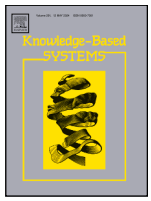
Knowledge-Based Systems 330 (2025) 114428

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



A unified multi-subgraph pre-training framework for spatio-temporal graph

Mingze Zhong [a,*], Zexuan Long [b], Xinglei Wang [c], Tao Cheng [c], Meng Fang [d], Ling Chen [a]

[a] Australian Artificial Intelligence Institute, University of Technology Sydney, Ultimo, Sydney, 2007, NSW, Australia
[b] Shenzhen University, Shenzhen, 518060, Guangdong, China
[c] University College London, Gower Street, London, WC1E 6BT, United Kingdom
[d] University of Liverpool, Ashton Street, Liverpool, L69 3BX, United Kingdom

ARTICLE INFO

Keywords:
Graph neural network
Pre-training
Spatio-temporal graph
Multi-subgraphs
Time decay.

ABSTRACT

Spatio-temporal graph (STG) learning has shown great potential in capturing complex spatio-temporal dependencies and has achieved significant success in various fields such as traffic flow prediction, climate forecasting, and epidemiological spread research. By learning general features from spatio-temporal graphs, pre-trained graph models can capture hidden semantic information in the data, thereby enhancing the learning effect of downstream tasks and improving overall model performance. However, most existing spatio-temporal graph learning methods use the entire graph for training, which may not fully capture local structure and feature information. In addition, existing methods usually adopt sequence modeling techniques without fully considering the time decay effect, i.e., the need to apply decaying attention to distant time steps. To address these issues, this paper proposes a unified dual-phase multi-subgraph pre-training spatio-temporal graph framework (UMSST). Specifically, in the first phase, the framework learns the global representation of the spatio-temporal graph and locates key graph nodes, while learning the "unit representations" of these key nodes. In the second phase, multiple spatio-temporal subgraphs are constructed based on these "unit representations" to further capture the implicit encoding information of more general features around the corresponding subgraphs, thereby helping the model make full use of general features. Experimental results on real datasets show that the proposed pre-trained spatio-temporal graph framework significantly improves the performance of downstream tasks and demonstrates its effectiveness in comparison with recent strong baseline models.

1. Introduction

Spatio-temporal graph (STG) modeling has become one of the most promising techniques due to its ability to learn complex spatial and temporal dependencies. Effective STG learning has achieved success in many real-world applications, such as traffic flow forecasting in intelligent transportation systems [33], epidemic prediction for public health management [5], urban crime prediction for public safety [22], etc. However, most existing STG learning methods adopt the entire graph for training, potentially failing to fully capture local regional and feature information. Furthermore, existing methods typically employ sequence modeling techniques without considering the time decay effect, which requires applying decaying attention to distant time steps [18,23,37,38].

The main challenge is how to effectively pretrain models to represent highly localized structures and temporal dynamics within large-scale STGs. Existing whole-graph approaches, while capturing global patterns, can overlook critical local events such as traffic hotspots or epidemic epicenters. Furthermore, they often treat all past observations equally, ignoring the crucial principle of temporal decay, where older events have diminishing relevance [1,17,30]. Moreover, full-graph pre-training on fine-grained data becomes computationally prohibitive. Our multi-subgraph approach directly confronts these issues by learning detailed representations within key local regions, which not only enhances expressive power by capturing complex local topologies but also enables scalable, parallelized training to reduce the computational burden.

Inspired by the latest progress of graph pre-training models in learning rich latent information from graph structures and node features [21,31], this paper proposes a unified and generalizable STG pre-training framework by exploring the following questions:

- How can we effectively pretrain the STG using large spatio-temporal datasets?
- How can we achieve better node representation learning on STGs, especially focusing on the representation of local regions in the graph,

* Corresponding author.
E-mail addresses: Mingze.Zhong@Student.uts.edu.au (M. Zhong), longzexuan2023@email.szu.edu.cn (Z. Long), xinglei.wang.21@ucl.ac.uk (X. Wang), tao.cheng@ucl.ac.uk (T. Cheng), Meng.Fang@liverpool.ac.uk (M. Fang), ling.chen@uts.edu.au (L. Chen).

https://doi.org/10.1016/j.knosys.2025.114428
Received 13 June 2025; Received in revised form 24 August 2025; Accepted 4 September 2025
Available online 13 September 2025

to address the issue that training the entire graph may not capture local regional and feature information?
- How can we account for the time decay effects?

To address these challenges, this paper proposes a Unified Multi-Subgraph pretraining framework for Spatio-Temporal graph (UMSST), which constructs multi-subgraphs in two phases to pretrain node representations. The motivation for this two-stage pre-training design is to enhance the modeling capability of local information and improve training efficiency. Specifically:

- **First Phase (Coarse Phase):** The graph embedding of the STG (whose dimension is consistent with the node embedding dimension) is computed through self-supervised learning, and the nodes with the highest correlation with the graph embedding are identified as subgraph centers. This phase is used to detect the locations in the graph that contain important information and their approximate scope. The basic assumption is that the higher the correlation, the more information of the entire STG the node and its vicinity contain. These preliminarily identified regions are considered the basis for "unit representations".
- **Second Phase (Fine Phase):** Based on the local regions determined in the first phase, multiple subgraphs are constructed and trained more finely to obtain more detailed implicit information about potentially more critical regions. This phase focuses on learning richer latent information from these subgraphs.

By combining the representations learned in the two phases, the final node representation is obtained. Since this pre-training framework is model-agnostic, the node representations can be utilized by any existing STG learning model to improve performance. For large-scale graphs, focusing on learning key localized regions rather than detailed learning of the entire graph can bring significant improvements in learning scale, complexity, and learning efficiency, which also indirectly clarifies the reason for adopting a two-stage pre-training method.

The contributions of this paper are summarized as follows:

- The proposed pre-training model framework is a general framework targeting different local regions of the STG and can be applied to any STG learning models in a plug-and-play manner, validated by experiments on multiple base models.
- The proposed multi-subgraph method can capture the underlying high-level semantics more comprehensively than a single subgraph.
- We reveal the decay characteristics of STG learning in the time dimension and experimentally verify its potential positive impact on model performance.
- Experiments on real-world datasets, including comparisons with recent strong baseline models, demonstrate the effectiveness of the proposed STG pre-training framework.
- A complexity analysis is provided, demonstrating the potential computational efficiency of the multi-subgraph method, especially for large-scale graphs.

2. Related work

2.1. Graph pre-training methods

Pre-training of graph models has made significant strides in prior research. A series of works employed transfer learning to enhance the model's expressive power for representing graph structure. Subsequent research implemented pre-training tasks with significantly larger scales of parameters. To bridge the gap between pre-training and downstream tasks, researchers proposed injecting enhanced knowledge into the model during pre-training [32], while other work simulated fine-tuning operations by introducing new tasks into the pre-training process [9]. These models were all applied to ordinary graphs, and it was only in recent years that pre-training models for spatio-temporal graphs (STGs) began to emerge.

For example, a model named STEP [2] proposed to incorporate long-term historical time series during pre-training. This design was proven effective but neglected abundant spatial dependencies. Another work adopted adversarial contrastive samples to enhance self-supervised learning (ST-SSL) on STGs [28]. However, these adversarial samples were constructed purely based on the STG's spatial structure, overlooking the temporal aspect. A more recent work proposed a generative pre-training model for STGs (GPT-ST) [17], utilizing a hierarchical hypergraph structure to capture spatial dependencies at different levels. Our work differs in that we construct multi-subgraphs instead of a hypergraph to capture spatial dependencies, focusing more on learning refined representations of local regions, whereas hypergraph methods typically focus more on global or predefined cluster representations. UMSST, through a two-stage process, first identifies globally important regions and then constructs and learns multiple (potentially overlapping) subgraphs on these regions, aiming to capture local features more finely.

2.2. Spatio-temporal graph learning

Recent years have seen a surge in spatio-temporal graph (STG) learning methods. Among the various models proposed, STSGNN [24] and STFGNN [15] stood out for their construction of STGs with temporally adjacent connections and the design of spatio-temporal synchronous graphs to adaptively capture spatio-temporal correlations. S2TAT [29] adopted a spatio-temporal synchronous Transformer framework, leveraging attention mechanisms to enhance learning capabilities. Auto-STG [11] and AutoSTS [16] integrated neural architecture search (NAS) methods into STG learning models to achieve optimal neural network structures. Another work utilizing NAS emphasized determining the optimal adjacency matrix for the graph. In contrast to previous work focusing on neural architecture design, our framework learns more fundamental STG representations through pre-training, which can benefit various neural architectures. UMSST aims to provide a general pre-training module to enhance the capabilities of these existing downstream models. Beyond pre-training, significant progress has been made in designing novel architectures for STG learning. For instance, COOL [13] proposes a conjoint framework that models heterogeneous graphs from both prior and posterior information to capture high-order spatio-temporal relationships, utilizing affinity and penalty graphs to refine node representations. Another line of research explores continuous-time dynamics. Methods like GDERec [20], which learns a graph ordinary differential equation (ODE), are designed for sequential recommendation on irregularly-sampled interaction data, modeling the continuous evolution of the user-item graph. These approaches introduce powerful, specialized architectures for specific problem settings. In contrast, our UMSST framework is designed to be model-agnostic. Instead of proposing a new end-to-end architecture, it serves as a general-purpose pre-training module that can enhance a wide variety of existing downstream models by providing them with richer, pre-trained representations of local structures.

2.3. Alternative graph structures in representation learning

The broader field of deep graph representation learning has explored structures beyond standard pairwise graphs to capture more complex relationships [12]. A notable example is the use of hypergraphs, where an edge can connect any number of nodes. Frameworks like HEAL [14] leverage hypergraphs for semi-supervised graph classification, designing learnable hypergraph structures to capture higher-order node dependencies. While powerful, these methods fundamentally alter the data representation. UMSST, however, operates on standard spatio-temporal graphs, making it directly applicable to a vast range of existing datasets and models without requiring a shift to a hypergraph formulation. Our multi-subgraph approach can be seen as an alternative strategy to capture complex local dependencies within the standard graph paradigm.

2.4. Graph data augmentation

Representation learning on sparse graphs is challenging. Typically, self-supervised learning (SSL) is combined with data augmentation to learn graph representations that benefit subsequent training. Currently, research on data augmentation primarily focuses on three aspects: **Feature Augmentation**: GraphMAE [9] employed a novel masking strategy and scaled cosine error for graph feature reconstruction. **Structure Augmentation**: CSSL [34] augmented graphs through a sequence of graph transformation operations and trained the graph encoder with self-supervised contrastive learning. **Label Augmentation**: DPGNN [36] applied an unbalanced label propagation mechanism to supervise unlabeled nodes; CGCN [35] combined a Gaussian mixture model with a variational graph autoencoder to generate pseudo-labels for nodes. Moreover, integrating various enhancement techniques has proven effective. For example, JOAO [10] dynamically augmented data of different scales and types by adopting an enhanced perception projection head mechanism. BGRL [25] achieved SSL by predicting inputs as enhanced information, thereby eliminating the need for constructing negative samples. Compared to previous work, the multi-subgraph method proposed in this paper, by capturing finer information about local regions within a graph, can be considered an implicit form of data augmentation as it exposes the model to diversified local contexts.

3. Preliminaries

3.1. Definitions

Graph. A graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}| = N$, and $\mathcal{E}$ is the set of edges. Its topological structure is typically represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$.

Spatio-Temporal Graph (STG). A spatio-temporal graph is a special graph structure where the attributes (or features) of the nodes change dynamically over time. Formally, an STG at a specific time step t can be represented as $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}, \mathbf{X}_t)$, where $\mathcal{V}$ is a fixed set of nodes, $\mathcal{E}$ is a fixed set of edges, and $\mathbf{X}_t \in \mathbb{R}^{N\times C}$ is the node feature matrix at time step t, with C being the feature dimension. A sequence of STG data can be denoted as $\{\mathcal{G}_{t-T+1}, \cdots, \mathcal{G}_t\}$, and the corresponding feature tensor is $\mathbf{X}_{t-T+1:t} \in \mathbb{R}^{T\times N\times C}$.

3.2. Problem statement

Given the historical graph data $\mathbf{X}_{t-T+1:t}$ up to the current time step t, the goal is to learn a predictive function f that can accurately estimate the node attributes at the future time step $t+1$, i.e., $\mathbf{X}_{t+1} \in \mathbb{R}^{|\mathcal{V}|\times C}$.

4. Methodology

This section delineates the architecture of our proposed Unified Multi-Subgraph Spatio-Temporal graph pre-training (UMSST) model, whose overarching architecture is depicted in Fig. 1. The architecture of our pre-training model comprises two phases.

4.1. First phase: global representation learning and key node identification

4.1.1. Initial spatio-temporal representation

To process raw data containing spatio-temporal information of an STG and concurrently model the sequential relationships in the temporal dimension and the correlations among nodes in the spatial dimension, we urgently require a method for encoding the raw data. With this objective in mind, we plan to integrate the techniques of temporal convolution and graph convolution to represent spatio-temporal relationships more effectively. In our approach, we employ 1-dimensional causal convolutions along the temporal dimension, supplemented by gated mechanisms, to encode information within the temporal dimension. Specifically, the temporal convolution operation accepts a tensor $\mathbf{X}_{t-T+1:t}$ as input. Let the tensor $\mathbf{H}^{\mathcal{T}}_{t-T_{\text{out}}:t}$ denote the outputs and f_{TC} denote the function of temporal convolution, constituting time-aware embeddings for each node within the temporal dimension. This methodology is designed to capture and encode the spatio-temporal dynamics inherent to each node, thereby facilitating a further understanding of spatio-temporal sequences:

$$\mathbf{H}^{\mathcal{T}}_{t-T_{\text{out}}:t} = f_{TC}(\mathbf{X}_{t-T+1:t}), \tag{1}$$

$$\mathbf{H}^{\mathcal{T}}_{t-T_{\text{out}}:t} = (\mathbf{H}^{\mathcal{T}}_{t-T_{\text{out}}}, \cdots, \mathbf{H}^{\mathcal{T}}_{t}), \tag{2}$$

where $\mathbf{H}^{\mathcal{T}}_t \in \mathbb{R}^{N\times D}$ denotes the embedding matrix of the node at time step t after convolutional operations on the temporal convolution encoder. Especially, the n-th row $\mathbf{h}^{\mathcal{T}}_{t,n} \in \mathbb{R}^D$ in $\mathbf{H}^{\mathcal{T}}_t$ denotes the embedding of node v_n at the time step t after convolutional operations in the temporal convolution encoder. Here, D denotes the embedding dimensionality, and T_{out} is the length of the output embedding sequence after convolutional operations in the temporal convolution encoder.

Let f_{SC} denote the function of spatial convolution. We design a spatial convolution encoder to capture spatial correlations in space. This encoder is based on a graph-based message-passing mechanism presented as follows:

$$\mathbf{H}^{S}_t = f_{SC}(\mathbf{H}^{\mathcal{T}}_t, \mathbf{A}), \tag{3}$$

where $\mathbf{A}$ is the adjacency matrix of graph G. Now, we can get more refined embeddings of all nodes by merging the spatial context after the spatial convolution encoder:

$$\mathbf{H}^{S}_{t-T_{\text{out}}:t} = (\mathbf{H}^{S}_{t-T_{\text{out}}}, \cdots, \mathbf{H}^{S}_{t}), \tag{4}$$

Let $f_{TC} \to f_{SC} \to f_{TC}$ denote a basic block unit. By stacking these basic block units multiple times, we get a sequence of the embedding matrix $\mathbf{H}' = (\mathbf{H}'^{\mathcal{T}}_{t-T'_{\text{out}}}, \cdots, \mathbf{H}'^{\mathcal{T}}_{t})$, after several convolutions. For the spatio-temporal encoder based on embedding propagation and aggregation, T'_{out} can change according to the parameters specified in the convolution process. After multiple convolution operations, T'_{out} eventually converges. Finally, we obtain the final embedding representation matrix $\mathbf{H}^{\mathcal{G}} \in \mathbb{R}^{N\times D}$ ($T'_{\text{out}} = t$), for the spatio-temporal encoder, in which each row $\mathbf{h}^{\mathcal{G}}_n \in \mathbb{R}^D$ denotes the final embedding of node v_n.

4.1.2. Decay of importance in the time dimension

In STG neural networks, the conventional practice involves forecasting data at the time step $t+1$ relying on the information encompassed within the period $[t-T+1, t]$. In the context of time sequences, research proposed that proximity to the present moment corresponds to heightened attention allocation. Consequently, data in closer temporal proximity are assigned a greater weight, where the weight ascribed to the data dictates its influence on the current temporal instant. Hence, we posit the presence of a comparable phenomenon within STG neural networks, as illustrated in Fig. 2.

Now, we consider that the data at the time step $t+1$ is based on information encompassed within the period $[t-T+1, t]$. So, we represent the data over previous T time steps in the time step t with a new tensor $\mathbf{X}^{D}_{t-T+1:t} \in \mathbb{R}^{T\times N\times C} = (\mathbf{X}^{D}(t-T+1), \cdots, \mathbf{X}^{D}(t))$. The information of all nodes $\mathcal{V}$ in the time step t is denoted as $\mathbf{X}^{D}(t) \in \mathbb{R}^{N\times C}$, where C denotes the initial embedding dimensionality of the nodes. We define $\mathbf{X}^{D}(t) \in \mathbb{R}^{N\times C}$ as follows:

$$\mathbf{X}^{D}(t) = \frac{\alpha_t \mathbf{X}(t)}{\sum_{t'=t-T+1}^{t} \alpha_{t'} \mathbf{X}(t')}, \tag{5}$$

where the weight $\alpha_{t'}$ satisfies $\alpha_{t'} \in \mathbb{R}_+$. There are several possible forms of the weight $\alpha_{t'}$, one can use the following forms of $\alpha_{t'}$ to model $\mathbf{X}^{D}(t)$:

$$\alpha_{t'} = \exp(-\xi(t-t')), \alpha_{t'} = \frac{1}{(t-t')^{\xi}}, \alpha_{t'} = \frac{1}{\xi \ln(t-t'+1)}$$

where $\xi \in \mathbb{R}$ is a hyperparameter in the time decay function to adjust the strength of the decay. It should be noted that ξ is treated as a hyperparameter chosen in a systematic and regular manner. Our primary

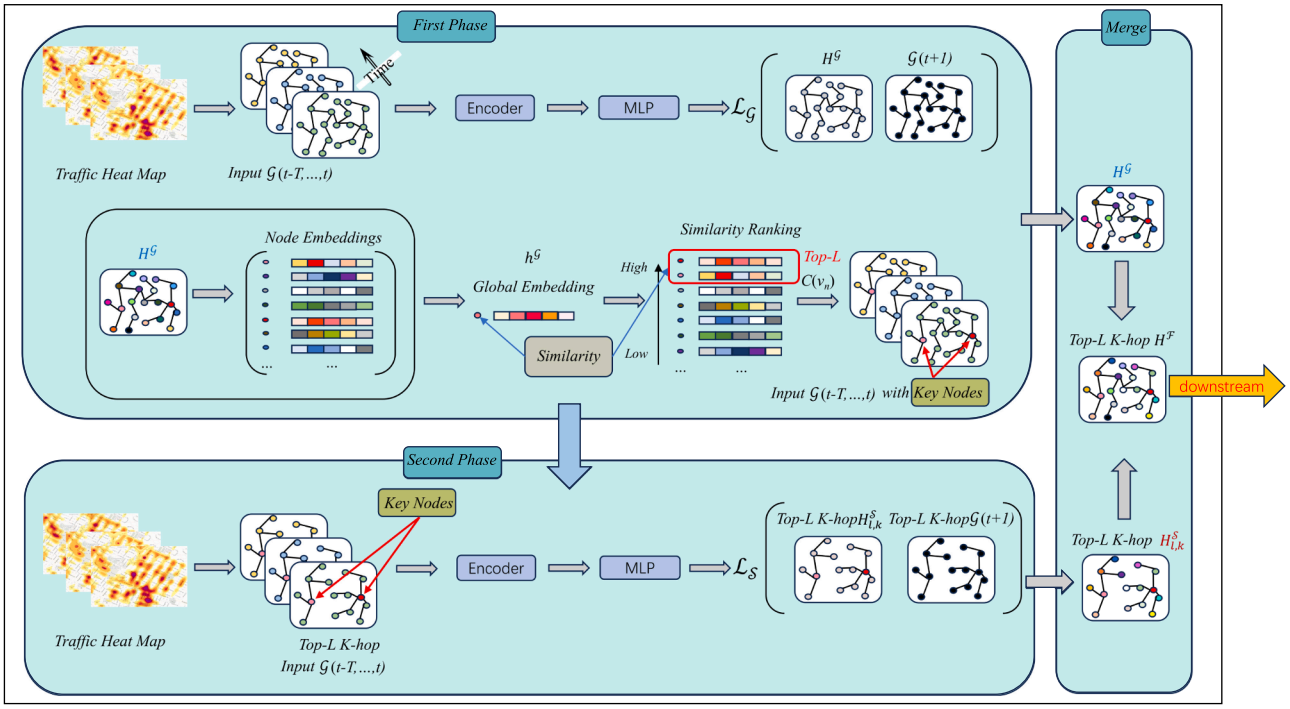


Fig. 1. Overall architecture of UMSST. **First Phase (Upper Left Parts):** Input original STG data $\mathcal{G}_{t-T+1:t}$, process through a spatio-temporal encoder (containing temporal convolution and graph convolution layers) to obtain initial node embeddings. Subsequently, learn global node embeddings $\mathbf{H}^{\mathcal{G}} \in \mathbb{R}^{N\times C}$ through self-supervised learning methods. Calculate the global graph embedding $\mathbf{h}^{\mathcal{G}} \in \mathbb{R}^{C}$, Sort nodes by comparing the similarity (e.g., cosine similarity) between each node embedding $\mathbf{h}^{\mathcal{G}}_{v} \in \mathbb{R}^{C}$ and the global embedding $\mathbf{h}^{\mathcal{G}} \in \mathbb{R}^{C}$, and select the Top-$L$ percentage of nodes with the highest similarity as key nodes (subgraph centers). **Second Phase (Lower Left Part):** Based on the top-L percentage key nodes identified in the first phase, construct their K-hop neighborhood subgraphs respectively. Reapply the spatio-temporal encoder and self-supervised learning methods to each subgraph data $\mathbf{X}_{l,k|t-T+1:t}$ to learn the node embeddings $\mathbf{H}^{S}_{l,k} \in \mathbb{R}^{N_{S_{l,k}}\times C}$ for each subgraph, where $N_{S_{l,k}}$ is the number of notes of the l-th subgraph with $K = k$. **Feature Fusion and Output (Right Part):** Replace or fuse the representations of corresponding nodes from the multi-subgraph embeddings $\mathbf{H}^{S}_{l,k}$ earned in the second phase into the corresponding positions in the global node embeddings $\mathbf{H}^{\mathcal{G}}$ obtained in the first phase, forming the final node representations $\mathbf{H}^{\mathcal{F}}_{l,k}$. These enhanced node representations are then used for prediction in downstream tasks, for example, predicting $\mathcal{G}_{t+1}$ through an MLP layer.

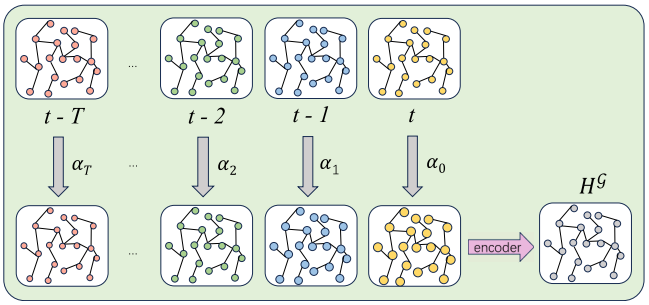


Fig. 2. Introducing weights to the temporal dimension data, with the intention that data closer to the current moment carries greater weight.

goal in the experiments is not to conduct an exhaustive search for the optimal ξ. but to demonstrate the general effectiveness and robustness of incorporating a temporal decay mechanism. The systematic trial of a few representative values serves to validate that performance gains are consistent across a reasonable range of decay strengths, rather than being an artifact of a single, finely-tuned value. So, we substitute $\mathbf{X}_{t-T+1:t}$ in Eq. (1) with $\mathbf{X}^{\mathcal{D}}_{t-T+1:t}$ to obtain the following equation:

$$\mathbf{H}^{\mathcal{T}}_{t-T_{\text{out}}:t} = f_{TC}(\mathbf{X}^{\mathcal{D}}_{t-T+1:t}). \tag{6}$$

Therefore, we can substitute Eq. (6) for (1) to get $\mathbf{H}^{\mathcal{G}}$. Hence, we establish a correlation between the decay of importance in the temporal dimension and the previous model.

4.1.3. Global embedding and key node selection

Through the spatio-temporal self-supervised learning process described above, we obtain the node embeddings $\mathbf{H}^{\mathcal{G}} \in \mathbb{R}^{N\times D}$. Subsequently, leveraging the learned node embeddings, we attain a global embedding with dimensions equivalent to the node dimensions. This embedding encapsulates the entirety of the information within the STG. Specifically, we obtain M final embeddings from a dataset $\mathbf{X}_{\text{samples}} = \{\mathbf{X}_{t_1-T+1:t_1}, \cdots, \mathbf{X}_{t_M-T+1:t_M}\}$ composed of M random samples through self-supervised training, where $\mathbf{X}_{t_m-T+1:t_m} \in \mathbb{R}^{T\times N\times C}$. These M final embeddings constitute a final embedding set $\mathbf{H}^{\mathcal{R}}_{\text{samples}} = \{\mathbf{H}^{\mathcal{G}}_1, \cdots, \mathbf{H}^{\mathcal{R}}_M\}$, where $\mathbf{H}^{\mathcal{G}}_m \in \mathbb{R}^{N\times D}$, and $\mathbf{h}^{\mathcal{G}}_{m,v_n} \in \mathbb{R}^{D}$ denote the embedding of node $v_n \in \mathcal{V}$ in $\mathbf{H}^{\mathcal{G}}_m$. Therefore, we can acquire node embeddings in the dataset $\mathbf{X}_{\text{samples}}$ by obtaining the graph embeddings corresponding to each node in the random samples. Let $\mathbf{h}^{\mathcal{G}}_{v_n} \in \mathbb{R}^{D}$ denote the average embedding of node $v_n \in \mathcal{V}$ from the M samples, and its specific form is as follows:

$$\mathbf{h}^{\mathcal{G}}_{v_n} = \frac{1}{M}\sum_{m=1}^{M} \mathbf{h}_{m,v_n}. \tag{7}$$

Furthermore, we can obtain a global embedding that represents the entire STG through the embeddings of the final nodes. Let $\mathbf{h}^{\mathcal{G}} \in \mathbb{R}^{D}$ denote the global embedding and let its specific form be as follows:

$$\mathbf{h}^{\mathcal{G}} = \frac{1}{|\mathcal{V}|}\sum_{v_n\in\mathcal{V}} \mathbf{h}^{\mathcal{G}}_{v_n}, \tag{8}$$

where $\mathbf{h}^{\mathcal{G}} \in \mathbb{R}^{D}$ encapsulates the complete spatio-temporal information of the graph. This mean aggregation strategy is chosen for its ability to provide a comprehensive and unbiased summary of the graph's overall state. Its effectiveness was validated against alternative strategies,

such as degree-weighted averaging and random node sampling, where it consistently led to superior performance in identifying globally representative key nodes for downstream tasks (see Appendix 2 for details).

The acquisition of key nodes is facilitated through the analysis of the similarity between the node embeddings $\mathbf{h}_{v_n}^{\mathcal{G}}$ and $\mathbf{h}^{\mathcal{G}}$. The nodes that exhibit the highest similarity are identified as key nodes. We use a fixed percentage of the total number of nodes as the criterion to select the key nodes. For example, consider a graph with 200 nodes where the average number of neighbors per node is 3. Selecting the top 2 % of the nodes as the subgraph centers is reasonable because the total number of nodes covered by their 1-hop neighborhoods ($3 \times 200 \times 2\,\% = 12$nodes) is moderate compared to the overall size of the graph. Even extending to 2-hop neighborhoods, ignoring overlaps, this approach would only cover about 36 nodes (12×3). Therefore, the framework would focus only on about 18 % (36 of 200) of the total nodes, significantly enhancing computational efficiency. This percentage-based selection method is more robust for graphs of different sizes. Let $C(\mathcal{V}_s)$ denote the set of selected key nodes:

$$C(\mathcal{V}_s) = \{v_n \in \mathcal{V}_s \mid \text{Similarity}(\mathbf{h}^{\mathcal{G}}, \mathbf{h}_{v_n}^{\mathcal{G}})\}, \tag{9}$$

where $\mathcal{V}_s$ represents the set of nodes corresponding to the top-L percentage of nodes embedding that have the highest similarity to $h^{\mathcal{G}}$, and $\text{Similarity}(\mathbf{h}^{\mathcal{G}}, \mathbf{h}_{v_n}^{\mathcal{G}})$ is the cosine function (We believe that more accurate methodologies for quantifying similarity, here primarily focused on substantiating the efficacy of the model). Selecting nodes most similar to the global embedding as representatives of key regions is based on the rationale that these nodes and their surrounding areas are more likely to contain information crucial to the overall graph structure and dynamics.

4.2. Second phase: multi-subgraph learning

4.2.1. Subgraph construction

The construction of subgraphs is based on the key nodes identified in the first phase. To address potential subgraph overlap and ensure subgraph diversity, key node selection employs a sequential filtering strategy. First, the node with the highest similarity to the global embedding is selected as the first key node. Subsequently, when selecting the next key node, its distance to already selected key nodes is considered. Specifically, the subgraph distance (e.g., shortest path hops) between a newly selected key node and all previously selected key nodes must be greater than $2 \times K$ (where K is the hop parameter for subgraph construction). This process is iterated until $L\,\%$ key nodes satisfying this distance constraint are selected. The corresponding subgraphs are then constructed around these filtered key nodes and their K-hop neighborhood nodes. The parameter K defines the scope of these local regions, which are presumed to contain significant graph-related information. The rationale for introducing K lies in its capacity to delineate the bounds within which crucial information (potentially dense or dispersed) is likely to be found. Adjusting the magnitude of K, it is possible to capture essential information exhibiting various distributional properties. The scale of K ought tobe carefully considered; an overly extensive K may inadvertently shift focus away from vital areas Therefore, the optimal value of K is data-dependent and is determined empirically. In our experimental setup, for each dataset, we treat K as a hyperparameter and select its optimal value by performing a grid search over a predefined range (e.g., $K \in \{1, 2, 3\}$) and evaluating the performance of the downstream task on a dedicated validation set. This standard practice ensures that the subgraph scale is appropriately adapted to the intrinsic spatial characteristics of the specific graph. This key point selection method aims to reduce redundancy among subgraphs while encouraging the model to explore different and representative local structures within the graph.

4.2.2. "Unit-representation" and learning from multi-subgraphs

Pre-trained models must learn features that are useful for diverse downstream tasks, whether at the node, edge, or graph level. This requires a representational approach that captures fundamental, transferable patterns. We introduce the concept of a "unit representation" to refer to a rich, localized knowledge module that serves as a basic building block for understanding the entire graph. Analogous to how words form the basis of language, these unit representations encapsulate fundamental local spatio-temporal events or motifs. We propose that subgraphs are the ideal structural candidates to serve as these "units". A subgraph, defined by a key node and its local neighborhood, inherently contains all relevant local information: the states of its constituent nodes, the relationships (edges) between them, and their collective topology. By learning a representation for each subgraph, the model effectively builds a vocabulary of these localized knowledge modules. This approach allows for a more granular and compositional understanding of the global graph dynamics. Extracting information from multiple subgraphs facilitates a more thorough exploration of local regions within the graph that contain critical information. Due to information being dispersed across disparate locales, representing the entire graph's information using only one subgraph is quite limiting, as crucial information in the graph may be scattered across different local regions. Extracting information from a single subgraph diminishes the effectiveness of information propagation on the graph and weakens the utilization of initial information by downstream tasks. Consequently, we advocate for a pioneering approach that entails extracting information from multi-subgraph, thereby facilitating a more thorough exploration of graph local regions with critical information.

Let the l-th k-hop subgraph be represented as follows:

$$\mathbf{X}_{l,k|t-T+1:t} \in \mathbb{R}^{T \times N_{S_{l,k}} \times C} = (\mathbf{X}_{l,k}(t-T+1), \cdots, \mathbf{X}_{l,k}(t)),$$

where $N_{S_{l,k}}$ is the number of notes of the l-th subgraph with $K = k$. For each subgraph, a spatio-temporal encoding process similar to that in the first phase (i.e., stacking of temporal and spatial convolutions) is applied again to learn its node embeddings. Let $\mathbf{H}_{l,k}^{\text{Sub}}$ denote the node embedding matrix learned for the l-th subgraph with $K = k$, where $\mathbf{h}_{l,k,v_n}^{\text{Sub}}$ is the embedding of node v_n in the subgraph. This learning process aims to capture more detailed local spatio-temporal patterns within each subgraph.

4.2.3. Model training: representation learning

In the initial representation phase (Phase One), the model is optimized by minimizing the following loss function to learn $\mathbf{H}^{\mathcal{G}}$:

$$\mathcal{L}_{\mathcal{G}} = \sum_{c=1}^{C} \sum_{n=1}^{N} \lambda_c \left| \hat{x}_{\mathcal{G},t+1,v_n}^{(c)} - x_{t+1,v_n}^{(c)} \right| \tag{10}$$

where $\hat{x}_{\mathcal{G},t+1,v_n}^{(c)}$ is the c-th dimension of the predicted result obtained via MLP based on $\mathbf{h}_{v_n}^{\mathcal{G}}$:

$$\hat{x}_{\mathcal{G},t+1,v_n}^{(c)} = MLP(\mathbf{h}_{v_n}^{\mathcal{G}}) \tag{11}$$

and $x_{t+1,v_n}^{(c)}$ is the c-th dimension of the ground truth. λ_c is a parameter to balance the data in different dimensions, and $\sum_{c=1}^{C} \lambda_c = 1$.

In the multi-subgraph phase (Phase Two), the model is optimized by minimizing the following loss function to learn the representations $\mathbf{H}_{l,k}^{\text{Sub}}$ of individual subgraphs:

$$\mathcal{L}_{\mathcal{S}} = \sum_{c=1}^{C} \sum_{n=1}^{N} \lambda_c \left| \hat{x}_{\mathcal{S},t+1,v_n}^{(c)} - x_{t+1,v_n}^{(c)} \right| \tag{12}$$

where $\hat{x}_{\mathcal{S},t+1,v_n}^{(c)}$ is the c-th dimension of the predicted result obtained via MLP based on $\mathbf{h}_{l,k,v_n}^{\text{Sub}}$:

$$\hat{x}_{\mathcal{G},t+1,v_n}^{(c)} = MLP(\mathbf{h}_{l,k,v_n}^{\text{Sub}}) \tag{13}$$

4.3. Feature merging and final representation

Similarly to the first phase, following the acquisition of STG representations for various subgraphs, self-supervised training (or representation learning oriented towards downstream tasks as described above) is

specifically employed on these STGs. The final embedding corresponding to the multi-subgraph STGs, denoted as $\mathbf{H}^{\text{Sub}}_{l,k}$, is obtained. Subsequently, the node embeddings in $\mathbf{H}^{\mathcal{G}}$ are replaced or enhanced with their respective corresponding node embedding representations from $\mathbf{H}^{\text{Sub}}_{l,k}$. Let $\mathbf{H}^{\mathcal{F}}$ represent the embedding after the merge of $\mathbf{H}^{\mathcal{G}}$ and all $\mathbf{H}^{\text{Sub}}_{l,k}$. The embedding $\mathbf{h}^{\mathcal{F}}_{v_n} \in \mathbb{R}^D$ of node v_n in $\mathbf{H}^{\mathcal{F}}$ is specifically formulated as follows:

$$\mathbf{h}^{\mathcal{F}}_{l,k,v_n} = \begin{cases} \mathbf{h}^{\mathcal{F}}_{l,k,v_n} = \mathbf{h}^{\text{Sub}}_{l,k,v_n}, & \mathbf{h}^{\text{Sub}}_{l,k,v_n} \in \mathbf{H}^{\text{Sub}}_{l,k} \\ \mathbf{h}^{\mathcal{F}}_{l,k,v_n} = \mathbf{h}^{\mathcal{G}}_{v_n}, & \mathbf{h}^{\text{Sub}}_{l,k,v_n} \notin \mathbf{H}^{\text{Sub}}_{l,k} \text{ and } \mathbf{h}^{\mathcal{G}}_{v_n} \in \mathbf{H}^{\mathcal{G}} \end{cases}$$

4.4. Complexity analysis

To evaluate the computational efficiency of the proposed UMSST framework, we analyze its complexity compared to refined training on the full graph.

• **Full Graph Training**: Standard graph convolution operations on the full graph typically have a spatial complexity of $O(N^2)$ (for dense adjacency matrices) or $O(N \times D + |\mathcal{E}| \times D)$ (for sparse adjacency matrices and message passing networks), where N is the number of nodes, $|\mathcal{E}|$ is the number of edges, and D is the feature dimension. The time complexity depends on the number of layers and specific operations.

• **UMSST Framework**:

- **Phase One**: Similar to full graph training, the complexity depends on the size of the entire graph.
- **Phase Two (Multi-Subgraph Training)**: Since the size of each subgraph is limited by the K-hop neighborhood, assuming an average subgraph size of N_{sub} (the subgraph nodes) where $N_{\text{sub}} \ll N$, the complexity of processing a single subgraph is significantly reduced. If L_{num} (the number of subgraphs) subgraphs are selected and their training can be parallelized, the total training time can be effectively controlled. The spatial complexity of each subgraph tends towards a smaller constant level (relative to the full graph). For example, if the K-hop constraint makes the number of subgraph nodes N_{sub} much smaller than N, its complexity is mainly determined by N_{sub} and the number of edges within the subgraph.
- **Parallelism**: The training process of multi-subgraphs naturally supports parallelization, which can further significantly improve training efficiency, especially when dealing with large-scale graphs. By "blurring" the training in the first phase (e.g., using a more lightweight model or fewer iterations to quickly locate key regions) and allocating more computational resources to the parallel subgraph training in the second phase, significant efficiency gains can be achieved.

This two-stage and multi-subgraph parallel processing design gives UMSST a potential efficiency advantage when processing large-scale spatio-temporal graphs.

However, it is important to acknowledge the overhead associated with parallel training. The additional costs primarily include: (1) Subgraph Construction Overhead: A one-time computational cost is incurred before training to extract all subgraphs based on the selected key nodes and the hop parameter K. (2) Data Communication and Synchronization: In a parallel environment, subgraph data must be distributed to different computational units (e.g., GPUs), and synchronization is required during or after training (e.g., for gradient updates), which introduces communication latency. Particularly, as K increases, the size of each subgraph and the degree of overlap between them also increase. This not only raises the processing time for individual subgraphs but can also reduce the efficiency of parallelization due to increased redundant computations across different units. Nevertheless, for large-scale graphs, the computational benefits of decomposing a large, complex problem into multiple smaller, manageable ones typically far outweigh these parallelization overheads.

5. Experiments

In this section, we evaluate the performance of UMSST in the task of spatio-temporal traffic exploration. The experimental evaluations are performed on four distinct real datasets; their detailed descriptions are provided in Table 1. The performance of UMSST is validated in these four real datasets, and a comparative analysis is performed against the current state-of-the-art method (GPT-ST) [17] and other important baseline methods.

5.1. Datasets

The experiments used four publicly available real-world traffic flow datasets: NYCBike1, NYCBike2, NYCTaxi, and BJTaxi. The statistics of these datasets are shown in Table 1.

5.2. Evaluation metrics and baseline models

Metrics. In the experiments, the mean absolute error (MAE) [26] and the mean absolute percentage error (MAPE) [7] were used as evaluation metrics for model performance, which are commonly used in current research on STG prediction. Their definitions are as follows (assuming that $\mathbf{x} = \{x_1, \cdots, x_N\}$ is the ground truth sequence, $\hat{\mathbf{x}} = \{\hat{x}_1, \cdots, \hat{x}_N\}$ is the predicted sequence, and N is the number of samples)

- Mean Absolute Error(MAE):

$$MAE(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \sum_{n=1}^{N} |x_n - \hat{x}_n| \tag{14}$$

- Mean Absolute Percentage Error(MAPE):

$$MAPE(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \sum_{n=1}^{N} \frac{|x_n - \hat{x}_n|}{x_n} \tag{15}$$

Baselines. To comprehensively evaluate the effectiveness of our proposed UMSST framework, we selected several advanced and advanced spatio-temporal prediction models as baselines.

Spatio-temporal prediction methods based on GNNs:

- **GWN**[27]: This method utilizes an adaptive adjacency matrix to learn latent spatial dependencies and incorporates graph diffusion convolution with gated temporal convolution to efficiently capture dependencies in long-term time series.
- **MSDR**[19]:This model proposes a variant of RNNs with Multi-Step Dependency Relation to make full use of historical time step information and combines it with GNNs to model long-range spatio-temporal dependence.
- **STFGNN**[15]: This work proposes a data-driven approach that utilizes a gated convolution method to generate spatio-temporal graphs. By learning spatial and temporal dependencies, the approach effectively captures the correlations within the data.
- **STGCN**[8]: This pioneering method combines graph convolutional networks (GCN) with gated temporal convolution to capture spatial and temporal dependencies, respectively.
- **STSGCN**[24]: This work captures complex localized spatio-temporal correlations by constructing a local spatio-temporal graph, enabling the synchronous modeling of these correlations.

Table 1
Statistics of datasets.

Data type	Bike rental		Taxi GPS	
Dataset	NYCBike1	NYCBike2	NYCTaxi1	BJTaxi
Time interval	1 h	30 min	30 min	30 min
Nodes	16 × 8	10 × 20	10 × 20	32 × 32
Bikes/Taxis	6.8k+	2.6m+	22m+	34k+

- **TGCN**[3]: This method integrates graph convolutional networks (GCNs) into a gated recurrent unit (GRU), forming a unified spatio-temporal graph model to synchronously model spatio-temporal correlations.
- **BGCN**[6]:This model introduces a Bayesian framework to graph convolutional networks for traffic prediction, enabling it to not only provide accurate predictions but also quantify the uncertainty associated with them, which is crucial for robust decision-making in intelligent transportation systems.
- **ST-SSL**[10]:This model performs the adaptive augmentation over the traffic flow graph data at both attribute-levels and structure-levels.

Attention-based spatio-temporal prediction methods:

- **STWA** (Spatio-Temporal Wave-Attention)[4]: This method integrates location-specific and time-varying parameters into the attention network to effectively capture dynamic spatio-temporal correlations.

5.3. Implementation details

The UMSST model is implemented using the PyTorch framework, with an embedding dimension D configured to 64. The temporal and spatial convolution kernel sizes of the spatio-temporal encoder are set to 3. The training phase uses the Adam optimizer with a batch size of 32. In the architecture of our UMSST model, the Spatio-Temporal (ST) encoder stacks two "sandwich" structured modules (e.g., $f_{TC} \rightarrow f_{SC} \rightarrow f_{TC}$) to elaborately capture the intricate dependencies between spatial and temporal dynamics. To further enhance the model's temporal dimension comprehension, an additional Temporal Convolution (TC) layer is appended to the end of the ST encoder. In addition, the downstream task processing method borrows from the ST-SSL model and maintains consistency with the parameter settings used in ST-SSL's downstream tasks. To ensure the robustness and stability of our findings, all reported results are averaged over five runs with different random seeds. All experiments were conducted on a server equipped with four NVIDIA GeForce RTX 4080 GPUs.

The hyperparameters K (number of hops) and L (percentage of key nodes for subgraph selection) are determined by screening the validation set of each dataset. After determining the optimal K and L on the validation set, we investigate the impact of the temporal decay parameter ξ. Our approach was to test a set of systematic, representative values to validate the robustness of the temporal decay mechanism, as shown below:

- $\alpha_{t'} = \exp(-\xi(t-t'))$: $\xi \in \{0.001, 0.01, 0.1\}$
- $\alpha_{t'} = \frac{1}{(t-t')^{\xi}}$: $\xi \in \{1, 1.5, 2\}$
- $\alpha_{t'} = \frac{1}{\xi \ln(t-t'+1)}$: $\xi \in \{1, 1.5, 2\}$

The consistent performance improvements observed across these settings, without exhaustive fine-tuning, strongly demonstrate the general effectiveness of incorporating a temporal decay mechanism. This suggests that the benefit is a fundamental property of the model architecture, rather than being contingent on a highly specific hyperparameter choice, although further gains could likely be achieved with a more granular search.

5.4. Performance comparison

In this section, we primarily investigate the performance improvement that our UMSST framework brings to downstream spatio-temporal forecasting tasks. To achieve this, we systematically evaluated the performance of several baseline models before and after applying UMSST on four real-world datasets. Furthermore, to validate the superiority of our framework, we conduct a direct comparison between UMSST and the current state-of-the-art pre-training model, GPT-ST. All experimental results are presented in Table 2.

The results clearly indicate that our proposed UMSST framework significantly improves the prediction performance of different downstream baseline models across all datasets, which effectively demonstrates the effectiveness and generalization ability of our framework. We analyze the promotion effect of UMSST from the following dimensions:

- We observe that UMSST provides consistent performance improvements for various types of baseline models (e.g., GCN-based, RNN-based, or attention-based), including STWA, GWN, MSDR, STFGNN, STGCN, STSGCN, BGCN and ST-SSL. This universal enhancement verifies that UMSST's effectiveness is not confined to a specific category of models. Instead, it learns general and transferable spatio-temporal knowledge that can empower a wide range of downstream models.
- In addition to evaluating the improvements over models without pre-training, we also compare UMSST against a strong pre-training baseline, GPT-ST. We chose TGCN as the base model and applied both UMSST and GPT-ST to it for a fair comparison across all four datasets. As shown in Table 2, while both pre-training methods enhance TGCN's performance, the gains from UMSST are substantially more significant.

5.5. Ablation study

To validate the effectiveness of key components within our proposed UMSST framework, we designed a series of rigorous ablation studies. This investigation aims to answer two essential questions: (1) ***Is our two-phase, multi-subgraph learning mechanism the primary driver of performance improvement?*** and (2) ***Does the temporal decay module provide further effective gains on top of it?*** Therefore, we compare the performance of four model configurations:

- **Baseline:** The baseline model without pre-training.
- **UMSST (First Phase Only):** Using only the first phase of our framework (global learning and key node identification).
- **w/UMSST (No Decay):** Applying the complete two-phase multi-subgraph framework but without the temporal decay module
- **w/UMSST (Full Model):** Completed UMSST model.

The experimental results are illustrated in Figs. 3, 4, 5 and 6 (each figure corresponds to the four metrics of a single dataset).

5.5.1. Effectiveness of the two-phase multi-subgraph framework

Our innovation lies in the two-phase learning paradigm: global exploration in the first phase, followed by local, multi-subgraph refinement in the second. To verify the necessity of the second phase, we compare the performance of **UMSST (First Phase Only)** against **w/UMSST (No Decay)**.

As clearly observed in the Figs. 3 to 6, the model using only the first pre-training phase generally yields performance that is inferior, and often significantly so, to the original baseline. This is expected, as the primary goal of fisrt phase is to identify critical spatio-temporal regions rather than to optimize for the final prediction task. However, once the second phase is introduced, the model is refined in learning the identified key subgraphs, and the model performs qualitatively. For instance, the result for First Phase Only is always worse than the baseline, whereas the No Decay model's performance always surpasses it. This obvious performance gap unequivocally proves that multi-subgraph learning in the second phase is the fundamental driver of performance gain in the UMSST framework. By focusing on high-value local regions, it successfully captures fine-grained spatio-temporal patterns that are difficult to learn from a single global view.

5.5.2. Effectiveness of the temporal decay module

We further investigate the contribution of the temporal decay module. We evaluated its effect by comparing **w/UMSST (No Decay)** with our **w/UMSST (Full Model)**.

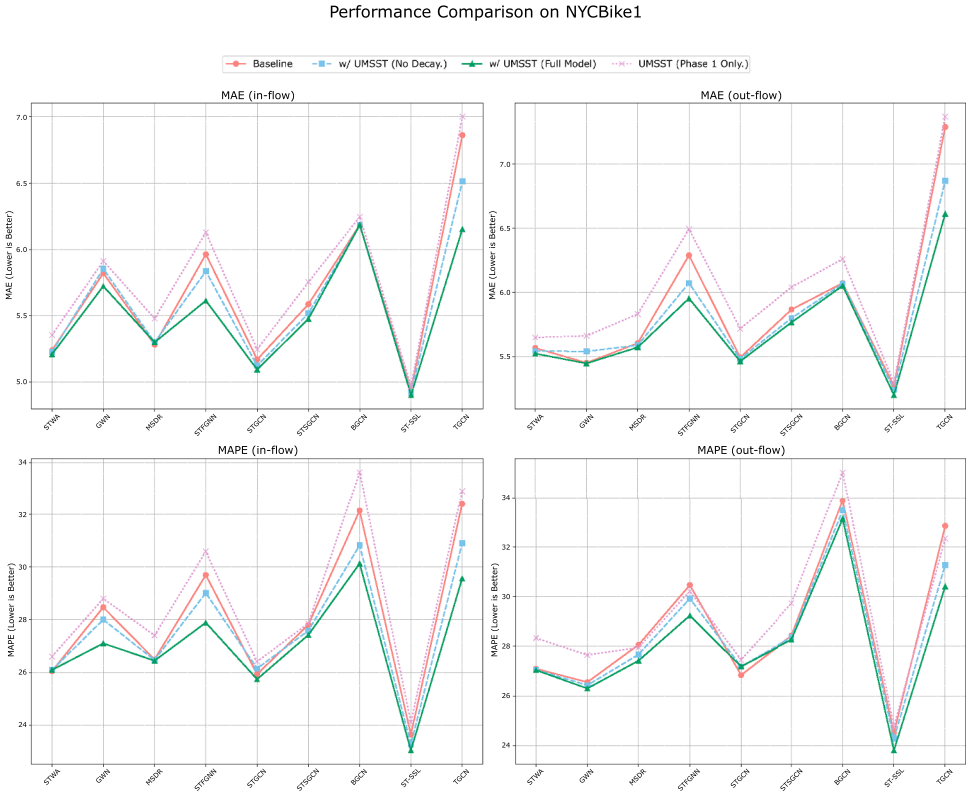


Fig. 3. Ablation study on the NYCBike1 dataset. The plot compares the performance of the baseline, Phase 1 only, UMSST without decay, and the full UMSST model.

Fig. 4. Ablation study on the NYCBike2 dataset, showing a performance comparison for the four model configurations.

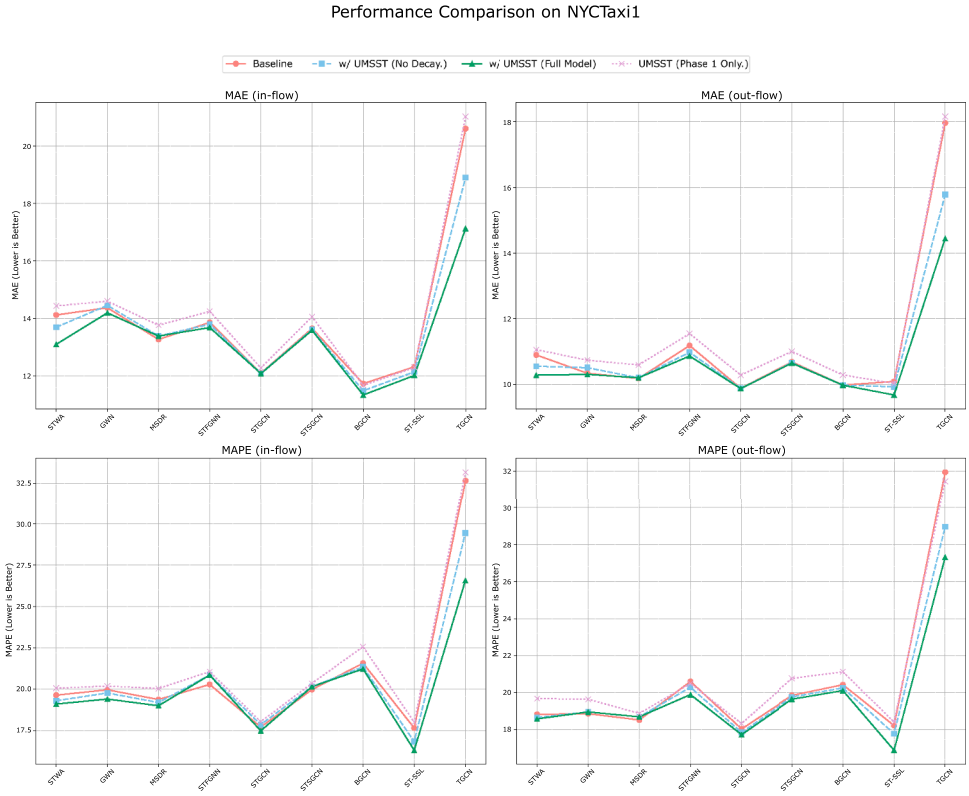


Fig. 5. Ablation study on the NYCTaxi1 dataset, demonstrating the effectiveness of the UMSST components.

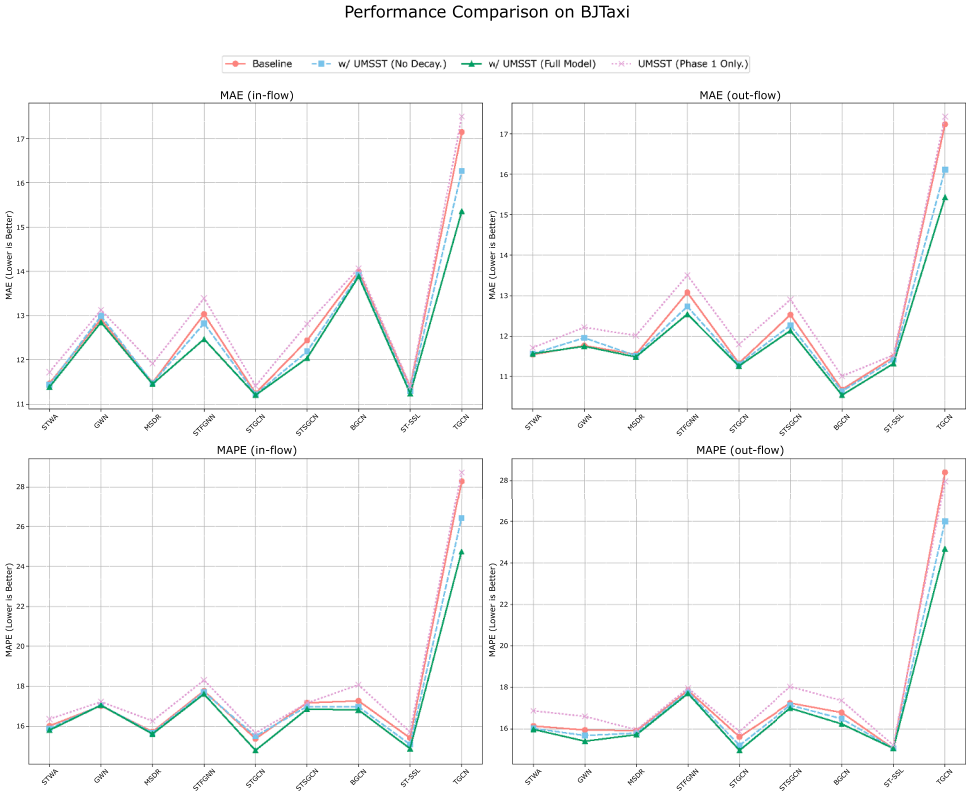


Fig. 6. Ablation study on the BJTaxi dataset. The results validate the contributions of both the multi-subgraph framework and the decay module.

Table 2
Overall performance comparison on different datasets in terms of *MAE* and *MAPE*. "in" and "out" in the Metrics column refer to the prediction of traffic in-flow and out-flow, respectively.

Model	Dataset	NYCBike1		NYCBike2		NYCTaxi1		BJTaxi	
	Metrics	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
in	STWA	5.2376	**26.0449 %**	5.4781	26.8128 %	14.1236	19.6344 %	11.4702	16.0336 %
	w/ UMSST	**5.2083**	26.0928 %	**5.4162**	**26.2362 %**	**13.1019**	**19.0974 %**	**11.3858**	**15.8207 %**
out	STWA	5.5670	27.0988 %	5.1822	25.1836 %	10.8994	18.8272 %	**11.5432**	16.1398 %
	w/ UMSST	**5.5234**	**27.0406 %**	**5.1812**	**24.8839 %**	**10.2826**	**18.5857 %**	11.5649	**15.9794 %**
in	GWN	5.8203	28.4713 %	5.8100	27.0552 %	14.3730	19.9583 %	12.9198	**17.0329 %**
	w/ UMSST	**5.7220**	**27.1013 %**	**5.7401**	**26.8175 %**	**14.1965**	**19.3990 %**	**12.8496**	17.0669 %
out	GWN	5.4513	26.5561 %	5.0779	24.9189 %	10.3416	**18.8708 %**	11.7650	15.9479 %
	w/ UMSST	**5.4471**	**26.3036**	**5.0400**	**24.9857 %**	**10.3080**	18.9623 %	**11.7491**	**15.4100 %**
in	MSDR	**5.2808**	26.4916 %	5.7164	27.2596 %	**13.2714**	19.3715 %	11.4857	15.7396 %
	w/ UMSST	5.2982	**26.4434 %**	**5.5458**	**26.8772 %**	13.3887	**18.9950 %**	**11.4564**	**15.6317 %**
out	MSDR	5.6043	28.0605 %	5.3382	26.0199 %	**10.1825**	**18.5269 %**	11.5478	15.9192 %
	w/ UMSST	**5.5723**	**27.4217 %**	**5.1344**	**25.1035 %**	10.2026	18.6975 %	**11.4850**	**15.7242 %**
in	STFGNN	5.9630	29.6953 %	5.8192	27.4402 %	13.8655	**20.2805 %**	13.0341	17.7794 %
	w/ UMSST	**5.6128**	**27.8877 %**	**5.6980**	**27.1494 %**	**13.6869**	20.8487 %	**12.4695**	**17.6252 %**
out	STFGNN	6.2891	30.4628 %	5.4666	26.3836 %	11.1849	20.6068 %	13.0777	17.8508 %
	w/ UMSST	**5.9533**	**29.2357 %**	**5.3710**	**25.6297 %**	**10.8659**	**19.8985 %**	**12.5430**	**17.7073 %**
in	STGCN	5.1684	25.9314 %	**5.4468**	25.4049 %	12.0995	17.7057 %	11.2467	15.3902 %
	w/ UMSST	**5.0928**	**25.7412 %**	5.4576	**25.0372 %**	**12.0867**	**17.4810 %**	**11.2058**	**14.8110 %**
out	STGCN	5.4936	**26.8405 %**	5.0981	**25.0902 %**	9.8825	18.0440 %	11.3294	15.6190 %
	w/ UMSST	**5.4634**	27.1840 %	**5.0650**	25.1730 %	**9.8751**	**17.7345 %**	**11.2615**	**14.9674 %**
in	STSGCN	5.5863	27.8069 %	5.6364	27.1371 %	13.6548	**19.9709 %**	12.4423	17.1820 %
	w/ UMSST	**5.4763**	**27.4207 %**	**5.5632**	26.4016 %	**13.5906**	20.1471 %	**12.0415**	**16.8708 %**
out	STSGCN	5.8664	28.4314 %	5.2677	26.0045 %	10.6868	19.8553 %	12.5289	17.2490 %
	w/ UMSST	**5.7665**	**28.2697 %**	**5.2068**	**25.6438 %**	**10.6508**	**19.6510 %**	**12.1364**	**17.0017 %**
in	BGCN	6.1852	32.1366 %	5.8226	28.3144 %	11.7362	21.5669 %	13.9987	17.2828 %
	w/ UMSST	**6.1844**	**30.1269 %**	**5.7329**	**27.6565 %**	**11.3426**	**21.2225 %**	**13.8867**	**16.8309 %**
out	BGCN	6.0728	33.8765 %	5.5352	29.9981 %	9.9801	20.4298 %	10.6806	16.7878 %
	w/ UMSST	**6.0522**	**33.1287 %**	**5.4998**	**29.7623 %**	**9.9773**	**20.1154 %**	**10.5433**	**16.2399 %**
in	ST-SSL	4.9403	23.6434 %	5.1004	22.9403 %	12.3220	17.6452 %	11.3907	15.4323 %
	w/ UMSST	**4.9001**	**23.0429 %**	**5.0204**	**22.3731 %**	**12.0203**	**16.3022 %**	**11.2389**	**14.9006 %**
out	ST-SSL	5.2704	24.5136 %	4.7406	21.5901 %	10.0924	18.2268 %	11.4806	15.0619 %
	w/ UMSST	**5.2003**	**23.8132 %**	**4.6702**	**21.2327 %**	**9.6804**	**16.8931 %**	**11.3210**	**15.0605 %**
in	TGCN	6.8620	32.3966 %	6.3801	28.7089 %	20.6009	32.6614 %	17.1461	28.2910 %
	w/ GPT-ST	6.6604	30.8634 %	6.3137	29.5952 %	19.9628	30.5810 %	16.7653	26.0843 %
	w/ UMSST	**6.1526**	**29.5561 %**	**6.2812**	**29.4436 %**	**17.1290**	**26.5617 %**	**15.3552**	**24.7278 %**
out	TGCN	7.2883	32.8440 %	5.9432	**27.1731 %**	17.9647	31.9601 %	17.2318	28.4041 %
	w/ GPT-ST	7.1236	31.6247 %	**5.7325**	27.1994 %	17.1668	30.2569 %	16.6719	26.6769 %
	w/ UMSST	**6.6112**	**30.4076 %**	5.7756	27.8133 %	**14.4488**	**27.3196 %**	**15.4305**	**24.6736 %**

The results shown in Figs. 3 to 6, based on the significant improvements from the two-phase framework, the addition of the temporal decay module provides a stable and consistent further optimization. Across nearly all models, datasets, and all four metrics, the full model achieves the best performance. This indicates that by assigning higher weights to more recent time steps, our model can better capture the temporal dynamics most relevant to the current prediction, thereby effectively making a final refinement to the results.

5.6. Efficiency comparison

To assess the computational efficiency of UMSST, we compare its pre-training time with that of GPT-ST across various datasets in Table 3. Theoretically, our multi-subgraph approach is designed for higher efficiency by distributing the computational load over smaller, localized graph structures, which also facilitates parallel processing.

The results in Table 3 provide strong empirical support for this theory. In particular, the data reveal a clear trend: the larger the dataset scale, the greater the percentage of efficiency improvement from UMSST. This trend demonstrates that our method not only reduces computational cost, but also scales more effectively than existing approaches, making it particularly well-suited for large-scale spatio-temporal graph applications.

Table 3
Efficiency comparison on different datasets (unit: ms).

Model / **Dataset**	NYCBike1	NYCBike2	NYCTaxi1	BJTaxi
UMSST	243,123	357,331	891,493	1,389,211
GPT-ST	301,663	522,996	1,442,752	2,362,553

6. Conclusion

In this paper, we propose UMSST, a novel two-phase, multi-subgraph pre-training framework designed to address the limitations of existing methods in capturing fine-grained local patterns in spatio-temporal

graphs (STGs). Our "coarse-to-fine" strategy first identifies key regions and then performs refined learning on their corresponding subgraphs.

Comprehensive experiments demonstrate that UMSST significantly enhances a wide range of downstream models. Notably, in controlled comparisons against SOTA pre-training methods like GPT-ST on the same base model, our framework demonstrated a stronger potential for performance enhancement, validating the effectiveness of our approach. Furthermore, ablation studies confirm that the multi-subgraph refinement is the primary performance driver, while a temporal decay module provides further optimization. Future work could extend UMSST to other domains and explore its generalization capabilities. Furthermore, while our current key node selection strategy is optimized for improving overall predictive performance, it may overlook critical but anomalous regions (e.g., traffic incident epicenters) whose representations differ significantly from the graph's average state. A promising direction for future research is to develop hybrid selection strategies that can identify and pre-train on both representative and anomalous subgraphs. This could enhance the framework's utility for specialized downstream tasks such as event detection and anomaly analysis.

CRediT authorship contribution statement

Mingze Zhong: Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Zexuan Long:** Writing – review & editing, Visualization, Validation, Software, Project administration, Methodology, Data curation, Conceptualization; **Xinglei Wang:** Writing – review & editing, Writing – original draft, Methodology, Conceptualization; **Tao Cheng:** Writing – review & editing, Methodology, Conceptualization; **Meng Fang:** Writing – review & editing, Methodology, Investigation, Formal analysis, Conceptualization; **Ling Chen:** Writing – review & editing, Methodology, Investigation, Formal analysis, Conceptualization.

Data availability

The datasets used in this study are publicly available. Code supporting this study's findings is available from the corresponding author upon reasonable request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This research did not receive specific grants from funding agencies in the public, commercial or non-profit sectors.

1. Summary of notations

We adopt calligraphic letters for sets and graphs, bold uppercase for matrices, and bold lowercase for vectors. A consolidated list of symbols used throughout the paper is provided in Table 4.

Table 4
Notations and description.

Notation	Description
$\mathcal{G}$	A graph, represented as $(\mathcal{V}, \mathcal{E}, \mathbf{A})$
$\mathcal{V}$	The set of nodes
$\mathcal{E}$	The set of edges
$\mathbf{A}$	Adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N\times N}$
N	Total number of nodes, $N = \lvert\mathcal{V}\rvert$
$\mathbf{X}_t$	Node feature matrix at time step t, $\mathbf{X}_t \in \mathbb{R}^{N\times C}$
C	Initial node feature dimension
T	Historical time step window length
$\mathbf{X}_{t-T+1:t}$	Historical feature tensor for T time steps
D	Node embedding dimension
f_{TC}	Temporal convolution encoder function
f_{SC}	Spatial convolution encoder function
$\mathbf{H}^T_{t-T_{\text{out}}:t}$	Node embeddings after temporal convolution
$\mathbf{H}^S_t$	Node embeddings after spatial convolution at time t
$\alpha_{t'}$	Time decay weight for time step t'
ξ	Hyperparameter for the time decay function
$\mathbf{X}^D_{t-T+1:t}$	Input feature tensor after applying time decay
$\mathbf{H}^{\mathcal{G}}$	Global node embedding matrix from Phase One
$\mathbf{h}^{\mathcal{G}}_{v_n}$	Average embedding of node v_n over samples
$\mathbf{h}^{\mathcal{G}}$	Global graph embedding vector
L	Percentage of key nodes selected as subgraph centers
K	Number of hops for subgraph neighborhood construction
$\mathbf{X}_{l,k\lvert t-T+1:t}$	Input feature tensor of the l-th k-hop subgraph
$N_{S_{l,k}}$	Number of nodes in the l-th k-hop subgraph
$\mathbf{H}^{\text{Sub}}_{l,k}$	Node embedding matrix of the l-th k-hop subgraph
$\mathbf{h}^{\text{Sub}}_{l,k,v_n}$	Embedding of node v_n within the l-th k-hop subgraph
$\mathcal{L}_{\mathcal{G}}$	Loss function for Phase One (global learning)
$\mathcal{L}_S$	Loss function for Phase Two (subgraph learning)
$\mathbf{H}^F$	Final merged node embedding matrix
$\mathbf{h}^F_{v_n}$	Final embedding of node v_n

2. Supplemental experimental validation

2.1. Justification for aggregation method in global embedding

To validate the choice of mean aggregation for computing the global graph embedding $\mathbf{h}^{\mathcal{G}}$ (Eq. 8), we conducted a comparative analysis against two other common aggregation strategies: (1) Degree-Weighted Aggregation, where each node's embedding is weighted by its normalized degree, giving more importance to highly connected nodes; and (2) Random Sampling Aggregation, where the global embedding is computed by averaging the embeddings of a randomly selected subset (e.g., 30 %) of nodes. We applied these three strategies within the UMSST framework and evaluated the final downstream prediction performance. The results consistently showed that mean aggregation provides the most stable and superior performance. Degree-weighted aggregation was sometimes biased towards hubs that might not be representative of the overall graph dynamics, while random sampling often failed to capture a complete picture, leading to suboptimal key node selection. Mean aggregation, by treating all nodes equally, generates a more robust and holistic representation of the graph's state, which is crucial for our goal of identifying globally significant regions.

2.2. Ablation study on key node selection strategy

To validate our key node selection strategy, which is based on similarity to the global graph embedding, we compared it against two alternative baseline strategies: (1) Random Selection(30 %), where subgraph centers are chosen uniformly at random from all nodes; and (2) Degree-based Selection, where nodes with the highest degree are selected as centers, assuming that hubs are important. We implemented these strategies within the UMSST framework and applied them to several downstream models on the NYCBike1 and BJTaxi datasets. The results, summarized in Table 5, demonstrate that our similarity-based method consistently yields the best performance. This suggests that for the goal of improving overall forecasting accuracy, nodes that are most representative of the graph's global state provide more valuable information for pre-training than randomly selected nodes or simple structural hubs. While degree-based selection performs better than random, it can be biased towards static structural properties, whereas our method dynamically identifies nodes that are central to the graph's current spatio-temporal state.

Table 5
Performance comparison of different key node selection strategies on datasets **NYCBike1** and **BJTaxi** in terms of *MAE* and *MAPE*. "in" and "out" in the Metrics column refer to the prediction of traffic in-flow and out-flow, respectively.

Model	Dataset	NYCBike1		BJTaxi	
	Metrics	MAE	MAPE	MAE	MAPE
STWA/in	Random(30 %)	5.2166	26.3977 %	11.7899	16.0208 %
	Degree-based	5.2923	28.1132 %	12.0818	16.5201 %
	Ours	**5.2083**	**26.0928 %**	**11.3858**	**15.8207 %**
STWA/out	Random(30 %)	5.5512	**27.0216 %**	11.7641	16.3220 %
	Degree-based	5.5431	27.0516 %	11.7172	**15.8881 %**
	Ours	**5.5234**	27.0406 %	**11.5649**	15.9794 %
STSGCN/in	Random(30 %)	5.4922	27.9901 %	**12.0396**	16.8909 %
	Degree-based	5.5152	**27.6107 %**	12.4416	16.9792 %
	Ours	**5.4763**	**27.4207 %**	12.0415	**16.8708 %**
STSGCN/out	Random(30 %)	5.7913	28.7691 %	12.1514	17.2111 %
	Degree-based	**5.7461**	**28.2491 %**	12.1441	17.3998 %
	Ours	5.7665	28.2697 %	**12.1364**	**17.0017 %**
TGCN/in	Random(30 %)	**6.1499**	29.8179 %	15.6152	24.9877 %
	Degree-based	6.6514	30.88812 %	16.8182	26.0843 %
	Ours	6.1526	**29.5561 %**	**15.3552**	**24.7278 %**
TGCN/out	Random(30 %)	6.6197	30.7002 %	15.5308	24.7763 %
	Degree-based	7.1431	31.8821 %	16.2322	25.7768 %
	Ours	**6.6112**	**30.4076 %**	**15.4305**	**24.6736 %**
ST-SSL/in	Random(30 %)	4.9292	24.0562 %	12.3398	15.4987 %
	Degree-based	4.9028	23.2486 %	**11.2123**	**14.8685 %**
	Ours	**4.9001**	**23.0429 %**	11.2389	14.9006 %
ST-SSL/out	Random(30 %)	5.1816	24.3988 %	11.9124	16.1020 %
	Degree-based	5.1896	24.1166 %	11.5201	15.1314 %
	Ours	**5.2003**	**23.8132 %**	**11.3210**	**15.0605 %**

References

[1] A. Abubaker, T. Maehara, M. Nimishakavi, V. Plachouras, Self-supervised pretraining for heterogeneous hypergraph neural networks, Technical Report, 2023.
[2] U. Bhattacharya, T. Mittal, R. Chandra, T. Randhavane, A. Bera, D. Manocha, Step: spatial temporal graph convolutional networks for emotion perception from gaits, Proc. AAAI Conf. Artif. Intell. 34 (2020) 1342–1350.
[3] B. Chen, W. Guo, R. Tang, X. Xin, Y. Ding, X. He, D. Wang, TGCN: tag graph convolutional network for tag-aware recommendation, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 155–164.
[4] R.-G. Cirstea, B. Yang, C. Guo, T. Kieu, S. Pan, Towards spatio-temporal aware traffic time series forecasting, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), IEEE, 2022, pp. 2900–2913.
[5] V. M. Croft, S.C.J.L. van Iersel, C. Della Santina, Forecasting infections with spatio-temporal graph neural networks: a case study of the Dutch SARS-CoV-2 spread, Front. Phys. 11 (2023) 1277052. Frontiers Media SA.
[6] J. Fu, W. Zhou, Z. Chen, Bayesian graph convolutional network for traffic prediction, Neurocomputing 582 (2024) 127507.
[7] P. Goodwin, R. Lawton, On the asymmetry of the symmetric MAPE, 15, Elsevier, 1999.
[8] H. Han, M. Zhang, M. Hou, F. Zhang, Z. Wang, E. Chen, H. Wang, J. Ma, Q. Liu, STGCN: a spatial-temporal aware graph learning method for POI recommendation, in: 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 1052–1057.
[9] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, J. Tang, Graphmae: self-supervised masked graph autoencoders, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 594–604.
[10] J. Ji, J. Wang, C. Huang, J. Wu, B. Xu, Z. Wu, J. Zhang, Y. Zheng, Spatio-temporal self-supervised learning for traffic flow prediction, Proc. AAAI Conf. Artif. Intell. 37 (2023) 4356–4364.
[11] G. Jin, H. Yan, F. Li, Y. Li, J. Huang, Dual graph convolution architecture search for travel time estimation, 14, ACM, New York, NY, New York, NY, 2023.
[12] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao, A comprehensive survey on deep graph representation learning, Neural Netw. 173 (2024) 106207.
[13] W. Ju, Y. Zhao, Y. Qin, S. Yi, J. Yuan, Z. Xiao, X. Luo, X. Yan, M. Zhang, Cool: a conjoint perspective on spatio-temporal graph neural network for traffic forecasting, Inform. Fusion 107 (2024) 102341.
[14] W. Ju, Z. Mao, S. Yi, Y. Qin, Y. Gu, Z. Xiao, Y. Wang, X. Luo, M. Zhang, Hypergraph-enhanced dual semi-supervised graph classification, 2024.
[15] M. Li, Z. Zhu, Spatial-temporal fusion graph neural networks for traffic flow forecasting, Proc. AAAI Conf. Artif. Intell. 35 (2021) 4189–4196.
[16] F. Li, H. Yan, G. Jin, Y. Liu, Y. Li, D. Jin, Automated spatio-temporal synchronous modeling with multiple graphs for traffic prediction, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 1084–1093.
[17] Z. Li, L. Xia, Y. Xu, C. Huang, GPT-ST: generative pre-training of spatio-temporal graph neural networks, Adv. Neural Inform. Process. Syst. 36 (2023) 70229–70246.
[18] C. Lin, X. Han, Z. Yu, J. Du, Integrating social and knowledge graphs with time decay mechanisms, in: International Conference on Intelligent Computing, Springer, 2024, pp. 137–149.
[19] D. Liu, J. Wang, S. Shang, P. Han, Msdr: multi-step dependency relation networks for spatial temporal forecasting, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1042–1050.
[20] Y. Qin, W. Ju, H. Wu, X. Luo, M. Zhang, Learning graph ODE for continuous-time sequential recommendation, IEEE Trans. Knowl. Data Eng. 36 (7) (2024) 3224–3236.
[21] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, J. Tang, Gcc: graph contrastive coding for graph neural network pre-training, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1150–1160.
[22] R. Roshankar, M. R. Keyvanpour, Spatio-temporal graph neural networks for accurate crime prediction, in: 2023 13th International Conference on Computer and Knowledge Engineering (ICCKE), IEEE, 2023, pp. 168–173.
[23] L. Sasal, D. Busby, A. Hadid, TempoKGAT: a novel graph attention network approach for temporal graph analysis, Technical Report, 2024.
[24] C. Song, Y. Lin, S. Guo, H. Wan, Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting, Proc. AAAI Conf. Artif. Intell. 34 (2020) 914–921.
[25] S. Thakoor, C. Tallec, M.G. Azar, M. Azabou, E.L. Dyer, R. Munos, P. Veličković, M. Valko, Large-scale representation learning on graphs via bootstrapping, Technical Report, 2021.
[26] C. J. Willmott, K. Matsuura, Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Res. 30 (2005) 79–82.
[27] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, Technical Report, 2019.
[28] Y. Wu, T. Zhang, W. Ke, S. Süsstrunk, M. Salzmann, Spatiotemporal self-supervised learning for point clouds in the wild, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 5251–5260.
[29] G. Xiao, H. Tong, Y. Shu, A. Ni, Spatial-temporal load prediction of electric bus charging station based on S2TAT, 164, Elsevier, 2025.
[30] M. Yang, Z. Liu, L. Yang, X. Liu, C. Wang, H. Peng, P.S. Yu, Instruction-based hypergraph pretraining, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 501–511.
[31] J. Yin, C. Li, H. Yan, J. Lian, S. Wang, Train once and explain everywhere: Pre-training interpretable graph neural networks, Adv. Neural Inform. Process. Syst. 36 (2023) 35277–35299.
[32] Y. You, T. Chen, Y. Shen, Z. Wang, Graph contrastive learning automated, in: International Conference on Machine Learning, PMLR, 2021, pp. 12121–12132.
[33] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, Technical Report, 2017.
[34] J. Zeng, P. Xie, Contrastive self-supervised learning for graph classification, Proc. AAAI Conf. Artif. Intell. 35 (2021) 10824–10832.
[35] S. Zhang, H. Wang, L. Wang, X. Han, Q. Tian, CGCN: context graph convolutional network for few-shot temporal action localization, 62, Elsevier, 2025.
[36] L. Zhou, W. Chen, D. Zeng, S. Cheng, W. Liu, M. Zhang, H. Qu, DPGNN: dual-perception graph neural network for representation learning, Knowledge-Based Systems, 268, Elsevier, 2023.
[37] M. Zhong, H. Xie, Q. Zhu, Quantifying assimilate-contrast effects in online rating systems: modeling, analysis and application, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 2369–2377.
[38] H. Xie, M. Zhong, X. Shi, X. Zhang, J. Zhong, M. Shang, Probabilistic modeling of assimilate-contrast effects in online rating systems, IEEE Transactions on Knowledge and Data Engineering, 36, 2, 795–808, IEEE, 2023.