Graph Edge Convolutional Neural Networks for Skeleton-Based Action Recognition

Xikun Zhang, Chang Xu, Xinmei Tian, *Member, IEEE*, and Dacheng Tao, *Fellow, IEEE*

***Abstract*— Body joints, directly obtained from a pose estimation model, have proven effective for action recognition. Existing works focus on analyzing the dynamics of human joints. However, except joints, humans also explore motions of limbs for understanding actions. Given this observation, we investigate the dynamics of human limbs for skeleton-based action recognition. Specifically, we represent an edge in a graph of a human skeleton by integrating its spatial neighboring edges (for encoding the cooperation between different limbs) and its temporal neighboring edges (for achieving the consistency of movements in an action). Based on this new edge representation, we devise a graph edge convolutional neural network (CNN). Considering the complementarity between graph node convolution and edge convolution, we further construct two hybrid networks by introducing different shared intermediate layers to integrate graph node and edge CNNs. Our contributions are twofold, graph edge convolution and hybrid networks for integrating the proposed edge convolution and the conventional node convolution. Experimental results on the Kinetics and NTU-RGB+D data sets demonstrate that our graph edge convolution is effective at capturing the characteristics of actions and that our graph edge CNN significantly outperforms the existing state-of-the-art skeleton-based action recognition methods.**

***Index Terms*— Action recognition, graph convolutional neural networks (CNNs), skeletal data.**

I. Introduction

HUMAN behavior analysis is a crucial and challenging problem in computer vision. Recently, there emerge different tasks to tackle this problem from distinct aspects, including human pose estimation to detect and localize major joints of the human body [1], [2], gait recognition to identify people's walking pattern [3]–[6], and human action recognition to classify human actions [7]–[9]. We focus on human action recognition considering its wide applications in video surveillance, virtual reality, human–computer interaction, and robotics. Classical monocular RGB video-based action recognition often has difficulties in comprehensively representing actions in the 3-D space [10], [11]. Given the fast development of low-cost devices to capture 3-D data (e.g., camera arrays and Kinect), an increasing number of studies are actively being conducted on 3-D action recognition [12]–[14].

Skeletons generated from depth maps are often invariant to viewpoint or appearance. Moreover, as a high-level abstraction of human actions, skeletal data greatly simplify the difficulty in representing and understanding different action categories. Recently, different technologies have been developed to estimate the temporal motion of a skeleton by tracking and analyzing the motion of human joints. One of the most straightforward approaches is to concatenate coordinates of joints into a long feature vector at each time step and then input it into temporal analysis models [15], [16]. However, the spatial relationship between the joints, as an indispensable part of a human skeleton, has been neglected. To exploit the connections between the joints, [17] used a covariance matrix for skeleton joint locations over time as a discriminative descriptor, and [15] proposed representing an action by a weighted summation of actionlets. Most recently, many deep neural network-based solutions are developed. For example, [18] used multiple recurrent neural networks (RNNs) in a tree-like hierarchy to categorize action classes. Zhang *et al.* [19] designed a view adaptive RNN with long short term memory (LSTM) architecture that can adapt to most suitable observation viewpoints from end to end. Yan *et al.* [20] developed the spatial–temporal graph convolutional networks to process human skeleton in both the spatial and temporal domains. Lee *et al.* [21] proposed the ensemble temporal sliding LSTM networks composed of short-, medium-, and long-term TS-LSTM to capture various temporal features. Li *et al.* [22] introduced a two-stream convolutional neural network (CNN) to process both raw coordinates and motion data obtained by subtracting joint coordinates in the consecutive frames. Ke *et al.* [23] transformed a skeleton sequence into three clips corresponding to three cylindrical coordinates of the skeleton sequence before applying deep CNN.

Taking human joints as graph nodes, skeleton data can be naturally handled by graph CNNs [12], [20]. Recently, there is increasing interest in extending deep learning for graph data. In general, existing graph CNNs can be divided into two categories: spatial graph convolutional network (GCN) [24], [25] and spectral GCN [26], [27]. GCNs designed in the spatial domain mimic the image-based convolution and perform

Manuscript received June 11, 2018; revised January 18, 2019 and July 19, 2019; accepted August 2, 2019. Date of publication September 17, 2019; date of current version August 4, 2020. This work was supported by the Australian Research Council under Project FL-170100117, Project DP-180103424, and Project DE-180101438. *(Corresponding author: Chang Xu.)*

X. Zhang, C. Xu, and D. Tao are with the UBTECH Sydney Artificial Intelligence Centre, The University of Sydney, Darlington, NSW 2008, Australia, and also with the Faculty of Engineering, School of Computer Science, The University of Sydney, Darlington, NSW 2008, Australia (e-mail: xzha0505@uni.sydney.edu.au; c.xu@sydney.edu.au; dacheng.tao@sydney.edu.au).

X. Tian is with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application Systems, University of Science and Technology of China, Hefei 230027, China (e-mail: xinmei@ustc.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2935173

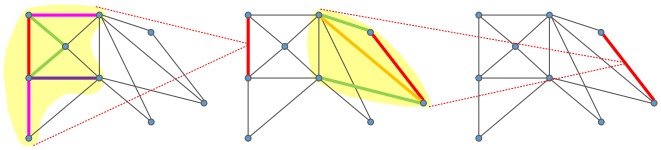


Fig. 1. Illustration of the edge convolution operation. In the left and middle graphs, the light yellow zones denote the receptive fields, and the red bold edge is the center edge in the receptive field. Other colored bold edges in addition to the center red edge are neighboring edges, and their differences in color indicate the different weights assigned to them.

convolution by integrating all neighboring nodes at each position. Spectral GCNs first transform graphs into their spectrum and then conduct convolution with convolutional kernel designed in the spectral domain according to the convolution theorem. These existing works often interpret the graph from the perspective of graph node, while neglecting the importance of graph edges, especially when graph edge possess notable physical meanings in some real-world scenario, e.g., limbs in the human skeleton.

Considering the task of skeleton-based action recognition, these methods have yielded impressive performance improvements on real-world action recognition data sets. They mostly focus on human joints, which are directly obtained from a pose estimation algorithm. However, besides joints, humans also explore the motions of limbs for understanding actions. Therefore, it is also instructive to pay attention to the dynamics of human limbs (corresponding to the edges in a skeleton graph) for skeleton-based action recognition.

Based on this observation, in this article, we propose a novel graph edge convolution operation to leverage the dynamics of human limbs. Considering a graph edge convolution operation in a single graph, an edge is represented by a weighted integration of its neighboring edges (see Fig. 1). For skeletal data in which each example is a temporal sequence consisting of multiple skeleton graphs, we extend our graph edge convolution to represent each edge by integrating both the spatial and temporal neighboring edges. Based on this, we develop an edge convolutional network to extract features from the graph edge perspective. Different from conventional graph convolutional models concentrating on graph nodes [20], [28], [29], we focus on invaluable information carried by graph edges, with graph nodes indicating only the connections between the edges. In this way, we are able to capture the relationship and dependences between human limbs, which correspond to edges in human skeleton graphs. In addition, considering the complementarity between the graph node and edge convolution, we further design two hybrid networks with different shared intermediate layers to integrate graph node- and edge-convolutional networks: one is with an extra shared fully connected layer directly merging the features from two networks by linear combination, while the other is equipped with two extra shared convolutional layers that include both nodes and edges in each convolution. We choose skeleton-based human action recognition as our task to demonstrate the effectiveness of our models, as human skeleton data are naturally in the form of graphs. We conduct experiments on two data sets: Kinetics and NTU-RGB+D, revealing the advantages of the proposed graph edge convolution over the existing state-of-the-art techniques. Further performance improvement can be derived by simultaneously exploiting graph edge and node convolutions.

The organization of this article is as follows. In Section II, we review some related works. In Section III, we introduce our graph edge CNNs (GECNNs) as well as two hybrid models integrating node and edge convolutions. Experimental results are presented in Section IV. Finally, we draw some conclusions in Section V.

II. Related Work

In this section, we briefly review the related works on human action recognition and graph CNNs.

A. Graph Convolutional Networks

Generalizing convolutional networks from images to graph-structured data has attracted the attention of the research community [22], [26], [27], [30]. These works mainly fall into two categories as follows.

1) *Spectral Perspective:* Graph data are converted into its spectrum, and CNNs are applied to the spectral domain. For example, [26] defined a spectral convolutional layer to apply filters in the frequency domain. Henaff *et al.* [27] extended spectral networks to incorporate a graph estimation procedure. However, there is an limitation shared by these spectral methods. The spectral construction is in a single domain, as the spectral filter coefficients are basis-dependent. If we learn a filter with respect to a certain set of bases, this filter cannot be applied to another domain with another set of bases. This problem can be solved if we construct compatible orthogonal bases across different domains. However, such construction requires the knowledge of some correspondence between the domains, which is extremely difficult to obtain in most cases.
2) *Spatial Perspective:* Works in this stream directly design convolution operation based on the spatial domain, which greatly resembles the convolution on images [24], [31]. For example, [25] utilized the polynomials of functions of a graph adjacency matrix as filters to conduct graph convolution and then applied the convolution to image classification and 3-D mesh classification. Reference [32] designed depthwise separable graph convolution and applied it to perform image classification on the CIFAR data sets and [30] formulated a convolution-like operation for graph signals that can be performed in the spatial domain, where filter weights are conditioned on edge labels and dynamically generated for each specific input sample.

B. Human Action Recognition

Traditional methods directly investigate RGB videos to capture the dynamics of human actions [9], [11], [33], [34]. Klaser *et al.* [35] developed a local descriptor based on histograms

of oriented 3-D spatial–temporal gradients. Wang *et al.* [36] proposed to describe videos by dense trajectories, which samples dense points from each frame and tracks them based on displacement information from a dense optical flow. Donahue *et al.* [37] developed a recurrent convolutional architecture, in which features of the video are first extracted by CNN along the temporal dimension and then fed into recurrent sequence models. Wang *et al.* [38] proposed the temporal segmentation network, which combines a sparse temporal sampling strategy and video-level supervision to enable efficient and effective learning using the whole action video. Zhou [8] proposed a mixed convolutional tube that integrates 2-D and 3-D CNNs to generate deeper and more informative feature maps. With the fast development of depth sensors, RGB-D video-based action recognition is attracting increasingly more interest. Wang *et al.* [39] proposed the extraction and use of scene flow for action recognition from the RGB-D data. Shi and Kim [40] proposed to learn an RNN driven by privileged knowledge that is only available during training. Rahmani and Bennamoun [41] first combined view-invariant body-part representations obtained from the skeletal and depth images and then learned temporal structures as well as relations among body parts and environment objects. Hu *et al.* [13] proposed a bilinear block consisting of two linear pooling layers to pool input tensor along the modality and temporal directions separately.

The development of highly accurate depth sensors and pose estimation algorithms [42], [43] makes it easy to obtain the skeletal data related to human action (see Fig. 2), and skeleton-based human action recognition is attracting more and more interest. The work on this topic mainly follows two streams.

1) Handcrafted Features: Work belonging to this stream leverages the dynamics of joint motion by handcrafted features. For example, [17] used a covariance matrix for skeleton joint locations over time as a discriminative descriptor for a sequence and used multiple covariance matrices to encode the temporal dependence of joint locations. Wang *et al.* [15] used an actionlet ensemble obtained by data mining to represent action and designed an local occupancy pattern (LOP) feature to overcome intraclass variance caused by the imperfectness of raw data. Xia *et al.* [44] utilized the histograms of 3-D joint locations (HOJ3D) as a compact representation of postures and then reprojected the HOJ3D using the linear discriminant analysis (LDA) and clustered it into k posture visual words. Vemulapalli *et al.* [45] proposed explicitly modeling the 3-D geometric relationships between various body parts using rotations and translations in the 3-D space to represent a human skeleton and then modeled human actions as curves in a Lie group and [46] described a new 3-D saliency prediction model, in which salient 3-D space–time regions in a video are detected and segmented and, then, the saliency strength of each segment is calculated using different attributes, including motion, disparity, texture, and the predicted degree of visual discomfort experienced.

2) Deep-Learning Approaches: These process skeletal data using deep-learning methods. Du *et al.* [47] divided the human skeleton into five parts and input them into five RNN networks. Shahroudy *et al.* [48] proposed using a new RNN structure to model the long-term temporal correlation of the features for each body part. Liu *et al.* [49] extended the RNN to spatio-temporal domains to analyze the hidden sources of action-related information within the input data over both domains concurrently. Zhu *et al.* [50] proposed a fully connected deep LSTM for skeleton-based recognition. Zhang *et al.* [51] selected a set of geometric features to describe joint relations and then applied LSTM. In addition to the RNN and LSTM, CNN has also been explored to recognize human action; [22] proposed a two-stream CNN in which one stream's input is the raw coordinates and the other stream's input is motion data obtained by subtracting the joint coordinates in every two consecutive frames. Ke *et al.* [23] first transformed a skeleton sequence into three clips corresponding to three cylindrical coordinates of the skeleton sequence and then applied a deep CNN to them. Kim and Reiter [28] proposed using temporal CNNs to explicitly learn readily interpretable spatio-temporal representations for 3-D human action recognition. Si *et al.* [52] captured a high-level spatial structural information in each frame by a residual graph neural network and modeled the temporal dynamics with a composition of multiple skip-clip LSTMs. Yan *et al.* [20] first constructed a spatio-temporal human skeleton graph and then applied CNN networks. Our model is also a CNN-based model that utilizes the skeletal data. However, unlike the existing models, our convolution is performed on the edges instead of nodes.

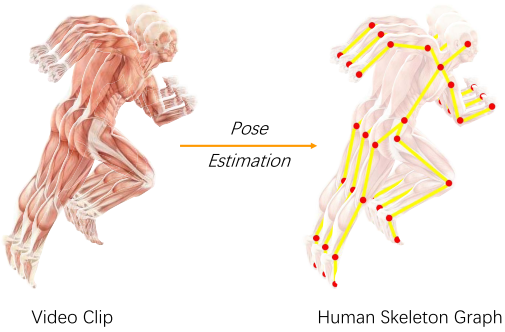


Fig. 2. Transformation from raw video clips into human skeleton graphs. Left: Raw video clip. Right: frames of the skeleton graph obtained by the pose estimation algorithm. Yellow edges in the right represent the edges (limbs), while red dots represent the joints.

III. Graph Edge Convolutional Neural Networks

In this section, we first review the classical convolution operation conducted on the images. Then, we move to our graph edge convolution and GECNNs as well as their application to skeleton-based action recognition. Finally, two hybrid models integrating edge and node convolutional networks are introduced.

A. Classical Image Convolution

To make our graph edge convolution more understandable and exhibit its connection to image convolution, we first present a reformulation of classical convolutions on image pixels. Given a grayscale image, we center the convolution kernel at pixel x_{ij}, where i and j denote the row index and

column index, respectively. The convolution output can be derived as follows:

$$x_{ij}^{\text{out}} = \sum_{x_{mn} \in \mathbf{N}(x_{ij})} x_{mn} w(l(x_{mn})) \tag{1}$$

where x_{ij}^{out} is the output of this convolution operation, $\mathbf{N}(x_{ij})$ represents the set of neighboring pixels of x_{ij}, and x_{mn} denotes the pixels in $\mathbf{N}(x_{ij})$. In practice, for a grayscale image, $\mathbf{N}(x_{ij})$ is composed of eight neighboring pixels x_{ij} as well as x_{ij} itself, which is the actual receptive field of this convolution. l is a labeling function to assign an order to each individual element in $\mathbf{N}(x_{ij})$. For example, given a 3×3 convolutional kernel on a grayscale image, $l(\cdot)$ will assign $\{1, 2, \cdots, 9\}$ to nine pixels in $\mathbf{N}(\cdot)$ from left to right and top to bottom. The weight function w will give each pixel a weight $w(l(\cdot))$ according to its order. Pixels are then multiplied by their corresponding weight and get summed to be the convolution output at the center pixel. A similar convolution operation can be applied for RGB color images. Since there are three different channels in an RGB image, the value of each pixel is a 3-D feature vector, and its corresponding weight becomes a 3-D weight vector as well. The multiplication between pixels and weights is extended to the inner product.

CNNs have achieved impressive performances in various scenarios, e.g., image classification and object detection. However, it is not straightforward to apply a pixel convolution to graph data, because graph data do not have the gridded array structure that images, video, and signal data do. Each pixel in gridded images will have the same number of neighbors and the same relationships with respect to a neighbor in a given direction. Non-gridded graphs do not have these limitations. A non-gridded graph can vary in the number of neighbors from node (edge) to node (edge), and there is not necessarily a geometrical interpretation for any given connection between two nodes (edges).

B. Graph Edge Convolution

We represent a graph by $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, in which $\mathbf{V} = \{v_i | i = 1, \ldots, N\}$ is a node set with N elements and $\mathbf{E} = \{e_{ij} | v_i, v_j \textit{ are connected }\}$ is an edge set containing all edges of $\mathbf{G}$. Several concepts have to be introduced to facilitate the explanation of graph convolution.

1) *Path Between Two Edges:* A path is a set of distinct nodes and edges connecting two edges in a graph [see Fig. 3(a)]. More than one path may exist between two edges.
2) *Length of Path Between Two Edges:* Given a pair of edges, different paths contain different numbers of nodes and edges may exist. For a certain path, the number of contained nodes is defined as its length. Paths with different lengths are shown in Fig. 3(b).
3) *Shortest Path Between Two Edges:* Given a pair of edges, among all paths connecting the two edges, the path with the smallest length is defined as the shortest path between the two edges. An illustration of the shortest path is given in Fig. 3(b).

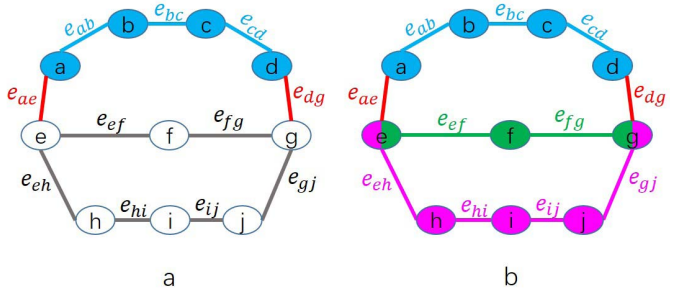


Fig. 3. (a) Blue parts (nodes a–d and edges e_{ab}, e_{bc}, and e_{cd}) form a path between e_{ae} and e_{dg} with a length of 4. (b) Parts in blue, green, and mauve depict three different paths between e_{ae} and e_{dg}. Node e and g have two colors as they are shared by two paths. The lengths of the blue, green, and mauve path are 4, 3, and 5, respectively. The green path is the shortest path between e_{ae} and e_{dg}. Thus, the length of the green path, which is 3, is the distance between e_{ae} and e_{dg}.

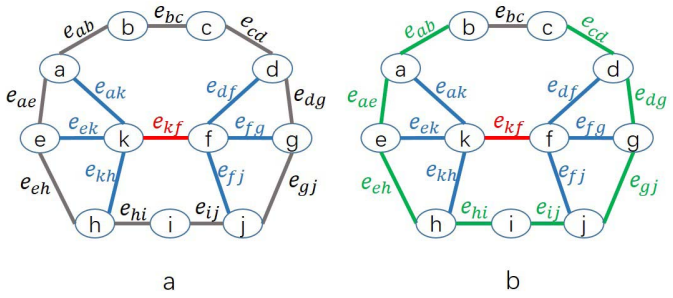


Fig. 4. (a) 1-hop neighbors of the edge e_{kf}. (b) Both 1- and 2-hop neighbors of edge e_{kf}. In (a) and (b), the root edge is e_{kf}, and the neighboring edges are colored blue or green. Blue edges in (a) are neighboring edges when $R = 1$. The blue and green edges in (b) are neighboring edges when $R = 2$, while the green ones are edges with distances of 2 from root edge e_{kf}.

4) *Distance Between Two Edges:* Given a pair of edges, the length of the shortest path is defined as the distance between the two edges, denoted as $\mathbf{D}(\cdot, \cdot)$. An illustration of the distance between a specific pair of edges is given in Fig. 3(b).

To conduct convolution, we need a receptive field, which is also called a neighborhood in the following. Here, we define the neighborhood of a certain edge e_{pq} as $\mathbf{N}(e_{pq}) = \{e_{kl} \mid e_{kl} \in \mathbf{E} \text{ and } \mathbf{D}(e_{pq}, e_{kl}) \leqslant R\}$, where $\mathbf{D}(e_{pq}, e_{kl})$ is the distance between root edge e_{pq} and edge e_{kl} and R is an integer denoting the maximal distance between the root edge and neighboring edges. Thus, we can also express the neighborhood of a root edge as edges within a certain distance R from the root edge. In our convolution, we set R as 1 [see Fig. 4(a)], and the resulting neighborhood contains only the edges that are directly connected to the root edge by one node, i.e., the neighborhood of an edge e_{pq} is $\mathbf{N}(e_{pq}) = \{e_{kl} \mid e_{kl} \in \mathbf{E} \textit{ and } \mathbf{D}(e_{pq}, e_{kl}) \leqslant 1\}$.

The convolution at a certain edge e_{pq} is a weighted summation on all the neighboring edges. Formally, the output of convolution at edge e_{pq} can be written as (see Fig. 1)

$$e_{pq}^{\text{out}} = \sum_{e_{kl} \in \mathbf{N}(e_{pq})} e_{kl} \cdot w(l(e_{kl})) \tag{2}$$

where e_{pq}^{out} is the output of the convolution at edge e_{pq}, e_{kl} represents all neighboring edges being summed, and $w(l(e_{kl}))$ are the corresponding weights.

Neighboring edges in a graph do not have an implicit spatial order, such as pixels in an image. Thus, to assign weights to neighboring edges, we have to first sort them and then match each edge with a corresponding weight in order and use labeling function l to achieve this. For each edge e_{kl} in the neighborhood, l will assign a labeling value $l(e_{kl})$ to it, denoting its order, and a weight will be assigned according to $l(e_{kl})$. Unlike in images, we cannot prepare a fixed number of weights for edges in a graph, as the number of neighboring edges may vary. Therefore, instead of giving each neighboring edge a unique labeling value, function l will map neighboring edges to a fixed number of subsets, and edges in the same subset will have the same labeling value, i.e., l: $\mathbf{N}(e_{pq}) \rightarrow \{1, \ldots, K\}$. Every edge in the neighborhood will be labeled with an integer ranging from 1 to K as its order. Thus, we see that even if the number of neighboring edges is not fixed, we can always assign K weight values to them. Actually, w in (2) is a weight function that assigns weights according to the labeling values. After multiplied by the weights, the summation is conducted for all the neighboring edges, after which the convolution operation terminates.

Similar to the image convolution mentioned earlier, by replacing the edge feature and weight with the edge feature vector and weight vector, we can easily extend our model to graphs with multiple channels.

To balance the contribution of edges with different weights, we add a normalizing term to (2)

$$e_{pq} = \sum_{e_{kl} \in \mathbf{N}(e_{pq})} \frac{1}{Z_{pq}(e_{kl})} e_{kl} \cdot w(l(e_{kl})). \tag{3}$$

$Z_{pq}(e_{kl})$ is the number of neighboring edges with the same labeling value as e_{kl} in edge e_{pq}'s neighborhood, that is

$$Z_{pq}(e_{kl}) = |\{e_{mn} | e_{mn} \in \mathbf{N}(e_{pq}) \quad \text{and } l(e_{mn}) = l(e_{kl})\}. \tag{4}$$

Thus, we can see that when the neighboring edges are not divided evenly into K subsets, $(1/Z_{pq}(e_{kl}))$ is the term used to balance the contribution of neighboring edges with different labeling values. Now, the convolution operation of edges has been elucidated, and we can use it to build our GECNNs (see Fig. 5).

C. Graph Edge Convolutional Neural Networks

Our edge convolution can be directly conducted on a single skeleton graph, while skeletal data are stored as sequences of skeleton graphs. Skeleton graphs in different frames have to be processed concurrently to capture the temporal dynamics. Thus, we have to redefine the neighborhood of an edge to incorporate edges in different frames to perform our convolution in both the spatial and temporal domains.

We denote a certain edge e_{pq} in frame $\tau 1$ as $e_{pq}^{\tau 1}$, and the neighboring edges of $e_{pq}^{\tau 1}$ are defined from two aspects: 1) in frame $\tau 1$, edges connected to $e_{pq}^{\tau 1}$ via only one node are its spatial neighbors, which is just the definition of a neighborhood mentioned earlier and 2) in another frame $\tau 2$, if $\tau 2$ is within a certain distance from $\tau 1$, i.e., $|\tau 2 - \tau 1| < K_t$, then the spatial neighbors of $e_{pq}^{\tau 2}$ are also regarded as neighbors of $e_{pq}^{\tau 1}$ and are called temporal neighbors of $e_{pq}^{\tau 1}$, that is

$$\mathbf{N}\left(e_{pq}^{\tau 1}\right) = \left\{e_{kl}^{\tau 2} | e_{kl}^{\tau 2} \in \mathbf{E}, |\tau 2 - \tau 1| \leqslant K_t \text{ and } \mathbf{D}\left(e_{kl}^{\tau 2}, e_{pq}^{\tau 2}\right) \leqslant 1\right\}.$$

Here, $\tau 1$ and $\tau 2$ denote two different frames. K_t is an integer called the temporal kernel size, which is used to restrict the temporal neighbors of $e_{pq}^{\tau 1}$ such that they cannot be more than K_t frames away. The constraint $\mathbf{D}(e_{kl}^{\tau 2}, e_{pq}^{\tau 2}) \leqslant 1$ is used to define spatial neighbors.

The labeling function defined in Section III-B is applicable to spatial neighbors in a single frame. Considering the temporal neighbors, the labeling function is rewritten as

$$l'\left(e_{kl}^{\tau 2}\right) = l\left(e_{kl}^{\tau 2}\right) + (\tau 2 - \tau 1 + K_t) \times K \tag{5}$$

where l' is the rewritten labeling function, l is the labeling function in a single frame for spatial neighbors, K_t is the temporal kernel size, and K is the number of subsets divided by the labeling function l mentioned earlier. Moreover, K_t is added to $\tau 1 - \tau 2$ to ensure that $(\tau 2 - \tau 1 + K_t)$ is nonnegative, and K is multiplied at the end to ensure that the labeling values for temporal neighbors are not the same as those for spatial neighbors.

As our model aims to convolve edges instead of nodes and the original skeletal data are the coordinates of joints (nodes), we have to transform the skeletal data from the node form into edge form. The data of each limb (edge) are calculated from the coordinates of the two joints on the limb's end. For each limb, we first obtain the coordinates of the limb's center by averaging the coordinates of the joints on both ends and then subtract one joint's coordinates from those of the other to obtain an orientation vector representing the length and orientation of the limb between the two joints. Denoting the limb (edge) as e_{pq} and the joints (nodes) on its two ends as n_p and n_q, we can formulate the center coordinates of e_{pq} as follows:

$$x_c(e_{pq}) = \frac{1}{2} \times (x(n_p) + x(n_q)) \tag{6}$$

$$y_c(e_{pq}) = \frac{1}{2} \times (y(n_p) + y(n_q)) \tag{7}$$

$$z_c(e_{pq}) = \frac{1}{2} \times (z(n_p) + z(n_q)) \tag{8}$$

where $x_c(e_{pq})$, $y_c(e_{pq})$, and $z_c(e_{pq})$ denote the 3-D coordinates of the center of limb e_{pq} and $x(n_p)$, $y(n_p)$, and $z(n_p)$ are the coordinates of node n_p; a similar format applies to node n_q. The orientation vector, whose length and orientation represent the length and orientation of limb e_{pq}, can be formulated as

$$Ori(e_{pq}) = \left(x(n_p) - x(n_q), y(n_p) - y(n_q), z(n_p) - z(n_q)\right). \tag{9}$$

Motions of human body parts can be roughly categorized as concentric and eccentric. Thus, we choose labeling function l for spatial neighbors as spatial configuration labeling. Specifically, for convolution at edge e_{pq}, this labeling function divides neighboring edges into three subsets: 1) edges that are closer to the center of gravity than e_{pq}; 2) edges that have the

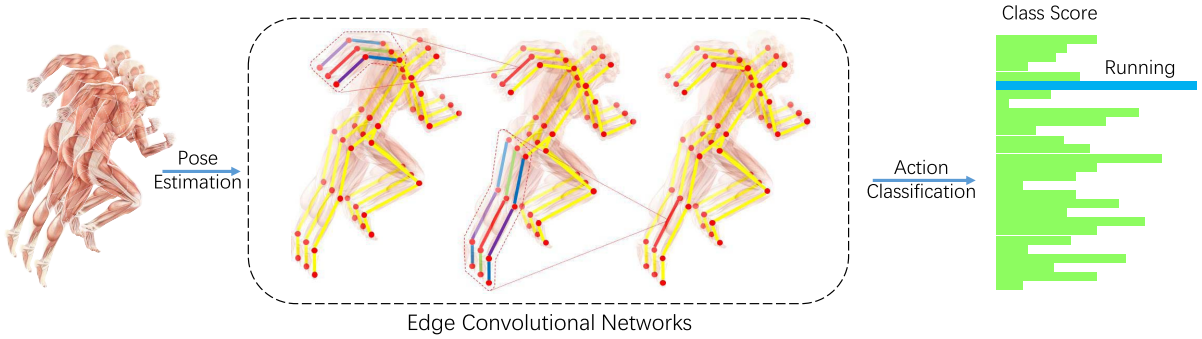


Fig. 5. Illustration of the edge convolution system. Middle: process of edge convolution. The edges to be convolved are colored, and different colors indicate different labeling values.

same distance from the center of gravity as e_{pq}, and 3) edges that are farther away from the center of gravity than e_{pq}, that is

$$l(e_{kl}) = \begin{cases} 0, & \text{if } \mathbf{d}(e_{kl}, G_c) = \mathbf{d}(e_{pq}, G_c) \\ 1, & \text{if } \mathbf{d}(e_{kl}, G_c) < \mathbf{d}(e_{pq}, G_c) \\ 2, & \text{if } \mathbf{d}(e_{kl}, G_c) > \mathbf{d}(e_{pq}, G_c). \end{cases} \tag{10}$$

In (10), G_c is the center of gravity of the human body, calculated by averaging the coordinates of all body parts, and $\mathbf{d}(e_{kl}, G_c)$ denotes the distance from edge e_{kl} to G_c. Having the labeling function l for spatial neighbors, the labeling function for temporal neighbors can be formulated according to (5).

D. Combining Graph Edge and Node Convolutions

The node and edge convolutional networks are used to extract features from different perspectives. Both edge convolution and node convolution have their unique advantages and thus are complementary to each other. As we mentioned earlier, an edge convolutional network leverages the dynamics of limbs, while a node convolutional network leverages the dynamics of joints.

Thus, we consider to design hybrid models integrating both two networks to utilize the advantages of both edge and node convolution. Specifically, we have two hybrid networks, including one using a shared fully connected layer and one equipped with two shared convolutional layers.

1) Sequence-Level Hybrid Model: Either edge- or node-convolutional network extracts one set of features for each input sequence, which is used for classification. As this is a representation for an entire sequence, we call it sequence-level features. Our first hybrid model directly integrates the two sets of sequence-level features from edge- and node-convolutional networks with a shared fully connected layer. Given the input sequence s_{in}, the output of the convolutional layer in the edge convolution stream can be formulated as

$$h^e_{\text{seq}} = f^e_{\text{conv}}(s_{\text{in}}) \tag{11}$$

in which f^e_{conv} represents the layers of edge convolution and h^e_{seq} is the output from the last layer of the edge convolution stream. Similarly, the output of the final layer in the node convolution stream can be formulated as

$$h^n_{\text{seq}} = f^n_{\text{conv}}(s_{\text{in}}). \tag{12}$$

We obtain two sets of features h^e_{seq} and h^n_{seq} from edge and node convolution streams. They are separately processed first by a global pooling to obtain a representation for the whole sequence and then fed into fully connected layers for two sets of class scores. We then concatenate them into a single tensor to obtain the input for the last fully connected layer, that is

$$I^{fc}_{\text{seq}} = f_{\text{pool}}\left(h^e_{\text{seq}}\right) \oplus f_{\text{pool}}\left(h^n_{\text{seq}}\right) \tag{13}$$

in which I^{fc}_{seq} denotes the input to the fully connected layer in the sequence-level hybrid model (SLHM), f_{pool} is the global pooling layer, and $\oplus$ is a concatenation operator. The final output of the whole model is

$$O_{\text{seq}} = f_{fc}\left(I^{fc}_{\text{seq}}\right) \tag{14}$$

in which f_{fc} denotes the fully connected layer and O_{seq} is the final output of the SLHM.

By concatenating the outputs from the edge and node convolution streams, the features extracted from both the edge and node convolutional networks contribute to the final classification result, i.e., the dynamics of both joints and limbs are leveraged in our classification.

In this model, the input to the shared layer is the features of the whole skeleton sequence, so we call the first hybrid model an SLHM.

2) Body-Part-Level Hybrid Model: In the SLHM, two sets of sequence-level features get linearly combined by a shared fully connected layer after being separately extracted by node- and edge-convolutional networks. Features directly extracted by the edge (node) convolutional layers are not sequence-level, but they are attached to each edge (node), and the sequence-level features are obtained by applying pooling layers to them. As the nodes and edges correspond to human body parts (joints and limbs), we call these body-part-level features. Besides directly merging the sequence-level features, we also want to merge the features in the body-part level to get the node- and edge-convolution supplement each other in a subtler way. Thus, we design another hybrid model, with the most distinctive part being the shared convolutional layer.

In a shared convolutional layer, our convolution operation is neither an edge convolution nor a node convolution but rather

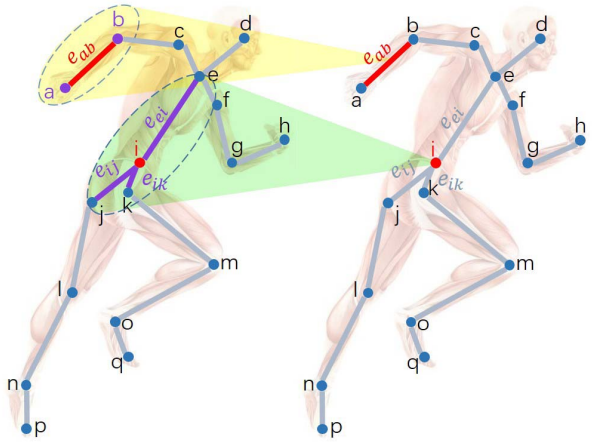


Fig. 6. Illustration of the convolution in the shared convolutional layer. A convolution at edge e_{ab} and a convolution at node i are shown. For the convolution at e_{ab} (with the light-yellow background), nodes a and b and edge e_{ab} are involved, and their weight summation is assigned to the kernel center e_{ab} in the next layer. For the convolution at node i (with the light-green background), edges e_{ei}, e_{ij}, and e_{ik} and node i are involved, and their weight summation is assigned to the kernel center node i in the next layer. In both convolutions, the center of the convolution kernel is colored red, just like e_{ab} and node i, while the other parts involved are colored purple.

a combination of both. As mentioned earlier, we conduct convolution on a human skeleton graph. When we perform edge convolution, the data are information regarding the limbs and attached to the edges, and when we perform node convolution, the data are information regarding the joints and attached to the nodes. However, in a shared convolutional layer, both nodes and edges have information attached to them. Thus, when conducting convolution, the information of the edges can be passed to the nodes, and vice versa. By this kind of convolution, the dynamics of limbs and joints can supplement each other at the body-part level in the shared convolutional layer. An illustration of the convolution operation in the shared convolutional layer can be found in Fig. 6.

Similar to the SLHM, the first part of the body-part-level hybrid model (BPLHM) involves two separated streams, including an edge and a node convolutional network, and the outputs are two sets of features denoted as h^e_{body} and h^n_{body}, which can be formulated as follows:

$$h^e_{\text{body}} = f^e_{\text{conv}}(s_{\text{in}}) \tag{15}$$

$$h^n_{\text{body}} = f^n_{\text{conv}}(s_{\text{in}}) \tag{16}$$

in which f^n_{conv}, f^e_{conv}, and s_{in} have exactly the same meanings as for the SLHM. Different from the SLHM, the subscripts are changed from *seq* to *body* to indicate that they are parts of the BPLHM. More specifically, the feature set extracted by the edge convolutional network contains feature vectors for each limb in each frame, and the feature set extracted by the node convolutional network contains feature vectors for each joint in each frame. After obtaining these two feature sets, we construct a new human skeleton graph and then fill in each edge and node with the corresponding feature vectors. In this new skeleton graph, both edges and nodes are filled with feature vectors, which are the high-level features of limbs and joints containing the dynamic information extracted by the two separate convolutional networks. Then, the shared convolutional layer is applied. The input can be formulated as

$$I^{s-\text{conv}}_{\text{body}} = h^e_{\text{body}} \oplus h^n_{\text{body}} \tag{17}$$

in which $I^{\text{shared}-\text{conv}}_{\text{body}}$ denotes the input to the shared convolutional layer in the BPLHM. In the shared convolutional layer, the high-level features of limbs and joints communicate with each other and are furnished by each other. The output of the shared convolutional layer is also the feature vectors of the limbs and joints, just like the output that we obtain from the two separate streams. The difference is that in the two separate streams, the features of joints and limbs are separately extracted, while in our shared convolutional layer, each time we perform the convolution operation on the human body graph, the information of the joints has the potential to flow to the neighboring limbs. Thus, the features of limbs are informed and are trimmed by the dynamics of joints, and vice versa, making the final features more representative. We can formally write this as

$$h^{s-\text{conv}}_{\text{body}} = f_{s-\text{conv}}\left(I^{s-\text{conv}}_{\text{body}}\right) \tag{18}$$

in which $h^{s-\text{conv}}_{\text{body}}$ represents the output from a shared convolutional layer in the body-part-level hybrid model and f_{s-conv} denotes the shared convolutional layer. After utilizing the two layers of shared convolution, we apply a global pooling layer and a fully connected layer to obtain the final classification result as in the SLHM

$$h^{\text{pool}}_{\text{body}} = f_{\text{pool}}\left(I^{s-\text{conv}}_{\text{body}}\right) \tag{19}$$

$$I^{fc}_{\text{body}} = h^{\text{pool}}_{\text{body}} \tag{20}$$

where $h^{\text{pool}}_{\text{body}}$ is the output from the global pooling layer and I^{fc}_{body} is the input to the final fully connected layer. The final output from the fully connected layer can be formulated as

$$O_{\text{body}} = f_{fc}\left(h^{\text{pool}}_{\text{body}}\right) \tag{21}$$

in which O_{body} is the output of the whole body-part-level hybrid model and f_{fc} denotes the fully connected layer.

Overall, the main difference between our SLHM and BPLHM is the final part that integrates node- and edge-convolutional networks. SLHM utilizes a fully connected layer to merge two streams, while BPLHM is more complex and requires two extra shared convolutional layers. SLHM integrates two flows in a rather straightforward way, but it runs faster as well. In contrast, BPLHM is more complex and can extract higher level features based on features already extracted by edge- and node-convolutional networks. Accordingly, higher complexity makes BPLHM cost more computational time compared to SLHM.

In our hybrid models, the high-level features extracted from both edge convolution and node convolution can corroborate each other. As we mentioned earlier, in several situations, the movements of joints are too subtle to capture, but the movements of limbs are always obvious. In these cases, the hybrid models can utilize the power of

edge convolution to classify human actions. However, among some classes of human action, the movements of limbs may be too similar to distinguish from each other. In these cases, the dynamics of joints may help the hybrid model to perform classification, that is, the dynamics of both joints and limbs are leveraged and fused together to recognize human action. Thus, the skeletal data are fully utilized in our hybrid models.

In our second hybrid model, the features that are fed into the shared layers are the features of each specific joint and limb, which are body-part-level features. Thus, we name this model the body-part-level hybrid model.

E. Complexity Analysis

The adjacency matrix of a skeleton graph with V nodes and M edges is of size $V \times V$ for nodes and $M \times M$ for edges. The graph convolution on sequence is implemented both in the spatial and temporal dimensions (see Section IV-C). The spatial graph convolution is conducted by multiplying edge feature vectors with a trainable matrix of shape $M \times M$ masked by the adjacency matrix. The trainable matrix is dismantled into K matrices to model different dependences on K neighboring edge subsets. Thus, the number of trainable parameters for spatial graph convolution is $K \times M^2$. The temporal convolution is implemented by a standard 2-D convolution with the kernel size of $1 \times K_t$. Suppose that the input tensor is of C_{in} channels and the output tensor is of C_{out} channels, and then, the number of trainable parameters included in a temporal convolution is $K_t \times C_{\text{in}} \times C_{\text{out}}$. Thus, the number of trainable parameters contained in an edge convolutional layer is

$$K_t \times C_{\text{in}} \times C_{\text{out}} + K \times M^2. \tag{22}$$

Denoting the length of a skeleton sequence as T, the computational complexity of spatial graph convolution measured in flops can be formulated as follows:

$$M^2 \times C_{\text{in}} \times T + M \times C_{\text{in}} \times T \times (M - 1) \tag{23}$$

where $M^2 \times C_{\text{in}} \times T$ and $M \times C_{\text{in}} \times T \times (M - 1)$ denote the number of multiplications and additions in spatial graph convolution, and the computational complexity for temporal convolution can be written as

$$M \times C_{\text{out}} \times T \times (2 \times K_t \times C_{\text{in}} - 1) \tag{24}$$

where $M \times C_{\text{out}} \times T \times K_t \times C_{\text{in}}$ is the number of multiplications and $M \times C_{\text{out}} \times T \times (K_t \times C_{\text{in}} - 1)$ is the number of multiplication and additions for temporal convolution.

Node convolution differs from edge convolution in which elements participating in the convolution are V joints instead of M edges. Thus, (22)–(24) can be rewritten by replacing M with V to represent the complexity for a node convolutional layer. In our skeleton graph (a tree in graph theory), $M = V - 1$ holds, and thus, our edge convolutional network is more efficient than node convolutional network in terms of both parameter and computational complexity.

Extra computational costs are brought about by integration layers in hybrid models. For the SLHM with a fully connected integration layer, suppose that the final output tensor is of O channels and number of classes is N_c, and then, the number of trainable parameters of the fully connected layer can be formulated as $N_c \times O$. For the shared convolutional layer in the BPLHM, the elements participating in the shared convolution are $M + V$, and thus, the number of parameters in a single shared convolutional layer to train is $K_t \times C_{\text{in}} \times C_{\text{out}} + K \times (V + M)^2$. Besides, the computational complexity of the extra fully layer can be formulated as

$$N_c \times M \times O + N_c \times M \times (O - 1) \tag{25}$$

where $N_c \times M \times O$ and $N_c \times M \times (O - 1)$ are the number of multiplications and additive operations required by the fully connected layers, respectively, and the flops for the shared convolutional layer can be obtained by simply replacing M in (23) and (24) with $V + M$. The additional complexity brought about by hybrid models will not cause many difficulties during training. As shown in Section IV-E, the convergence performances of hybrid models are actually better than the node convolutional model.

IV. Experiments

First, we explore the influence of temporal kernel size on action recognition on the Kinetics data set and choose a proper temporal kernel size. Then, we evaluate our model's performance on two data sets: the Kinetics human action data set (Kinetics) [53], which is currently the largest unconstrained action recognition data set, and the NTU-RGB+D data set [48], the largest in-house captured human action recognition data set. Then, we compare our performance with that of the previous state-of-the-art algorithms. On Kinetics, we compare our results with the feature encoding approach [16], Deep LSTM [48], Temporal ConvNet [28], and ST-GCN [20]. On NTU-RGB+D, we make a comparison with the Lie group model [45], H-RNN [47], Deep LSTM [48], PA-LSTM [48], ST-LSTM + TS [49], Temporal Conv [28], C-CNN + MTLN [23], and ST-GCN [20]. To further analyze the improvement brought by our edge convolution, we visualize the classification accuracies for each class on the NTU-RGB+D data set and compare the accuracy between the node convolution model and the edge convolution model on some representative classes.

A. Data Sets and Settings

1) Kinetics: Kinetics contains about 300 000 videos retrieved from YouTube, with all samples falling into 400 classes. To obtain the skeletal data, [20] used the OpenPose toolbox [43] to extract 2-D coordinates of joints in a video. Each located joint has the form X, Y, and C, which are 2-D coordinates and confidence score, and each body is represented by 18 joints (see Fig. 7).

In Kinetics, each limb is represented by five values, including 2-D coordinates, confidence score, and an orientation vector. The 2-D coordinates and confidence score are obtained by averaging the corresponding values of the joints on the limb's two ends, and the orientation vector is obtained by subtracting the coordinates of the joints on the limb's ends to denote the length and orientation of the limb.

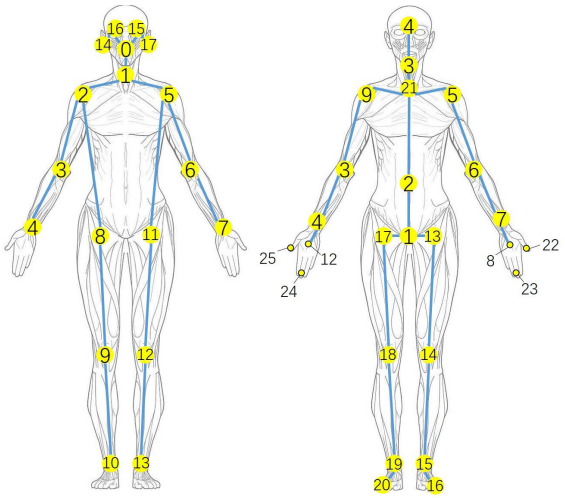


Fig. 7. Illustration of the human skeleton graphs in two data sets. Left: skeleton graph of the Kinetics data set. Right: skeleton graph of the NTU-RGB+D data set.

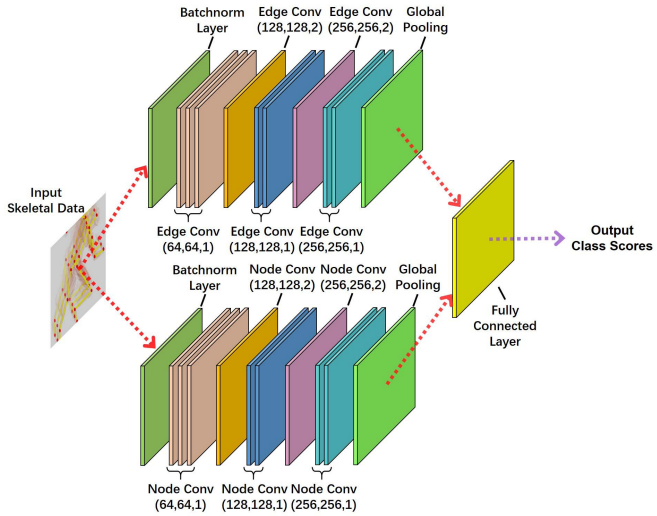


Fig. 8. Network structure of the SLHM. A single skeletal sequence is first input into two separated flows. Then, the output of the two flows is concatenated, and the two flows become one.

The whole skeletal data set retrieved from Kinetics contains 266 440 samples, with 246 534 samples for training and 19 906 samples for testing. On Kinetics, we report both the top-1 and top-5 accuracies.

2) NTU-RGB+D: NTU-RGB+D is the largest data set with 3-D joint annotations captured in [48] using three Microsoft Kinetic v.2 cameras concurrently. It contains 56 880 action samples in the form of RGB videos, depth map sequences, 3-D skeletal data, and infrared videos. All samples in this data set fall into 60 action classes. Among the multiple modalities, we only use the skeletal version, in which each human body is represented by 3-D coordinates of 25 joints (see Fig. 7).

Similar to Kinetics, each limb in this data set is represented by a 6-D vector containing 3-D coordinates of its center and an orientation vector.

The author of this data set recommends dividing the data in two ways, including cross-subject (X-sub) and cross-view (X-view). X-sub splits the data set into 40 320 training samples performed by one group of actors and 16 560 testing samples performed by the other group of actors, while X-sub 37 920 samples captured by one set of cameras set for training and 18 960 samples captured by the other camera set for testing. On the NTU-RGB+D data set, we report only the top-1 accuracy.

B. Pipeline

1) Graph Edge Convolutional Neural Network: Data input to our model are first normalized by a batch normalization layer. Next, center coordinates and orientations of limbs are calculated. Then, the data are fed into a nine-layer GECNN. The first three layers have 64 channels for output, the next three layers have 128 channels for output, and the last three layers have 256 channels for output. In the fourth and seventh layers, the stride is set to 2 for pooling. The final output tensor from convolutional layers is then fed into a global pooling layer. The global pooling layer is used to obtain a 256-D feature vector for each video sequence. After the global pooling layer, the output tensor is fed into a fully connected layer to obtain the class scores for each video sequence.

2) Sequence-Level Hybrid Model: As mentioned earlier, the first part of the SLHM involves two separated streams, including an edge convolutional stream and a node convolutional stream. The edge convolutional stream includes a normalization layer, followed by nine edge convolutional layers and a global pooling layer. The node convolution stream has the same structure, including a normalization layer, nine node convolutional layers, and a global pooling layer.

After the global pooling layer in the first part, each sequence is represented as a 256-D feature vector. In the second part of the SLHM, we concatenate the two 256-D output tensors from the global pooling layer into a single tensor and then input it into a fully connected layer to obtain the final classification result, which is the class scores for each sequence, indicating the probabilities of classifying a sequence into each class (see Fig. 8).

3) Body-Part-Level Hybrid Model: Body-part-level hybrid model is also divided into two parts. The first part is the same as the first part of the SLHM, except that the global pooling layer is removed. The second part of the body-part-level hybrid model includes four layers. In the second part, the output tensors of the two former streams are first input into two layers of shared convolutional layers. Then, global pooling is conducted on the output tensors of the shared convolutional network. Finally, we feed the result into a fully connected layer to obtain the final classification result (see Fig. 9).

C. Implementation of Edge Convolution

The implementing of edge convolution actually consists of two stages, including a temporal convolution and a spatial convolution. Each input sequence can be represented as a tensor of the form (C, E, and T), in which C denotes the multiple channels of the feature vector, E is the number of edges, and T is the length of the sequence. We first perform

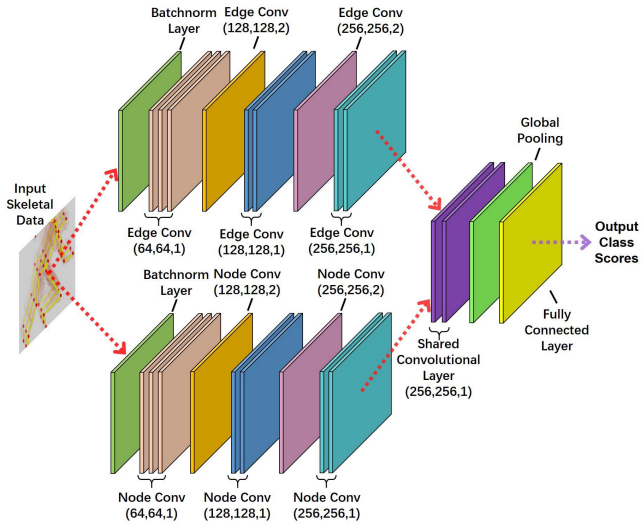


Fig. 9. Network structure of the body-part-level hybrid model.

TABLE I

ABLATION STUDY ON THE KINETICS DATA SETS. BOTH TOP-1 AND TOP-5 ACCURACIES ARE REPORTED

Temporal kernel size	top-1	top-5
3	29.30%	51.71%
5	30.58%	52.84%
7	31.17%	53.95%
9	31.41%	53.90%
11	31.49%	54.38%
13	31.43%	53.86%

traditional 2-D convolution with a kernel size of $1 \times K_t$ on the third dimension to conduct temporal convolution. Then, we multiply the resulting tensor by the adjacency matrix of the human skeleton graph to conduct spatial convolution. The operation of conducting spatial convolution by multiplying the tensor by the adjacency matrix is inspired by [54], in which the concept of graph shift corresponds to spatial convolution on the graphs.

D. Hyperparameter Analysis

To determine to what extent incorporating temporal neighbors will improve the performance of action recognition, we choose six different temporal sizes for our edge convolution model and compare the results on the Kinetics data set.

From Table I, we see that when the temporal kernel size rises from 3 to 9, the top-1 accuracy increases significantly, while further enlargement to 11 and 13 does not yield much improvement. This demonstrates the necessity to incorporate a sufficient number of consecutive frames in a sequence to capture motion when conducting a convolution. However, incorporating too many frames is not promising and will require more computational resources. Thus, based on the comparison, we set the temporal kernel size to 9 to achieve both satisfying accuracy and a lower computational resource cost.

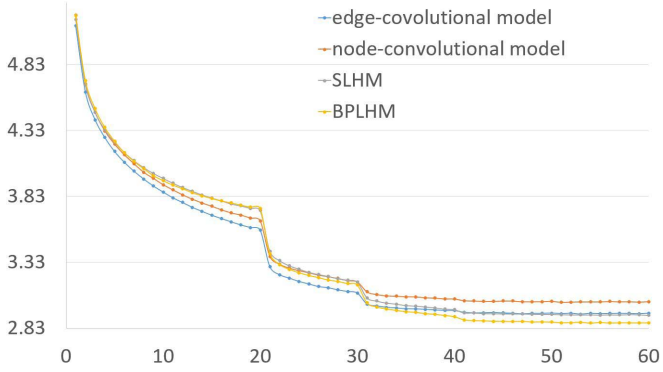


Fig. 10. Loss decreasing curves of edge- and node-convolutional networks over 60 epochs on the Kinetics data set.

E. Training Performance

In this section, we show the convergence performance of our models and node convolutional model. As shown in Fig. 10, our edge convolutional model converges a little faster than the node convolutional model and achieves a lower training loss eventually. There are two abrupt drops of loss in the curve, which is where we shrink the learning rate. We also compare the training performance of our hybrid models with that of node convolutional model. In Fig. 10, we can see that although our hybrid models need to train more parameters, they still converge faster than the sole node convolutional model. In the experiment, we do not use pre-trained edge- or node-convolutional networks but train the entire hybrid network concurrently.

Similar results on training performance can also be found on the NTU-RGB+D data set with both cross-subject and cross-view protocols.

F. Experiment Results

1) Kinetics: We compare our GECNN model, SLHM, and BPLHM with four other characteristic skeleton-based models: feature encoding approach [16], Deep LSTM [48], Temporal ConvNet [28], and ST-GCN [20]. In Table II, we see that our model achieves superior performance compared to that of the other approaches.

On this data set, our edge convolution network achieves superior classification accuracy over the previous state-of-the-art models by 0.7% in top-1 accuracy and 1.1% in top-5 accuracy. Our SLHM further improves the performance by 0.4% in top-1 accuracy and 0.3% in top-5 accuracy. The body-part-level hybrid model performs the best, with 2.7% higher top-1 accuracy and 3.4% higher top-5 accuracy.

2) NTU-RGB+D: On this data set, we compare our edge convolution model with eight other state-of-the-art models: Lie group [45], H-RNN [47], Deep LSTM [48], PA-LSTM [48], ST-LSTM + TS [49], Temporal Conv [28], GCA-LSTM [55], D-Pose T-Conv [12], C-CNN + MTLN [23], Zolfaghari *et al.* [56], ST-GCN [20], Pose-attention [57], and Deep Bilinear [13]. In Table III, we can see the result and the great improvement compared with the other models. On this data set, only the top-1 accuracy is reported.

TABLE II

COMPARISON BETWEEN DIFFERENT MODELS ON THE KINETIC DATA SETS. BOTH TOP-1 AND TOP-5 ACCURACIES ARE REPORTED

Model	top-1	top-5
Feature Enc. [16]	14.9%	25.8%
Deep LSTM [48]	16.4%	35.3%
Temporal ConvNet [28]	20.3%	40.0%
ST-GCN [20]	30.7%	52.8%
GECNN	31.4%	53.9%
SLHM	31.8%	54.2%
BPLHM	**33.4**%	**56.2**%

TABLE III

COMPARISON BETWEEN DIFFERENT MODELS ON THE NTU-RGB+D DATA SET. TOP-1 ACCURACY IS REPORTED ON TWO SEPARATE PARTS OF THE DATA SET

Model	Data	X-Sub	X-View
Lie group [45]	SKL	50.1%	52.8%
H-RNN [47]	SKL	59.1%	64.0%
Deep LSTM [48]	SKL	60.7%	67.3%
PA-LSTM [48]	SKL	62.9%	70.3%
ST-LSTM+TS [49]	SKL	69.2%	77.7%
Temporal Conv [28]	SKL	74.3%	83.1%
GCA-LSTM[55]	SKL	74.4%	82.8%
D-Pose T-Conv[12]	SKL	76.8%	84.9%
C-CNN+MTLN[23]	SKL	79.6%	84.8%
Zolfaghari *et al.* [56]	RGB+SKL	80.8%	-
ST-GCN[20]	SKL	81.5%	88.3%
Pose-attention[57]	RGB+SKL	82.5%	88.6%
Deep Bilinear[13]	RGB+SKL	83.0%	87.1%
GECNN	SKL	84.0%	89.4%
SLHM	SKL	84.7%	89.7%
BPLHM	SKL	**85.4**%	**91.1**%

With the X-sub protocol on NTU-RGB+D, our edge convolution model outperforms the previous state-of-the-art model by 2.5%. Moreover, our SLHM and BPLHM further improve the accuracy by 0.7% and 1.4%, respectively. Overall, the BPLHM performs the best, achieving 3.9% higher accuracy than that of the previous state-of-the-art model.

With the X-View protocol, our edge convolution network outperforms the previous state-of-the-art model by 1.1%. Moreover, our SLHM and BPLHM further improve the accuracy by 0.3% and 1.7%, respectively. The BPLHM still performs the best, with a 2.8% higher accuracy than the previous state-of-the-art model.

Overall, on all data sets and in all different settings, our GECNN model outperforms the previous state-of-the-art models. This result preliminarily corroborates our conjecture that the movements of human limbs indeed deserve our attention, and on average, recognizing human action by analyzing bone dynamics is more efficient than analyzing the joints. Our hybrid models further improve upon the performance of the graph edge convolutional network model, which means that combining edge and node convolution does help to improve the performance. The fact that the body-part-level hybrid model performs better than the SLHM validates our assumption that instead of simply combining the sequence-level features, merging the body-part-level features enables the two models to help each other in a subtler way, resulting in higher performance. To further validate our analysis, we analyze the classification accuracies on each class of the NTU-RGB+D data set.

a) Advantages of GECNN on certain classes: In Fig. 11, we reported the classification accuracy for each class. Looking into certain classes in which the edge convolution model greatly outperforms the node convolution model, we can obtain more interesting findings, leading us to realize more advantages of recognizing human action from the dynamics of limbs. For example, in classes 3, 10, 44, 47, and 53, the edge convolution model significantly outperforms the node convolution model. We can see that in these action classes, joints' movements may be too subtle to capture, e.g., in classes 3 (brushing teeth) and 47 (neckache), the hand movements are nearly invisible (see Fig. 12).

In these cases, although dynamics of some joints are difficult to capture, the limb movements are always easier to observe. For example, in the second row of Fig. 12, we visualize several frames of a certain sample in class "brushing teeth." As we see, the hand joints and head joint are nearly stable, causing difficulty in capturing their dynamics and recognizing the corresponding action class. In contrast, the location changes of upper arms and forearms are more significant or at least visible.

In the previous paragraph, we analyzed the advantages of the edge convolution model by determining in which classes it outperforms other models significantly and then analyzed the characteristics of these classes. A similar analysis can also be conducted in other classes to find other advantages of different models, but we are not going to discuss this matter further, as this kind of analysis is not very rigorous and requires further validation to be fully convincing. Our discussion in the previous paragraph simply provides a possible way to analyze the characteristics of different models; more detailed validation is left for future work.

b) Model comparison on class accuracies: Besides the accuracies reported in Table III, we can also compare the performances of the four models from another point of view. In Fig. 11, we see that in 25 classes, our body-part-level hybrid model outperforms the other three models; in 18 classes, the SLHM performs the best; in 15 classes, the edge convolution model achieves the highest accuracies; and in only 8 classes, the node convolution model obtains better results than those of the other three models. (In some classes, two of the three models may achieve a draw.)

The hybrid models outperform the two models that are based solely on joints or limbs in terms of both overall accuracy and number of best performance classes, which demonstrates that our hybrid models successfully inherit the advantages from both the limb-based and joint-based models.

G. Computational Time

In this section, we list the computational time cost by each model, as listed in Table IV. The reported computational time is averaged on multiple test epochs.

From Table IV, we see that there is no huge difference between the time costs of edge- and node-convolutional networks, which accords with our complexity analysis in former parts. On Kinetics, our edge convolutional model is slightly faster, and on NTU-RGB+D, it is a little bit

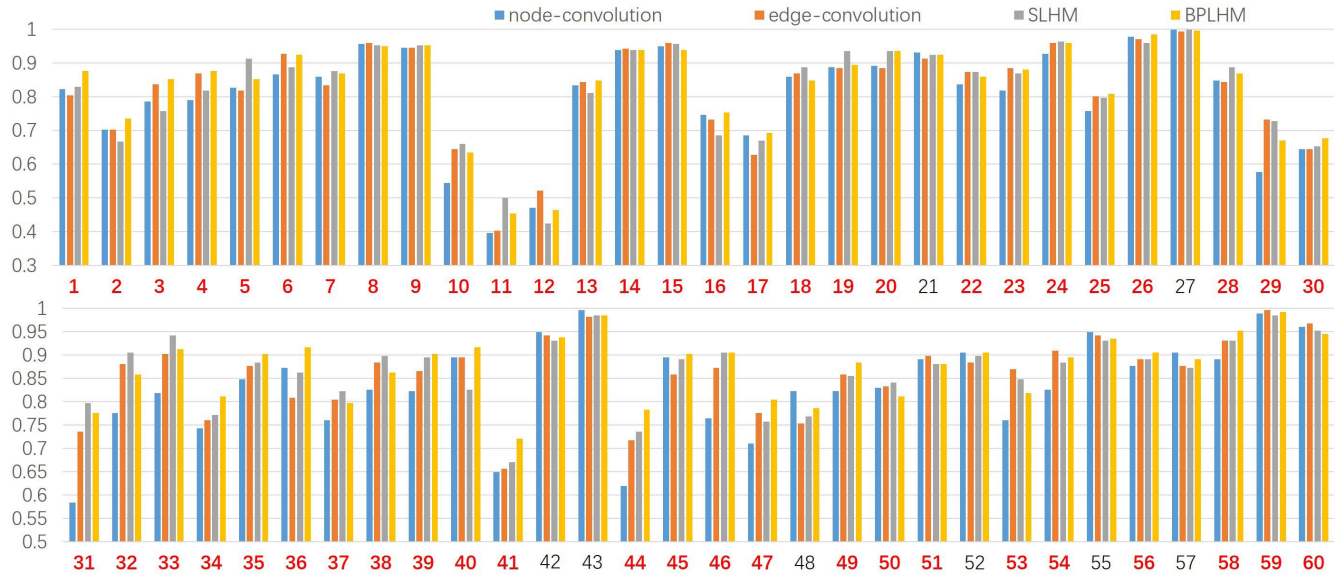


Fig. 11. Class accuracy on the NTU-RGB+D data set with cross-subject separation. For each class, we represent the accuracies of four models, with the corresponding bars in four different colors. We provide only the number denoting each class with red and bold numbers denoting the classes on which our model performs better; for the detailed name of each class, please refer to https://github.com/shahroudy/NTURGB-DNTU-RGB+D.

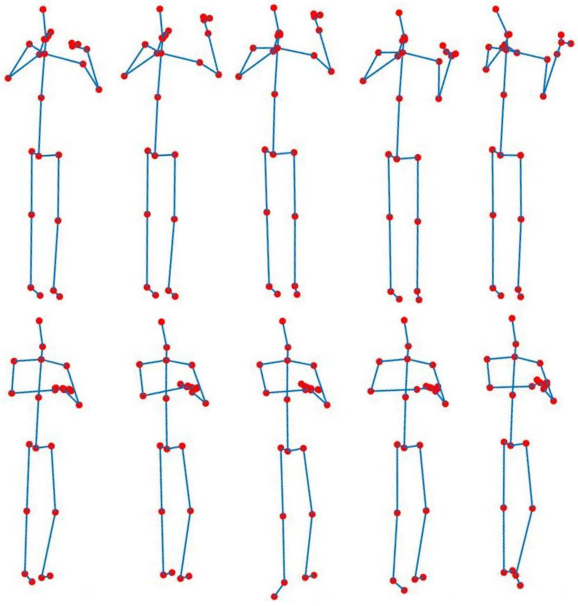


Fig. 12. Top row: 5 frames extracted from an example in the 47th class "touch neck (neckache)"; we can see that the joints of the hands seriously overlap the joint around the neck. Bottom row: 5 frames extracted from the initial part of an example in the third class "brushing teeth"; we can see that the joints of the two hands overlap each other greatly.

slower. Our hybrid models cost more time than the sole node- or edge-convolutional model because of higher complexity. However, from Tables II and III, we can see that the performance improvement brought by the hybrid models is significant. On Kinetics, the highest improvement achieved by the hybrid models is 2% compared to GECNN and 2.6% compared to ST-GCN, and on NTU-RGB+D, the corresponding improvement is 1.4% and 2.4%. Thus, the extra time cost by hybrid models is worthwhile.

TABLE IV
COMPUTATIONAL TIME OF DIFFERENT MODELS ON DIFFERENT DATA SETS

Model \ Dataset	Kinetics	X-Sub (NTU)	X-View (NTU)
Node ConvNet	87 sec	91 sec	87 sec
Edge ConvNet	81 sec	102 sec	93 sec
SLHM	137 sec	129 sec	150 sec
BPLHM	164 sec	188 sec	180 sec

V. CONCLUSION

Considering the lack of attention on human limb dynamics for skeleton-based action recognition, in this article, we propose a novel graph edge convolution that represents each edge by integrating its neighboring edges. A GECNN is then constructed based on the edge convolution operation. Our model captures the relationship and dependences between human limbs by conducting convolution on edges of human skeleton graphs. We apply our model to human action recognition tasks on two data sets, resulting in remarkably superior performance over the existing state-of-the-art methods. Considering the complementarity of node- and edge-convolutional networks, we further seek to combine node convolution and edge convolution in two hybrid models that inherit the advantages of both the models. The experiment results show that our hybrid models can further improve the performance.

REFERENCES

[1] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1799–1807.

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," 2016, *arXiv:1611.08050*. [Online]. Available: https://arxiv.org/abs/1611.08050

[3] Y. Makihara, A. Suzuki, D. Muramatsu, X. Li, and Y. Yagi, "Joint intensity and spatial metric learning for robust gait recognition," in *Proc. 30th IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5705–5715.

[4] Z. Wu, Y. Huang, L. Wang, X. Wang, and T. Tan, "A comprehensive study on cross-view gait based human identification with deep CNNs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 2, pp. 209–226, Mar. 2017.

[5] A. Mansur, Y. Makihara, R. Aqmar, and Y. Yagi, "Gait recognition under speed transition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 2521–2528.

[6] S. Lombardi, K. Nishino, Y. Makihara, and Y. Yagi, "Two-point gait: Decoupling gait from body shape," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 1041–1048.

[7] L. Niu, X. Xu, L. Chen, L. Duan, and D. Xu, "Action and event recognition in videos by learning from heterogeneous Web sources," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1290–1304, Jun. 2017.

[8] Y. Zhou, X. Sun, Z.-J. Zha, and W. Zeng, "MICT: Mixed 3D/2D convolutional tube for human action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 449–458.

[9] H. Liu, N. Shu, Q. Tang, and W. Zhang, "Computational model based on neural network of visual cortex for human action recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1427–1440, May 2018.

[10] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B. E. Shi, and S. Savarese, "Lattice long short-term memory for human action recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2147–2156.

[11] X. Chen, J. Weng, W. Lu, J. Xu, and J. Weng, "Deep manifold learning combined with convolutional neural networks for action recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 3938–3952, Sep. 2018.

[12] J. Weng, M. Liu, X. Jiang, and J. Yuan, "Deformable pose traversal convolution for 3D action and gesture recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 136–152.

[13] J.-F. Hu, W.-S. Zheng, J. Pan, J. Lai, and J. Zhang, "Deep bilinear learning for RGB-D action recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 335–351.

[14] M. Guo, E. Chou, D.-A. Huang, S. Song, S. Yeung, and L. Fei-Fei, "Neural graph matching networks for fewshot 3D action recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 653–669.

[15] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 1290–1297.

[16] B. Fernando, S. Gavves, O. Mogrovejo, J. Antonio, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *Proc. CVPR*, Jun. 2015, pp. 5378–5387.

[17] M. E. Hussein, M. Torki, M. A. Gowayyed, and M. El-Saban, "Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations," in *Proc. IJCAI*, vol. 13, 2013, pp. 2466–2472.

[18] W. Li, L. Wen, M.-C. Chang, S. N. Lim, and S. Lyu, "Adaptive RNN tree for large-scale human action recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1444–1452.

[19] 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 Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2117–2126.

[20] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," 2018, *arXiv:1801.07455*. [Online]. Available: https://arxiv.org/abs/1801.07455

[21] I. Lee, D. Kim, S. Kang, and S. Lee, "Ensemble deep learning for skeleton-based action recognition using temporal sliding LSTM networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1012–1020.

[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] 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.

[24] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," 2015, *arXiv:1511.05493*. [Online]. Available: https://arxiv.org/abs/1511.05493

[25] F. P. Such *et al.*, "Robust spatial filtering with graph convolutional neural networks," *IEEE J. Sel. Topics Signal Process.*, vol. 11, no. 6, pp. 884–896, Sep. 2017.

[26] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*. [Online]. Available: https://arxiv.org/abs/1312.6203

[27] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*. [Online]. Available: https://arxiv.org/abs/1506.05163

[28] T. S. Kim and A. Reiter, "Interpretable 3D human action analysis with temporal convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jul. 2017, pp. 20–28.

[29] B. Du, Q. Wei, and R. Liu, "An improved quantum-behaved particle swarm optimization for endmember extraction," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 6003–6017, Aug. 2017.

[30] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. CVPR*, Jul. 2017, pp. 3693–3702.

[31] L. Zhang, L. Song, B. Du, and Y. Zhang, "Nonlocal low-rank tensor completion for visual data," *IEEE Trans. Cybern.*, to be published.

[32] G. Lai, H. Liu, and Y. Yang, "Learning graph convolution filters from data manifold," 2017, *arXiv:1710.11577*. [Online]. Available: https://arxiv.org/abs/1710.11577

[33] Z. Xu *et al.*, "Semisupervised discriminant multimanifold analysis for action recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.

[34] B. Du, Y. Wang, C. Wu, and L. Zhang, "Unsupervised scene change detection via latent Dirichlet allocation and multivariate alteration detection," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 11, no. 12, pp. 4676–4689, Oct. 2018.

[35] A. Klaser, M. Marszałek, and C. Schmid, "A spatio-temporal descriptor based on 3D-gradients," in *Proc. 19th Brit. Mach. Vis. Conf. (BMVC)*, 2008, pp. 1–275.

[36] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Apr. 2011, pp. 3169–3176.

[37] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2625–2634.

[38] L. Wang *et al.*, "Temporal segment networks: Towards good practices for deep action recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 20–36.

[39] P. Wang, W. Li, Z. Gao, Y. Zhang, C. Tang, and P. Ogunbona, "Scene flow to action map: A new representation for RGB-D based action recognition with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 595–604.

[40] Z. Shi and T.-K. Kim, "Learning and refining of privileged information-based RNNs for action recognition from depth sequences," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3461–3470.

[41] H. Rahmani and M. Bennamoun, "Learning action recognition model from depth and skeleton videos," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5832–5841.

[42] J. Shotton *et al.*, "Real-time human pose recognition in parts from single depth images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 1297–1304.

[43] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. CVPR*, vol. 1, no. 2, Jul. 2017, p. 7.

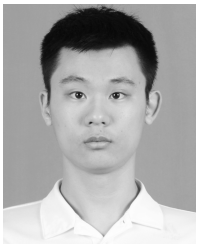
[44] L. Xia, C.-C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3D joints," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2012, pp. 20–27.

[45] 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.*, Jun. 2014, pp. 588–595.

[46] H. Kim, S. Lee, and A. C. Bovik, "Saliency prediction on stereoscopic videos," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1476–1490, Apr. 2014.

[47] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1110–1118.

[48] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3d human activity analysis," 2016, *arXiv:1604.02808*. [Online]. Available: https://arxiv.org/abs/1604.02808
[49] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 816–833.
[50] W. Zhu *et al.*, "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks," in *Proc. AAAI*, vol. 2, 2016, pp. 1–8.
[51] S. Zhang, X. Liu, and J. Xiao, "On geometric features for skeleton-based action recognition using multilayer LSTM networks," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 148–157.
[52] 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.
[53] W. Kay *et al.*, "The kinetics human action video dataset," 2017, *arXiv:1705.06950*. [Online]. Available: https://arxiv.org/abs/1705.06950
[54] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.
[55] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, "Global context-aware attention LSTM networks for 3D action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 7, Jul. 2017, pp. 1647–1656.
[56] 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. ICCV*, Jun. 2017, pp. 2904–2913.
[57] F. Baradel, C. Wolf, and J. Mille, "Human action recognition: Pose-based attention draws focus to hands," in *Proc. ICCV Workshop Hands Action*, Jul. 2017, pp. 604–613.



Xikun Zhang received the bachelor's degree in applied physics from the University of Science and Technology of China, Hefei, China. He is currently pursuing the master's degree with the School of Computer Science, The University of Sydney, Darlington, NSW, Australia.

His current research interests include graph data classification and skeleton-based human action recognition.

Chang Xu received the Ph.D. degree from Peking University, Beijing, China.

He is currently a Lecturer and an ARC DECRA Fellow with the School of Computer Science, The University of Sydney, Darlington, NSW, Australia. He has published over 50 papers in prestigious journals and top tier conferences, including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON IMAGE PROCESSING, International Conference on Machine Learning (ICML), Conference on Neural Information Processing Systems (NIPS), International Joint Conference on Artificial Intelligence (IJCAI), and Association for the Advancement of Artificial Intelligence (AAAI). His current research interests include machine learning algorithms and related applications in computer vision.

Dr. Xu regularly served as a (senior) Program Community Member for many conferences, including NIPS, ICML, Conference on Computer Vision and Pattern Recognition (CVPR), International Conference on Computer Vision (ICCV), IJCAI, and AAAI. He has received several paper awards, including the Distinguished Paper Award in IJCAI 2018. He has been recognized as the Top Ten Distinguished Senior PC at IJCAI 2017.

Xinmei Tian (M'13) received the B.E. and Ph.D. degrees from the University of Science and Technology of China (USTC), Hefei, China, in 2005 and 2010, respectively.

She is currently an Associate Professor with the CAS Key Laboratory of Technology in Geo-Spatial Information Processing and Application System, USTC. Her current research interests include multimedia information retrieval and machine learning.

Dr. Tian received the Excellent Doctoral Dissertation of the Chinese Academy of Sciences Award in 2012 and the Nomination of the National Excellent Doctoral Dissertation Award in 2013.

Dacheng Tao (F'15) is currently a Professor of computer science and an ARC Laureate Fellow with the School of Computer Science and the Faculty of Engineering and the Inaugural Director of the UBTECH Sydney Artificial Intelligence Centre, The University of Sydney, Darlington, NSW, Australia. His research results in artificial intelligence have expounded in one monograph and over 200 publications at prestigious journals and prominent conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, International Journal of Computer Vision (IJCV), Journal of Machine Learning Research (JMLR), AAAI, IJCAI, NIPS, ICML, CVPR, ICCV, European Conference on Computer Vision (ECCV), International Conference on Data Mining (ICDM), and SIGKDD Conference on Knowledge Discovery and Data Mining, with several best paper awards.

Prof. Tao is also a fellow of the Australian Academy of Science. He received the 2018 IEEE ICDM Research Contributions Award.