Skeleton-Based Action Recognition With Multi-Stream Adaptive Graph Convolutional Networks

Lei Shi, Yifan Zhang, *Member, IEEE*, Jian Cheng, *Member, IEEE*, and Hanqing Lu, *Senior Member, IEEE*

***Abstract*— Graph convolutional networks (GCNs), which generalize CNNs to more generic non-Euclidean structures, have achieved remarkable performance for skeleton-based action recognition. However, there still exist several issues in the previous GCN-based models. First, the topology of the graph is set heuristically and fixed over all the model layers and input data. This may not be suitable for the hierarchy of the GCN model and the diversity of the data in action recognition tasks. Second, the second-order information of the skeleton data, i.e., the length and orientation of the bones, is rarely investigated, which is naturally more informative and discriminative for the human action recognition. In this work, we propose a novel multi-stream attention-enhanced adaptive graph convolutional neural network (MS-AAGCN) for skeleton-based action recognition. The graph topology in our model can be either uniformly or individually learned based on the input data in an end-to-end manner. This data-driven approach increases the flexibility of the model for graph construction and brings more generality to adapt to various data samples. Besides, the proposed adaptive graph convolutional layer is further enhanced by a spatial-temporal-channel attention module, which helps the model pay more attention to important joints, frames and features. Moreover, the information of both the joints and bones, together with their motion information, are simultaneously modeled in a multi-stream framework, which shows notable improvement for the recognition accuracy. Extensive experiments on the two large-scale datasets, NTU-RGBD and Kinetics-Skeleton, demonstrate that the performance of our model exceeds the state-of-the-art with a significant margin.**

***Index Terms*— Skeleton-based action recognition, graph convolutional network, adaptive graph, multi-stream network.**

I. Introduction

ACTION recognition has been widely researched in decades since it plays a significant role in many real-world applications [1]–[5]. Recently, compared with conventional methods that use RGB videos for recognition, the skeleton-based method draws increasingly more attention due to its strong adaptability to the dynamic circumstance and complicated background [6]–[9]. In latest works, the GCN-based methods have shown impressive performance for skeleton-based action recognition, where the most famous one is the spatiotemporal graph convolutional network (ST-GCN) [10]. They proposed to construct a spatiotemporal graph to encode the skeleton sequences and stacked a set of spatiotemporal graph convolutions to extract features and make predictions.

However, there are three limitations for ST-GCN: (1) The skeleton graph used in ST-GCN is heuristically predefined based on the physical structure of the human body. Thus it is not guaranteed to be optimal for the action recognition task. For example, the relationship between the two hands is important for recognizing classes such as "clapping" and "reading." However, it is difficult for ST-GCN to capture the dependencies between the two hands since they are located far away from each other in the predefined human-body-based graphs. (2) Since the attention mechanism has been demonstrated the effectiveness and necessity in many computer vision tasks [11]–[13], it is necessary to investigate it for the skeleton-based action recognition. From the spatial perspective, a certain kind of action is usually associated with and characterized by a key subset of the joints. From the temporal perspective, an action flow may contains multiple stages where different sub-stages or frames have different degrees of importance for the final recognition. From the feature perspective, multiple channels of a convolutional feature map contain multiple levels of semantics. Each channel plays different roles for different actions and data samples. However, these attentions, which are essential for action recognition, are ignored and not exploited in ST-GCN. (3) The feature vector attached to each vertex in ST-GCN only contains the 2D or 3D coordinates of joints. However, from the human perspective, the lengths and directions of bones between two joints, are naturally more informative and discriminative for action recognition. Besides, the optical flow field have been demonstrated a useful information to model the temporal evolutions for RGB-based action recognition [2], [5]. The similar motion information should also be important for skeleton-based action recognition. However, neither the bones nor motions are investigated in ST-GCN.

To address these problems, we propose a novel multi-stream attention-enhanced adaptive graph convolutional network (MS-AAGCN) in this work. Specifically, there are mainly three improvements in MS-AAGCN: (1) A novel adaptive graph convolutional layer is introduced and two kinds of

Manuscript received August 21, 2019; revised February 22, 2020, June 4, 2020, and August 14, 2020; accepted September 19, 2020. Date of publication October 9, 2020; date of current version October 20, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61876182 and Grant 61872364 and in part by the Jiangsu Leading Technology Basic Research Project under Grant BK20192004. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Giacomo Boracchi. *(Corresponding author: Yifan Zhang.)*

The authors are with the NLPR and AIRIA, Institute of Automation, Chinese Academy of Science, Beijing 100049, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: lei.shi@nlpr.ia.ac.cn; yfzhang@nlpr.ia.ac.cn; jcheng@nlpr.ia.ac.cn; luhq@nlpr.ia.ac.cn).

Digital Object Identifier 10.1109/TIP.2020.3028207

adaptive graphs are parameterized for graph convolution, namely, the global graph and the individual graph. The global graph determines the basic graph topology for the action recognition task, or in other words provides a global regularization for graph learning. The individual graph provides more flexibility of the model and brings more generality to adapt to various data samples. The two kinds of graphs are fused using a gating mechanism, which can adaptively adjust their importance in each of the model layers. Note that both of the graphs are optimized individually across different layers, thus it can better fit the hierarchical structure of the neural networks. (2) A spatial-temporal-channel (STC) attention module is introduced to adaptively re-calibrate the activations of the joints, frames and channels for different data samples. It can provide different attentions along both the spatial, temporal and feature perspective based on the current situations to better understand the human actions. The module is plugged in every graph convolutional layers, with small amount of parameters yet encouraging performance improvement. (3) We refer the 2D/3D coordinates of the skeleton data as the the first-order information and propose to exploit the second-order information, i.e., the bones between two joints. In detail, the bone information is reformulated as a vector pointing from its source joint to its target joint. Besides, inspired by the optical flow field, we propose to extract the coordinate differences of the joints and the bones between two consecutive frames as the motion information to help modeling the temporal evolution of the action. Finally, both the joint information and the bone information, together with their motion information, are integrated in a multi-stream framework. The fusion of the four streams shows notable improvement compared with previous single-stream method.

To verify the superiority of the proposed MS-AAGCN, extensive experiments are performed on two large-scale datasets: NTU-RGBD [14] and Kinetics-Skeleton [15]. Our model achieves the state-of-the-art performance on both of the datasets for skeleton-based action recognition. The learned adaptive graphs, attention maps of the model and the complementary of the four streams are further visualized and discussed. Finally, we provide a detailed comparison between the RGBs and skeletons and propose a skeleton-guided cropping strategy to fuse them, which shows remarkable improvement.

Overall, the main contributions of our work lie in four folds: (1) An adaptive graph convolutional network is proposed to adaptively learn the topology of the graph in an end-to-end manner, which can better suit the action recognition task, the hierarchical structure of the GCNs and the diverse skeleton samples. (2) A STC-attention module is introduced and embedded in every graph convolutional layers, which can help model learn to selectively focus on discriminative joints, frames and channels. (3) We propose to explicitly extract and fuse the first-order information (joints), the second-order information (bones), and the corresponding motion information of the skeleton data in a multi-stream framework, which can provide a more comprehensive understanding of the human actions. (4) Compared with ST-GCN, our MS-AAGCN obtains significant improvements of +7.9% and +8.5% on the CV and CS benchmarks of the NTU-RGBD dataset, respectively. By combing it with the skeleton-guided cropped RGB data, it obtains additional improvements of +2.8% and +6.1%[1].

This paper is an extended version of our previous work [16] in a number of aspects. (1) The composition scheme of the proposed global and individual graphs is optimized and a gating mechanism is introduced to adaptively adjust the importance of the two graphs. (2) A STC-attention module is proposed to help the model paying attention to important joints, frames and features. (3) The previous model is extended to a multi-stream framework, which integrates the motion information for both of the joints and the bones. (4) More experiments and quantitative results are provided to demonstrate the effectiveness and the necessity of the proposed modules and streams. (5) An effective pose-guided cropping strategy is presented to fuse the skeletons with the RGBs, which exhibits excellent performance.

The rest of paper is organized as follows. In Sec. II, we introduce the related works. In Sec. III, the ST-GCN is used as an example to formulate the basic GCNs for the skeleton-based action recognition. In Sec. IV, the components of our proposed MS-AAGCN are introduced in detail. In Sec. V, we show the ablation studies and the comparisons with the state-of-the-art methods. In Sec. VI, more quantitative results and analysis are provided. In Sec. VII, we provide more discussions for RGBs and skeletons. Finally, the paper is concluded in Sec. VIII.

II. Related Work

A. Skeleton-Based Action Recognition

Skeleton-based action recognition has been studied extensively in recent years. Here, we only review the works related to our approach. In early stage, the works for skeleton-based action recognition usually design handcrafted features to model the human body [6], [17]. For example, Vemulapalli *et al.* [6] encode the skeletons with their rotations and translations in a Lie group. Fernando *et al.* [17] use the rank pooling method to represent the data with the parameters of the ranker. In recent years, deep learning have achieved big success in many computer vision tasks, which also becomes the mainstream methods for skeleton-based action recognition. There are mainly two streams for these data-driven methods: the RNN-based methods and the CNN-based methods. RNN-based methods usually model the skeleton data as a sequence of the coordinate vectors along both the spatial and temporal dimensions, where each of the vectors represents a human body joint [7]–[9], [18], [19]. Du *et al.* [7] use a hierarchical bidirectional RNN model to identify the skeleton sequence, which divides the human body into different parts and sends them to different sub-networks. Song *et al.* [8] embed a spatiotemporal attention module in LSTM-based model, so that the network can automatically pay attention to the discriminant spatiotemporal region of the skeleton sequence. Zhang *et al.* [18] introduce the mechanism of view transformation in an LSTM-based model, which automatically translates the skeleton data into a more advantageous angle for action recognition. Si *et al.* [19] propose a model with spatial

[1]The code is released on https://github.com/lshiwjx/2s-AGCN

reasoning (SRN) and temporal stack learning (TSL), where the SRN can capture the structural information between different body parts and the TSL can model the detailed temporal dynamics.

CNN-based methods model the skeleton data as a pseudo-image based on the manually designed transformation rules [20]–[24]. The CNN-based methods are generally more popular than the RNN-based methods because the CNNs have better parallelizability and are easier for training. Kim and Reiter [20] use a one-dimensional residual CNN to identify skeleton sequences where the coordinates of joints are directly concatenated. Liu *et al.* [21] propose 10 kinds of spatiotemporal images for skeleton encoding, and enhance these images using visual and motion enhancement methods. Li *et al.* [23] use multi-scale residual networks and various data-augmentation strategies for the skeleton-based action recognition. Cao *et al.* [24] design a permutation network to learn an optimized order for the rearrangement of the joints.

However, the skeleton data are naturally embedded in the form of graphs rather than a vector sequence or a 2D grid. Both the RNNs and CNNs fail to fully represent the graph structure of the skeleton data. Very recently, Yan *et al.* [10] propose a spatiotemporal graph convolutional network (ST-GCN) to directly model the skeleton data as a spatiotemporal graph. It eliminates the requirement for designing handcrafted transformation rules to transform the skeleton data into vector sequences or pseudo-images, thus achieves better performance. Based on this, Tang *et al.* [25] further propose a selection strategy of the key frames with the help of reinforcement learning. Si *et al.* [19] propose a AGC-LSTM to embed the graph convolutional module into a multi-layer LSTMs to model the relations between joints. It is further enhanced with an spatial attention module in [26]. Li *et al.* [27] integrate the pose prediction into the action recognition task to help capture more detailed action patterns through self-supervision. Shi *et al.* [16] propose three types of graphs to adaptively learn the graph topology for action recognition, which is the basis of this work.

B. Graph Convolutional Neural Networks

The input of traditional CNNs is usually low-dimensional regular grids, such as images and videos. However, it is not straightforward to model the graph data with CNNs because the size and shape of the graphs are usually irregular, which is not suitable for gridded convolutional kernels. How to generalize CNNs to data with graph structures has been explored extensively over decades and now the most popular solution is to use the graph convolutional networks (GCNs) [28]–[37].

The principle of constructing GCNs mainly follows two streams: spatial perspective and spectral perspective. Spatial perspective methods directly perform convolution on the graph vertexes and their neighbors. The key lies in how to construct the locally connected neighborhoods from the graph which misses the implicit order of vertexes and edges. These methods always extracts the neighbors based on the manually designed rules [30], [33], [35]–[38]. For example, Niepert *et al.* [30] sample the neighborhoods for each of the vertexes based on their distances in the graph. A normalization algorithm is proposed to crop excess vertexes and pad dummy vertexes. Wang and Gupta [38] represent the video as a graph containing persons and detected objects for action recognition. The neighborhood of each vertex is defined according to the feature similarity and the spatial-temporal relations. In contrast to the spatial perspective methods, spectral perspective methods use the eigenvalues and eigen vectors of the graph Laplace matrices. These methods perform graph convolution in the frequency domain with the help of the graph Fourier transform [28], which does not need to extract locally connected regions from graphs at each convolution step [29], [32], [34]. Defferrard *et al.* [34] propose to use recurrent Chebyshev polynomials as the filtering scheme which is more efficient than previous polynomial filter. Kipf and Welling [32] further simplified this approach using the first-order approximation of the spectral graph convolutions. This work follows the spatial perspective methods.

There are also some works discussing the optimal strategy for graph construction of GCNs. Li *et al.* [39] assume that the optimal graph topology is a small shifting from the original graph topology and propose to learn a residual graph given the current data. Veličković *et al.* [40] propose to learn different attention weights for different nodes in a neighborhood. Guo *et al.* [41] also propose to learn an attention matrix according to the node features, which is then multiplied to the original adjacency matrices to dynamic adjust the impacting weights between nodes. Wu *et al.* [42] combine the pre-defined graphs and the learnable graph for graph convolutional layer. The learnable graph is fixed for all of the data samples. These works either learn attention weights for graph edges or use predefined human-body-based graph. However, the attention mechanism can not produce new edges to meet the specific task requirements because the attention map is element-wise multiplied to the original adjacency matrix. Besides, for skeleton data, we found that the predefined graph limits the flexibility of the learning process and is harmful for the performance. Instead, we propose two types of adaptive graphs, namely, the global graph and individual graph, which avoid the above limitations and are designed more suitable for skeleton-based action recognition task.

As for the related works for multi-modal data fusion with GCNs, Zhao *et al.* [43] propose a SemGCN to learn channel-wise weights for edges as priors implied in the graph. The image features are pooled according to the 2D coordinates to help predicting 3D coordinates. Wang *et al.* [58] represent the mesh as a graph and propose a cascaded mesh deformation network. The image features from different layers of the CNN network are pooled and used to gradually deforming the initial ellipsoid mesh. In this work, we also make a discussion about how to utilize the RGB features. Instead of pooling patch features or fusing multi-layer RGB features, we crop the human areas of the RGB image and feed it into another independent stream. Finally, only the $SoftMax$ scores are fused to help recognizing actions. Our method is more simple but is effective as showed in experiments. It not only reduces the interference of the cluttered background, but also largely saves the details of the human body.

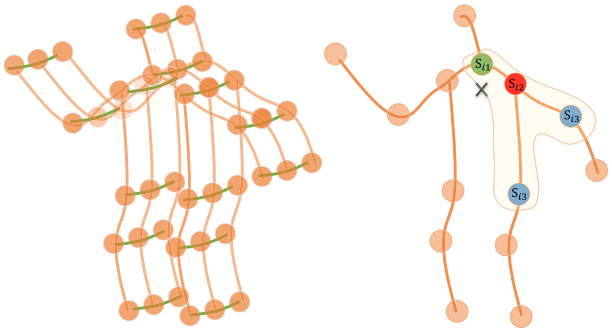

Fig. 1. Left: Illustration of the spatiotemporal graph. Right: Illustration of the mapping strategy. Different colors denote different subsets.

III. ST-GCN Revisited

In this section, we use the spatiotemporal graph convolutional network [10] (ST-GCN) as an example to introduce the previous GCN-based method for skeleton-based action recognition.

A. Spatiotemporal Graph Convolution for Skeleton Data

The raw skeleton data in one frame is usually represented by a sequence of vectors. Each vector represents the 2D/3D coordinates of the corresponding human joint. A complete action contains multiple frames with different lengths for different samples. In ST-GCN, a spatiotemporal graph is used to model the structured information among these joints along both the spatial and temporal dimensions. Here, the spatial dimension refers to the joints in the same frame, and the temporal dimension refers to the same joints over all of the frames. The left sub-figure in Fig. 1 presents an example of the constructed spatiotemporal skeleton graph, where the joints are represented as the vertexes and their natural connections in the human body are represented as the spatial edges (the orange lines in Fig. 1, left). For the temporal dimension, the corresponding joints in two consecutive frames are connected with temporal edges (the green lines in Fig. 1, left). The coordinate vector of each joint is set as the attribute of the corresponding vertex. Since the graph is intrinsic and is built based on the natural connectivity of the human body, we refer it as the human-body-based graph.

Given the graph defined above, for the spatial dimension, the graph convolution operation on vertex v_i is formulated as:

$$f_{out}(v_i) = \sum_{v_j \in \mathcal{B}_i} \frac{1}{Z_{ij}} f_{in}(v_j) \cdot w(l_i(v_j)) \tag{1}$$

where f denotes the feature map and v denotes the vertex of the graph. $\mathcal{B}_i$ denotes the sampling area of the convolution for v_i, which is defined as the 1-distance neighboring vertexes (v_j) of the target vertex (v_i). w is the weighting function similar to the traditional convolution operation, which provides a weight vector based on the given input. Note that the number of weight vectors of convolution is fixed, while the number of vertexes in $\mathcal{B}_i$ is varied. So a mapping function l_i is required to map all neighboring vertexes into a fix-numbered subsets, each of which is associated with a unique weight vector. The right sub-figure in Fig. 1 shows this mapping strategy, where $\times$ represents the center of gravity of the skeleton. $\mathcal{B}_i$ is the sampling area enclosed by the curve. In detail, the kernel size is empirically set to 3 and the $\mathcal{B}_i$ is naturally divided into 3 subsets: $\mathcal{S}_{i2}$ is the vertex itself (the red circle in Fig. 1, right); $\mathcal{S}_{i1}$ is the centripetal subset, which contains the neighboring vertexes that are closer to the center of gravity (the green dot); $\mathcal{S}_{i3}$ is the centrifugal subset, which contains the neighboring vertexes that are farther from the center of gravity (the blue dot). Z_{ij} denotes the cardinality of $\mathcal{S}_{ik}$ that contains v_j. It aims to balance the contribution of each subset.

B. Implementation of ST-GCN in Code

The skeletal sequence in code is actually a 3-order tensor whose shape is $C \times T \times N$, where C, T and N denote the number of channels, frames and joints, respectively. For spatial dimension, to implement the ST-GCN in code, Eq. 1 should also be transformed into the tensor format. We use $\mathcal{X} \in \mathbb{R}^{C_{in} \times T \times N}$ and $\mathcal{Y} \in \mathbb{R}^{C_{out} \times T \times N}$ to denote the input and output feature maps, i.e., f_{in} and f_{out} in Eq. 1, respectively. To perform multiplication for tensors, the elements of the tensor should be reordered into a matrix. Here we make use of the tensor matricization to achieve this operation, also known as unfolding or flattening [44]. In detail, given a tensor $\mathcal{V} \in \mathbb{R}^{C_1 \times C_2 \times \cdots \times C_n}$, its k-mode matricization is an (S_1, S_2)-reshaping in which $S_1 = (k)$ and $S_2 = (1, 2, \cdots, k-1, k+1, \cdots, n)$. Since the matricization strategy is already known, we also define the reverse matricization as $|*|_{\mathfrak{S}}$, which reorders the matrix to the tensor with the shape as $\mathfrak{S}$. For example, the 3-mode matricization of $\mathcal{X} \in \mathbb{R}^{C_{in} \times T \times N}$, called $\mathbf{X}_{(3)}$, is an $N \times C_{in}T$ matrix and its reverse matricization $|\mathbf{X}_{(3)}|_{C_{in} \times T \times N}$ is the original $C_{in} \times T \times N$ tensor.

Apart from the tensor multiplication, the mapping function l_i of Eq. 1 also need to be defined. It is realized with the help of a adjacency-like matrix $\bar{\mathbf{A}}_k \in \mathbb{R}^{N \times N}$, whose element $\bar{\mathbf{A}}_k^{ij}$ indicates whether the vertex v_j is in the subset S_{ik} of the vertex v_i (1 or 0). By multiplying it with the input tensor in the joint dimension, it can filter out the connected vertexes in a particular subset for the corresponding weight vector. Since the contribution of each subset is balanced by dividing Z_{ij} according to Eq. 1, $\bar{\mathbf{A}}_k$ is also normalized to $\mathbf{A}_k = \Lambda_k^{-\frac{1}{2}} \bar{\mathbf{A}}_k \Lambda_k^{-\frac{1}{2}}$. $\Lambda_k^{ii} = \sum_j (\bar{\mathbf{A}}_k^{ij}) + \alpha$ is the normalized diagonal matrix, which keeps the magnitude the same as before after the graph convolution. α is set to 0.001 to avoid empty rows.

With these definitions, Eq. 1 is transformed into

$$\hat{\mathcal{X}}_k = |\mathbf{M}_k \odot \mathbf{A}_k \mathbf{X}_{(3)}|_{C_{in} \times T \times N} \tag{2}$$

$$\mathcal{Y} = \sum_k^{K_v} |\mathbf{W}_k \hat{\mathbf{X}}_{k(1)}|_{C_{out} \times T \times N} \tag{3}$$

where $\mathbf{M}_k \in \mathbb{R}^{N \times N}$ is an attention map added by ST-GCN, which represents the strength of connections. $\odot$ denotes the element-wise multiplication. Eq. 2 corresponds to the mapping function l of Eq. 1. K_v denotes the kernel size of the spatial dimension. With the mapping strategy designed above, K_v is

set to 3. $\mathbf{W}_k \in \mathbb{R}^{C_{out} \times C_{in}}$ is the weight vector that corresponds to the w of Eq. 1.

For temporal dimension, since the number of neighbors for each vertex is fixed as 2 (corresponding joints in the two adjacent frames), it is straightforward to perform the graph convolution similar to the classical convolution operation. Concretely, we perform a $K_t \times 1$ convolution on the $T \times N$ output feature map calculated above, where K_t is the kernel size of the temporal dimension.

With above formulations, multiple layers of spatiotemporal graph convolution operations are applied on the graph to extract the high-level features. The global average pooling layer and the $SoftMax$ classifier are then used to predict the action categories based on the extracted features.

IV. Multi-Stream Attention-Enhanced Adaptive Graph Convolutional Network

In this section, we introduce the components of the proposed multi-stream attention-enhanced adaptive graph convolutional network (MS-AAGCN) in detail.

A. Adaptive Graph Convolution

The spatiotemporal graph convolution for the skeleton data described in Sec. III is calculated based on a intrinsic human-body-based graph, which may not be the best choice as explained in Sec. I. To solve this problem, we propose an adaptive graph convolutional layer, where the graph topology is optimized together with other parameters of the network in an end-to-end learning manner. The graph is unique for different layers and samples, which greatly increases the flexibility of the model. Meanwhile, it is designed as a residual branch, which can guarantee the stability of the original model.

In detail, according to the Eq. 2, the graph is actually determined by the adjacency matrix and the mask, i.e., $\mathbf{A}_k$ and $\mathbf{M}_k$, respectively. $\mathbf{A}_k$ determines whether there are connections between two vertexes and $\mathbf{M}_k$ determines the strength of the connection. To make the graph topology adaptive, we modify the Eq. 2 into the following form:

$$\hat{\mathcal{X}}_k = |(\mathbf{B}_k + \alpha \mathbf{C}_k)\mathbf{X}_{(3)}|_{C_{in} \times T \times N} \tag{4}$$

The main difference lies in the adjacency matrix of the graph, which is divided into two sub-graphs: $\mathbf{B}_k$ and $\mathbf{C}_k$.

The first sub-graph ($\mathbf{B}_k$) is the global graph learned from the data. It represents the graph topology that is more suitable for the action recognition task. It is initialized with the adjacency matrix of the human-body-based graph, i.e., $\mathbf{A}_k$ in Eq. 2. Different with $\mathbf{A}_k$, the elements of $\mathbf{B}_k$ are parameterized and updated together with other parameters in the training process. There are no constraints on the value of $\mathbf{B}_k$, which means that either creating new edges or removing old edges are allowed for $\mathbf{B}_k$. With this data-driven method, the model can learn graphs that are fully targeted to the recognition task. Besides, $\mathbf{B}_k$ is unique for each layer, thus is more individualized for different levels of semantics contained in different layers.

The second sub-graph ($\mathbf{C}_k$) is the individual graph which learns a unique topology for each sample. To determine whether there is a connection between two vertexes and how strong is the connection, we use the normalized embedded Gaussian function to estimate the feature similarity of the two vertexes as:

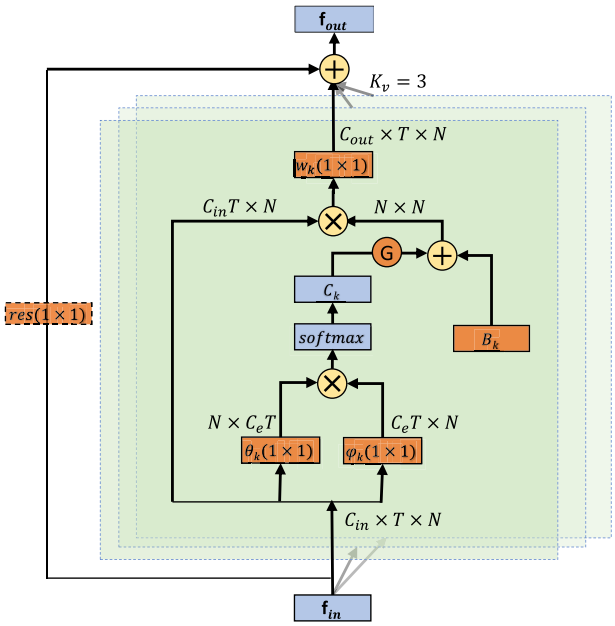

Fig. 2. Illustration of the adaptive graph convolutional layer (AGCL). There are two kinds of graphs in each layer, i.e., $\mathbf{B}_k$ and $\mathbf{C}_k$. The orange box indicates that the part is the parameter of the network and is updated during the training process. θ and ϕ are two embedding functions whose kernel size is (1×1). K_v denotes the number of subsets. $\oplus$ denotes the element-wise addition. $\otimes$ denotes the matrix multiplication. G is the gate that controls the importance of the two kinds of graphs. The residual box (dotted line) is only needed when C_{in} is not the same as C_{out}.

$$f(v_i, v_j) = \frac{e^{\theta(v_i)^\dagger \phi(v_j)}}{\sum_{j=1}^{N} e^{\theta(v_i)^\dagger \phi(v_j)}} \tag{5}$$

where N is the number of vertexes. $\dagger$ denote the transpose operation. We use the dot product to measure the similarity of the two vertexes in an embedding space. In detail, given the input feature map $\mathcal{X} \in \mathbb{R}^{C_{in} \times T \times N}$, we first embed it into the embedding space $\mathbb{R}^{C_e \times T \times N}$ with two embedding functions, i.e., θ and ϕ. Here, through extensive experiments, we choose the 1×1 convolutional layer as the embedding function. The two embedded features maps, i.e., $\mathcal{M}_{\theta k}$ and $\mathcal{M}_{\phi k}$, are multiplied to obtain a similarity matrix $\mathbf{C}_k \in \mathbb{R}^{N \times N}$, whose element $\mathbf{C}_k^{ij}$ represents the similarity between the vertex v_i and vertex v_j. The value of the matrix is normalized to $0-1$, which is used as the soft edge of the two vertexes. Since the normalized Gaussian is equipped with a $SoftMax$ operation, the $\mathbf{C}_k$ can be computed based on Eq.5 as follows:

$$\begin{aligned} \mathcal{M}_{\theta k} &= |\mathbf{W}_{\theta k}\mathbf{X}_{(1)}|_{C_e \times T \times N} \\ \mathcal{M}_{\phi k} &= |\mathbf{W}_{\phi k}\mathbf{X}_{(1)}|_{C_e \times T \times N} \\ \mathbf{C}_k &= SoftMax(\mathbf{M}_{\theta k(3)}\mathbf{M}_{\phi k(3)}^{\dagger}) \end{aligned} \tag{6}$$

where $\mathbf{W}_\theta$, $\mathbf{W}_\phi \in \mathbb{R}^{C_e \times C_{in}}$ are the parameters of the embedding functions θ and ϕ, respectively.

Gating mechanism: The global graph determines the basic graph topology for the action recognition task. It can be

considered as providing a global regularization for graph learning. It is learned from the whole training data and is irrelevant with the input features. The individual graph provides more flexibility of the model and adds individuality according to the various sample features. It is calculated based on the similarity of the embedded features and is distinct for different samples. For bottom layers of the network, the receptive field is small and the features are mostly low-level features, which limits the ability of learning the graph topology from diverse samples. Thus the global graph should be more important in these layers because it is irrelevant with the input features. But for top layers, the model gathers more comprehensive information and the features are more semantic, which provides more diversity and requires more individuality of the graph topology. Thus the individual graph should be more important because it is constructed based on the input features and is individual for each of the samples.

Based on these observations, we propose a gating mechanism to adjust the importance of the individual graph for different layers. In detail, the $\mathbf{C}_k$ is multiplied with a parameterized coefficient α, which is unique for each layer and is updated in the training process.

Initialization: In the experiments we found that the graph topology changes dramatically in the early stage of the training process, thus causes unstable and affect the convergence of the model. To stabilize the training, we propose two strategies. The first strategy is using $\mathbf{A}_k + \alpha\mathbf{B}_k + \beta\mathbf{C}_k$ as the adjacency matrix where $\mathbf{A}_k$ is the human-body-based graph which is fixed. The $\mathbf{B}_k$, $\mathbf{C}_k$, α and β are initialized to be 0, thus the $\mathbf{A}_k$ will dominate the early stage of the training. The second strategy is initializing the $\mathbf{B}_k$ with $\mathbf{A}_k$ and blocking the propagation of the gradient for $\mathbf{B}_k$ at the early stage of the training process to stabilize the training process. Finally, the second strategy is verified slightly better as shown in Sec. V-C.

The overall architecture of the adaptive graph convolutional layer (AGCL) is shown in Fig. 2. The kernel size of the graph convolution (K_v) is set to 3. w_k is the weighting function introduced in Eq. 1. A residual connection, similar to [45], is added for each layer, which allows the layer to be inserted into any existing models without breaking its initial behavior. If the number of input channels is different from the number of output channels, a 1×1 convolution (orange box with dashed line in Fig. 2) will be inserted in the residual path to transform the input to match the output in the channel dimension. G is the gate that controls the semantics of the two kind of graphs.

B. STC-Attention Module

There have been many formulations for attention modules [11]–[13], [46]. Here, with extensive experiments, we propose a STC-attention module as showed in Fig. 3. It contains three sub-modules: spatial attention module, temporal attention module and channel attention module. The input feature maps are fed into the three modules in sequence, which generates the attention maps along different dimensions. These attention maps are then multiplied to the original feature maps to strengthen the corresponded features. A residual connection is added for all attention modules to stabilize the training.

Spatial attention module (SAM) helps model strengthening the features of a key set of joints that are important for characterizing a certain kind of actions. It is computed as:

$$\mathcal{M}_s = \sigma(g_s(AvgPool_t(\mathcal{X})) \tag{7}$$

where $\mathcal{X} \in \mathbb{R}^{C\times T\times N}$ is the feature map. It is averaged over all of the frames by $AvgPool_t$, i.e., $AvgPool_t(\mathcal{X}) \in \mathbb{R}^{C\times 1\times N}$. g_s represents the 1D convolution along the spatial dimension. The number of output channels of g_s is 1. It is used to generate the attention values that encodes where to emphasize or suppress. σ denotes the $Sigmoid$ activation function. It should be note that instead of employing $SoftMax$ to restrict the summation of attention values to one, we found that using $Sigmoid$ obtains better performance. It may because the $SoftMax$ function restricts the learning of the attention weights because one value is related with other values. Instead, $Sigmoid$ function can provide more flexibility, which is more suitable for the data-driven neural networks. The attention map $\mathcal{M}_s \in \mathbb{R}^{1\times 1\times N}$ is then dot multiplied to the input feature map in a residual manner for adaptive feature refinement.

Temporal attention module (TAM) is similar with the SAM. It helps model paying more attention on the frames that are in more important stages for action recognition. It is computed as:

$$\mathcal{M}_t = \sigma(g_t(AvgPool_s(\mathcal{X})) \tag{8}$$

where $\mathcal{M}_t \in \mathbb{R}^{1\times T\times 1}$. $AvgPool_s$ denotes averaging the feature map along the spatial dimension. g_t represents the 1D convolution along the temporal dimension.

Channel attention module (CAM) helps model strengthening the discrimitive features (channels) according to the input samples. It generates the attention maps as follows:

$$\mathcal{M}_c = \sigma(g_{c2}(\delta(g_{c1}(AvgPool_{st}(\mathcal{X}))))) \tag{9}$$

where $\mathcal{M}_c \in \mathbb{R}^{C\times 1\times 1}$. $AvgPool_{st}$ denotes averaging the feature map along both the spatial and temporal dimensions. g_{c1} and g_{c2} are two linear functions operated along the channel dimension. δ denotes the $ReLu$ activation function.

Arrangement of the attention modules: the three sub-modules introduced above can be placed in different manners: the parallel manner or the sequential manner with different orders. Finally we found that the sequential manner is better, where the order is SAM, TAM and CAM. This is shown in Sec. V-C

C. Network Architecture

Fig. 4 shows a basic block of MS-AAGCN, which is the series of one spatial GCN (ConvS), one STC-attention module (STC) and one temporal GCN (ConvT). The convolution along the temporal dimension is the same as the ST-GCN, i.e., performing the $K_t \times 1$ convolution on the $C \times T \times N$ feature maps. Both the spatial GCN and the temporal GCN are followed by a batch normalization (BN) layer and a ReLU layer. To stabilize the training and ease the gradient propagation, a residual connection is added for each basic block.

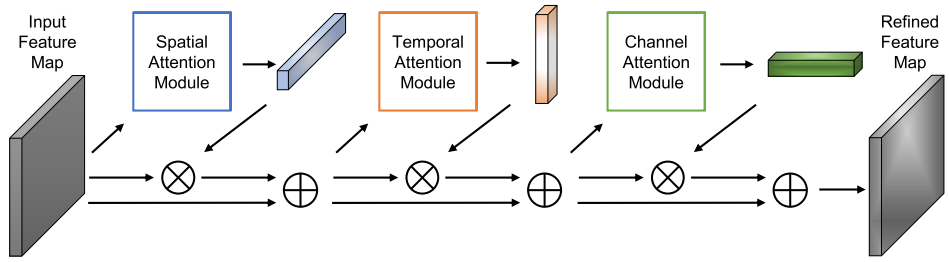

Fig. 3. Illustration of the STC-attention module. Three sub-modules are arranged in the orders of SAM, TAM and CAM. ⊗ denotes the element-wise multiplication. ⊕ denotes the element-wise addition. The generated attention maps are multiplied to the original feature maps. A residual connection is added for all attention modules to stabilize the training.

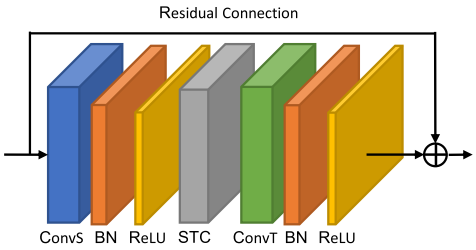

Fig. 4. Illustration of the basic block. ConvS represents the spatial AGCL, and ConvT represents the temporal AGCL, both of which are followed by a BN layer and a ReLU layer. STC represents the STC-attention module. Moreover, a residual connection is added for each block.

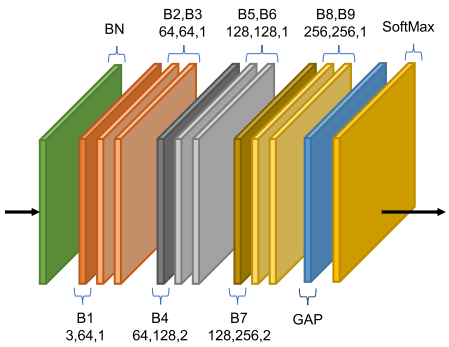

Fig. 5. Illustration of the network architecture. There are a total of 9 basic blocks (B1-B9). The three numbers of each block represent the number of input channels, the number of output channels and the stride, respectively. GAP represents the global average pooling layer.

Fig. 5 shows the overall architecture of the network, which is the stack of the basic blocks introduced in Fig. 4. There are a total of 9 blocks the same as ST-GCN. The numbers of output channels for each block are 64, 64, 64, 128, 128, 128, 256, 256 and 256. A data BN layer is added at the beginning to normalize the input data. A global average pooling layer is performed at the end to pool feature maps of different samples to the same size. The final output is sent to a $SoftMax$ classifier to obtain the prediction.

D. Multi-Stream Framework

As introduced in Sec. I, both the first-order information (the coordinates of the joints) and the second-order information (the direction and length of the bones), as well as their motion information, are worth to be investigated for the skeleton-based action recognition task. In this work, we model these four types of data in a multi-stream framework.

In particular, we define that the joint closer to the center of gravity of the skeleton is the source joint and the joint farther away from the center of gravity is the target joint. Each bone is represented as a vector pointing from its source joint to its target joint. For example, given a bone in frame t with its source joint $\mathbf{v}_{i,t} = (x_{i,t},\ y_{i,t},\ z_{i,t})$ and its target joint $\mathbf{v}_{j,t} = (x_{j,t},\ y_{j,t},\ z_{j,t})$, the vector of the bone is calculated as $\mathbf{e}_{i,j,t} = (x_{j,t} - x_{i,t},\ y_{j,t} - y_{i,t},\ z_{j,t} - z_{i,t})$. Since there are no cycles in the graph of the skeleton data, each bone can be assigned with a unique target joint. Note that the number of joints is one more than the number of bones because the root joint is not assigned to any bones. To simplify the design of the network, we assign an empty bone with its values as 0 to the root joint. Thus both the graph and the network of bones can be designed the same as that of joints. We use J-stream and B-stream to represent the networks of joints and bones, respectively.

As for the motion information, it is calculated as the difference between the same joints or bones in two consecutive frames. For example, given a joint in frame t, i.e., $\mathbf{v}_{i,t} = (x_{i,t},\ y_{i,t},\ z_{i,t})$ and the same joint in frame $t+1$, i.e., $\mathbf{v}_{i,t+1} = (x_{i,t+1},\ y_{i,t+1},\ z_{i,t+1})$, the motion information between the $\mathbf{v}_{i,t}$ and $\mathbf{v}_{i,t+1}$ is represented as $\mathbf{m_{i,t,t+1}} = (x_{i,t+1} - x_{i,t},\ y_{i,t+1} - y_{i,t},\ z_{i,t+1} - z_{i,t})$.

The overall architecture (MS-AAGCN) is shown in Fig. 6. The joints, bones and their motions are fed into four streams. Finally, the $SoftMax$ scores of the four streams are fused using weighted summation to obtain the action scores and predict the action label.

V. Experiments

To perform a head-to-head comparison with ST-GCN, our experiments are conducted on the same two large-scale action recognition datasets: NTU-RGBD [14] and Kinetics-Skeleton [10], [15]. First, since the NTU-RGBD dataset is smaller than the Kinetics-Skeleton dataset, we perform exhaustive ablation studies on it to verify the effectiveness of the proposed model components based on the recognition performance. Then, the final model is evaluated on both of

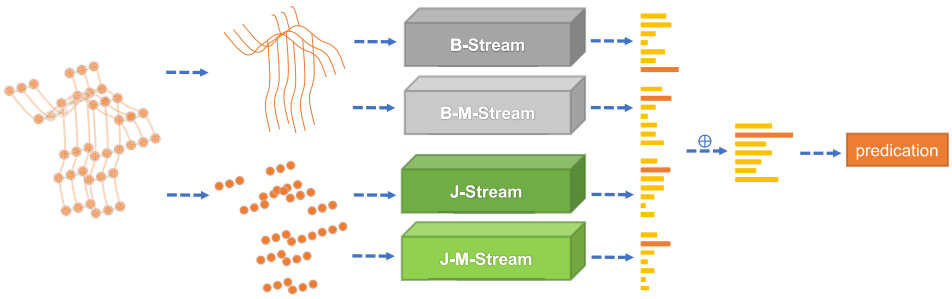

Fig. 6. Illustration of the overall architecture of the MS-AAGCN. The $SoftMax$ scores of the four streams are fused using weighted summation to obtain the final prediction. J denotes the joint information. B denotes the bone information. M denotes the motion information.

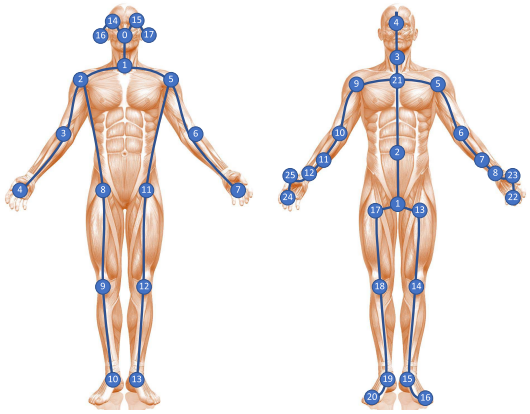

Fig. 7. The left sub-graph shows the joint labels of the Kinetics-Skeleton dataset and the right sub-graph shows the joint labels of the NTU-RGBD dataset.

the datasets to verify the generality and is compared with the other state-of-the-art approaches for skeleton-based action recognition task. The definitions of joints and their natural connections in the two datasets are shown in Fig. 7.

A. Datasets

NTU-RGBD: NTU-RGBD [14] is currently the most popular and authoritative dataset for skeleton-based action recognition. It contains 56,000 action clips in 60 action classes. Each action is captured by 3 cameras at the same height but from different horizontal angles: $-45°, 0°, 45°$. This dataset provides 3D joint locations of each frame detected by Kinect-V2 depth sensors. There are 25 joints for each subject in the skeleton sequences, while each video has no more than 2 subjects. The original paper [14] of the dataset recommends two benchmarks: (1) Cross-subject (CS): the dataset in this benchmark is divided into a training set (40,320 videos) and a test set (16,560 videos), where the subjects in the two subsets are different. (2) Cross-view (CV): the training set in this benchmark contains 37,920 videos that are captured by cameras 2 and 3, and the test set contains 18,960 videos that are captured by camera 1. We follow this convention and report the top-1 accuracy on both benchmarks.

Kinetics-Skeleton: Kinetics [15] is a large-scale human action dataset that contains 300,000 videos clips in 400 classes. The video clips are sourced from YouTube videos and have a great variety. It only provides raw video clips without skeleton data. ST-GCN [10] estimates the locations of 18 joints on every frame of the clips using the publicly available OpenPose toolbox [47]. Two peoples are selected for multi-person clips based on the average joint confidence. We use their released data (Kinetics-Skeleton) to evaluate our model. The training set contains 240,000 clips and the test set contains 20,000 clips. Following the evaluation method in [10], we train the models on the training set and report the top-1 and top-5 accuracies on the test set.

B. Training Details

All experiments are conducted on the PyTorch deep learning framework [48]. Stochastic gradient descent (SGD) with Nesterov momentum (0.9) is applied as the optimization strategy. The batch size is 64. Cross-entropy is selected as the loss function to back-propagate gradients. The weight decay is set to 0.0001. Warm up is used with 5 epochs.

For the NTU-RGBD dataset, there are at most two peoples in each sample of the dataset. If the number of bodies in the sample is less than 2, we pad the second body with 0. The max number of frames in each sample is 300. For samples with less than 300 frames, we repeat the samples until it reaches 300 frames. The learning rate is set as 0.1 and is divided by 10 at the 30_{th} epoch and 40_{th} epoch. The training process is ended at the 50_{th} epoch.

For the Kinetics-Skeleton dataset, the size of the input tensor of Kinetics is set the same as [10], which contains 150 frames with 2 bodies in each frame. We perform the same data-augmentation skills as in [10]. In detail, we randomly choose 150 frames from the input skeleton sequence and slightly disturb the joint coordinates with randomly chosen rotations and translations. The learning rate is also set as 0.1 and is divided by 10 at the 45_{th} epoch and 55_{th} epoch. The training process is ended at the 65_{th} epoch.

C. Ablation Study

We verify the effectiveness of the proposed components in MS-AAGCN in this section using the NTU-RGBD dataset.

1) Learning Rate Scheduler and Data Preprocessing: In the original paper of ST-GCN, the learning rate is multiplied by 0.1 at the 10_{th} and 50_{th} epochs. The training process is ended

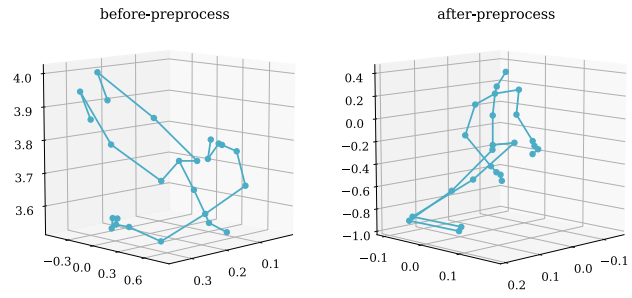

Fig. 8. Example of the data preprocessing on the NTU-RGBD dataset. The left is the original skeleton, and the right is the preprocessed skeleton.

TABLE I

COMPARISONS OF THE ACTION RECOGNITION PERFORMANCE USING REARRANGED LEARNING-RATE SCHEDULER AND DATA PREPROCESSING

Methods	CS (%)	CV (%)
original performance in [10]	81.5	88.3
before preprocessing	82.4	90.1
after preprocessing	84.3	92.7

in 80 epochs. We rearrange the learning rate scheduler from [10, 50, 80] to [30, 40, 50] and obtain the better performance (shown in Tab. I, "before preprocessing").

Moreover, we use some preprocessing strategies on the NTU-RGBD dataset. The body tracker of Kinect is prone to detecting more than 2 bodies, some of which are objects. To filter the incorrect bodies, we first select two bodies in each sample based on the body energy. The energy is defined as the average of the skeleton's standard deviation across each of the channels. Subsequently, each sample is normalized to make the distribution of the data for each channel unified. In detail, the coordinates of each joint are subtracted from the coordinates of the "spine joint" (the 2_{nd} joint of the left sub-graph in Fig. 7). Finally, as different samples may be captured in different viewpoints, similar to [14], we translate the original 3D location of the body joints from the camera coordinate system to body coordinates. For each sample, we perform a 3D rotation to fix the X axis parallel to the 3D vector from the "right shoulder" (5_{th} joint) to the "left shoulder" (9_{th} joint), and the Y axis toward the 3D vector from the "spine base" (21_{st} joint) to the "spine" (2_{nd} joint). Fig. 8 shows an example of the preprocessing. The performance after the preprocessing is referred as "after-preprocessing" in Tab. I.

Tab. I shows that the preprocessing considerably helps the recognition. It is because that the original data are too noisy.

2) Adaptive Graph Convolutional Block: For fair comparison, we use the "after preprocessing" version of the ST-GCN shown in Tab. I as the baseline method, which is denoted as "STGCN*" in the following paper. AGCN-A uses only the pre-defined graph and removes the attention maps used in STGCN*. It shows the performance largely drops without the attention maps, which can somewhat reflect the importance of adaptive graph learning. As introduced in Section IV-A, there are 2 kinds of sub-graphs in the our proposed adaptive graph convolutional layer (AGCL), i.e., the global graph B and the individual graph C. We test the performance of using each

TABLE II

COMPARISONS OF THE ACTION RECOGNITION PERFORMANCE ON THE NTU-RGBD DATASET. STGCN* DENOTE THE RESULT OF "AFTER PREPROCESSING" IN TAB. I. A DENOTES THE ADJACENCY MATRIX OF THE BODY-BASED GRAPH SHOWN IN EQ. 2. B AND C DENOTE THE GLOBAL GRAPH AND THE INDIVIDUAL GRAPH INTRODUCED IN SEC IV-A, RESPECTIVELY. G DENOTES USING THE GATING MECHANISM

Methods	CS (%)	CV (%)
STGCN*	84.3	92.7
AGCN-A	83.7	91.1
AGCN-B	86.4	93.6
AGCN-C	86.1	93.5
AGCN-ABC	86.6	93.7
AGCN-BC	87.0	94.1
AGCN-BC-G	87.4	94.4

of the graphs along and combining them together. The results are shown as AGCN-B, AGCN-C and AGCN-BC in Tab. II, respectively. It shows that both of the two designed graphs brings notable improvement for the action recognition task. With two graphs added together, the model obtains the best performance.

Besides, the performance of initial strategies introduced in Sec. IV-A is tested and shown as AGCN-ABC and AGCN-BC in Tab. II. It suggests the second strategy is better.

Moreover, we verify the effectiveness of adding the gating mechanism (AGCN-BC-G), which also brings encouraging improvement. Overall, the complete AGCL brings improvements of $+3.1\%$ and $+1.7\%$ compared with STGCN* on CS and CV benchmarks, respectively.

3) Attention Module: In this section, we test the effectiveness of the proposed STC-attention module introduced in Sec. IV-B. The results are shown in Tab. III. We first separately test the contributions of three sub-modules (SAM, TAM and CAM) based on the STGCN*, shown as ASTGCN-S, ASTGCN-T and ASTGCN-C, respectively. It shows that all of the three sub-modules can help improving the performance. Then we test the performance of adding each of the sub-modules as well as concatenating them sequentially, shown as STGCN-ADD and STGCN-STC, respectively. It suggests that concatenating the three sub-modules is slightly better. Finally, we embed the STC-attention module into the AGCN and obtains the similar results. Note that the improvement brought by the attention module for AGCN ($+0.6\%$ and $+0.7\%$) is less significant than that for STGCN* ($+2.8\%$ and $+1.5\%$). We argue that it is because the AGCN is powerful and the accuracy is already very high, thus the effect of the attention module is limited.

4) Multi-Stream Framework: Finally we test the performance of using four proposed streams and show the results in Tab. IV. Here, J, B, J-M and B-M denote the joint stream, the bone stream, the motion of joint and the motion of bone introduced in Sec. IV-D, respectively. Clearly, the multi-stream method outperforms the single-stream methods. For single-stream methods, the bone stream (B-AAGCN) performs slightly better than the joint stream (J-AAGCN) for CS benchmark. As for the CV benchmark, the result is reversed. This suggests the complementarity of the two

TABLE III

COMPARISONS OF THE ACTION RECOGNITION PERFORMANCE ON THE NTU-RGBD DATASET FOR EACH OF THE ATTENTION SUB-MODULES AND DIFFERENT ARRANGEMENT STRATEGIES. STGCN* AND AGCN DENOTE THE STGCN* AND AGCN-BC-G IN TAB. II, RESPECTIVELY. STC MEANS CONCATENATING THE THREE SUB-MODULES SEQUENTIALLY

Methods	CS (%)	CV (%)
STGCN*	84.3	92.7
ASTGCN-S	86.1	93.1
ASTGCN-T	86.6	93.6
ASTGCN-C	86.5	93.7
ASTGCN-ADD	86.8	94.1
ASTGCN-STC	87.1	94.2
AGCN	87.4	94.4
AAGCN-ADD	87.7	94.8
AAGCN-STC	88.0	95.1

TABLE IV

COMPARISONS OF THE ACTION RECOGNITION PERFORMANCE WITH DIFFERENT INPUT STREAMS ON THE NTU-RGBD DATASET. JB REPRESENTS USING THE JOINT STREAM AND BONE STREAM. MS REPRESENTS USING ALL OF THE FOUR STREAMS. AAGCN DENOTES THE AAGCN-STC IN TAB. III

Methods	CS (%)	CV (%)
J-AAGCN	88.0	95.1
B-AAGCN	88.4	94.7
J-M-AAGCN	85.9	93.0
B-M-AAGCN	86.0	93.1
JB-AAGCN	89.4	96.0
MS-AAGCN	90.0	96.2

streams. By combing the joints and bones (JB-AAGCN), it brings notable improvement as expected.

The performance of motion stream (J-M-AAGCN and B-M-AAGCN) is generally lower than the performance of the joint and bone stream. However, adding them together still brings improvement.

D. Comparisons With the State-of-the-Art Methods

We compare the final model with the state-of-the-art skeleton-based action recognition methods on both the NTU-RGBD dataset and Kinetics-Skeleton dataset. The results (mean±std) are shown in Tab. V and Tab. VI, respectively. The methods used for comparisons include the handcraft-feature-based methods [6], [17], RNN-based methods [7]–[9], [14], [18], [19], [49], CNN-based methods [20]–[23], [50] and GCN-based methods [10], [16], [25]–[27]. Our model achieves the state-of-the-art performance with a large margin on both of the datasets, which suggests the superiority of our model.

E. Runtime Performance

We provide the analysis for the parameters and the runtime performance of the proposed components in one stream as shown in Tab. VII. The input size is set to $1 \times 3 \times 300 \times 25$, which denotes 1 batch, 3 channels, 300 frames and 25 joints. T denotes the time required for 100 model runs using the Intel(R) E5-2630 CPU (2.20GHz) and TITAN XP GPU. It shows that both the adaptive graph convolutional layer and the STC-attention module brings extra parameters and computational costs. The extra computation mainly comes from the calculations of the individual graph, which needs many matrix multiplications. However, the extra cost is still a small part of the original model (around 16% for parameters and 12% for GFLOPS.) All models can run in real time using only the CPU, and when using GPU, it will be much faster (around $\times 50 - \times 100$ faster).

TABLE V

COMPARISONS OF THE ACTION RECOGNITION PERFORMANCE WITH STATE-OF-THE-ART METHODS ON THE NTU-RGBD DATASET. CS AND CV DENOTE THE CROSS-SUBJECT AND CROSS-VIEW BENCHMARKS, RESPECTIVELY. WE COMPARED OUR METHODS WITH FOUR TYPES OF METHODS THAT ARE SEPARATED WITH A HORIZONTAL LINES: HANDCRAFT-FEATURE-BASED METHODS, RNN-BASED METHODS, CNN-BASED METHODS AND GCN-BASED METHODS (FROM TOP TO BOTTOM)

Methods	CS (%)	CV (%)
Lie Group [6]	50.1	82.8
HBRNN [7]	59.1	64.0
Deep LSTM [14]	60.7	67.3
ST-LSTM [49]	69.2	77.7
STA-LSTM [8]	73.4	81.2
VA-LSTM [18]	79.2	87.7
Ind-RNN [9]	81.8	88.0
SRN+TSL [19]	84.8	92.4
TCN [20]	74.3	83.1
Clips+CNN+MTLN [50]	79.6	84.8
Synthesized CNN [21]	80.0	87.2
CNN+Motion+Trans [22]	83.2	89.3
3scale ResNet152 [23]	85.0	92.3
ST-GCN [10]	81.5	88.3
DPRL+GCNN [25]	83.5	89.8
ASGCN [27]	86.8	94.2
AGCN [16]	88.5	95.1
AGC-LSTM [26]	89.2	95.0
MS-AAGCN (ours)	90.0± 0.109	96.2± 0.095

TABLE VI

COMPARISONS OF THE ACTION RECOGNITION PERFORMANCE WITH STATE-OF-THE-ART METHODS ON THE KINETICS-SKELETON DATASET. J, B AND M DENOTE USING THE JOINT INFORMATION, BONE INFORMATION AND MOTION INFORMATION, RESPECTIVELY. JB REPRESENTS USING THE JOINT AND BONE STREAM. MS REPRESENTS USING ALL OF THE FOUR STREAM

Methods	Top-1 (%)	Top-5 (%)
Feature Enc. [17]	14.9	25.8
Deep LSTM [14]	16.4	35.3
TCN [20]	20.3	40.0
ST-GCN [10]	30.7	52.8
ASGCN [27]	34.8	56.5
AGCN [16]	36.1	58.7
J-AAGCN (ours)	36.0	58.4
B-AAGCN (ours)	34.7	57.5
J-M-AAGCN (ours)	31.7	54.6
B-M-AAGCN (ours)	29.7	50.0
JB-AAGCN (ours)	37.4	60.4
MS-AAGCN (ours)	37.8± 0.084	61.0± 0.132

VI. VISUALIZATION AND DISCUSSION

A. Adaptive Graphs

There are two kinds of graphs in our model: the global graph and the individual graph. Fig. 9 shows an example of the learned adjacency matrices of the global graph for different subsets and different layers. The first and second rows show

TABLE VII

THE ANALYSIS FOR THE PARAMETERS AND THE RUNTIME PERFORMANCE OF THE PROPOSED COMPONENTS. T DENOTES THE TIME REQUIRED FOR 100 MODEL RUNS USING CPU/GPU

Methods	#Params(M)	GFLOPS	T(ms) on CPU/GPU
ST-GCN	3.12	8.37	107/1.0
AGCN	3.47	9.34	145/2.7
AAGCN	3.78	9.35	156/3.2

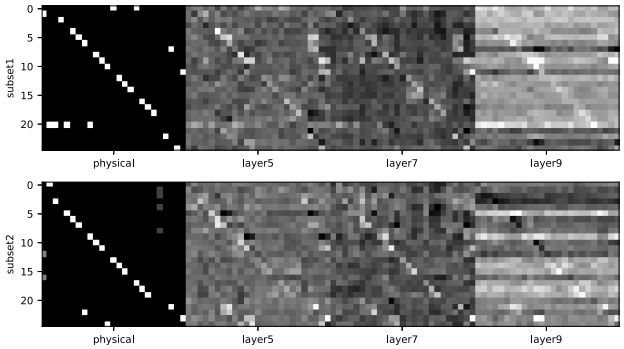

Fig. 9. Example of the learned adjacency matrices of the global graph. Different rows shows different subsets. The first column is the adjacency matrices of the human-body-based graph in the NTU-RGBD dataset. Others are examples of the learned adaptive adjacency matrices of different layers learned by our model.

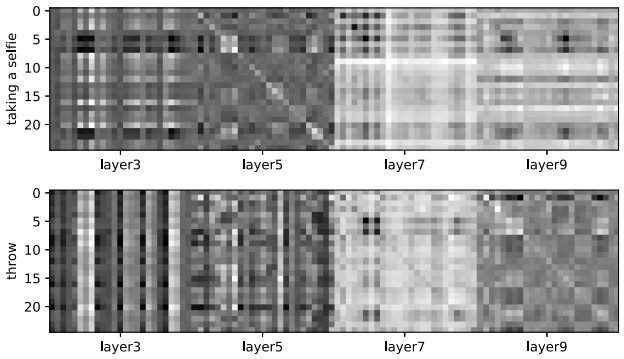

Fig. 10. Examples of the learned adjacency matrices of the individual graph. The first and second rows show different samples. Different columns represent different layers.

the adjacency matrices of the centripetal subset ($\mathcal{S}_{i2}$) and the centrifugal subset ($\mathcal{S}_{i3}$) introduced in Sec III, respectively. The first column shows the graph structure defined based on the natural connectivity of the human body, i.e., $\mathbf{A}$ in Eq. 2. Others are the adjacency matrices of the global graph in different layers. The gray scale of each element in the matrix represents the strength of the connection.

It shows that the topology of the learned graph is updated based on the human-body-based graph but has many changes. It confirms that the human-body-based graph is not the optimal choice for the action recognition task. Besides, it shows that the graph topology of the higher layers changes more than the lower layers. It is because the information contained in higher layers is more semantic, thus the required topology of graph is more different from the human-body-based graph.

Similarly, Fig. 10 shows some examples of the learned adjacency matrices of the individual graph for two different

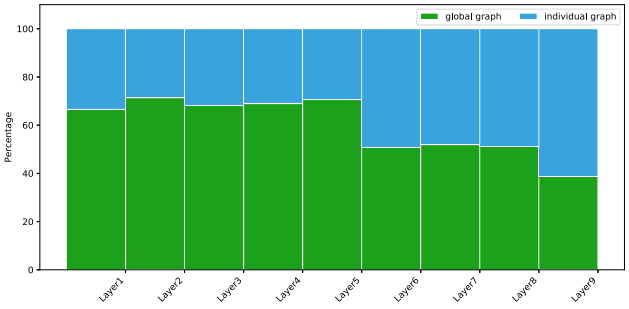

Fig. 11. Visualization for the importance of the two kinds of graphs in each of the layers.

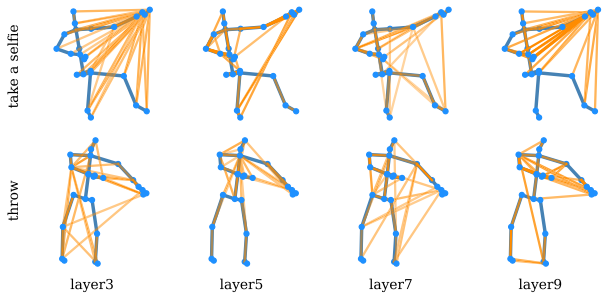

Fig. 12. Examples of the learned graph topologies. The orange lines represent the connections whose values are in the top 50.

samples. It shows that different samples and layers need different graph topologies, which confirms our motivation.

The two kinds of the graphs are fused using the gating mechanism. We visualize the importance of the two kinds of graphs in each layer in Fig. 11. It shows that the individual graph is more important on top layers. This is because the individual graph is learned based on the features of the data samples. The receptive fields of the top layers are larger and their features are more informative. Thus it is easier for top layers to learn the topology of the individual graph than lower layers.

To see the learned graph topologies clearly, we draw the connections between joints on the skeleton sketches as shown in Fig. 12. The orange lines represent the connections whose values are in the top 50. The alpha value of the lines represents the strength of the connections. It shows that the learned graph topology is accorded with the human intuition. For example, the relationships between the hand and the head should be more important to recognize the action of "take a selfie", and the learned graphs did have more connections between them as shown in the first row. This confirms the effectiveness and necessity of the proposed adaptive graph convoutional layer.

B. Attention Module

There are three sub-modules for our proposed STC-attention module: the spatial attention module, the temporal attention module and the channel attention module. For the spatial attention, we show the learned attention map for different samples and different layers in Fig. 13. The size of the circle

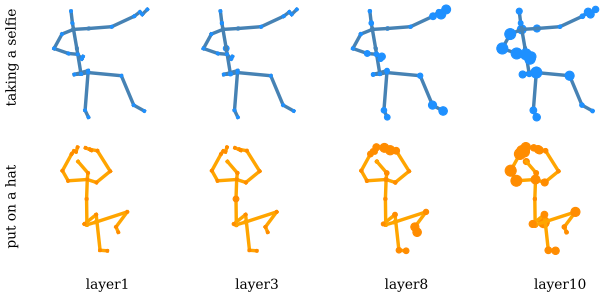

Fig. 13. Examples of the learned spatial attention maps. The size of the circle represents the importance of the corresponding joint.

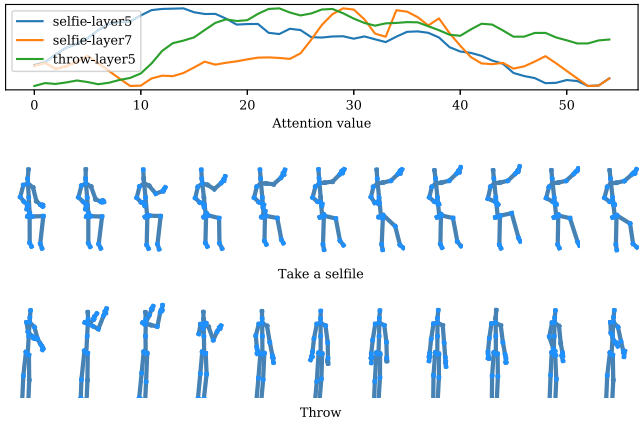

Fig. 14. Visualization of the temporal attention map. The first row shows the learned temporal attention weights for each of the frames for different layers and samples. The second and third rows show the corresponding skeleton sketches.

represents the importance of the corresponding joint. It shows that the model focus more on the joints of hands and head. Besides, the attention is not obvious in the lower layers. It is because the receptive fields of lower layers are relatively smaller, thus is hard to learn good attention maps.

For the temporal attention, we show an example of the learned attention weights for each of the frames and the corresponding skeleton sketches in Fig. 14. For the sample of taking a selfie, it shows the model focus more on the process of raising people's hand in the fifth layer, and cares more about the final posture of the selfie in the seventh layer. For the sample of throwing, although in the same fifth layer, it learns different structure and pays more attentions on the frames where the hands are in the lower position. This shows the effectiveness and the adaptability of the designed module.

VII. More Discussions for RGBs and Skeletons

The skeleton data is robust to the dynamic circumstance and complicated background. However, it lacks the appearance information. For example, if a person is eating something, it is hard to judge whether he is eating an apple or a peal using only the skeleton data.

In this section, we investigate the necessity of fusing both the skeleton data and the RGB data for the action recognition task on the NTU-RGBD dataset. We use a two-stream framework where one of the stream models the RGB data with the 3D convolutional networks and another stream models the skeleton data with our MS-AAGCN. The details of skeleton-stream is the same as Sec V-B. For RGB-stream, inspired by [51], we use the ResNeXt3D-101 model that is pre-trained on ImageNet and Kinetics. When training, we randomly crop a clip from the whole video whose length is random sampled from $[32, 64, 128]$. The crop position is randomly selected from four corners and one center. Then each image is cropped based on the crop ratio sampled form $[1, 0.875, 0.75]$. Note that the width and height of the cropped image can be different. Finally the cropped sequence of images are normalized and resized to $[16, 224, 224]$, which denotes the length, height and width of the clip. The learning rate is initialized with 0.01 and is multiplied with 0.1 after the accuracy saturates. We use four TITANXP-GPUs and the batch size is set to 32. Momentum-SGD is used as the optimizer and the weight decay is set to 0.0005. When testing, we crop clips with different lengths and sizes in different positions. The result is the average of these clips.

Moreover, to get rid of the interference of the background, we propose to crop the persons from the original images and use only the person part for the recognition. In detail, we calculate the border of the persons in each image and use their union as the crop box. This is referred as the pose-guided cropping strategy.

Tab. VIII shows the results of our methods as well as the previous methods that also use the RGBs. Here, C denotes using the pose-guided cropping strategy. It shows that the pose-guided cropping strategy brings notable improvement when using only the RGBs (RNX3D101 vs RNX3D101-C). This suggests that the model using only RGB data is easily misguided by the complicated background. However, the improvement decreases when fusing the skeleton data and the RGB data (RNX3D101+MS-AAGCN vs RNX3D101+MS-AAGCN-C). The reason is that the skeleton data can effectively avoid the interference of the circumstance, thus the contribution of the cropping strategy becomes not as obvious as before. Our best model, i.e., RNX3D101+MS-AAGCN-C, achieves 96.1% for CS benchmark and 99.0% for CV benchmark. As shown in Tab. VIII, it exceeds the previous methods with a large margin.

TABLE VIII
Comparisons of the Action Recognition Performance Using RGBs on the NTU-RGBD Dataset. CS and CV Are Two Benchmarks. C Denote Using the Pose-Guided Cropping Strategy

Methods	Pose	RGB	CS (%)	CV (%)
DSSCA-SSLM [52]	✓	✓	74.9	-
Chained Network [53]	✓	✓	80.8	-
RGB + 2D Pose [54]	✓	✓	85.5	-
Glimpse Clouds [55]		✓	86.6	93.2
PEM [56]	✓	✓	91.2	95.3
I3D+RNN+Attention [57]	✓	✓	93.0	95.4
MS-AAGCN	✓		90.0	96.2
RNX3D101		✓	85.3	92.6
RNX3D101-C		✓	94.6	97.0
RNX3D101+MS-AAGCN	✓	✓	95.5	98.0
RNX3D101+MS-AAGCN-C	✓	✓	96.1	99.0

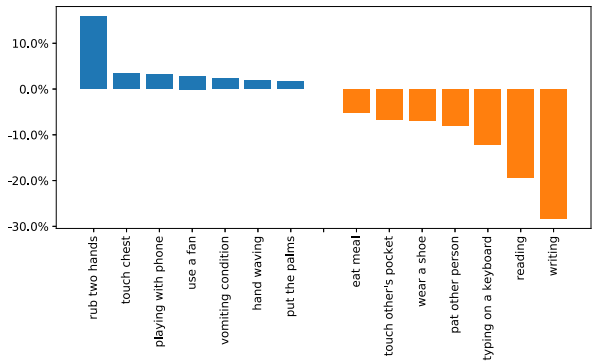

Fig. 15. Accuracy difference between the skeletons and the RGBs for different classes, i.e., ACC(skeleton)-ACC(RGB).

Fig. 16. Two examples for class "reading" and class "writing". The white lines represent the skeletons. The two examples are hard to distinguish with only skeletons.

To show the complementarity between RGBs and skeletons, we plot the accuracy difference for some of the classes. We use the CV benchmark of the NTU-RGBD dataset.

Fig. 15 shows the accuracy differences between the skeletons and the RGBs. It shows that the skeletons helps RGBs a lot for "rub the hands" class and the RGBs helps the skeletons a lot for "reading" and "writing" classes. We find two examples for the class "reading" and class "writing" as shown in Fig. 16. The skeletons of these two examples are very similar, thus are hard to tell apart. But with the help of the RGB data, they can be distinguished according to whether there is a pen in the hands. This example illustrates the complementarity between the skeletons and RGBs.

VIII. Conclusion

In this work, we propose a novel multi-stream attention-enhanced adaptive graph convolutional neural network (MS-AAGCN) for skeleton-based action recognition. The graph topology of the skeleton data used in this model is parameterized and embedded into the network to be jointly learned and updated with the other parameters. This data-driven approach increases the flexibility and generalization capacity of the model. It is also confirmed that the adaptively learned topology of graph is more suitable for the action recognition task than the human-body-based graph. Besides, an STC-attention module is embedded in every graph convolutional layers, which helps the model paying more attention to the important joints, frames and features. Moreover, we propose to explicitly model the joints, bones and the corresponding motion information in a unified multi-stream framework, which further enhances the performance. The final model is evaluated on two large-scale action recognition datasets, i.e., the NTU-RGBD and the Skeleton-Kinetics. It achieves the state-of-the-art performance on both of them. In addition, we fuse the skeleton data with the skeleton-guided cropped RGB data, which brings additional improvement. Future works can focus on how to better fuse the RGBs and skeletons. It is also worth to study the combination of the skeleton-based action recognition algorithms and the pose estimation algorithms in a unified framework.

References

[1] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 3551–3558.
[2] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.
[3] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3D convolutional networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4489–4497.
[4] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6299–6308.
[5] Y. Wang, L. Zhou, and Y. Qiao, "Temporal hallucinating for action recognition with few still images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 5314–5322.
[6] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3D skeletons as points in a lie group," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 588–595.
[7] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1110–1118.
[8] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Proc. AAAI*, 2017, pp. 4263–4270.
[9] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (IndRNN): Building a longer and deeper RNN," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 5457–5466.
[10] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. AAAI*, 2018, pp. 1–9.
[11] K. Xu *et al.*, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 2048–2057.
[12] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates, 2017, pp. 6000–6010.
[13] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi, "Gather-excite: Exploiting feature context in convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9423–9433.
[14] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1010–1019.
[15] W. Kay *et al.*, "The kinetics human action video dataset," 2017, *arXiv:1705.06950*. [Online]. Available: https://arxiv.org/abs/1705.06950
[16] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12026–12035.
[17] B. Fernando, E. Gavves, M. Jose Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5378–5387.
[18] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Oct. 2017, pp. 2117–2126.

[19] C. Si, Y. Jing, W. Wang, L. Wang, and T. Tan, "Skeleton-based action recognition with spatial reasoning and temporal stack learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 103–118.

[20] T. S. Kim and A. Reiter, "Interpretable 3D human action analysis with temporal convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1623–1631.

[21] M. Liu, H. Liu, and C. Chen, "Enhanced skeleton visualization for view invariant human action recognition," *Pattern Recognit.*, vol. 68, pp. 346–362, Aug. 2017.

[22] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with convolutional neural networks," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2017, pp. 597–600.

[23] B. Li, Y. Dai, X. Cheng, H. Chen, Y. Lin, and M. He, "Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2017, pp. 601–604.

[24] C. Cao, C. Lan, Y. Zhang, W. Zeng, H. Lu, and Y. Zhang, "Skeleton-based action recognition with gated convolutional neural networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 11, pp. 3247–3257, Nov. 2019.

[25] Y. Tang, Y. Tian, J. Lu, P. Li, and J. Zhou, "Deep progressive reinforcement learning for skeleton-based action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 5323–5332.

[26] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, "An attention enhanced graph convolutional LSTM network for skeleton-based action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1227–1236.

[27] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *Proc. ICCV*, 2019, pp. 3595–3603.

[28] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.

[29] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–14.

[30] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2016, pp. 2014–2023.

[31] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, 2016, pp. 1–14.

[33] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.

[34] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[35] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[36] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5115–5124.

[37] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 2693–2702.

[38] X. Wang and A. Gupta, "Videos as Space-Time Region Graphs," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 399–417.

[39] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proc. AAAI*, 2018, pp. 1–8.

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–12.

[41] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI*, Jul. 2019, pp. 922–929.

[42] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph WaveNet for deep spatial-temporal graph modeling," in *Proc. IJCAI*, Aug. 2019, pp. 1907–1913.

[43] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas, "Semantic graph convolutional networks for 3D human pose regression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3420–3430.

[44] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[46] F. Wang *et al.*, "Residual attention network for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3164–6156.

[47] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7291–7299.

[48] A. Paszke *et al.*, "Automatic differentiation in PyTorch," in *Proc. Adv. Neural Inf. Process. Syst. Workshops*, 2017, pp. 1–4.

[49] J. Liu *et al.*, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, vol. 9907, 2016, pp. 816–833.

[50] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "A new representation of skeleton sequences for 3D action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4570–4579.

[51] L. Shi, Y. Zhang, J. Hu, J. Cheng, and H. Lu, "Gesture recognition using spatiotemporal deformable convolutional representation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2019, pp. 1900–1904.

[52] A. Shahroudy, T.-T. Ng, Y. Gong, and G. Wang, "Deep multimodal feature analysis for action recognition in RGB+D videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1045–1058, May 2018.

[53] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox, "Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2904–2913.

[54] D. C. Luvizon, D. Picard, and H. Tabia, "2D/3D pose estimation and action recognition using multitask deep learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5137–5146.

[55] F. Baradel, C. Wolf, J. Mille, and G. W. Taylor, "Glimpse clouds: Human activity recognition from unstructured feature points," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 469–478.

[56] M. Liu and J. Yuan, "Recognizing human actions as the evolution of pose estimation maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 1159–1168.

[57] S. Das, A. Chaudhary, F. Bremond, and M. Thonnat, "Where to focus on for human action recognition?" in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 71–80.

[58] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang, "Pixel2Mesh: Generating 3D mesh models from single RGB images," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 52–67.