter adaptor to consolidate old knowledge from data and parameter perspectives.

**Experience-replay strategy.** In the realm of continuous learning, the effectiveness of replaying data samples from previous tasks in consolidating old knowledge has been demonstrated [Wang et al., 2023a]. Taking inspiration from this approach, we adopt a similar strategy in our framework. After each training iteration, we select a subset of nodes from the subgraph and add them to the node container $B$. In the subsequent training process, the nodes in the container $B$, along with their N-hop neighbors, are utilized to fine-tune the saved model alongside nodes from a new subgraph.

In order to improve the consolidation effect, these replay nodes should representatively cover what the model has learned in the previous task. Naturally, the model is compatible with these nodes. Based on the above analysis, during $\tau - 1$-th learning step, we sort the model’s prediction performance for all nodes only within $G_{\tau-1}$ in the validation dataset. Then we select top $K_{\%} \%$ nodes with small prediction errors as the replay nodes. However, some nodes with large prediction errors need to be re-learned, which can improve the prediction accuracy. Therefore, $K_{\%} \%$ nodes with large errors are also selected. After training, these nodes are added into $B$. When learning subgraph $G_{\tau}$, these nodes are trained together with the new subgraph. In fact, these cross-subgraph nodes facilitate information interaction between different subgraphs.

**Feature adaptor.** Striking a balance between extracting new knowledge and consolidating previous knowledge poses a challenge in continuous learning. To tackle this issue, we propose the development of individualized feature adaptors for each subgraph. These adaptors are designed to extract personalized features from each subgraph, empowering the model to effectively capture and comprehend the specific spatio-temporal information within each subgraph.

Specifically, we construct an individual adaptor, $\theta_i$, for $i$-th subgraph, which is a learnable parameter matrix. The shapes of these matrices are consistent with the ones of each parameter matrix in the spatio-temporal feature extractor $\phi$. Then this adaptor $\theta_i$ is used to scale the parameters $\phi$ by dot product: $\theta_i \circ \phi$. In essence, the adaptor can be viewed as a learnable attention weight that characterizes the importance of spatio-temporal features for this subgraph, i.e. personalized spatio-temporal features. And adaptors are trained end-
Algorithm 1 Large-scale spatio-temporal graph learning

Input: A large-scale spatio-temporal graph $G$, observation historical graph data $X$.
Parameter: spatio-temporal feature extractor $\phi$, feature adaptor $\theta$, Predictor $\varphi$.
Output: A spatio-temporal prediction function with optimal parameters $\omega(\phi, \theta, \varphi)$.

1: Using METIS algorithm to divide $G$ into $k$ overlapping subgraphs $\{g_1, ..., g_k\}$.
2: while $\{g_1, ..., g_k\}$ do
3:     if $\tau == 1$ then
4:         Random initialization parameters $\omega_1(\phi, \theta_1, \varphi_1)$.
5:     end if
6:     Initial replay node container $B$.
7:     Train the model with the data of the subgraph $g_1$.
8: else
9:     Random initialization parameters $\theta_\tau$ and $\varphi_\tau$.
10:    Load the saved parameters $\theta$ in $\omega_{\tau-1}$.
11:   end if
12: end while
13: for $\{g_1, ..., g_k\}$ do
14:     Add replay nodes in $g_\tau$ into the container $B$.
15:     Make predictions for the subgraph $g_\tau$ with $\omega_{\tau}(\phi, \theta_\tau, \varphi_\tau)$.
16: end for

5 Experiment

In this chapter, we evaluate the effectiveness of LarSTL on a large-scale network dataset with thousands of nodes and answer the following underlying concerns: Q1. What is the prediction performance and efficiency of the model? Please refer to Section 5.2. Q2. Is every component of the method valid? Please refer to Section 5.3. Q3. How sensitive is the model to the number of subgraphs and the graph partition algorithm? Please refer to Section 5.4.

5.1 Experiment Setting

Data. We evaluate the validity of the model on three large traffic datasets, GLA, GBA, and CA, proposed by from [Liu et al., 2023b]. These three emerging datasets, with thousands of nodes, are far larger than those previously used in the traffic forecasting community [Liu et al., 2023b]. All datasets are collected from the California Department of Transportation (CalTrans) Performance Measurement System (PeMS) and aggregate the traffic readings from 5-minute intervals into 15-minute windows. The details of traffic networks in each dataset are shown in Table 1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CA</th>
<th>GLA</th>
<th>GBA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>8,600</td>
<td>3,834</td>
<td>2,352</td>
</tr>
<tr>
<td>Edges</td>
<td>98,703</td>
<td>201,363</td>
<td>61,246</td>
</tr>
<tr>
<td>Degree</td>
<td>23.4</td>
<td>25.7</td>
<td>26</td>
</tr>
<tr>
<td>Time Range</td>
<td>2018</td>
<td>2019</td>
<td>2020</td>
</tr>
<tr>
<td>Train frames</td>
<td>21,010</td>
<td>21,010</td>
<td>21,010</td>
</tr>
<tr>
<td>Val/test frames</td>
<td>7,003/7,004</td>
<td>7,003/7,004</td>
<td>7,003/7,004</td>
</tr>
</tbody>
</table>

Table 1: The details of the three datasets.

Restricted GPU device. All models are trained on a V100 GPU with 16G memory. We give priority to running a model with the optimal hyperparameters from their official codes. And we specify a maximum batch size of 64. If a model is unable to run with this setting, we would reduce the hyperparameter and the batch size of the models until it fully occupies the memory on this V100 GPU. Eventually, if the model still doesn’t work, we consider this model to be out of memory.

Experiment setting. We chronologically split the data into train, validation, and test sets, with a ratio of 6:2:2 for all datasets. AdamW [Loshchilov and Hutter, 2017] is used as the optimizer with an initial learning rate of 0.001. We use an early stopping strategy to accelerate the training process. We divide the entire graph into three subgraphs. To evaluate the performance of our framework, we adopt Mean Absolute Errors (MAE), Root Mean Squared Errors (RMSE), and Mean Absolute Percentage Errors (MAPE) as the metrics for effectiveness. The replay node numbers $K_r$ and $K_c$ are set to 1%. We report the overall efficiency by considering the total training time and evaluation time of all models, along with the average training time per epoch.

Baseline. We compare some advanced spatio-temporal prediction models as follows:

- LSTM is an RNN variant to model long-term temporal dependencies.
- TCN is a lightweight temporal modeling framework using causal convolution.
- DCRNN [Li et al., 2017] introduces graph diffusion graph convolutional networks and LSTM to capture long-term dependencies.
- Z-GCNETs [Chen et al., 2021] propose to enhance DL architectures with the most salient time-conditioned topological information of the data.
- AGCRN [Bai et al., 2020] is also a RNN-based spatio-temporal graph learning model.
- ASTGCN [Guo et al., 2019] focuses on adaptive modeling of spatial dependencies using attention mechanism.
- HGNCN [Guo et al., 2021] designs hierarchical GCN to capture both the spatial relationships of microscopic nodes and relationships between macroscopic regions.
### GBA

<table>
<thead>
<tr>
<th>Model</th>
<th>Horizon 3</th>
<th>Horizon 6</th>
<th>Horizon 12</th>
<th>All Time</th>
<th>Train Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCN</td>
<td>26.78</td>
<td>42.45</td>
<td>21.87</td>
<td>34.99</td>
<td>49.13</td>
</tr>
<tr>
<td>LSTM</td>
<td>25.64</td>
<td>39.15</td>
<td>22.42</td>
<td>31.59</td>
<td>48.95</td>
</tr>
<tr>
<td>HGCN</td>
<td>25.57</td>
<td>37.93</td>
<td>21.43</td>
<td>27.19</td>
<td>44.26</td>
</tr>
<tr>
<td>Z-GCNETs</td>
<td>25.62</td>
<td>43.27</td>
<td>19.56</td>
<td>30.16</td>
<td>47.53</td>
</tr>
<tr>
<td>ASTGCN</td>
<td>24.71</td>
<td>36.42</td>
<td>22.73</td>
<td>26.09</td>
<td>40.57</td>
</tr>
<tr>
<td>DCRNN</td>
<td>19.17</td>
<td>31.30</td>
<td>17.72</td>
<td>23.74</td>
<td>36.91</td>
</tr>
<tr>
<td>AGCRN</td>
<td>19.02</td>
<td>29.64</td>
<td>17.26</td>
<td>23.61</td>
<td>34.92</td>
</tr>
<tr>
<td>STGODE</td>
<td>19.32</td>
<td>29.48</td>
<td>20.11</td>
<td>24.26</td>
<td>36.69</td>
</tr>
<tr>
<td>D2STGNN</td>
<td>21.42</td>
<td>31.10</td>
<td>19.74</td>
<td>23.75</td>
<td>40.10</td>
</tr>
<tr>
<td>Ours</td>
<td>19.28</td>
<td>30.45</td>
<td>17.84</td>
<td>23.15</td>
<td>35.96</td>
</tr>
</tbody>
</table>

### GLA

<table>
<thead>
<tr>
<th>Model</th>
<th>Horizon 3</th>
<th>Horizon 6</th>
<th>Horizon 12</th>
<th>All Time</th>
<th>Train Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCN</td>
<td>27.23</td>
<td>44.21</td>
<td>18.81</td>
<td>35.77</td>
<td>50.46</td>
</tr>
<tr>
<td>LSTM</td>
<td>24.86</td>
<td>37.92</td>
<td>13.94</td>
<td>32.97</td>
<td>48.48</td>
</tr>
<tr>
<td>HGCN</td>
<td>25.75</td>
<td>43.77</td>
<td>14.93</td>
<td>31.35</td>
<td>46.14</td>
</tr>
<tr>
<td>Z-GCNETs</td>
<td>23.04</td>
<td>37.05</td>
<td>13.68</td>
<td>28.28</td>
<td>43.56</td>
</tr>
<tr>
<td>ASTGCN</td>
<td>24.73</td>
<td>41.27</td>
<td>14.97</td>
<td>31.18</td>
<td>51.31</td>
</tr>
<tr>
<td>DCRNN</td>
<td>22.02</td>
<td>33.95</td>
<td>12.10</td>
<td>27.67</td>
<td>41.74</td>
</tr>
<tr>
<td>AGCRN</td>
<td>21.01</td>
<td>32.88</td>
<td>11.51</td>
<td>26.80</td>
<td>40.94</td>
</tr>
<tr>
<td>STGODE</td>
<td>20.74</td>
<td>32.41</td>
<td>11.48</td>
<td>27.06</td>
<td>41.21</td>
</tr>
<tr>
<td>DSTAGNN</td>
<td>23.08</td>
<td>36.01</td>
<td>12.78</td>
<td>29.96</td>
<td>48.16</td>
</tr>
<tr>
<td>Ours</td>
<td>20.35</td>
<td>31.92</td>
<td>11.20</td>
<td>26.16</td>
<td>40.27</td>
</tr>
</tbody>
</table>

### CA

<table>
<thead>
<tr>
<th>Model</th>
<th>Horizon 3</th>
<th>Horizon 6</th>
<th>Horizon 12</th>
<th>All Time</th>
<th>Train Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCN</td>
<td>26.28</td>
<td>42.19</td>
<td>21.72</td>
<td>35.77</td>
<td>51.34</td>
</tr>
<tr>
<td>LSTM</td>
<td>27.12</td>
<td>36.41</td>
<td>20.65</td>
<td>31.26</td>
<td>49.23</td>
</tr>
<tr>
<td>HGCN</td>
<td>25.51</td>
<td>38.74</td>
<td>19.51</td>
<td>30.04</td>
<td>47.52</td>
</tr>
<tr>
<td>Z-GCNETs</td>
<td>23.69</td>
<td>36.04</td>
<td>18.33</td>
<td>28.41</td>
<td>43.92</td>
</tr>
<tr>
<td>ASTGCN</td>
<td>26.13</td>
<td>40.46</td>
<td>19.77</td>
<td>30.98</td>
<td>49.02</td>
</tr>
<tr>
<td>DCRNN</td>
<td>22.12</td>
<td>35.16</td>
<td>17.00</td>
<td>26.95</td>
<td>42.71</td>
</tr>
<tr>
<td>STGODE</td>
<td>21.93</td>
<td>34.34</td>
<td>16.89</td>
<td>26.82</td>
<td>42.23</td>
</tr>
<tr>
<td>DSTAGNN</td>
<td>25.76</td>
<td>41.29</td>
<td>19.70</td>
<td>31.03</td>
<td>48.99</td>
</tr>
<tr>
<td>Ours</td>
<td>21.42</td>
<td>34.15</td>
<td>16.46</td>
<td>26.17</td>
<td>41.42</td>
</tr>
</tbody>
</table>

Table 2: Prediction performance and efficiency of each model. 'All Times' means total training time and evaluation time (in hours). We also report time cost per epoch (in seconds) for efficiency comparison. The best results are bold, and the second best results are underlined.
• DGCRN [Li et al., 2023] leverages and extracts dynamic characteristics from node attributes, while the parameters of dynamic filters are generated at each time step.
• STGODE [Fang et al., 2021] uses neural ordinary differential equations to effectively model the continuous changes of traffic signals.
• DSTAGNN [Lan et al., 2022] uses a dynamic spatial-temporal aware graph to replace the predefined static graphs used in traditional graph convolution.
• D^2STGNN [Shao et al., 2022b] proposes a unique estimation gate with residual decomposition mechanism to decouple traffic signals into diffusion and inherent traffic information.

5.2 Performance and Efficiency Analysis (Q.1)
The prediction performance of each model on the three datasets is presented in Table 2, showcasing the competitive efficiency and prediction accuracy of our model.

Comparing different methods on the GBA, we observe that temporal models, such as TCN and LSTM, perform poorly because they fail to capture spatial information efficiently. DGCRN and D^2STGNN demonstrate impressive performance on the GBA datasets, highlighting the importance of considering the dynamic features of spatial topology. However, their performance can not meet expectations because the device’s memory capacity limited the size of their parameters, resulting in inadequate learning of the spatio-temporal information. Meanwhile, their complex model designs limit their scalability when applied to larger datasets like GLA and CA datasets. On the other hand, RNN-based models, such as DCRNN and AGCRN, maintain good predictive performance across multiple datasets, which can be attributed to their ability to effectively capture spatio-temporal information. However, when running in the limited GPU, these models all struggle to effectively handle the large road network, resulting in a rapid deterioration of prediction performance. In terms of efficiency, TCN-based approaches such as Z-GCNETs and HGCN are generally faster than other baselines, thanks to their parallel computation of temporal convolution operations. Among the RNN-based models, DCRNN and AGCRN are slower than TCN-based methods due to the use of RNNs as decoders.

Our model achieves superior performance and efficiency, especially for large-scale datasets, thanks to the proposed divide-and-conquer strategy, which reduces the learning complexity of spatio-temporal information.

5.3 Ablation Experiment (Q.2)
In this section, we evaluate the effectiveness of each component. We make the following variants: w/o tem means that we remove the temporal module in LarST. w/o rep means that we remove the experience-replay strategy for consolidating old knowledge learned from the previous subgraphs. w/o adp means that we directly fine-tune the saved model each time instead of configuring feature adaptors for each subgraph. The prediction results of each variant are shown in Figure 2.

We can see that each variant is more or less inferior to our proposed model, proving the validity of each component of the model. The prediction performance of w/o tem is not good, this is because it does not model temporal correlations and has not learned sufficient spatiotemporal information, resulting in suboptimal prediction performance. The prediction error of w/o adp is high, which indicates that the feature adaptor can effectively help the model consolidate the old knowledge. At the same time, it shows that learning the personalized features of each subgraph improves the model’s prosperity in the new subgraph.

5.4 Hyperparameter Experiment (Q.3)
We assess the impact of the number of graph partitions on clustering and its subsequent effect on model prediction efficacy. The outcomes derived from a 60-horizon analysis on the GLA dataset are presented in Table 3, revealing that optimal model performance is achieved when $K$ is equal to 3. This because that a higher value of $K$ results in clusters with fewer nodes, which fails to furnish adequate spatio-temporal data, leading to model underfitting. Moreover, such a setup causes the model to readily forget previously learned information when it encounters new knowledge within current subgraphs. Conversely, a small value of $K$ implies a larger number of nodes per cluster, and the scale of model parameters is constrained by GPU memory, resulting in underfitting of the model.

<table>
<thead>
<tr>
<th>$K$</th>
<th>MAE</th>
<th>RMSE</th>
<th>MAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>34.13</td>
<td>53.32</td>
<td>20.72</td>
</tr>
<tr>
<td>3</td>
<td><strong>34.01</strong></td>
<td><strong>52.89</strong></td>
<td><strong>20.26</strong></td>
</tr>
<tr>
<td>4</td>
<td>34.62</td>
<td>54.07</td>
<td>21.48</td>
</tr>
<tr>
<td>5</td>
<td>35.49</td>
<td>55.64</td>
<td>22.63</td>
</tr>
</tbody>
</table>

Table 3: Sensitivity analysis of hyperparameter $K$.

6 Conclusion
We propose an efficient approach for large-scale spatio-temporal graph learning, employing a divide-and-conquer strategy. Our method partitions a large graph into smaller subgraphs, allowing individual processing and analysis. To enhance the learning process, we introduce two strategies: experience replay and feature adaptator. Experience replay strategy can consolidate knowledge from preceding subgraphs, while the feature adaptator strategy ensures scalability for incorporating new knowledge from subsequent subgraphs. We comprehensively evaluate our approach on datasets featuring large traffic networks, demonstrating its effectiveness and efficiency in large-scale spatio-temporal graph learning under limited GPUs.
Acknowledgments

This paper is supported by the National Natural Science Foundation of China (No.62072427, No.12227901), the Project of Stable Support for Youth Team in Basic Research Field, CAS (No.YSBR-005), and Academic Leaders Cultivation Program, USTC.

References


[Li et al., 2023] Fuxuan Li, Jie Feng, Huan Yan, Guangyin Jin, Fan Yang, Funing Sun, Depeng Jin, and Yong Li. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. ACM Transactions on Knowledge Discovery from Data, 2023.


[Miao et al., 2022a] Hao Miao, Yan Fei, Senzhang Wang, Fang Wang, and Danyan Wen. Deep learning based origin-


