Signal Processing

Learning Spatial-Temporal Feature with Graph Product

--Manuscript Draft--

Manuscript Number:	SIGPRO-D-23-00171R1
Article Type:	Research Paper
Keywords:	Graph Convolutional Network; Graph Product; Spatial-Temporal Data
Corresponding Author:	Bin Liu, PH.D. Southwestern University of Finance and Economics CHINA
First Author:	Zhuo Tan
Order of Authors:	Zhuo Tan
	Yifan Zhu
	Bin Liu
Abstract:	Earlier works on dynamic spatial-temporal data modelling prefer using spatialtemporal factorized graph convolutional networks (GCNs), which are easy to interpret but fail to capture joint spatial-temporal correlations. Thus, lots of subsequent research focus on constructing a localized adjacent matrix to capture joint features from both spatial and temporal dimension simultaneously. However, their ways of building the adjacent matrices are usually heuristic, which makes the models difficult to interpret. Meanwhile, the lack of theoretical explanations hinders the model's generalization. We introduce a general framework to model dynamic spatial-temporal graph data from the view of graph product. With the power of graph product, we propose a systematical way of constructing the spatial-temporal adjacent graphs, which can not only improve the model's interpretability but increase the spatial-temporal receptive field. Under the novel framework, the existing methods can be taken as special cases of our model. Extensive experiments on multiple large-scale real-world datasets, NTU-RGB+D60, NTU-RGB+D120, UAV-Human, PEMS03, PEMS04, PEMS07, and PEMS08, demonstrate that the proposed model can generalize to most of the scenarios with a performance improvement in a significant margin compared to the state-of-the-art methods.
Suggested Reviewers:	Guosheng Yin gyin@hku.hk
	Shudong Huang huangsd@scu.edu.cn
Response to Reviewers:	see the attached file

conflict of Interest Statement

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.

Learning Spatial-Temporal Feature with Graph Product

Zhuo Tan, Yifan Zhu, Bin Liu*

Center of Statistical Research, School of Statistics, Southwestern University of Finance & Economics Chengdu 610000, China

Abstract

Earlier works on dynamic spatial-temporal data modelling prefer using spatial-temporal factorized graph convolutional networks (GCNs), which are easy to interpret but fail to capture joint spatial-temporal correlations. Thus, lots of subsequent research focus on constructing a localized adjacent matrix to capture joint features from both spatial and temporal dimension simultaneously. However, their ways of building the adjacent matrices are usually heuristic, which makes the models difficult to interpret. Meanwhile, the lack of theoretical explanations hinders the model's generalization. We introduce a general framework to model dynamic spatial-temporal graph data from the view of graph product. With the power of graph product, we propose a systematical way of constructing the spatial-temporal adjacent graphs, which can not only improve the model's interpretability but increase the spatial-temporal receptive field. Under the novel framework, the existing methods can be taken as special cases of our model. Extensive experiments on multiple large-scale real-world datasets, NTU-RGB+D60, NTU-RGB+D120, UAV-Human, PEMS03, PEMS04, PEMS07, and PEMS08, demonstrate that the proposed model can generalize to most of the scenarios with a performance improvement in a significant margin compared to the state-of-the-art methods.

Keywords: Graph Convolutional Network, Graph Product, Spatial-Temporal Data.

*Corresponding author

1. Introduction

Spatial-temporal data is popular in real-world applications, such as traffic forecasting [1], skeleton-based action recognition [2], etc. In traffic forecasting, a traffic network equipped with speed sensors can be modeled as an undirected weighted distance graph, where edge weights represent the relative distance between pairs of sensors. The traffic flow on certain roads can be affected by both their adjacent roads' current flow and historical records, making it necessary to consider both spatial and temporal dependencies when forecasting traffic status.

Lots of efforts have endeavored for spatial-temporal data analysis. Spatial-temporal graph neural networks (STGNNs) [3, 4] play a significant role in modeling the spatial-temporal feature. Methods under this category aim to model the dynamic node inputs and time-varying graph topology under the assumption of inter-dependency between connected nodes. However, most of the STGNNs-based approaches consider the spatial feature and temporal feature separately. Typically, they employ two separate components: a spatial dependencies module and a temporal correlation module to extract spatial and temporal features, respectively [1, 3, 4].

While such spatial-temporal-factorized methods can effectively model the long-range dependencies between graph nodes in the specific spacetime, they impede the direct flow of information across spacetime, limiting their ability to capture complex regional spatial-temporal joint dependencies. For instance, in the field of action recognition, most body gestures have co-occurring motions, such as locomotory movement, walking, which causes arm swinging. Similarly, in traffic prediction, the traffic speed or volume is always influenced by the traffic condition of upstream and similar roads across both space and time, with downstream roads closely correlating to the former and current situation of upstream roads. These information, which is critical to the task, is difficult to effectively captured by factorized model only. Therefore, the ability to leverage those complex cross-spacetime relationships is necessary for effective spatial-

temporal modelling.

To capture more complex spatial-temporal relationships, researchers have started to explore joint spatial-temporal learning. Instead of focusing solely on either spatial or temporal information, models such as the spatial-temporal synchronous graph convolutional network (STSGCN) proposed by [5] consider both by defining a large spatial-temporal graph manually. [6] build upon STSGCN by accounting for temporal patterns shared between disconnected nodes through a spatial-temporal fusion graph. Additionally, [7] generate directed edges to subsequent states along the temporal dimension, constructing a heterogeneous spatial-temporal adjacent matrix that offers a unique perspective on information diffusion to model both spatial and temporal features jointly. These approaches allow for a more comprehensive understanding of spatial-temporal interactive features, beyond the traditional spatial-only and temporal-only modes.

Though existing approaches can effectively capture complex spatio-temporal correlations, the constructed big adjacent matrix only has a limited spatial-temporal receptive field [2, 5, 7]. Besides, the unified spatial-temporal modelling is usually heuristic without any theoretical explanations, which makes their experimental results cannot support the motivations directly. In essense, how the spatial-temporal context configuration (spatial-temporal coinciding graphs) affects the model performance remains unknown.

In this work, we propose a general framework, the spatial-temporal graph product convolutional network (STGPCN), to learn the spatial-temporal and cross-spacetime depending features jointly. Our framework is built on the concept of graph product [8], where the model takes the spatial and temporal graph as inputs and automatically produces a new large coinciding spatial-temporal graph using graph product, without any manual intervention. The most commonly used forms of graph product are Cartesian product and Kronecker product, which represent different patterns of information exchange. With an ensemble pattern, a node can aggregate information not only from spatial neighbors, temporal neighbors but also from spatial-temporal neighbors. In other words, the graph product captures the community structure and the cross-spacetime dependencies.

We can obtain different cross-spacetime graphs by defining different forms of graph product in STGPCN, thus it can capture complex cross-spacetime node relationships. Note that, with a given static spatial graph, a joint spatial-temporal feature can be constructed by a graph product between a spatial graph and a well-defined temporal evolution graph.

By fine-tuning the hyper-parameters of STGPCN, we can effectively control its spatial and temporal receptive field, allowing us to capture complex cross-spacetime relationships by operating convolution on a different cross-spacetime graph defined by graph product operation. Importantly, this approach has the added advantage of producing a sparse adjacency matrix, significantly reducing computational costs without sacrificing performance. Indeed, our experimental results demonstrate that STGPCN can significantly improve the performance of spatial-temporal unified convolutions, making it an important tool for the analysis of massive spatial-temporal graph data.

The contributions of this work are as follows: 1) We propose a general framework that firstly unifies most of the existing spatial-temporal GNNs including the factorized models and spatial-temporal unified. Under this framework, the existing works can be taken as a special case of STGPCN. For example, STSGCN [5] is a special case in our framework by choosing the Cartesian graph product. Alternatively, setting an upper triangular adjacency matrix (a masked Cartesian product) highly resembles ISTD [7]. In MS-G3D [2], a large adjacent matrix was constructed based on different graph connectivity settings. It is also a special case of STGPCN by choosing Cartesian or Lexicographical graph product; 2) Traditional spatial-temporal models are only designed to handle one task, the skeleton-based action recognition or traffic prediction. The proposed STGPCN can adapt to both of them; 3) We evaluate the effectiveness of our model with extensive experiments on several real-world datasets.

The rest of the paper is organized as follows. Section II introduces the related work. Section III formulates the basic graph convolutional models (GCNs) for skeleton-based action recognition and traffic prediction. The components of our proposed STGPCN are introduced in detail in Section IV. The ablation study

and the comparison with the state-of-the-art methods are shown in Section V. Section VI provides some quantitative results and discussions. Section VII concludes the paper.

2. Related Work

2.1. Graph convolutional neural networks

Convolutional neural networks (CNNs) have proven to be effective in processing data residing in the Euclidean space. However, non-Euclidean data, such as graphs and manifolds, remains a challenge for CNNs [9, 10, 11]. Graph data processing has been extensively studied over the decades, and the most established solution is the graph convolutional networks (GCNs) based method, which extends CNNs from regular grids in the Euclidean space to graphs of arbitrary size and shape. Despite the efforts in this rapidly developing deep learning field, existing works can mainly be categorized into two perspectives: spectral-based and spatial-based methods. Methods under the former category perform graph convolution based on the graph spectrum, which stems from the perspective of graph signal processing [12]. Those methods define graph Fourier transform by decomposing the Laplacian matrix and perform graph convolution. Bruna et al. [9] firstly propose the spectral GCNs. They assume that multi-scale information can be learned by higher-order graph polynomial convolution filters. To reduce the model complexity, Defferrard et al. [13] replace the polynomial filters with truncated recursive Chebyshev polynomials filters. Kipf et al. [14] then use the first-order approximation of the Chebyshev polynomials as the graph convolution filter, which further decreases model parameters and achieves considerable performance on multiple node classification benchmarks.

The spatial-based approaches directly define the convolutions by aggregation of graph spatial neighbors. For example, Hamilton et al. [15] generate node embedding based on the sampling and aggregating operation of local neighborhood features of every node. They use different aggregation functions including mean, max, sum, LSTM and achieve consistently comparable performance. Veličković

et al. [10] adopt the attention mechanism to assign relatively large weights on more important nodes for specific tasks during neighborhoods' information aggregation process. Our work follows the spatial perspective methods.

2.2. Skeleton-based action recognition

Skeleton-based action recognition is a crucial application of graph convolutional networks (GCNs). In this framework, the joints of the human body are represented as vertices and the natural physical connection between joints as edges. The spatial-temporal graph convolutional networks (ST-GCN) proposed by Yan et al. [16] is the first work to directly model the skeleton data as the graph structure, where spatial graph convolutions and temporal 1-D convolutions are used interleavingly for spatial-temporal modelling.

To overcome the disadvantages of ST-GCN that the skeleton graph is heuristically predefined regardless of the diversity of data and stays fixed during the training process, Shi et al. [17] adopt a self-attention mechanism and a learnable graph residual mask to increase the flexibility of the graph construction and bring adaptability to various data samples. Besides, they use two-stream architecture to ensemble the second-order information of skeleton data, i.e., bone information to boost recognition accuracy. Li et al. [18] construct generalized skeleton graphs with actional links and structural links to capture action-specific latent dependencies and higher-order relationships. Shi et al. [19] represent the skeleton data as a directed acyclic graph (DAG) on the basis of the kinetics correlation between joints and bones of the human body and propose a novel spatial aggregation scheme. Cheng et al. [20] extend shift operation [21] to graph domain that can adjust the receptive field flexibly and outperform some state-of-the-art with much less computational complexity. However, it is worth noting that the above approaches focus primarily on spatial modelling, but they all ignore that the factorized way of spatial and temporal modelling hinders direct information flow across spacetime for more accurate feature learning.

2.3. Traffic prediction

Traffic forecasting is another popular application for GCNs. Recently, in order to model the spatial correlations of spatial-temporal network data more precisely, researchers try to utilize graph convolution. DCRNN [22] introduces graph convolutional networks into spatial-temporal data prediction tasks, which employs a diffusion graph convolution to depict the information diffusion process in spatial networks. Besides, it uses RNN to model temporal correlations. STGCN [3] uses a similar implementation of graph convolution [13, 14] to extract spatial features and gated-CNNs to extract temporal features, respectively. ASTGCN [4] combines the spatial-temporal attention mechanism and the spatial-temporal convolution, including graph convolutions in the spatial dimension and standard convolutions in the temporal dimension, to capture the dynamic spatial-temporal characteristics. Graph WaveNet [23] designs a self-adaptive matrix to take the variations of the influence between nodes and their neighbors into account. It uses dilated casual convolutions to model the temporal correlations with the exponentially increased receptive fields. However, the above methods only use two different components to capture spatial dependencies and temporal correlations separately. Differ from the aforementioned, STG2Seq [24] tries to model spatial-temporal correlations simultaneously by using a gated residual GCN module with two attention mechanisms. However, to some extent, concatenating features of each node in different time steps obscures spatial-temporal correlations and cannot capture the heterogeneity of spatial-temporal data.

3. Preliminaries

In this section, we provide a detailed formulation of the graph convolution models in the field of two types of application, which is the basis of our proposed model.

3.1. Notations

A spatial graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the node set, those nodes are connected by $|\mathcal{E}|$ edges and the connection between

edges are encoded into an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. Under the assumption that $\mathbf{A}$ is defined as a connection matrix, then the initial value of $\mathbf{a}_{i,j}$ is equal to 1 if there exits an edge between node v_i and node v_j, 0 otherwise. If $\mathbf{A}$ is defined as a distance matrix, then the initial value of $\mathbf{a}_{i,j}$ is calculated by some distance function. In this work, $\mathbf{A}$ is a symmetric matrix since the graph $\mathcal{G}$ we constructed is undirected. The spatial graph $\mathcal{G}$ represents the relationship between the nodes in the graphic space, and its topology stays fixed and does not change with time.

Graph signals $\mathcal{X} = \{x_{t,n} \in \mathbb{R}^C \mid t, n \in \mathbb{Z}, 1 \leq t \leq T, 1 \leq n \leq N\}$ are represented by a feature tensor $\mathbf{X} \in \mathbb{R}^{T \times N \times C}$, where $x_{t,n} := \mathbf{X}_{t,n,:}$ is the C dimensional feature vector for node v_n at time t over total T time steps. The graph signal matrix $\mathbf{X}_t \in \mathbb{R}^{N \times C}$ represents the observations of the spatial network $\mathcal{G}$ at time t. The input is thus adequately described by $\mathbf{A}$ structurally and $\mathbf{X}$ feature-wise. Suppose $\mathbf{W}^{(l)} \in \mathbb{R}^{C_l \times C_{l+1}}$ denotes a trainable weight matrix at the l-th layer, then the spatial-temporal data classification problem can be modeled as learning a classifier $\mathcal{F}_1$ to predict the category of a spatial-temporal network series $(\mathbf{X}_{t-T+1}, \mathbf{X}_{t-T+2}, \ldots, \mathbf{X}_t)$. Another task is the spatial-temporal network data forecasting, which is a regression problem. That is, learn a mapping function $\mathcal{F}_2$ to predict the future observations $\mathbf{X}_{t+1}, \mathbf{X}_{t+2}, \ldots, \mathbf{X}_{t+T'}$ based on the historical observations $\mathbf{X}_{t-T+1}, \mathbf{X}_{t-T+2}, \ldots, \mathbf{X}_t$, where T and T' are the length of historical series and target spatial-temporal network series to be forecasted respectively. Note that the prediction is made iteratively in this paper. Though tasks are different, the essence is exact the same, that is, to capture spatial-temporal dependencies.

3.2. Graph construction

Graph construction is the first and crucial step for utilizing graph-based deep learning methods to solve real tasks. Spatial graph $\mathcal{G}$ is generally composed of two parts, the first one is adjacency matrix $\mathbf{A} = \{a_{ij}\}_{N \times N}$ which is determined by the node set $\mathcal{V}$, the connection relationship between node pairs $\mathcal{E}$ and the relative importance of edges a_{ij}. The construction of the adjacency matrices

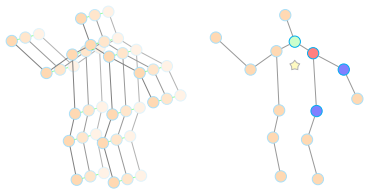


Figure 1: Left: illustration of the factorized spatial-temporal graph. Right: illustration of the mapping strategy. Different colours denote different subsets.

depends on both of the datasets and the assumption of node relationships, which could be static or dynamic. The second part is node feature matrix $\mathbf{X}_t$, which changes with different datasets. We first introduce how to construct the node sets and the corresponding node features. And a systematic introduction about the commonly-used mode of adjacency matrix construction will be stated subsequently.

3.2.1. Node feature matrix construction

Generally, there is no unified standard to get node features. We have to deal with it case by case in practice. For example, in action classification, the raw skeleton data can be represented as sequential snapshots of 2D or 3D crucial body joints coordinates. These joints of a human body are represented by nodes of the skeleton graph. The feature of each node is its coordinates that are collected by the equipped sensors or extracted from raw video by a specific body pose algorithm [25]. In traffic prediction, however, the traffic measurements (e.g., traffic speed) are generally collected during a short time interval by the sensors deployed on a road network. The nodes of a traffic graph are the sensors located at different roads and the node features are the traffic measurements collected by the corresponding sensors.

3.2.2. Adjacency matrix construction

The adjacency matrix $\mathbf{A} = \{a_{ij}\}_{N\times N} \in \mathbb{R}^{N\times N}$ captures the explicit spatial dependencies that are valuable for prediction. Its element a_{ij} (unweighted or weighted) represents a pairwise relationship between nodes. However, there are different assumptions for node relationships in different scenarios. Hence, the adjacency matrix could be constructed in various ways, e.g., fixed matrix or adaptive matrix. Some existing works [14, 26, 27] assume that the correlations between nodes are static over time. Then they manually construct fixed adjacency matrices to capture predefined correlations between nodes, such as citation relationship [14], function similarity [26] and semantic connection [27], etc.

One of the most frequently used adjacency matrix is the connection matrix. It depicts the connectivity between nodes, and the entry of the matrix is defined as $a_{ij} = 1$ (connected) or $a_{ij} = 0$ (disconnected). For instance, in [16, 21], the authors use the natural connection between human joints to define the adjacency matrix of a skeleton-based graph (Figure 1 left). In transportation prediction, the adjacent relationship among nodes is built based on their geographical distance. Thus, a_{ij} is defined as a function of the relative distance between two nodes, and usually calculated by a Gaussian Kernel as follows,

$$a_{ij} = \begin{cases} exp(-\frac{d_{ij}^2}{\sigma^2}) & if\ i \neq j \text{ and } d_{ij} \geq \epsilon \\ 0 & if\ i = j \text{ and } d_{ij} < \epsilon \end{cases}$$

where d_{ij} is the distance between node i and node j, hyper-parameters σ and ϵ are thresholds to control the distribution and sparsity of the matrix $\mathbf{A}$.

The aforementioned two types of the adjacency matrix are predefined, it is easy to construct but does not necessarily reflect the intrinsic dependencies among nodes due to defective prior knowledge or incomplete data [28]. Several methods [17, 29, 30, 23] have also been proposed to design a novel adaptive adjacency matrix that can be inferred from data, and corresponding experiments in those methods show that adaptive matrix does capture the hidden spatial dependency more precisely.

3.3. Factorized spatial-temporal graph convolution

Most of the existing methods use spatial-only (e.g., GCNs) and temporal-only (e.g., CNNs) modules to model spatial [31, 16] and temporal correlations [16, 17, 3, 4, 29] separately. In this section, we will briefly review them.

3.3.1. Spatial-only module

One of the most significant spatial-only models is ST-GCN [16]. Let B_i denotes the 1-hop neighbor set of a vertex v_i of a graph $\mathcal{G}$. Similar to the traditional convolution operation, w is the weighting parameter that provides a weight vector for a given input of ST-GCN. The cardinality of B_i is usually varied, but the number of weight vectors is associated with a predefined mapping $l_i : B_i \to \{0, \ldots, K-1\}$ has to be fixed[16]. ST-GCN [16] assigns the nodes in B_i with a fixed-numbered subset of labels, and each of them is associated with a unique learnable weight vector. For example, in the field of action classification, to capture more refined location information, three kinds of special B_i have been explored: 1) $\mathcal{P}_{i1}$ is the root node itself (the red circle in Figure1 right); 2) $\mathcal{P}_{i2}$ is the centripetal group (the green vertex in Figure1 right), which contains the neighboring nodes nearest to the gravity center of the skeleton (the yellow star in Figure1 right) than the root node; 3) $\mathcal{P}_{i3}$ is the centrifugal group (the blue dots in Figure1 right), which contains the neighboring nodes that are farther from the gravity center.[1]

The left panel of Figure 1 illustrates the graph $\mathcal{G}$ in ST-GCN. With the aforementioned neighbor set B_i, the graph convolution operation on vertex v_i of ST-GCN can be calculated as,

$$f^{(l+1)}(v_i) = \sum_{v_j \in B_i} \frac{1}{Z_{ij}} f^{(l)}(v_j) \cdot w(l_i(v_j)),$$

[1] Coordinate of the gravity center is the average coordinate of all joints in the skeleton at a frame.

where f denotes the feature map and $l_i(\cdot)$ is defined as follow,

$$l_i(v_j) = \begin{cases} 0 & if\ r_j = r_i \\ 1 & if\ r_j < r_i \\ 2 & if\ r_j > r_i. \end{cases}$$

In action recognition, $l_i(\cdot)$ is partly inspired by the fact that motion of body parts can be broadly categorized as concentric and eccentric motions, and r_i is the average distance from gravity center to joint i over all frames in the training set. $Z_{ij} = |\{v_k | l_i(v_k) = l_i(v_j)\}|$ denotes the cardinality of $\mathcal{P}_{ik}$ that contains v_j, which is added to balance the contributions of different subsets.

It is worth noting that this formulation can resemble the standard 2D convolution if we take an image as a regular 2D grid. For instance, to resemble a 3×3 convolution operation, we have a local 3×3 grid around the center pixel. The neighbor set could be partitioned into 9 subsets, and each contains one pixel.

Let $\mathbf{X}_t^{(l)} \in \mathbb{R}^{N \times d_{in}}$ be the input features of all joints at time t in the l-th layer, where d_{in} is the dimension of input feature; $\mathbf{X}_t^{(l+1)} \in \mathbb{R}^{N \times d_{out}}$ denotes the output features obtained from the spatial graph convolution, where d_{out} is the dimension of output features. Then the spatial graph convolution can be formulated as,

$$\mathbf{X}_t^{(l+1)} = \sum_{p \in \mathcal{P}} \mathbf{M} \odot \tilde{\mathbf{A}}^{(p)} \mathbf{X}_t^{(l)} \mathbf{W}_{(p)}^{(l)}, \tag{1}$$

where $\mathcal{P}$ is the partition strategy we discussed above. $\tilde{\mathbf{A}}^{(p)} = {\mathbf{\Lambda}^{(p)}}^{-\frac{1}{2}} (\mathbf{A}^{(p)} + \mathbf{I}) {\mathbf{\Lambda}^{(p)}}^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$ is a normalized adjacency matrix where $\mathbf{A}^{(p)}$ is similar to the adjacency matrix and $\mathbf{\Lambda}_{ii}^{(p)} = \sum_j (\mathbf{A}^{(p)} + \mathbf{I})$ is a normalized diagonal matrix. The element $\mathbf{A}_{ij}^{(p)}$ indicates whether the vertex v_j is in the set $\mathcal{P}_{ip}$. $\mathbf{W}_{(p)}^{(l)} \in \mathbb{R}^{d_{in} \times d_{out}}$ are trainable weights of the convolution for each partition that aim to capture the feature importance in l-th layer. $\mathbf{M} \in \mathbb{R}^{N \times N}$ is an learnable attention map that indicates the importance of edges, and $\odot$ is the Hadamard product.

In other scenarios, we do not have to assign different weights to different subsets of nodes. Therefore, the mapping l_i equals to 0 for any neighbor of the

specific node v_i and the node v_i itself. It means that all neighbor nodes belong to the same partition and share the same weight, so the Eq. 1 degenerates to

$$\mathbf{X}_t^{(l+1)} = \mathbf{M} \odot \tilde{\mathbf{A}} \mathbf{X}_t^{(l)} \mathbf{W}^{(l)}. \tag{2}$$

3.3.2. Temporal-only module

After capturing the spatial neighboring information, the factorized spatial-temporal graph convolution extracts temporal dependency sequentially. Specifically, the classic 1-D used in skeleton-based action recognition or casual convolutions used in traffic prediction with kernel size K_t are stacked to update the node signal by merging the temporal correlations at the neighboring time step, thus the number of temporal neighbors for each node is fixed as 2 and K_t can be adjusted to control the receptive field of temporal dimension [16, 17, 3, 4, 29]. With the above spatial-only and temporal-only modules, one layer factorized spatial-temporal graph convolution module is obtained. Then, stacked factorized spatial-temporal convolution modules can extract high-level semantic features based on the input graph.

3.4. Unified spatial-temporal graph convolution

A unified spatial-temporal graph convolution aims to capture the influence of each node on its neighbors that belongs to both the current time step and the adjacent time step directly. By connecting all nodes with themselves at the previous and the next moment, Song et al. [5] proposed to construct a localized spatial-temporal graph $\mathbf{A}' \in \mathbb{R}^{3N \times 3N}$ manually (as shown in Eq. 3). As we see, the new adjacency matrix $\mathbf{A}'$ contains $3N$ nodes. The diagonal blocks of the adjacency matrix are the adjacency matrices of the spatial networks along three continuous time steps. The off-diagonal entries of $\mathbf{A}'$ indicate the connectivity of each node to itself that belongs to the adjacent time steps. It is consistent with the Cartesian product.

$$\mathbf{A}' = \begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \tilde{\mathbf{A}} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \tilde{\mathbf{A}} \end{bmatrix}_{3N \times 3N} \tag{3}$$

With the manual crafted spatial-temporal adjacent matrix $\mathbf{A}'$ (Eq. 3), the correlations between each node and its spatial-temporal neighbors can be captured simultaneously. A graph convolutional operation over the vertex domain is further used to aggregate localized spatial-temporal features. The input of the graph convolution is the graph signal matrix of the localized spatial-temporal graph. Within the graph convolutional operation, each node aggregates the features of itself and neighbors at an adjacent time step, which is formulated as,

$$\mathbf{h}^{(l+1)} = \sigma(\mathbf{M}' \odot \mathbf{A}'\mathbf{h}^{(l)}\mathbf{W}) \in \mathbb{R}^{3N \times C'},$$

where $\mathbf{h}^{(l)} \in \mathbb{R}^{3N \times C^l}$ is the input of the $(l+1)$-th graph convolutional layer. Specifically, $\mathbf{h}^{(0)} \in \mathbb{R}^{3N \times C}$ is the concatenated features from $3N$ nodes. $\mathbf{W} \in \mathbb{R}^{C \times C'}$ is learnable parameter and $\mathbf{M}' \in \mathbb{R}^{3N \times 3N}$ denotes the adaptive mask.

4. Graph product convolutional network

4.1. Graph products

In mathematics, graph product is a binary operation on two graphs [32]. It accepts two graphs $\mathcal{G}_1, \mathcal{G}_2$ as inputs and outputs a new large graph $\mathcal{G}$ with the following two properties: 1) the vertex set of $\mathcal{G}$ is $\mathcal{V}_1 \times \mathcal{V}_2$, where $\mathcal{V}_1$ and $\mathcal{V}_2$ are the vertex sets of $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively; 2) two vertices (a_1, a_2) and (b_1, b_2) of $\mathcal{G}$ are connected by an edge if and only if a certain condition about a_1, b_1 in $\mathcal{G}_1$ and a_2, b_2 in $\mathcal{G}_2$ is met.

Figure 2 demonstrates some examples of the graph product with a spatial graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1, \mathbf{A}_1)$ with $N_1 = |\mathcal{V}_1|$ nodes and three different kind of temporal graphs (un-directed, directed, and recurrent) $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2, \mathbf{A}_2)$ with $N_2 = |\mathcal{V}_2|$ nodes (Figure 2a). For better to visualize the process of graph product, we assign nodes in spatial graph with different colors, and nodes in temporal graph with increasing grayscale to denote time evolution. Let $\diamond$ denotes the graph product operation, then the result of graph product between $\mathcal{G}_1$ and $\mathcal{G}_2$ is,

$$\mathcal{G} := \mathcal{G}_1 \diamond \mathcal{G}_2 = (\mathcal{V}, \mathcal{E}, \mathbf{A}_\diamond),$$

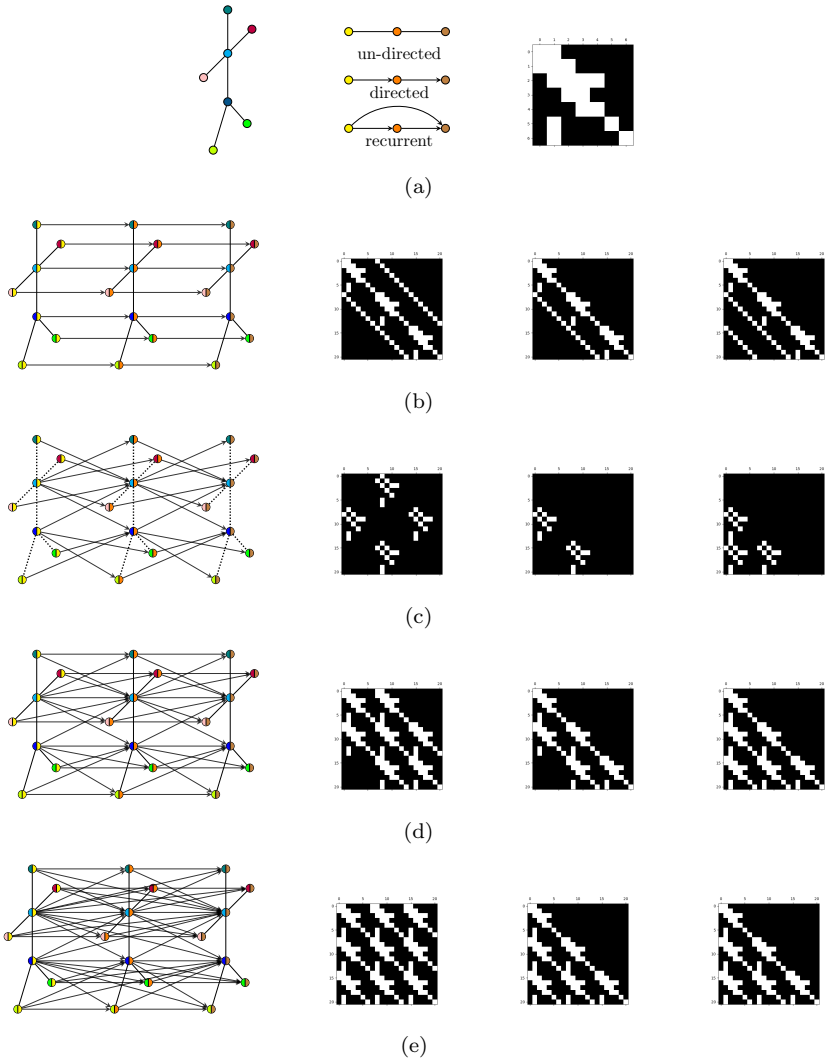


Figure 2: Left column: Illustration of different graph products. Right column: Visualization of the corresponding adjacency matrices. The white color indicates the connections. (a) Factorized spatial graph (seven nodes located in different spatial positions are assigned different colors) and three types of temporal graphs (three nodes along the timeline are assigned colors in yellow, orange, and brown). (b), (c), (d), and (e) represent the adjacency matrices generated by Cartesian product, Kronecker product, Strong product, and Lexicographical product with three temporal graphs respectively. The solid line denotes real connection, dashed line denotes nonexistent connection. Each node has two colors of product graph, the left-half color of each node represents the spatial position and the right-half one represents the temporal position.

with $|\mathcal{V}| = N_1 N_2$ nodes and an appropriately defined adjacency matrix $\mathbf{A}_\diamond \in \mathbb{R}^{N_1 N_2 \times N_1 N_2}$.

Figure 2b 2c 2d 2e visualize the product graphs from $\mathcal{G}_1$ and $\mathcal{G}_2$ with different graph products as well as different type of temporal graph (un-directed, directed and recurrent). Each node in the product graphs splits in half and contains two colors which represent its spatial position and time evolution step respectively. With this setting, the resulted graph illuminates how to deploy graph products to produce the spatial-temporal graphs from $\mathcal{G}_1$ and $\mathcal{G}_2$.

For instance, Figure 2d visualizes the Strong product (one kind of graph product) of $\mathcal{G}_1$ and $\mathcal{G}_2$, each node of the resulted $\mathcal{G}$ has its own left-half color which is same as the spatial graph $\mathcal{G}_1$ (Figure 2a left), while nodes in the same time step have the same right-half color. According to the definition of Strong product in Table 1, there is a spatial-temporal connection between blue-yellow and blue-orange nodes because the blue nodes are the same node in the spatial graph and the yellow and orange nodes are temporal 1-hop neighbors. Meanwhile, there is a spatial-temporal connection between blue-yellow and red-orange because the blue and red nodes are spatial 1-hop neighbors and the yellow and orange ones are temporal 1-hop neighbors. In particular, four commonly studied graph products are the Cartesian product, Kronecker product, Strong product, and the Lexicographical product.

Cartesian graph product. The Cartesian graph product of $\mathcal{G}_1$ and $\mathcal{G}_2$, denoted as $\mathcal{G} = \mathcal{G}_1 \times \mathcal{G}_2$, for which vertices (a_1, a_2) and (b_1, b_2) are adjacent of $\mathcal{G}$ precisely if $a_1 = b_1$ and the temporal edge $a_2 b_2 \in \mathcal{E}_2$, or the spatial edge $a_1 b_1 \in \mathcal{E}_1$ and $a_2 = b_2$ [32]. One demo of Cartesian graph product is as shown in Figure 2b, which connects the spatial graph $\mathcal{G}_1$ by adding edges between the same spatial nodes of adjacent time steps. Its adjacency matrix is,

$$\mathbf{A}_\times = \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2,$$

where the $\otimes$ is the Kronecker product.

Kronecker graph product. The Kronecker graph product of $\mathcal{G}_1$ and $\mathcal{G}_2$, denoted as $\mathcal{G} = \mathcal{G}_1 \otimes \mathcal{G}_2$, and for which vertices (a_1, a_2) and (b_1, b_2) are adjacent

of $\mathcal{G}$ precisely if $a_1b_1 \in \mathcal{E}_1$ and $a_2b_2 \in \mathcal{E}_2$ [32]. As shown in Figure 2c, the product graph deletes spatial connections in each time step (dashed line in Figure 2c) and generates new spatial-temporal edges for node pairs which are not only one-hop spatial neighbor but one-hop temporal neighbor. Thus the adjacency matrix is obtained by the matrix Kronecker product of adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$,

$$\mathbf{A}_{\otimes} = \mathbf{A}_1 \otimes \mathbf{A}_2,$$

Strong graph product. For the Strong product, which can be seen as a combination of the Kronecker and Cartesian products (Figure 2d), denoted as $\mathcal{G} = \mathcal{G}_1 \boxtimes \mathcal{G}_2$, the adjacency matrix is [32],

$$\mathbf{A}_{\boxtimes} = \mathbf{A}_1 \otimes \mathbf{A}_2 + \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2. \tag{4}$$

Lexicographical graph product. The Lexicographical product of $\mathcal{G}_1$ and $\mathcal{G}_2$ is a graph $\mathcal{G}$, denoted as $\mathcal{G} = \mathcal{G}_1 \circ \mathcal{G}_2$. Two vertices (a_1, a_2) and (b_1, b_2) are adjacent of $\mathcal{G}$ precisely if $a_1 = b_1$ and $a_2b_2 \in \mathcal{E}_2$, or $a_1b_1 \in \mathcal{E}_1$ (illustrated in Figure 2e). The increment of Lexicographical graph product on the basis of Strong product is that it considers spatial neighbors among all time steps. Thus, the adjacency matrix is [32],

$$\mathbf{A}_{\circ} = \mathbf{A}_1 \circ \mathbf{A}_2,$$

where the operation $\circ$ will be explained in section 4.2.

Table 1 summarizes the definitions of those four graph products. An intuitive and appropriate application of the product graphs is the spatial-temporal data collected by a deep sensor/camera network. Graph signals $\mathcal{X}$ which consists of measurements of all sensors at all time steps reside on the product of the spatial and the temporal graph. As the example illustrated in Figure 2e, the Lexicographical product relates the measurement of the n-th sensor at time step k to its measurements at all time steps, as well as to measurements of its spatial neighbors at all time steps. Depending on the choice of product, the graph signal of one sensor is related to the signals of this sensor and its spatial neighbors at the same time and adjacent time steps, even all time steps.

Name	Condition for $(a_1, a_2) \sim (b_1, b_2)$
Cartesian product ($\mathcal{G}_1 \times \mathcal{G}_2$)	$a_1 = b_1$ and $a_2 \sim b_2$ or $a_1 \sim b_1$ and $a_2 = b_2$
Kronecker product ($\mathcal{G}_1 \otimes \mathcal{G}_2$)	$a_1 \sim b_1$ and $a_2 \sim b_2$
Strong product ($\mathcal{G}_1 \boxtimes \mathcal{G}_2$)	$a_1 = b_1$ and $a_2 \sim b_2$ or $a_1 \sim b_1$ and $a_2 = b_2$ or $a_1 \sim b_1$ and $a_2 \sim b_2$
Lexicographical product ($\mathcal{G}_1 \cdot \mathcal{G}_2$)	$a_1 \sim b_1$ or $a_1 = b_1$ and $a_2 \sim b_2$

Table 1: Definitions of four graph products, where $(a_1,\ a_2)$ and $(b_1,\ b_2)$ are nodes of the augmented graph generated by graph product. a_i, b_i are nodes of graph $\mathcal{G}_i$. $\sim$ denotes that there exists an edge connecting two corresponding nodes.

4.2. Unified spatial-temporal graph product convolution

We deem that factorized spatial-temporal methods hinder the direct information flow across spacetime for capturing complex regional spatial-temporal joint dependencies. In the ideal scenario, if the constructed graph topology reflects connectivity between node pairs, then the information propagation mechanism will make those node pairs incorporate a considerable portion of each other's features after several layers of the spatial-temporal factorized module. In effect, by stacking the spatial-temporal factorized convolution modules, updated features of node pairs do not effectively reflect the connectivity because an increasingly larger spatial-temporal receptive field introduces redundant information of irrelevant nodes. Furthermore, though some nodes reside on different time steps, they may have strong connections and the information propagation

among those should be direct and effective. To overcome the above problem, we use the proposed graph product approach to learn cross-spacetime direct dependencies, which can automatically capture space-time correlations rather than a manual-crafted design (Eq. 3) [5].

Consider a size m window of the temporal dimension sliding along the input graph sequence. At each slides, it obtains a spatial-temporal subgraph $\mathcal{G}_{(m)} = (\mathcal{V}_{(m)}, \mathcal{E}_{(m)}, \tilde{\mathbf{A}}_{(m)})$, where $\mathcal{V}_{(m)} = \mathcal{V}_1 \cup \cdots \cup \mathcal{V}_m$ is the union of all node sets across the m time steps within the time window. The initial edge set $\mathcal{E}_{(m)}$ is defined based on an augmented graph adjacency matrix $\tilde{\mathbf{A}}_{(m)}$ which depends on the chosen graph products.

4.2.1. Cartesian graph product

$$\tilde{\mathbf{A}}_{(m)} = \begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \tilde{\mathbf{A}} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \tilde{\mathbf{A}} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{A}} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \tilde{\mathbf{A}} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & \tilde{\mathbf{A}} \end{bmatrix}_{mN \times mN}$$

where $\tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$ represents normalized adjacency matrix with self loop and $\mathbf{I} \in \mathbb{R}^{N \times N}$ is an identity matrix. Intuitively, this setting unfolds the spatial graphs within the time window to a larger graph by simply introducing temporal self-edges between adjacent time steps in the window.

4.2.2. Kronecker graph product

$$\tilde{\mathbf{A}}_{(m)} = \begin{bmatrix} \mathbf{0} & \mathbf{A} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A} & \mathbf{0} & \mathbf{A} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{A} & \mathbf{0} & \mathbf{A} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{A} & \mathbf{0} \end{bmatrix}_{mN \times mN}$$

where $\mathbf{A}$ represents adjacency matrix without self-loop, which intuitively represents that every node in $\mathcal{V}_i$ is only connected to its 1-hop spatial neighbors in $\mathcal{V}_j$.

4.2.3. Strong graph product

$$\tilde{\mathbf{A}}_{(m)} = \begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \end{bmatrix}_{mN \times mN}$$

where $[\tilde{\mathbf{A}}_{(m)}]_{i,j} = \tilde{\mathbf{A}},\ \forall i, j = 1, \dots, m$ intuitively suggests that every node in $\mathcal{V}_i$ is connected to itself and its 1-hop spatial neighbors in $\mathcal{V}_j$ and $[\tilde{\mathbf{A}}_{(m)}]_{i,i} = \tilde{\mathbf{A}}$ means the original spatial connections at i time step. Thus, each node within $\mathcal{G}_{(m)}$ is densely connected to itself and its 1-hop spatial neighbors between two consecutive time steps.

4.2.4. Lexicographical graph product

$$\tilde{\mathbf{A}}_{(m)} = \begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \end{bmatrix}_{mN \times mN}$$

The augmented adjacency matrix $\tilde{\mathbf{A}}_{(m)}$ generated by Lexicographical product resembles the one generated by Strong product, the only difference lies in that nodes within $\mathcal{G}_{(m)}$ are not only densely connected to itself and its 1-hop spatial neighbors between two consecutive time steps, but all m time steps.

Besides the constructed spatial-temporal matrix $\tilde{\mathbf{A}}_{(m)}$, the features of corresponding T windows (each size is m) should also be stacked, then we can easily obtain $\mathbf{X}_{(m)} \in \mathbb{R}^{T \times mN \times C}$ using the same size m sliding window over $\mathbf{X}$ with zero padding to construct the T windows. Using Eq. 2, we thus arrive at a unified spatial-temporal graph convolutional operator for the t-th temporal window,

$$\mathbf{X}_{(m)}^{(l+1)} = \sigma(\tilde{\mathbf{A}}_{(m)} \mathbf{X}_{(m)}^{(l)} \mathbf{W}^{(l)}) \tag{5}$$

A multi-scale aggregation scheme can also be integrated into our model for reasoning directly in the spatial-temporal domain. We thus derive STGPCN from Eq. 5 as,

$$\mathbf{X}_{(m)}^{(l+1)} = \sum_{k=0}^{K} \sigma(\tilde{\mathbf{A}}_{(m)}^{k} \mathbf{X}_{(m)}^{(l)} \mathbf{W}_{(k)}^{(l)}) \tag{6}$$

where $\tilde{\mathbf{A}}_{(m)}^{k}$ is the k-order power of $\tilde{\mathbf{A}}_{(m)}$.

To improve the flexibility of STGPCN which performs homogeneous neighborhood averaging, we add a simple learnable, unconstrained graph residual mask $\mathbf{A}^{res}$ inspired by [17] [19] to every $\tilde{\mathbf{A}}_{(m)}^{k}$ to strengthen, weaken, add, or remove edges dynamically. The intuition behind this is the predefined graph

topology may not be perfect, adding $\mathbf{A}^{res}$ helps the model to learn an adaptive adjacency matrix based on the predefined one in a data-driven manner. Besides, adding $\mathbf{A}^{res}$ can also improve the robustness to noises. Then Eq. 6 is updated to

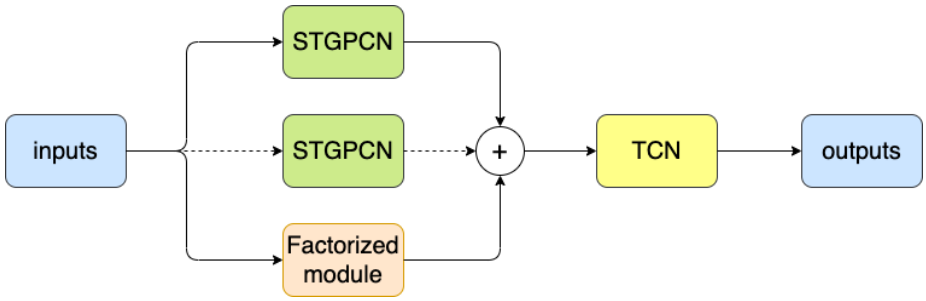
$$\mathbf{X}_{(m)}^{(l+1)} = \sum_{k=0}^{K} \sigma((\tilde{\mathbf{A}}_{(m)}^{k} + \mathbf{A}^{res})\mathbf{X}_{(m)}^{(l)}\mathbf{W}_{(k)}^{(l)}) \tag{7}$$

$\mathbf{A}^{res}$ is initialized with random values around zero, allowing each multi-scale context (either spatial or spatial-temporal) to select the best-suited mask.

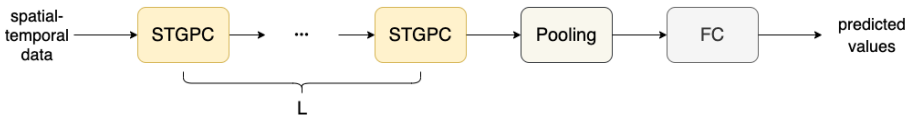
In short, the proposed STGPCN is designed to capture complex localized spatial-temporal but not long-range dependencies. Although increasing the window size m will have a larger temporal receptive field but will involve some irrelevant information and increase the computational complexity as well. In contrast, the factorized spatial-temporal graph convolution is good at capturing the long-range dependencies. Therefore, we combine the unified spatial-temporal graph product convolution in different window sizes with factorized spatial-temporal graph convolution to capture both complex localized and long-range dependencies. The experimental results also support us to select this final combined architecture as well.

4.3. Graph product convolutional networks

Figure 3b illustrate the overall architecture of our model, which is a cascade of L spatial-temporal graph product convolutional (STGPC) blocks to extract features from spatial-temporal data, a subsequent pooling layer and fully connected layer aims at mapping the features into a prediction vector or an additional softmax classifier. Each STGPC block (shown in Figure 3a contains three paths to simultaneously capture complex spatial-temporal cross dependencies and long-range spatial-temporal relationships. The STGPCN path first constructs spatial-temporal windows, performs unified graph product convolutions (Eq. 6), and then collapses them for window feature readout. The two STGPCN paths have different window sizes. The ST-GCN [31] path first performs factorized graph convolutions (Eq. 2) on inputs features to capture spatial relationships,



(a)



(b)

Figure 3: STGPC block and model architecture. (a) Spatial-temporal graph product convolutional networks (STGPC) block. The "Factorized module" denotes classic factorized spatial-temporal graph convolution and TCN denotes 1-D convolutional networks. The "Factorized module" could be an STGCN or ST-GCN (section 3.3 introduced) or a gated-CNN in practice. The STGPCN pathway aims to capture spatial-temporal cross relationships. For large datasets, one optional STGPCN pathway with different window sizes is added to gain different coarse-grained or fine-grained information. (b) The overall architecture of our model. It consists of a stack of STGPC blocks that can extract outstanding features from sequences, followed by a global average pooling layer and a full-connected layer.

and then followed by a TCN, i.e., classical 1-D convolution or casual convolution, to capture the long-range temporal contexts. The outputs from all pathways are aggregated (added) and then fed into a temporal convolution as the output of STGPC block.

4.4. Connections with existing methods

Our method unifies both factorized model and spatial-temporal unified model.

4.4.1. Factorized models

Factorized approaches deploy spatial-only (e.g., GCNs) and temporal-only (e.g., CNNs) modules separately [16], [3]. Most of factorized approaches [16], [3]

are special cases in our framework by dropping spatial-temporal cross edges as below.

$$\tilde{\mathbf{A}}_{(m)}^{\text{factorized}} = \begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{A}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \tilde{\mathbf{A}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}} \end{bmatrix}_{mN \times mN}$$

With $\tilde{\mathbf{A}}_{(m)}^{\text{factorized}}$, we can perform spatial convolution over m time steps, which is similar to the mini-batching in PyG [2] (a widely used GNN library built upon PyTorch). However, we have to deploy the additional temporal module to capture the temporal dependencies after the spatial convolution. Moreover, it can not capture spatial-temporal cross dependencies.

4.4.2. Unified models

Existing unified approaches simultaneously capture the localized spatial-temporal correlations directly. Most of them [5] [7] [2] are special cases in our framework by choosing different graph products.

For example, **ISTD-GCN** [7] connects adjacent snapshots by generating edges to its subsequent state along with the temporal dimension. The heterogeneous spatial-temporal graph is defined as follows,

[2] https://pytorch-geometric.readthedocs.io/en/latest/

$$\tilde{\mathbf{A}}^{\mathrm{ISTD}}_{(m)} = \begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{A}} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{A}} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \tilde{\mathbf{A}} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}} \end{bmatrix}_{mN \times mN}$$

where $\tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$ represents normalized adjacency matrix with self loop, and $\mathbf{I} \in \mathbb{R}^{N \times N}$ is an identity matrix that denotes the generated edges that connecting all snapshots. It is actually the upper triangular version of Cartesian product, which reflects that the generated edges are directed since the time series is uni-directional, hence it is a special case of STGPCN.

STSGCN [5] connects all nodes with themselves at the adjacent time steps and gets a localized spatial-temporal graph. The adjacency matrix of the localized spatial-temporal graph is,

$$\tilde{\mathbf{A}}^{\mathrm{STSGCN}}_{(m)} = \begin{bmatrix} \tilde{\mathbf{A}} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \tilde{\mathbf{A}} & \mathbf{I} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \tilde{\mathbf{A}} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \tilde{\mathbf{A}} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & \tilde{\mathbf{A}} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} & \tilde{\mathbf{A}} \end{bmatrix}_{mN \times mN}$$

STSGCN sets window size $m = 3$ to directly capture the influence of each node on its one-hop neighbors which belong to both the current time step and the adjacent time steps. It is exactly the Cartesian product.

MS-G3D [2] connects all nodes with themselves and their 1-hop neighbors across all m time steps. Its adjacency matrix is the Lexicographical product

form defined as follow,

$$\tilde{\mathbf{A}}_{(m)}^{\text{MS-G3D}} = \begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \dots & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} & \tilde{\mathbf{A}} \end{bmatrix}_{mN \times mN}$$

In conclusion, our framework unifies the most of existing spatial-temporal graph convolution network models both in skeleton-based action recognition and traffic prediction.

4.5. Complexity analysis

According the definition of STGPCN as shown in Eq.5, where $\tilde{\mathbf{A}}_{(m)} \in \mathbb{R}^{mN \times mN}$, $\mathbf{X}_{(m)}^{(l)} \in \mathbb{R}^{mN \times d_l}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ and $\mathbf{X}_{(m)}^{(l+1)} \in \mathbb{R}^{mN \times d_{l+1}}$, hence, the time complexity of STGPCN over T time steps is $T \cdot \mathcal{O}(m^2 N^2 d_l + mN d_l d_{l+1})$. The time complexity of the subsequent collapse operation (mean) in the m dimension is $\mathcal{O}(mN d_{l+1}) \cdot T$. Therefore, the time complexity of single pathway of STGPCN is $T \cdot \mathcal{O}(m^2 N^2 d_l + mN(d_l + 1)d_{l+1})$.

The factorized module (ST-GCN) is $\mathbf{X}_t^{(l+1)} = \tilde{\mathbf{A}} \mathbf{X}_t^{(l)} \mathbf{W}^{(l)}$, where $\tilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$, $\mathbf{X}_t^{(l)} \in \mathbb{R}^{N \times d_l}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l+1}}$ and $\mathbf{X}_t^{(l+1)} \in \mathbb{R}^{N \times d_{l+1}}$. The complexity of factorized module over T time steps is $T \cdot \mathcal{O}(N^2 d_l + N d_l d_{l+1})$. After that, the complexity of the temporal convolution operation (1-D CNN) in the T dimension is $\mathcal{O}(N d_{l+1} \Gamma(T - \Gamma + 1))$, where Γ is the kernel size of CNN (the stride is 1). So, the time complexity of factorized pathway is $\mathcal{O}(T \cdot (N^2 d_l + N d_l d_{l+1}) + N d_{l+1} \Gamma(T - \Gamma + 1))$.

The time complexity of STGPC block is dominated by the STGPCN, and we can obtain its approximate time complexity, which is $\mathcal{O}(T \cdot (m^2 N^2 d_l + mN d_l d_{l+1} + mN d_{l+1}) + N d_{l+1} \Gamma(T - \Gamma + 1))$. And the main calculation lies in L STGPC block, and the time complexity of the pooling layer and subsequent fully-connected layer can be omitted.

Therefore, the time complexity of our model is $L \cdot \mathcal{O}(T \cdot (m^2 N^2 d_l + mN d_l d_{l+1} + mN d_{l+1}) + N d_{l+1} \Gamma (T - \Gamma + 1))$, where L is the number of STGPC block (i.e., 3), N is the number of nodes (e.g., 25 for NTU-RGB+D dataset and 358 for PEMS03 dataset), m is window size (i.e., 3, 5 in the skeleton-based action recognition and 4 in the traffic prediction), d_l is the hidden dimension of l-th layer (e.g., 64), T is the total time steps (i.e., 300 in the skeleton-based action recognition and 288 in the traffic prediction), and Γ is the kernel size in the convolution operation (e.g., 3 or 2). To sum up, the total time complexity is acceptable.

5. Experiment

We first perform exhaustive ablation studies on the proposed model. Then, to perform head-to-head comparisons with SOTA methods of spatial-temporal graph modelling, we conduct experiments on two real-world applications, the skeleton-based action recognition and the traffic prediction.

5.1. Datasets, evaluation metrics and baseline models

For the skeleton-based action recognition task, we use two widely-used large-scale indoor-captured action recognition dataset, including NTU-RGB+D60 [33] and NTU-RGB+D120 [34] and a new large-scale UAV-based (unmanned aerial vehicle) dataset, UAV-Human [35]. We follow the recommended standards and report the top-1 accuracy on both three datasets. The SOTA baseline models includes Lie Group [36], ST-LSTM [37], ST-GCN [16], ASGCN [18], 2s-AGCN [17], AGC-LSTM [38], DGNN [19], Ms-AAGCN [29], Shift-GCN [20], ST-TR [39], and FGCN [40].

For the traffic prediction task, we use the traffic datasets released by STSGCN[5], including PEMS03, PEMS04, PEMS07, and PEMS08. We use several criteria to evaluate STGPCN in the field of traffic prediction, including the Mean Absolute Percentage Errors (MAPE), the Mean Absolute Errors (MAE), and the Root Mean Squared Errors (RMSE). All of them are widely used in the traffic predic-

tion tasks. The SOTA baseline models includes FC-LSTM [41], DCRNN [22], STGCN [3], ASTGCN(r) [4], GraphWaveNet [23], STSGCN [5].

5.1.1. Skeleton-based action recognition datasets

NTU-RGB+D60, NTU-RGB+D120. NTU-RGB+D60 [33] is a widely-used large-scale indoor-captured action recognition dataset, which contains total 56,578 skeleton samples from 60 different action classes. Each skeleton graph contains 25 body joints as nodes, and their 3D locations are initialized as features. For evaluating the classification accuracy, we follow the benchmark evaluating criteria proposed by dataset authors, which are the Cross-Subject (X-Sub) and Cross-View (X-View). X-Sub perspective splits all samples into training and testing groups according to different subjects, generating 40,091 and 16,487 training and testing examples, respectively. X-View divides all samples into 37,646 training samples and 18,932 testing samples by different camera views.

NTU-RGB+D120 [34] extends NTU-RGB+D60 by adding additional 57,367 skeleton sequences from 60 extra action classes, which contains total 113,945 samples over 120 classes from 106 different subjects and 32 camera setups. We follow the two criteria that the authors recommend, the first X-Sub is the same used for NTU-RGB+D60, where 63,026 samples from a selected group of half subjects are for training and the rest 50,919 samples for testing. And the second one replaces the X-View with a Cross-Setup (X-Set) setting, where 54,468 samples collected from half of the camera setups are for training and the rest 59,477 samples for testing.

UAV-Human [35] is a new large-scale UAV-based (unmanned aerial vehicle) dataset for human action, pose, and behavior understanding. It contains 23,031 samples (17 major body joints per sample) over 155 action classes from 119 distinct subjects, in which 89 subjects are used for training and other 30 subjects for testing. There are two evaluation benchmarks, Cross-Subject-v1 (CS1) and Cross-Subject-v2 (CS2) according to different subject partitions.

5.1.2. Traffic prediction datasets

PEMS03, PEMS04, PEMS07 and PEMS08. We also verify the performance of STGPCN on the above-mentioned traffic datasets released by STSGCN [5]. Those traffic data are collected by California Transportation Agencies (CalTrans) Performance Measurement System (PEMS) [42] and aggregated into 5-minutes windows, which means there are 288 points in the traffic flow for one day.

5.1.3. Baseline methods

As we stated before, lots of the spatial-temporal models focus on either skeleton-based action recognition or traffic prediction. Unfortunately, there are no existing works that can hit the two birds with one stone. To verify the superiority of STGPCN, we conduct exhausted comparisons with the following baseline methods of the two fields separately.

Skeleton-based action recognition models:

- Lie Group [36]: Hand-crafted body part-based feature extraction techniques.
- ST-LSTM [37]: Spatio-temporal long short-term memory, which extends temporal modelling to spatio-temporal domains, along with a new gating mechanism within long short-term memory (LSTM) to handle the noise and occlusion in 3D skeleton data for action recognition.
- ST-GCN [16]: Spatial-temporal graph convolutional networks, which are constructed by spatial-only graph convolution and temporal-only 1-d convolution.
- ASGCN [18]: Actional-structural graph convolutional networks, which generate human poses to augment the spatial graph convolution to capture spatial relationships, along with 1-D convolution to model temporal dynamics.

- 2s-AGCN [17]: Two-stream adaptive graph convolutional networks, which introduce graph adaptiveness with self-attention along with a freely learned graph residual mask. Besides, it also uses a two-stream architecture to fuse skeleton bone features.

- AGC-LSTM [38]: Attention enhanced graph convolutional LSTM networks, which combine graph convolution and LSTM.

- DGNN [19]: Directed graph neural networks, which model the skeleton graph as a directed acyclic graph and uses an alternative spatial aggregation scheme to update the joint and bone features simultaneously.

- Ms-AAGCN [29]: Multi-stream adaptive graph convolutional network, which explicitly models the joints, bones, and the corresponding motion information and fuses the skeleton data with the skeleton-guided cropped RGB data in a unified multi-stream framework.

- Shift-GCN [20]: Extend shift operation [21] to graph which can adjust the receptive field flexibly and outperform some state-of-the-art methods with much less computational complexity.

- ST-TR [39]: Spatial temporal transformor network, which uses spatial self-attention module to capture the interactions between different body parts in a certain frame, and temporal self-attention module to model correlations between different frames.

- FGCN [40] Feedback graph convolutional network, which proposes a multistage sampling strategy to select key frames from the input skeleton sequences.

Traffic prediction models:

- FC-LSTM [41]: Long short-term memory network, which is a recurrent neural network with fully connected LSTM hidden units.

- DCRNN [22]: Diffusion convolution recurrent neural network, which integrates graph convolution into an encoder-decoder gated recurrent unit.
- STGCN [3]: Spatio-temporal graph convolutional networks, which integrates graph convolution into a 1D convolution unit.
- ASTGCN(r) [4]: Attention-based spatial-temporal graph convolutional networks, which introduce spatial and temporal attention mechanisms into the model. Only recent components of modelling periodicity are taken to keep fair comparison.
- GraphWaveNet [23]: A framework that combines adaptive adjacency matrix into graph convolution with 1D causal and dilated convolution.
- STSGCN [5]: Spatial-temporal synchronous graph convolutional networks, which utilize localized spatial-temporal subgraph module to model localized correlations independently.

5.2. Implementation details

All experiments are conducted on the PyTorch platform, under a computer environment with one Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz CPU and four NVIDIA Titans V GPU cards.

The NTU-RGB+D datasets have at most two people in each sample. If this number is less than 2, we pad the second body with 0. And each sample contains at most 300 frames. For samples whose frames are less than 300, we pad them with duplication just as ST-GCN [16]. The initial learning rate is set as 0.05 and is divided by 10 at epochs $\{30, 40\}$ in totally 50 epochs and $\{30, 50\}$ in totally 60 epochs for NTU-RGB+D60 and NTU-RGB+D120, respectively. The weight Decay is set to 0.0005. The batch size is 32. The hidden dimensions in the three STPGC blocks are 96, 192, and 384, respectively. The initial learning rate is set as 0.1 and is divided by 10 at epochs $\{30, 40\}$ in totally 50 epochs for UAV-Huam. The weight Decay is set to 0.0001. The batch size is 56.

For traffic datasets, PEMS03, PEMS04, PEMS07, and PEMS08, we split data into $6 : 2 : 2$ for training, validation, and testing respectively as STSGCN

[5]. We use 12 continuous-time steps historical observations (1 hour) to predict the next hour's 12 records. The proposed models for PEMS datasets only have one STGPCN pathway (window size $m = 4$) in which three graph convolution filters are all 64. And the factorized module is the gated-CNN. The batch size is 32, the learning rate is 0.001. We choose Huber loss as the loss function following the experiment settings of STSGCN [5].

5.3. Ablation study

We first conduct an ablation study on two relatively small datasets (NTU-RGB+D60 and PEMS08) to analyse the individual components and their configurations in the final architecture. Unless specifically stated, the performance is reported as classification accuracy on the cross-view setting of NTU-RGB+D60 using only the joint data and the MAE, MAPE(%) and RMSE of PEMS08 dataset.

5.3.1. Effectiveness of STGPC block

To validate the efficacy of the STGPC block in capturing complex spatial-temporal features, we build up the model incrementally with STGPC blocks. The ablation results elaborate the experimental motivation why we combine the unified STGPCN with the factorized ST-GCN. We only present the model performance with the best hyper-parameter configurations.

Skeleton-based action recognition. Table 2 and Table 4 illustrate the ablation study on NTU-RGB+D60. We see that a single factorized ST-GCN (i.e., 'ST-GCN Only' in row 2 of Table 2) achieved 86.6% accuracy with 1.4 million parameters. We simply duplicate the factorized way (i.e., 'Dual ST-GCN Pathways' in row 4 of Table 2) to learn from different feature spaces, leading limited performance gains. Similarly, one STGPCN with window size $m = 3$ (row 6) achieves 84.1% with only 0.7 million parameters and dual STGPCN with window size $3, 5$ (row 8) achieves 91.2%. Row 9 illustrates 7.1% performance gain of the combination of one ST-GCN and one STGPCN. It indicates that efficient long-range modelling of factorized mode is a supplement of STGPCN.

When plugging in another STGPCN pathway with $m = 5$, considerable gains are achieved, verifying STGPCN's extraordinary ability to capture complex localized spatial-temporal correlations than factorized way. Best performance is recorded when one factorized block is combined with two STGPCN pathways with different window sizes, especially by further adding a TCN (the last row in Table 2). Besides, the selection of window sizes m and the numbers of STGPCN depend on quantities of comparison experiments. The detailed experimental result can refer to relevant contents in Appendix.

Although STGPCN only (row 6) and Dual STGPCN pathways (row 8) perform worse than the counterpart in row 2 and row 4 (ST-GCN only and dual ST-GCN pathways), but adding TCN module increases performance which makes the unified model reach the performance of the factorized model. And the combination of one factorized pathway along with two unified pathways with different window sizes achieves the best performance. We also validate the effectiveness of TCN in Table 4. The results indicate that applying TCN after STGPC does increase the performance and verifies its necessity.

Traffic prediction. Table 3 describe the results of ablation study on PEMS08. The basic experimental process is analog to action recognition. The only differences reside in two aspects. First, we replace ST-GCN with gated-CNN to adapt to traffic data (followed by STGCN [3]). Second, only one STGPCN pathway with window size $m = 4$ is used to capture the spatial-temporal dependencies. We find that the setting of two STGPCN pathways with different window sizes requires more than doubled parameter numbers and drops the performance (row 5 of Table 3). With those two changes, we observed the best model configuration (row 7 of Table 3).

5.3.2. Effectiveness of learnable residual mask

We conduct detailed experiments on the necessity of $\mathbf{A}^{res}$ and how the different initialization policies of $\mathbf{A}^{res}$ influence the model performance. The experimental results are presented in Table 5 and Table 6. In Table 5, 'wo $\mathbf{A}^{res}$' means without $\mathbf{A}^{res}$, 'STGPCN+normal' means initialize $\mathbf{A}^{res}$ with standard

Model Configurations	Window Size (m)	Params	Acc (%)
ST-GCN Only	/	1.4M	86.6
ST-GCN Only + TCN	/	1.4M	94.0
Dual ST-GCN Pathways	/	2.8M	88.5
Dual ST-GCN Pathways + TCN	/	2.8M	94.4
STGPCN Only	3	0.7M	84.1
STGPCN Only + TCN	3	1.0M	93.2
Dual STGPCN Pathways	3, 5	1.8M	91.2
Dual STGPCN Pathways + TCN	3, 5	2.0M	93.8
ST-GCN + STGPCN	3	2.1M	94.5
ST-GCN + 2 STGPCN	3, 5	3.2M	94.9
ST-GCN + 2 STGPCN + TCN	3, 5	3.2M	95.0

Table 2: Model performance (NTU-RGB+D60) with various configurations for STGPC Block.

Model Configurations	Window Size (m)	Params	MAE	MAPE (%)	RMSE
STGCN Only	/	0.4M	21.06	13.51	31.88
Gated-CNN Only	/	0.4M	19.99	12.60	30.84
STGPCN Only	4	1.7M	18.57	11.96	28.61
Dual STGPCN Pathways	3, 5	4.5M	19.04	12.11	29.51
STGCN + STGPCN	4	1.8M	17.91	11.45	27.62
Gated-CNN + STGPCN	4	1.7M	16.67	10.49	26.16
STGCN + 2 STGPCN	3, 5	4.0M	18.21	11.67	28.39
Gated-CNN + 2 STGPCN	3, 5	4.5M	19.93	12.63	30.21

Table 3: Model performance (PEMS08) with various configurations for STGPC Block.

normal distribution, and 'STGPCN+uniform' means initialization with uniform distribution (interval -1e-6 to 1e-6). We can see that the model performance improves as the $\mathbf{A}^{res}$ are added in the vast majority of cases, suggesting that adding an adaptive matrix $\mathbf{A}^{res}$ is always a good strategy, and STGPCN performs best no matter what kind of graph product is chose. The idea behind this is

Model Configurations	Different Connectivity	Params	Acc (%)
STGPCN wo TCN	Cartesian Product	2.9M	94.86
STGPCN	Cartesian Product	3.2M	95.00
STGPCN wo TCN	Kronecker Product	2.9M	94.87
STGPCN	Kronecker Product	3.2M	94.90
STGPCN wo TCN	Strong Product	2.9M	94.89
STGPCN	Strong Product	3.2M	94.90
STGPCN wo TCN	Lexicographical Product	2.9M	94.76
STGPCN	Lexicographical Product	3.2M	94.95

Table 4: Model performance (NTU-RGB+D60) with various configurations for TCN.

that the predefined graph topology may not be perfect, adding $\mathbf{A}^{res}$ helps the model to learn an adaptive adjacency matrix based on the predefined one in a data-driven manner. Lots of papers (including 2s-AGCN[17], MS-AAGCN[29]) use the similar strategy. On the other hand, adding $\mathbf{A}^{res}$ can improve the robustness to noises.

5.3.3. Different temporal graphs

We compare three different temporal graphs in Fig 2a, un-directed temporal graph, directed temporal graph, and directed and recurrent temporal graph as shown in Table 7. The un-directed temporal graph is the 1-D temporal convolution commonly used in previous studies [16]. We see that the two directed temporal graphs perform relatively better than the un-directed one. Among the two directed settings, the directed and recurrent temporal graph achieves the highest classification accuracy.

5.3.4. Exploring graph product configurations

We also compare the four graph products under two kinds of best model configurations: **STGCN + 2 STGPCN + TCN** w.r.t window sizes 3 and 5 for action recognition and **Gated-CNN + STGPCN** with window size 4 for traffic prediction. We observe that different connectivities of graph product (STGPCN

Model Configurations	Different Connectivity	Params	Acc (%)
STGPCN wo $\mathbf{A}^{res}$	Cartesian Product	2.8M	94.04
STGPCN+normal	Cartesian Product	3.2M	94.32
STGPCN+uniform	Cartesian Product	3.2M	95.00
STGPCN wo $\mathbf{A}^{res}$	Kronecker Product	2.8M	94.23
STGPCN+normal	Kronecker Product	3.2M	94.31
STGPCN+uniform	Kronecker Product	3.2M	94.90
STGPCN wo $\mathbf{A}^{res}$	Strong Product	2.8M	94.37
STGPCN+normal	Strong Product	3.2M	94.40
STGPCN+uniform	Strong Product	3.2M	94.90
STGPCN wo $\mathbf{A}^{res}$	Lexicographical Product	2.8M	94.09
STGPCN+normal	Lexicographical Product	3.2M	94.37
STGPCN+uniform	Lexicographical Product	3.2M	94.95

Table 5: Model performance (NTU-RGB+D60) with various configurations for residual mask $\mathbf{A}^{res}$.

Model Configurations	Different Connectivity	Params	MAE	MAPE(%)	RMSE
STGPCN wo $\mathbf{A}^{res}$	Cartesian Product	1.3M	16.73	10.80	25.99
STGPCN	Cartesian Product	1.7M	16.98	10.77	26.42
STGPCN wo $\mathbf{A}^{res}$	Kronecker Product	1.3M	16.90	10.75	26.30
STGPCN	Kronecker Product	1.7M	16.41	10.43	25.60
STGPCN wo $\mathbf{A}^{res}$	Strong Product	1.3M	16.91	10.83	26.27
STGPCN	Strong Product	1.7M	16.61	10.45	25.94
STGPCN wo $\mathbf{A}^{res}$	Lexicographical Product	1.3M	19.14	12.32	28.99
STGPCN	Lexicographical Product	1.7M	16.67	10.49	26.16

Table 6: Model performance (PEMS08) with various configurations for residual mask $\mathbf{A}^{res}$.

with different connectivity in Table 5 and Table 6) consistently outperform the baseline (STGCN Only in Table 2 and Gated-CNN Only in Table 3), which confirming the stability of STGPCN as a robust feature extractor.

Different temporal Graph	Acc(%)
Un-directed Temporal Graph	94.66
Directed Temporal Graph	94.79
Directed and Recurrent Temporal Graph	95.03

Table 7: Model performance with diverse temporal graph on NTU-RGB+D60 X-view.

5.4. Comparisons with the state-of-the-art methods

5.4.1. Action recognition

We process detailed performance comparisons with aforementioned baseline methods on NTU-RGB+D60 [33], NTU-RGB+D120 [34], and UAV-Human [35] dataset as shown in Table 8, Table 9, and Table 10, respectively. We observe that the proposed model surpasses other baselines in most of the metrics except for the X-view setting of the NTU-RGB+D60 dataset. Notably, our method applies a multi-pathway design to learn both long-range spatial-temporal dependencies and complex regional spatial-temporal correlations from skeleton sequences, and the results verify its effectiveness.

5.4.2. Traffic prediction

We compare the proposed model with the baseline methods on PEMS datasets. The results of the comparison are as shown in Table 11. We see that STGPCN outperforms the baseline models on all four datasets, verifying our model's superiority on spatial-temporal modelling in the field of traffic prediction (especially Strong product and Kronecker product). We found that the best performance on different datasets varies depending on which type of graph product is used, probably because different data adapt to varying connectivity. Although the best results are inconsistent, STGPCN with Strong product or Kronecker product outperforms the baseline models on all four datasets.

Methods	X-sub	X-view
Lie Group	50.1	82.8
ST-LSTM	69.2	77.7
ST-GCN	81.5	88.3
ASGCN	86.8	94.2
2s-AGCN	88.5	92.1
AGC-LSTM	89.2	95.0
DGNN	89.9	96.1
Ms-AAGCN	90.0	96.2
STGPCN (Joint Only)	89.4	95.0
STGPCN (Bone Only)	89.8	95.1
STGPCN (Ensembled)	91.2	96.0

Table 8: Comparisons of classification top-1 accuracy (%) with state-of-the-art methods on NTU-RGB+D60.

Methods	X-sub	X-set
ST-LSTM	55.7	57.9
2s-AGCN	82.9	84.9
ST-TR	85.1	87.1
Shift-GCN	85.3	86.6
FGCN	85.4	87.4
STGPCN (Joint Only)	82.2	84.4
STGPCN (Bone Only)	85.2	86.8
STGPCN (ensembled)	85.9	88.1

Table 9: Comparisons of classification top-1 accuracy (%) with state-of-the-art methods on NTU-RGB+D120.

6. Visualization and discussion

There are two kinds of graphs in our model: the factorized graph and the product graph. Figure 4 shows an example of the learned adjacency matrices of the skeleton graph with different construction in different epoch. Figure

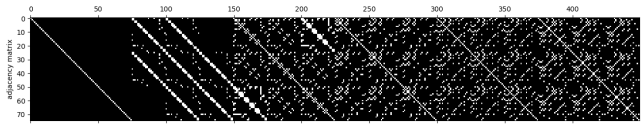
Methods	CS1	CS2
ST-GCN	38.9	63.9
2s-AGCN	40.0	66.2
STGPCN (Cartesian)	41.1	67.5
STGPCN (Kronecker)	41.1	68.0
STGPCN (Strong)	41.2	68.0
STGPCN (Lexicographical)	41.5	67.8

Table 10: Comparisons of classification accuracy (%) with state-of-the-art methods on UAV-Human dataset.

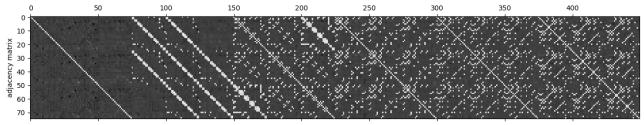
Datasets	Metrics	FC-LSTM	DCRNN	STGCN	ASTGCN (r)	Graph WaveNet	STSGCN	STGPCN (Cartesian)	STGPCN (Strong)	STGPCN (Kronecker)	STGPCN (Lexicographical)
PEMS 03	MAE	21.33	18.18	17.49	17.69	19.85	17.48	17.24	17.24	**17.11**	17.88
	MAPE(%)	23.33	18.91	17.15	19.40	19.31	16.78	**16.37**	17.03	16.48	17.45
	RMSE	35.11	30.31	30.12	29.66	32.94	29.21	**28.91**	29.41	28.99	30.74
PEMS 04	MAE	27.14	24.70	22.70	22.93	25.45	21.19	21.55	21.02	**20.96**	21.11
	MAPE(%)	18.20	17.12	14.59	16.56	17.29	13.90	14.29	**13.73**	13.78	13.86
	RMSE	41.59	38.12	35.55	35.22	39.70	33.65	34.00	33.41	**33.35**	33.43
PEMS 07	MAE	29.98	25.30	25.38	28.05	26.85	24.26	24.24	**23.73**	24.02	23.92
	MAPE(%)	13.20	11.66	11.08	13.92	12.12	10.21	10.25	**9.92**	10.08	10.04
	RMSE	45.94	38.58	38.78	42.57	42.78	39.03	38.69	**38.37**	38.77	38.84
PEMS 08	MAE	22.20	17.86	18.02	18.61	19.13	17.13	17.02	16.61	**16.41**	16.67
	MAPE(%)	14.20	11.45	11.40	13.08	12.68	10.96	10.92	10.45	**10.43**	10.49
	RMSE	34.06	27.83	27.83	28.16	31.05	26.80	26.38	25.94	**25.60**	26.16

Table 11: Comparisons of prediction error with baseline methods on PEMS datasets.

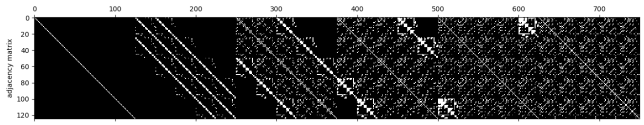
4a and Figure 4c are the visualization of the original adjacency matrices (i.e., $\tilde{\mathbf{A}}_{(m)}$ in Eq. 6) of the graph product convolution in STGPCN with window sizes of 3 and 5 respectively. Figure 4e show the factorized adjacency matrix of spatial-only convolution in ST-GCN which has 7 scales (i.e., K in Eq. 6). Figure 4b, Figure 4d and Figure 4f are the adjacency matrices learned adaptively (e.g., $\tilde{\mathbf{A}}_{(m)} + \mathbf{A}^{res}$ in Eq. 7) in different epochs. The greyscale of each element in the matrix represents the strength of the connection. We observe that the topology of the learned graph is updated based on the human-body-based graph but has many changes, it is the $\tilde{\mathbf{A}}_{(m)} + \mathbf{A}^{res}$. It corroborates that the human-body-based graph is not the optimal choice for the action recognition task. On the other



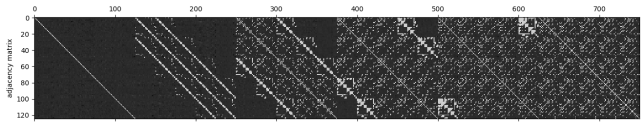
(a) $\tilde{\mathbf{A}}^k_{(m)},\ m = 3, k = 0, 1, 2, 3, 4, 5$



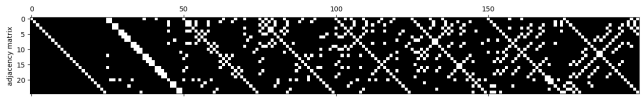
(b) $\tilde{\mathbf{A}}^k_{(m)} + \mathbf{A}^{res},\ m = 3, k = 0, 1, 2, 3, 4, 5$



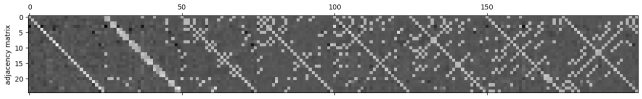
(c) $\tilde{\mathbf{A}}^k_{(m)},\ m = 5, k = 0, 1, 2, 3, 4, 5$



(d) $\tilde{\mathbf{A}}^k_{(m)} + \mathbf{A}^{res},\ m = 5, k = 0, 1, 2, 3, 4, 5$



(e) $\tilde{\mathbf{A}}^k,\ k = 0, 1, 2, 3, 4, 5, 6, 7$



(f) $\tilde{\mathbf{A}}^k + \mathbf{A}^{res},\ k = 0, 1, 2, 3, 4, 5, 6, 7$

Figure 4: Learned adjacency matrices with different window size m and scale K (Eq. 6). The grey value indicates the strength of the connection. The brighter a pixel, the corresponding value is more close to 1. (a) and (c) are initial graph product adjacency matrices (window size = 3 and 5, scale = 5) respectively, and (e) represents factorized adjacency matrix (scale = 7). (b), (d), (f) are corresponding updated matrices. Comparing these results, we can see that (b), (d), and (f) have many greyscale elements but still preserve the basic mode of the human-body-based graph, which shows that the topology of learned graphs has been updated during the training process.

hand, adding $\mathbf{A}^{res}$ can improve the robustness to noises.

7. Conclusion

In this paper, we introduce a novel spatial-temporal graph product convolutional framework that unifies various existing approaches to spatial-temporal graph learning. Our framework leverages graph product and different temporal graphs to capture multi-scale spatial-temporal features that were previously neglected by spatial-temporal factorized models. By modeling the spatial-temporal dependencies from graph sequences generated by different graph product configurations, experimental results demonstrate that our framework achieves better performance in capturing complex spatial-temporal patterns in two real-world applications.

References

[1] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: International Conference on Learning Representations, 2018.
URL https://openreview.net/forum?id=SJiHXGWAZ

[2] Z. Liu, H. Zhang, Z. Chen, Z. Wang, W. Ouyang, Disentangling and unifying graph convolutions for skeleton-based action recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 143–152.

[3] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in: The International Joint Conference on Artificial Intelligence, IJCAI, 2018, pp. 3634–3640.

[4] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: The AAAI Conference on Artificial Intelligence, AAAI, Vol. 33, 2019, pp. 922–929.

[5] C. Song, Y. Lin, S. Guo, H. Wan, Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network

data forecasting, in: The AAAI Conference on Artificial Intelligence, AAAI, Vol. 34, 2020, pp. 914–921.

[6] M. Li, Z. Zhu, Spatial-temporal fusion graph neural networks for traffic flow forecasting, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 35, 2021, pp. 4189–4196.

[7] Y. Xie, Y. Xiong, Y. Zhu, Istd-gcn: Iterative spatial-temporal diffusion graph convolutional network for traffic speed forecasting, CoRR abs/2008.03970 (2020).

[8] W. Imrich, S. Klavzar, Product graphs: structure and recognition, Wiley New York, 2000.

[9] J. Bruna, W. Zaremba, A. Szlam, Y. Lecun, Spectral networks and locally connected networks on graphs, in: The International Conference on Learning Representations, ICLR, April, 2014.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: The International Conference on Learning Representations, ICLR, 2018.

[11] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.

[12] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, in: The IEEE Signal Processing Magazine, Vol. 30, 2013, pp. 83–98.

[13] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, NIPS, 2016, p. 3844–3852.

[14] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: The International Conference on Learning Representations, ICLR, 2017.

[15] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, NIPS, 2017, pp. 1024–1034.

[16] S. Yan, Y. Xiong, D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in: The AAAI Conference on Artificial Intelligence, AAAI, 2018, pp. 7444–7452.

[17] L. Shi, Y. Zhang, J. Cheng, H. Lu, Two-stream adaptive graph convolutional networks for skeleton-based action recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 12026–12035.

[18] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, Q. Tian, Actional-structural graph convolutional networks for skeleton-based action recognition, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 3595–3603.

[19] L. Shi, Y. Zhang, J. Cheng, H. Lu, Skeleton-based action recognition with directed graph neural networks, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 7912–7921.

[20] K. Cheng, Y. Zhang, X. He, W. Chen, J. Cheng, H. Lu, Skeleton-based action recognition with shift graph convolutional network, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 183–192.

[21] B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, K. Keutzer, Shift: A zero flop, zero parameter alternative to spatial convolutions, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2018, pp. 9127–9135.

[22] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, in: The International Conference on Learning Representations, ICLR, 2018.

[23] Z. Wu, S. Pan, G. Long, J. Jiang, C. Zhang, Graph wavenet for deep spatial-temporal graph modeling, The International Joint Conference on Artificial Intelligence, IJCAI (2019) 1907–1913.

[24] L. Bai, L. Yao, S. Kanhere, X. Wang, Q. Sheng, et al., Stg2seq: Spatial-temporal graph to sequence model for multi-step passenger demand forecasting, in: The International Joint Conference on Artificial Intelligence, IJCAI, 2019, pp. 1981–1987.

[25] Z. Cao, T. Simon, S.-E. Wei, Y. Sheikh, Realtime multi-person 2d pose estimation using part affinity fields, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2017, pp. 7291–7299.

[26] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, Y. Liu, Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 33, 2019, pp. 3656–3663.

[27] Y. Wang, H. Yin, H. Chen, T. Wo, J. Xu, K. Zheng, Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1227–1235.

[28] J. Ye, J. Zhao, K. Ye, C. Xu, How to build a graph-based deep learning architecture in traffic domain: A survey, IEEE Transactions on Intelligent Transportation Systems (2020).

[29] L. Shi, Y. Zhang, J. Cheng, H. Lu, Skeleton-based action recognition with multi-stream adaptive graph convolutional networks, in: The IEEE Transactions on Image Processing, Vol. 29, IEEE, 2020, pp. 9532–9545.

[30] Z. Diao, X. Wang, D. Zhang, Y. Liu, K. Xie, S. He, Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting, in: The AAAI Conference on Artificial Intelligence, AAAI, Vol. 33, 2019, pp. 890–897.

[31] Y. Yan, J. Xu, B. Ni, W. Zhang, X. Yang, Skeleton-aided articulated motion generation, in: The ACM International Conference on Multimedia, 2017, pp. 199–207.

[32] R. Hammack, W. Imrich, S. Klavžar, Handbook of product graphs, CRC press, 2011.

[33] A. Shahroudy, J. Liu, T.-T. Ng, G. Wang, Ntu rgb+ d: A large scale dataset for 3d human activity analysis, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 1010–1019.

[34] J. Liu, A. Shahroudy, M. L. Perez, G. Wang, L.-Y. Duan, A. K. Chichung, Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding, in: The IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 42, 2019, pp. 2684–2701.

[35] T. Li, J. Liu, W. Zhang, Y. Ni, W. Wang, Z. Li, Uav-human: A large benchmark for human behavior understanding with unmanned aerial vehicles, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16266–16275.

[36] R. Vemulapalli, F. Arrate, R. Chellappa, Human action recognition by representing 3d skeletons as points in a lie group, in: The IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2014, pp. 588–595.

[37] J. Liu, A. Shahroudy, D. Xu, G. Wang, Spatio-temporal lstm with trust gates for 3d human action recognition, in: The European Conference on Computer Vision, ECCV, 2016, pp. 816–833.

[38] C. Si, W. Chen, W. Wang, L. Wang, T. Tan, An attention enhanced graph convolutional lstm network for skeleton-based action recognition, in: The

IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 1227–1236.

[39] C. Plizzari, M. Cannici, M. Matteucci, Spatial temporal transformer network for skeleton-based action recognition, in: International Conference on Pattern Recognition, Springer, 2021, pp. 694–701.

[40] H. Yang, D. Yan, L. Zhang, Y. Sun, D. Li, S. J. Maybank, Feedback graph convolutional network for skeleton-based action recognition, IEEE Transactions on Image Processing 31 (2021) 164–175.

[41] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, NIPS, Vol. 27, 2014, pp. 3104–3112.

[42] C. Chen, K. Petty, A. Skabardonis, P. Varaiya, Z. Jia, Freeway performance measurement system: mining loop detector data, in: Transportation Research Record, Vol. 1748, 2001, pp. 96–102.

Responses to the reviewer's comments

We thank reviewers for your invaluable and insightful comments. We have addressed all your concerns and have revised our manuscript accordingly.

Reviewer #3:

Comment:
First of all, it is a comprehensive and general study. I sincerely congratulate the authors. However, I have some suggestions for the article.

R: We would like to thank you for your careful reading and thoughtful comments of our manuscript submitted before.

Q1. The authors used different datasets, "NTURGB+D60, NTU-RGB+D120, UAV-Human, PEMS03, PEMS04, PEMS07, and PEMS08". They only mentioned the "NTU-RGB+D" dataset (on page 27-55, line) among these. How many samples are there in total, and how many samples are there in other datasets? Did they use all of the datasets, or did they use randomly selected images from these datasets while using them?

R1: Sorry for your confused. We do not include detailed information about datasets and experimental setting in the manuscript because we follow the common setting the same as other works.

For skeleton-based action recognition, we use NTU-RGB+D60, NTU-RGB+D120 and UAV-Human dataset.
NTU-RGB+D60 [1] is a widely-used large-scale indoor-captured action recognition dataset, which contains total 56,578 skeleton samples from 60 different action classes. Each skeleton graph contains 25 body joints as nodes, and their 3D locations are initialized as features. For evaluating the classification accuracy, we follow the benchmark evaluating criteria proposed by dataset authors, which are the Cross-Subject (X-Sub) and Cross-View (X-View). X-Sub perspective splits all samples into training and testing groups according to different subjects, generating 40,091 and 16,487 training and testing examples, respectively. X-View divides all samples into 37,646 training samples and 18,932 testing samples by different camera views.
NTU-RGB+D120 [2] extends NTU-RGB+D60 by adding additional 57,367 skeleton sequences from 60 extra action classes, which contains total 113,945 samples over 120 classes from 106 different subjects and 32 camera setups. We follow the two criteria that the authors recommend, the first X-Sub is the same used for NTU-RGB+D60, where 63,026 samples from a selected group of half subjects are for

training and the rest 50,919 samples for testing. And the second one replaces the X-View with a Cross-Setup (X-Set) setting, where 54,468 samples collected from half of the camera setups are for training and the rest 59,477 samples for testing.
UAV- Human [3] is a new large- scale UAV-based (unmanned aerial vehicle) dataset for human action, pose, and behavior understanding. It contains 23,031 samples (17 major body joints per sample) over 155 action classes from 119 distinct subjects, in which 89 subjects are used for training and other 30 subjects for testing. There are two evaluation benchmarks, Cross- Subject- v1 (CS1) and Cross- Subject- v2 (CS2) according to different subject partitions.

For Traffic prediction, we use PEMS03, PEMS04, PEMS07 and PEMS08 datasets. We also verify the performance of STGPCN on the above-mentioned traffic datasets released by STSGCN *Error! Reference source not found.*. ***Those traffic data are collected by California Transportation Agencies (CalTrans) Performance Measurement System (PEMS) and aggregated into 5-minutes windows, which means there are 288 points in the traffic flow for one day. In the traffic dataset we used in our manuscript, the numbers of node are listed below,***

Datasets	#Nodes	#Edges	#Timesteps	Time Range
PEMS03	358	547	26208	09/01/2018-11/30/2018
PEMS04	307	340	16992	01/01/2018-02/28/2018
PEMS07	883	866	28224	05/01/2017-08/31-2017
PEMS08	170	295	17856	07/01/2016-08/31/2016

Thank you for proposing this concern, it really helped us for clearer statements of our manuscript.

Q2. The authors also say that they conducted experiments on the PyTorch platform, using Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz CPU and four NVIDIA Titans V GPU cards, and presented their accuracy (%) values in tables (Table 2, 3, 4, ...). I suggest that they also show the MSE error graph for the highest performance results obtained in a visually comparative manner.

R2: For skeleton-based action recognition dataset, the benchmark evaluating criteria is classification accuracy in two perspectives, the results in table 2,4,5,8,9 is capable of verifying the model performance of our proposed model.
For traffic prediction task, the benchmark evaluating criteria is RMSE, MAE, MAPE. We follow the commonly-used evaluation criterion which is using one hour historical data to predict the next hour's data, which means using the past 12 continuous time steps to predict the future 12 continuous time steps. Table 10 is the detailed results of our proposed model and other SOTA model on four PEMS dataset.

Q3. Finally, in section 6 (page 35-29, line) of the Discussion, the authors did not adequately discuss the findings they obtained. They briefly mentioned through Figure 4. The advantages compared to the studies in the literature have not been fully explained. This section may need further improvement. Also, if this section is being interpreted through Figure 4, why is this figure located in section 7, "Conclusion"? Would it not be more appropriate to have it in this section?

R3: Thanks for your carefully review. In Figure 4, we just want to show that our model can adaptively learn the adjacency matrix of the skeleton graph. In addition, we are so sorry that the placement problem in Figure 4 is simply due to the paper layout.

Reviewer #4:

Summary:

This study developed general framework to model dynamic spatial-temporal graph data from the view of graph product. The manuscript brings to readers following highlights:

- The systematical way of constructing the spatial-temporal adjacent graphs;
- Extensive experiments on multiple large-scale real-world datasets, NTURGB+D60, NTU-RGB+D120, UAV-Human, PEMS03, PEMS04, PEMS07, and PEMS08;
- The model can generalize to most of the scenarios with a performance improvement in a significant margin compared to the state-of-the-art methods

The theoretical foundation and computer simulations presented in this manuscript demonstrate high credibility and feasibility. However, the study appears to lack sufficient depth, and the framework to model dynamic spatial-temporal graph data from the view of graph product is not a novel concept. Therefore, the originality of this work is limited, and certain sections of the manuscript lack clarity. As an expert in this field, I recommend a significant revision of this manuscript. I have several suggestions for the authors to carefully edit and expand their research:

R：Thanks for your review work.

Q1. The "Introduction" section of the manuscript needs major revision due to several issues. Firstly, the content lacks focus and wanders off-topic, which requires a significant reorganization to ensure its relevance. Secondly, the number of references cited in the manuscript is insufficient to provide adequate support for the study. Therefore, the authors need to include additional references that provide relevant background information. Thirdly, the reference list needs to be rearranged to conform to the journal's guidelines. Finally, the language quality of the manuscript is poor, with several sections being difficult to comprehend or expressed unclearly. Thus, the manuscript requires significant improvement to enhance its readability.

R1: Thank you for your feedback on the "Introduction" section of our manuscript. We appreciate your insights and agree that the section requires major revisions. We have already reorganized the content to ensure its relevance and providing additional references to support our study adequately. We have also rearranged the reference list to conform to the journal's guidelines. Additionally, we have made lots of efforts on improving the language quality of the manuscript to enhance its readability. We value your comments and hope that our revised manuscript meets the required standards. Thank you again for your feedback.

Q2. The "Conclusion" section of the manuscript requires revision to address several issues. Firstly, the items in the study are not presented in a logical order, which makes the conclusion confusing and difficult to comprehend. The authors need to restructure the presentation of the results to ensure clarity and coherence. Secondly, the study results are limited and do not provide significant new insights compared to similar research in the field. The authors should highlight the significance of their findings more clearly or consider including additional analyses to provide greater insights. Finally, the authors need to better demonstrate the strengths of their study, as the current presentation in the manuscript is confusing. By addressing these issues, the authors can improve the clarity and impact of the "Conclusion" section.

R2: Thank you for your review on the "Conclusion" section of our manuscript. We appreciate your comments and have made revisions accordingly to improve the clarity of our presentation. Regarding the issue of the logical order of the study

items, we have restructured the presentation of our results to ensure a more logical flow that is easier to comprehend. We have also revised our conclusion parts to highlight the significance of our findings.

Q3. It is recommended that the authors provide a more detailed explanation of the results presented in Figure 2. This will help readers better understand the findings and the relevance of the figures to the overall manuscript. If providing further details is not possible, the authors may consider omitting the figures altogether, as they do not seem to be directly relevant to the topic of the manuscript.

R3: Thanks for your concern on Figure 2. The caption in Figure 2 has already provided a detailed explanation about how to generate different graph product and the difference between four mentioned graph product. It is a way of connection the spatial graph and temporal graph to generate a spatial-temporal large graph so that the proposed model can capture the spatial-temporal dependencies simultaneously. And we have added more figures and corresponding captions to make the figure more understandable.

Q4. It is recommended that the authors compare their manuscript with other disability assessment models used in current studies. This is important because it will provide a basis for evaluating the novelty and significance of the proposed model. The authors should carefully check the literature and identify relevant models to compare with their own. By doing so, the manuscript will have better quality and relevance to the field.

R4: Thanks for your carefully review. We have conducted comparisons of classification accuracy in table 9 in the manuscript with ST-GCN and 2s-AGCN model. The robustness against the noisy input data is wildly studied in the action recognition field and there are lots of excellent works. We demonstrate that the proposed model not only surpass those two models in skeleton-based action recognition tasks, but have better model performance and model robustness against the noisy input.

Q5. In this study, the authors need to show us what the input value is? What is the output value? Stop condition and activation function?.... How is the signal-to-noise ratio and noise separation method performed in this study? This article should be

restructured to improve the content and help readers better understand the scope of the study, the work carried out, and the quality of the research.

R5:

For action recognition task, the input is adjacency matrix A and node features X, the dimensions of A is $N \times N$, the dimensions of X is $T \times N \times C$, where T is number of frames, N is the number of nodes, and C is the number of channels (equals 3, three-dimensional coordinates of each node), the output is the predicted action class.

For traffic prediction task, the input is adjacency matrix A and node features X, the dimensions of A is $N \times N$, the dimensions of X is $T \times N \times C$, where T is time-steps, N is the number of nodes, and C is the number of channels (equals 1, the number of vehicles passed the senser during the past 5 minutes), the output is the predicted number of vehicles passed each sensor, the dimension is $N \times 1$.

The activation function is set as ReLu.

The training epochs is set as 50, 60, 50 and 200 for NTU-RGB+D 60, 120, UAV-Human and PEMS dataset.

The point of our paper is that we present a unified spatial-temporal graph convolutional framework, which unifies many existing works of spatial-temporal graph learning. And we haven't performed any special signal-to-noise ratio and noise separation method in this work.

Reference

[1]. Shahroudy, A., Liu, J., Ng, T. T., & Wang, G. (2016). Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1010-1019).

[2]. Liu, J., Shahroudy, A., Perez, M., Wang, G., Duan, L. Y., & Kot, A. C. (2019). Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. IEEE transactions on pattern analysis and machine intelligence, 42(10), 2684-2701.

[3]. Li, T., Liu, J., Zhang, W., Ni, Y., Wang, W., & Li, Z. (2021). Uav-human: A large benchmark for human behavior understanding with unmanned aerial vehicles. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 16266-16275).

CRediT Author Statement

Zhuo Tan: Implementation, Visualization,Writing- Original draft preparation.

Yifan Zhu: Implementation, Writing- Original draft preparation,Writing- Reviewing and Editing.

Bin Liu:Conceptualization, Methodology, Writing- Original draft preparation,Writing-Reviewing and Editing, Supervision.

Highlights

- Introduces a new spatial-temporal graph product convolutional framework.
- Unifies various existing approaches to spatial-temporal graph learning.
- Captures multi-scale spatial-temporal features that were previously neglected.
- Achieves better performance in capturing complex spatial-temporal patterns in two real-world applications.