Skeleton-Based Action Recognition with Directed Graph Neural Networks

Lei Shi[1,2] Yifan Zhang[1,2*] Jian Cheng[1,2,3] Hanqing Lu[1,2]
[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]CAS Center for Excellence in Brain Science and Intelligence Technology
{lei.shi, yfzhang, jcheng, luhq}@nlpr.ia.ac.cn

Abstract

The skeleton data have been widely used for the action recognition tasks since they can robustly accommodate dynamic circumstances and complex backgrounds. In existing methods, both the joint and bone information in skeleton data have been proved to be of great help for action recognition tasks. However, how to incorporate these two types of data to best take advantage of the relationship between joints and bones remains a problem to be solved. In this work, we represent the skeleton data as a directed acyclic graph (DAG) based on the kinematic dependency between the joints and bones in the natural human body. A novel directed graph neural network is designed specially to extract the information of joints, bones and their relationships and make prediction based on the extracted features. In addition, to better fit the action recognition task, the topological structure of the graph is made adaptive based on the training process, which brings notable improvement. Moreover, the motion information of the skeleton sequence is exploited and combined with the spatial information to further enhance the performance in a two-stream framework. Our final model is tested on two large-scale datasets, NTU-RGBD and Skeleton-Kinetics, and exceeds state-of-the-art performance on both of them.

1. Introduction

Action recognition, which plays an essential role in video surveillance and human-computer interaction, has been widely investigated but not yet fully addressed [27, 31, 36, 5, 32, 38]. Compared with conventional processes that use RGB images or videos for recognition, skeleton-based action recognition has drawn increasingly more attention since it is robust against changes in body scales, motion speeds, camera viewpoints and interference of backgrounds. The skeleton data represent the human body as a sequence of coordinates of the major body joints, which can be easily captured by the depth sensors (e.g., Kinetics) or the pose estimation algorithms [4, 10].

Conventional methods for skeleton-based action recognition focus mainly on designing handcrafted features to represent the skeleton [30, 8]. With the development of deep-learning-based methods, the data-driven methods have become the mainstream [7, 25, 20, 28, 37, 18, 19, 14, 13, 21, 17, 16, 34, 29, 3]. The most widely used models in deep-learning-based methods are recurrent neural networks (RNNs), convolutional neural networks (CNNs) and graph convolutional networks (GCNs), where the coordinates of joints are represented as vector sequences, pseudo-images and graphs, respectively.

Recently, bone information, which represents the directions and lengths of bones, has been proved to be a good modality for skeleton-based action recognition [26, 18]. This information is intuitive since humans naturally evaluate action according to the directions and positions of bones in the human body rather than the positions of joints. Moreover, it has been proved that joint and bone information are complementary to each other and combining them can lead to further improvement of recognition performance. For the natural human body, joints and bones are strongly coupled, and the position of each joint (bone) is actually determined by their connected bones (joints). For example, the position of the elbow joint depends on the location of the upper arm bone, which also determines the location of the forearm bone at the same time. Existing graph-based methods usually represent the skeleton as an undirected graph and model the bones and joints with two separate networks, which cannot fully exploit these dependencies between joints and bones. To solve this problem, we represent the skeleton as a directed acyclic graph with joints as vertexes and bones as edges, where the dependencies between the joints and bones can be easily modeled by the directed edges of the graph. Furthermore, a novel directed graph neural network (DGNN) is designed to model the constructed directed graph, which can propagate the infor-

*Corresponding Author

mation in adjacent joints and bones and update their associated information in each layer. The final extracted features contain not only the information of each joint and bone but also their dependencies which can facilitate action recognition.

Another problem is that the original skeleton is designed by hand according to the structure of the human body, which may be not optimal for the action recognition tasks. For example, the two hands have strong dependencies in some action classes such as clapping and hugging, but this connection does not exist in the graph constructed based on the human body structure. We solved this problem by applying an adaptive graph instead of a fixed graph inspired by [26], which means that the topology of graph is parameterized and is optimized during the learning process. Since there is no constraint in the learned graph, the approach described in [26] adds a fixed manually set graph to stabilize the training process, which somewhat loses the flexibility at the same time. In this work, we propose a simple yet effective method to not only ensure the stability of training process but also avoid losing flexibility, which brings a notable improvement.

Two-stream-based architecture, a widely used method for RGB-based action recognition, extracts the optical flow field of the video to model the temporal dependency between frames [27, 31]. This approach is effective since some classes strongly rely on the sequential information of the action, such as "waving a hand to the left" versus "waving a hand to the right". Inspired by this method, we extract the motion information from both joints and bones to aid in recognition. A two-stream framework is proposed to fuse the results of the spatial stream and motion stream to further enhance the performance.

The final model is evaluated on two large-scale datasets for skeleton-based action recognition tasks, i.e., NTU-RGBD and Skeleton-Kinetics, and exceeds state-of-the-art performance on both. The main contributions of our work can be included as follows: (1) To the best of our knowledge, this is the first work to represent the skeleton data as a directed acyclic graph to model the dependencies between joints and bones. A novel directed graph neural network is designed specially to extract these dependencies for the final action recognition task. (2) An adaptively learned graph structure, which is trained and updated jointly with model parameters in the training process, is used to better suit the action recognition task. (3) The motion information between consecutive frames is extracted for temporal information modeling. Both the spatial and motion information are fed into a two-stream framework for the final recognition task. (4) On the two large-scale datasets for skeleton-based action recognition, our model exceeds the state-of-the-art performance with a significant margin.

2. Related work

2.1. Skeleton-based action recognition

Conventional methods for skeleton-based action recognition usually use handcrafted features to represent the human body, which present challenges during design and result in unsatisfactory performance [30, 8]. Recently, deep-learning-based methods have been shown to be superior over conventional methods. There are mainly three frameworks for deep-learning-based methods: sequence-based methods, image-based methods and graph-based methods.

Sequence-based methods represent the skeleton data as a sequence of joints based on the designed traversal strategy, which is then modeled with RNN-based architectures [25, 20, 28, 37, 18, 3]. Another framework, involving the image-based methods, represents the skeleton data as a pseudo-image to implement CNNs applied successfully in the field of image classification [6, 14, 21, 16] . Instead of representing skeleton data as sequences or pseudo-images, graph-based methods model the data as a graph with joints as vertexes and bones as edges [34, 29, 26]. Compared with the sequence-based methods and image-based methods, the graph-based methods are more intuitive since the human body is naturally organized as a graph rather than a sequence or an image.

2.2. Graph networks

Graph is a more general data structure than image and sequence, which cannot be directly modeled by conventional deep learning modules such as CNNs and RNNs. Approaches for operating directly on graphs and solving graph-based problems have been explored extensively for several years [15, 9, 33, 24, 1, 11, 2]. For example, Kipf et al. [15] propose an unsupervised neural relational inference model that can infer the interactions and learn the dynamics from the observational data on physical simulations. Gilmer et al. [9] propose a message passing network to solve chemical prediction problems, which can directly extract features from molecular graphs and is invariant to the graph isomorphism. Wang et al. [33] represent videos as space-time region graphs to model the temporal dynamics and relationships between humans and objects, which can then be used to understand human action.

3. Method

Typically, the raw skeleton data are a sequence of frames, each of which contains a set of joint coordinates. Given a skeletons sequence, we first extract the bone information according to the 2D or 3D coordinates of the joints. Then, the joints and bones (the spatial information) in each frame are represented as the vertexes and edges within a directed acyclic graph, which is fed into the directed graph neural network (DGNN) to extract features for action recognition.

Finally, the motion information, which is represented with the same graph structure that used for spatial information, is extracted and combined with the spatial information in a two-stream framework to further improve the performance.

3.1. Bone Information

Previous works have shown the importance of combining the joint information and bone information together for skeleton-based action recognition [26, 18]. The bone is represented as the difference of coordinates between two connected joints. Take the 3D skeleton data as an example: the joint in raw data is represented as a vector with three elements, i.e., its x-coordinate, y-coordinate and z-coordinate. Given two joints $\mathbf{v_1} = (x_1, y_1, z_1)$ and $\mathbf{v_2} = (x_2, y_2, z_2)$, the bone linked from $\mathbf{v_1}$ to $\mathbf{v_2}$ is formulated as the difference of the two joint vectors, i.e., $\mathbf{e}_{\mathbf{v_1},\mathbf{v_2}} = (x_1 - x_2, y_1 - y_2, z_1 - z_2)$.

3.2. Graph Construction

Conventional methods always model the skeleton data as a sequence of vectors or a pseudo-image to be processed by RNNs or CNNs. However, these representations ignore the kinematic dependencies between joints and bones. In human parsing, the skeleton data are always modeled as a tree-based pictorial structure [39, 35] according to the physical structure of the human body. In this work, we represent the skeleton data as a directed acyclic graph (DAG) with the joints as vertexes and bones as edges. The direction of each edge is determined by the distance between the vertex and the root vertex, where the vertex closer to the root vertex points to the vertex farther from the root vertex. Here, the root vertex is defined as the center of gravity of the skeleton. Figure 1 shows an example of a skeleton and its corresponding directed graph representation, where the vertex one is the root vertex. This representation is intuitive since the human body is naturally an articulated system. The joints farther from the center of the human body are always physically controlled by an adjacent joint which is closer to the center. For example, the wrist position is determined by the position of the elbow and the shape of the forearm. In this way, we represent the forearm as an directed edge pointing to the wrist from the elbow.

Formally, for each vertex $\mathbf{v}_i$, we define the edge heading to it as the incoming edge $\mathbf{e}_i^-$ and the edge emitting from it as the outgoing edge $\mathbf{e}_i^+$. Similarly, for a directed edge $\mathbf{e}_j$, we define that it is a vector from the source vertex $\mathbf{v}_j^s$ to the target vertex $\mathbf{v}_j^t$. If $\mathbf{v}_i$ is the target (source) vertex of $\mathbf{e}_j$, then $\mathbf{e}_j$ is the incoming (outgoing) edge of $\mathbf{v}_i$, and vice versa. For example, as shown in Figure 2 (a), $\mathbf{v}_1$ and $\mathbf{v}_2$ are the source and target vertexes of $\mathbf{e}_1$, respectively. $\mathbf{e}_1$ is the incoming edge of $\mathbf{v}_2$. $\mathbf{e}_2$ and $\mathbf{e}_3$ are the outgoing edges of $\mathbf{v}_2$. Note that each edge has only one source vertex and one target vertex. For one vertex, however, the number of its incoming edges and outgoing edges is varied. We use $\mathcal{E}_i^-$ and $\mathcal{E}_i^+$ to denote the set of incoming edges and the set of outgoing edges of vertex $\mathbf{v}_i$, respectively. In this way, a skeleton-based frame can be formulated as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertexes (joints) and $\mathcal{E}$ is a set of directed edges (bones). A skeleton-based video is a sequence of frames that can be formulated as $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_T\}$, where T denotes the length of the video.

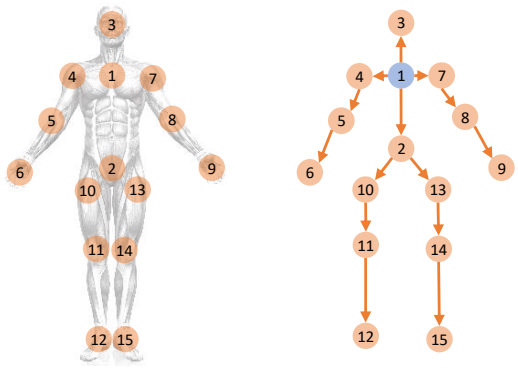

Figure 1. Illustration of the graph construction for skeleton data. The blue circle indicates the root vertex.

3.3. Directed graph neural network

Since we have represented the skeleton data as a directed graph, the problem now lies in how to extract the information contained in the graph for action classification, specially how to make use of the dependencies between the joints and bones in the graph. In this work, we propose a directed graph neural network (DGNN) to solve this problem. The network contains multiple layers, each of which is fed with a graph containing the attributes of the vertexes and edges, and outputs the same graph with updated attributes. Here, the attributes denote the properties of the vertexes and edges that are encoded as vectors. In each layer, the attributes of the vertexes and edges are updated according to its adjacent edges and vertexes. On the bottom layers, each vertex or edge can receive attribute only from its adjacent edge or vertex. The model in these layers aims to extract the local information of the vertexes and edges when updating the attributes. For example, the model can extract the angle information of a joint, which needs only the information of one joint and its two connected bones. On the top layers, messages from the joints and bones farther from each other can be accumulated together. Thus, the extracted information is more global and semantic for the recognition task. This concept is similar to the principle of the convolutional neural networks, i.e., the hierarchical representation and locality. In contrast to CNNs, the DGNN

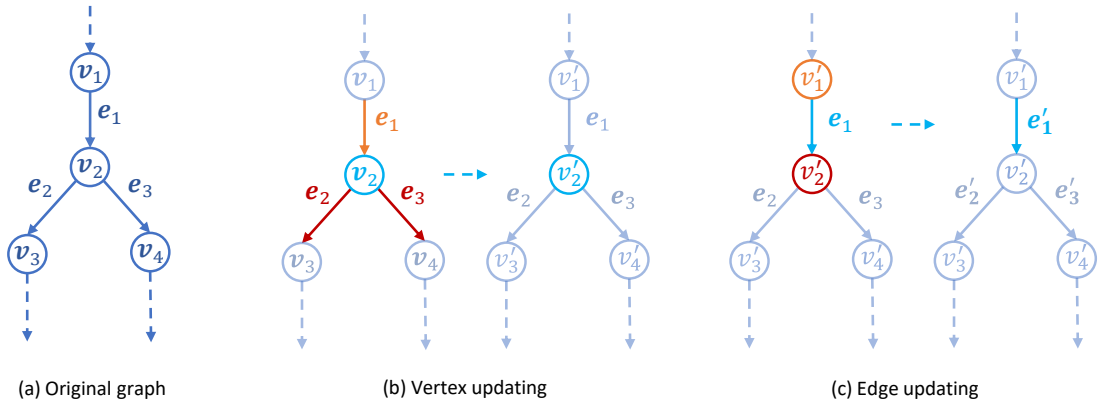

Figure 2. (a) is the original graph. (b) shows the procedure of vertex updating, where the attribute of the vertex itself (v_2) and the attributes of its incoming edge (e_1) and outgoing edges (e_2 and e_3) are combined to obtain an updated vertex ($v_2{}'$). (c) shows the procedure of edge updating, where the attribute of the edge itself (e_1) and the attributes of its source vertex ($v_1{}'$) and target vertex ($v_2{}'$) are combined to obtain an updated edge ($e_1{}'$). The blue circle represents the edge (or vertex) that is being updated. The orange circle and red circle represent the source vertex (or incoming edge) and target vertex (or outgoing edge) that are involved in the update, respectively.

is designed for the directed acyclic graphs that can model the tree-based structure of the skeleton data. This design is also similar to the "body parts" conception in previous works for skeleton-based action recognition, which aims to restrict the modeling of joints in a local part of the human body [6, 25, 16]. However, our method does not need the process of manually designing the segmentation strategies and achieves better performance than these methods.

3.3.1 Directed graph network block

The directed graph network (DGN) block is the basic block for a directed graph neural network; it contains two updating functions, $h^{\mathbf{v}}$ and $h^{\mathbf{e}}$, and two aggregation functions, $g^{\mathbf{e}^-}$ and $g^{\mathbf{e}^+}$. The updating function is used to update the attributes of vertexes and edges based on their connected edges and vertexes. The aggregation function is used to aggregate the attributes contained in multiple incoming (outgoing) edges connected to one vertex. It is because the number of incoming (outgoing) edges connected to each vertex is varied whereas the number of parameters is fixed. Because there are no apparent orders for these edges, the aggregation function should be invariant to the permutation of its inputs and can take variable numbers of arguments, such as the average pooling, max pooling and elementwise summation. Formally, this process is formulated as follows:

$$\begin{aligned}
\bar{\mathbf{e}}_i^- &= g^{\mathbf{e}^-}(\mathcal{E}_i^-) \\
\bar{\mathbf{e}}_i^+ &= g^{\mathbf{e}^+}(\mathcal{E}_i^+) \\
\mathbf{v}_i' &= h^{\mathbf{v}}([\mathbf{v}_i, \bar{\mathbf{e}}_j^-, \bar{\mathbf{e}}_j^+]) \\
\mathbf{e}_j' &= h^{\mathbf{e}}([\mathbf{e}_j, \mathbf{v}_j^{s\prime}, \mathbf{v}_j^{t\prime}])
\end{aligned} \tag{1}$$

where $[\cdot]$ denotes the concatenation operation. $\mathbf{v}'$ and $\mathbf{e}'$ are the updated versions of $\mathbf{v}$ and $\mathbf{e}$, respectively. The process involves four steps:

1. For each vertex $\mathbf{v}_i$, all of the edges that point to it are processed by the incoming aggregation function $g^{\mathbf{e}^-}$, which returns the aggregated result $\bar{\mathbf{e}}_i^-$.

2. Similar to step 1, all of the edges that emit from $\mathbf{v}_i$ are processed by the outgoing aggregation function $g^{\mathbf{e}^+}$, which returns the aggregated result $\bar{\mathbf{e}}_i^+$.

3. $\mathbf{v}_i$, $\bar{\mathbf{e}}_i^-$ and $\bar{\mathbf{e}}_i^+$ are concatenated and fed into the vertex-update function $h^{\mathbf{v}}$, which returns $\mathbf{v}_i'$ as the updated version of $\mathbf{v}_i$.

4. For each edge $\mathbf{e}_j$, its source vertex, target vertex and itself are concatenated and processed by the edge-update function $h^{\mathbf{e}}$. The function returns $\mathbf{e}_j'$, which is the updated version of edge $\mathbf{e}_j$.

The process can be also summarized as a vertex-update process followed by an edge-update process as shown in Fig. 2. With extensive experiments, we have chosen the average pooling as the aggregation functions for both the incoming edges and outgoing edges and chosen the single fully-connected layer as the update functions in this work.

3.3.2 Implementation of the DGN block

When implementing the DGN block, the input data of the vertexes actually form a $C \times T \times N_v$ tensor $\mathbf{f}_v$, where C is the number of the channels and T is the number of the frames. N_v denotes the number of the vertexes in a skeleton

graph. Similarly, the data of the edges form a $C \times T \times N_e$ tensor $\mathbf{f}_e$, where N_e is the number of the edges in the graph. It is not satisfactory to implement the DGN block with this form of input data. According to the last section, the key for implementing DGN block is to find the incoming edges and outgoing edges for each vertex (i.e., $\mathcal{E}_i^-$ and $\mathcal{E}_i^+$), and find the source vertex and target vertex for each edge (i.e., $\mathbf{v}_j^s$ and $\mathbf{v}_j^t$). To this end, we use the incidence matrix of the graph. Given a directed graph with N_v vertexes and N_e edges, the incidence matrix of A is an $N_v \times N_e$ matrix whose element $(A_{ij}, i = 1, \cdots, N_v; j = 1, \cdots, N_e)$ indicates the relationship between the corresponding vertex ($\mathbf{v_j}$) and edge ($\mathbf{e_i}$). In detail, if $\mathbf{v}_i$ is the source vertex of $\mathbf{e}_j$, then $A_{ij} = -1$. If $\mathbf{v}_i$ is the target vertex of $\mathbf{e}_j$, then $A_{ij} = 1$. If there are no connections between $\mathbf{v}_i$ and $\mathbf{e}_j$, then $A_{ij} = 0$.

To separate the source vertexes and target vertexes, we use A^s to denote the incidence matrix of source vertexes, which contains only the absolute value of the elements of A that are smaller than 0. Similarly, we define A^t as the incidence matrix of target vertexes, which contains only the elements of A that are greater than 0. For example, Eq. 2 shows the incidence matrix and its corresponding A^s and A^t for the graph shown in Figure 1 (a).

$$
\begin{aligned}
A &= \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}^\intercal \\
A^s &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}^\intercal \\
A^t &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^\intercal
\end{aligned}
\tag{2}
$$

where $\intercal$ denotes the transpose operation of the matrix. Given an input tensor and incidence matrix, we can now filter the required edges and vertexes and perform the aggregation function by matrix multiplication. For example, given $\mathbf{f}_v$ and A^s, we first reshape $\mathbf{f}_v$ into a $CT \times N_v$ matrix; then, the multiplication of $\mathbf{f}_v$ and A^s can provide a $CT \times N_e$ tensor. According to the definition of matrix multiplication, each element of this tensor corresponds to the summation of the source vertexes for the corresponding edge. Note that the aggregation function used in this work is the average pooling operation and that the incidence matrix needs to be normalized. In detail, we define $\tilde{A} = A\Lambda^{-1}$ as the normalized version of A, where Λ is a diagonal matrix and $\Lambda_{ii} = \sum_j A_{ij} + \epsilon$. ϵ is a small number to avoid division by zero. With these modifications, Eq. 1 is transformed into

$$
\begin{aligned}
\mathbf{f}'_v &= H_v([\mathbf{f}_v, \mathbf{f}_e\tilde{A}^{s\intercal}, \mathbf{f}_e\tilde{A}^{t\intercal}]) \\
\mathbf{f}'_e &= H_e([\mathbf{f}_e, \mathbf{f}_v\tilde{A}^s, \mathbf{f}_v\tilde{A}^t])
\end{aligned}
\tag{3}
$$

where H denotes the single-layer fully connected layer, i.e., the updating function in Eq. 1. Similar to the conventional convolutional layer, we add a BN layer and a ReLU layer after each DGN block.

3.3.3 Adaptive DGN block

The input graph of the DGN block is manually designed according to the natural structure of the human body. We suggest that this configuration may be not suitable for the action recognition task. For example, there are no connections between the left hand and the right hand; however, for many actions such as clapping and hugging, the relations between two hands are important for recognition. To give more flexibility to graph construction, conventional methods aim to construct an adaptive graph by learning the topology of the graph structure in the training process. For example, Yan et al. [34] apply an attention map on the original adjacency matrix to assign different levels of importance to different edges. If we use A_o to denote the original adjacency matrix, the new adjacency matrix A is calculated by $A = PA_o$, where the elements of P are initialized as 1 and are updated during training process. However, the multiplication operation cannot change the elements that are 0 in the original adjacency matrix, which means that this approach can change only the importance of existing edges and cannot add new edges, e.g., an edge between two hands. Different from ST-GCN, Shi et al. [26] directly set the adjacency matrix as the parameter of networks. To stabilize the training process, they set $A = A_o + P$, where P has the same size as A_o and is initialized with 0. In this way, new edges can be added through the parameter P in the learning process if necessary. Nevertheless, since A_o is unmodifiable, we cannot remove the edges we do not want, which also reduces the flexibility of the model. However, if we remove A_o, directly learning the graph structure without any restriction will degrade the performance.

In this work, we found that the difference between the cases with or without A_o in [26] lies mainly in the beginning of the training process (note that A here denotes the incidence matrix rather than the adjacency matrix as in previous work). This result is intuitive since there is more uncertainty in the beginning of the training process; thus, the model with less restrictions but a large number of parameters easily converges to the local optimum. Adding a graph with fixed topology is equivalent to regularizing the model based on the prior knowledge of the human body, which can help the model converge to the global optimum. Based on this observation, we propose a simple and effective strategy to solve the problem. We directly set A as the parameter of the model but fix it at the first several training epochs. Fixing the graph structure in the early stage can ease the training and unfixing it afterwards can provides more flexibility for graph construction.

3.3.4 Temporal information modeling

Typically, an action is recorded as a sequence of skeleton-based frames. The DGN blocks introduced above can process the spatial information of a single frame only; thus, we now advance to the task of modeling the temporal dynamics within the skeleton sequence. The pseudo-3D CNN [23] has shown its superiority in the RGB-based action recognition field, which models the spatial information with the 2D convolutions and then models the temporal information with the 1D convolutions. By decoupling the spatial and temporal dimensions, the pseudo-3D CNN can model the spatiotemporal information in a more economic and effective way. Inspired by this approach, after updating the spatial information of joints and bones in each DGN block, we apply a 1D convolution along the temporal dimension to model the temporal information. This is easy to achieve since the same joints or bones in all of the frames can be naturally organized as a 1D sequence.

Similar to the DGN block, each 1D convolutional layer is followed with a BN layer and a ReLU layer to form a temporal convolutional block (TCN). The overall architecture of the directed graph neural network (DGNN) has 9 units, each containing one DGN block and one TCN block. The output channels of the units are 64,64,64,128,128,128,256,256 and 256. A global-average-pooling layer followed by a $softmax$ layer is added at the end for class prediction.

3.3.5 Two-Stream Framework

Some actions such as "standing up" versus "sitting down" are difficult to recognize from spatial information along. Conventional RGB-based action recognition methods usually use optical flow fields to describe the motion information of a video [27, 31, 5], which calculates the pixel movement information between the consecutive frames. Inspired by these methods, we extract both the movements of joints and the deformations of bones in this work to help the recognition. Since the skeleton data are represented as the coordinates of the joints, the motion of joints is easily calculated as the difference of coordinates along the temporal dimension. Similarly, the deformation of bones is represented as the difference of the vectors for the same bone in consecutive frames. Formally, the movement of joint $\mathbf{v}$ in time t is calculated as $\mathbf{m}_{\mathbf{v}_t} = \mathbf{v}_{t+1} - \mathbf{v}_t$. The deformation of bones is defined similarly as $\mathbf{m}_{\mathbf{e}_t} = \mathbf{e}_{t+1} - \mathbf{e}_t$. As with the spatial information modeling, the motion information is formulated as a sequence of directed acyclic graphs $\mathcal{S}^m = \{\mathcal{G}_1^m, \mathcal{G}_2^m, \cdots, \mathcal{G}_T^m\}$, where $\mathcal{G}^m = (\mathcal{V}^m, \mathcal{E}^m)$, $\mathcal{V}^m = \{\mathbf{m}_{\mathbf{v}_j}\}_{j=0,\cdots,N_v}$ and $\mathcal{E}^m = \{\mathbf{m}_{\mathbf{e}_i}\}_{i=0,\cdots,N_e}$. Then, the motion graphs are fed into another DGNN to make the prediction for the action label. Two networks are finally fused by adding the output scores of the $softmax$ layer.

4. Experiments

To validate our method, we conducted extensive experiments on two skeleton-based action recognition datasets: NTU-RGBD [25] and Skeleton-Kinetics [34]. Both of these datasets have been widely used in previous work for skeleton-based action recognition. We performed ablation studies on the NTU-RGBD dataset to validate the effectiveness of the proposed model components since it is smaller than Skeleton-Kinetics. Finally, the model was evaluated on both the NTU-RGBD dataset and Skeleton-Kinetics dataset to make a comparison with the state-of-the-art methods.

4.1. Datasets

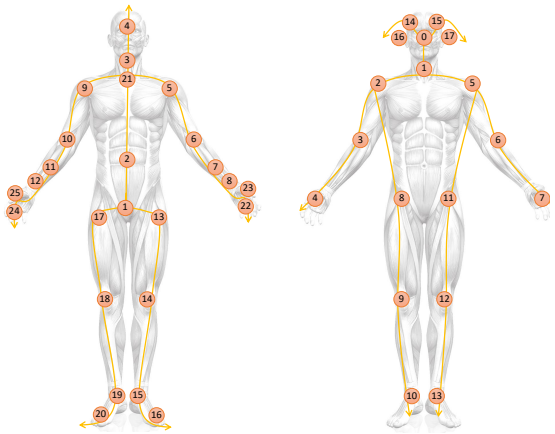

Figure 3. Illustration of the human skeleton graphs in NTU-RGBD dataset (left) and Skeleton-Kinetics dataset (right).

NTU-RGBD: NTU-RGBD is currently the most widely used dataset for skeleton-based action recognition; it contains 56,000 videos, each containing an action. There are a total of 60 classes including single-person action, e.g., drinking water, and two-person action, e.g., kicking another person. The dataset contains 4 different modalities of data: RGB videos, depth map sequences, 3D skeleton data and infrared videos. Here, we use only the skeleton data. These data are captured by Microsoft Kinect V2 at 30 fps. The actions are performed by 40 volunteers aging from 10 to 35. There are three cameras for every action, set at the same height but aimed from different horizontal angles: $-45°, 0°, 45°$. The camera can provide 25 3D locations of joints as labeled and shown in Figure 3, left. We follow the convention of the original paper [25] describing the dataset, which recommends two benchmarks: 1). Cross-subject (CS): The persons in the training and validation sets are different. The training set contains 40,320 videos, and validation set contains 16,560 videos. 2).Cross-view (CV): The horizontal angles of the cameras used in the training and validation sets are different. The training set $(0°, 45°)$

contains 37,920 videos, and the validation set ($-45°$) contains 18,960 videos. Top-1 accuracy is reported on both of the two benchmarks.

Skeleton-Kinetics: The original Deepmind Kinetics human action dataset [12] contains no skeleton data, instead containing approximately 300,000 video clips retrieved from YouTube. There are 400 human action classes, with at least 400 video clips for each action. Each clip lasts approximately 10 seconds. The actions cover a large range of classes focusing on human actions. The skeleton data in Skeleton-Kinetics [34] are extracted using the OpenPose [4] toolbox. All videos are resized to a resolution of 340×256 and are converted to a frame rate of 30 fps. The Openpose toolbox can predict 18 joints for each person, as labeled and shown in Figure 3, right. The toolbox provides 2D coordinates (X, Y) of the predicted joints in the image coordinate system and their corresponding confidence score C. Yan et al.[34] represent each joint with a tuple of (X, Y, C); we followed this approach to enable comparison of the results. If there are more than two persons, the persons with lower confidence are ignored. The released data pad every clips to 300 frames. Top-1 and Top-5 recognition accuracies are reported as the recommendation. The dataset is split into training and validation sets containing 240000 clips and 20000 clips, respectively.

4.2. Training details

All of the models are trained with the same batch size (32), learning schedule (SGD with an initial learning rate as 0.1 and reduced by 10 in epoch 60 and 90) and training epochs (120) with the Pytorch [22] framework. In addition, we performed some preprocessing for 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 wrong bodies, we first define the energy of each bodies as the summation of the skeleton's standard deviation across each channel. We then select two bodies in each sample according to their body energies. Subsequently, each sample is normalized and translated to the central perspective, which is the same approach as that used earlier [25].

4.3. Ablation Study

In this section, we examine the effectiveness of proposed DGN block, adaptive graph strategy and two-stream framework. The recognition accuracy is used as the evaluation indicator.

4.3.1 DGN block

First, we evaluate the necessity of applying the DGN block to combine the bone information and joint information. Table 1 shows the results. We use ST-GCN [34] as the baseline method. Due to the adjustment of the learning-rate scheduler and data preprocessing, we obtain a higher recognition accuracy (92.7%) than the results in the original paper (88.7%). 2s-ST-GCN indicates that the joint information and bone information are modeled with two ST-GCNs separately and are fused by adding the predicted scores of the $softmax$ layers. This approach achieves better performance than using only the joint information, which shows the importance of using bone information. We also test the addition of a fully-connected layer or pooling-based methods to fuse the $softmax$ scores, which results in similar accuracies as adding them directly. 1s-ST-GCN indicates that the joint information and bone information are concatenated along the channel dimension and fed into the ST-GCN, whose number of channels in each layer is twice the original number. Better performance is obtained than with 2s-ST-GCN, possibly because of the deep fusion of two modalities of information caused by concatenation. Then we test our DGNN model in the same condition. Since the graph structure in ST-GCN is multiplied by a mask, we also fix the graph structure of DGNN and multiply using a mask with the incidence matrix for fair comparison. The resulting model is called masked DGNN. The final result shows that our masked DGNN model achieves better performance than 1s-ST-GCN. Thus, the superiority of our fusion strategy, which fully exploit the graph structures of skeletons and the dependencies between joints and bones, is verified.

Method	Accuracy
ST-GCN	92.7
2s-ST-GCN	93.4
1s-ST-GCN	93.7
Masked DGNN	95.0

Table 1. Comparisons of the recognition accuracy (%) for ST-GCNs and the masked DGNN.

4.3.2 Adaptive DGN blocks

We test four strategies to make the graph adaptive: (1) Similar to ST-GCN, we multiply a mask P by the original incidence matrix A, which is set as the model parameter and initialized to 1 (marked as PA in Table 2); (2) We set P as a residual connection, which is initialized as 0 and added to A (marked as $P + A$); (3) We directly set the incidence matrix as the parameter P, which is initialized with A (marked as P_0); and (4) Similar to (3), the incidence matrix is set as the parameter of the model and initialized with A but fixed at the first 10 epochs (marked as P_{10}). We also test the performance without the adaptive graph strategy (marked as A). Table 2 shows the results; the P_{10} strategy is found to provide the best performance. This supports our design strategy as described in Section 3.3.3.

Method	A	PA	$P+A$	P_0	P_{10}
Accuracy	94.4	95.0	95.3	95.2	95.5

Table 2. Comparisons of the recognition accuracy (%) for different adaptive graph strategies.

4.3.3 Two-Stream framework

To test the necessity of using the motion information, we compare the performance of using spatial information and motion information separately with the performance of fusing two streams in both the NTU-RGBD dataset and Skeleton-Kinetics dataset as shown in Table 3. We found that fusing the spatial information and motion information improves the performance on all of the benchmarks, which verifies the superiority of the proposed method.

Method	NTU(cv)	NTU(cs)	SK(t1)	SK(t5)
Spatial	95.5	89.2	36.1	58.7
Motion	93.8	86.8	31.8	54.8
Fusion	96.1	89.9	36.9	59.6

Table 3. Comparisons of the recognition accuracy (%) using spatial information, motion information, and the fusion of two modalities. SK denotes the Skeleton-Kinetics dataset; t1 and t5 denote the top-1 and top-5 accuracy, respectively.

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

To show the superiority and generality of our method, the model is compared with state-of-the-art methods using both the NTU-RGBD dataset and Skeleton-Kinetics dataset. We divide these methods into four classes, including handcraft-feature-based methods, RNN-based methods, CNN-based methods and GCN-based methods, and split them with a horizontal line in the table of results. With the NTU-RGBD dataset, our model is tested on both cross-view (CV) and cross-sub (CS) benchmarks as shown in Table 4. The performance of the deep-learning-based methods is generally better than that of handcraft-feature-based methods, and CNN-based methods are generally better than RNN-based methods. Our model outperforms these methods with a large margin, which verifies the superiority of our model for skeleton-based action recognition.

The Skeleton-Kinetics dataset is larger and more challenging than the NTU-RGBD dataset due to the diversity of videos collected from YouTube. We report the top-1 and top-5 recognition accuracies in Table 5. The results are identical to the experiments on NTU-RGBD, where our model shows the best performance. Such results confirm the generality capability of our model for large-scale datasets.

Method	CS	CV
Lie Group [30]	50.1	82.8
HBRNN [7]	59.1	64.0
Deep LSTM [25]	60.7	67.3
ST-LSTM [20]	69.2	77.7
STA-LSTM [28]	73.4	81.2
VA-LSTM [37]	79.2	87.7
ARRN-LSTM [18]	80.7	88.8
TCN [14]	74.3	83.1
Clips+CNN+MTLN [13]	79.6	84.8
Synthesized CNN [21]	80.0	87.2
3scale ResNet152 [16]	85.0	92.3
ST-GCN [34]	81.5	88.3
DPRL+GCNN [29]	83.5	89.8
DGNN (ours)	89.9	96.1

Table 4. Comparisons of the recognition accuracy (%) with the state-of-the-art methods on the NTU-RGBD dataset.

Method	Top-1 (%)	Top-5 (%)
Feature Encoding [8]	14.9	25.8
Deep LSTM [25]	16.4	35.3
TCN [14]	20.3	40.0
ST-GCN [34]	30.7	52.8
DGNN (ours)	36.9	59.6

Table 5. Comparisons of recognition accuracy (%) with the state-of-the-art methods on the Skeleton-Kinetics dataset.

5. Conclusion

In this work, we represent both joint and bone information as a directed acyclic graph and design a customized novel directed graph neural network (DGNN) to predict action based on the constructed graph. In addition, we make the graph structure adaptive to better fit the multilayer architecture and the recognition task. Furthermore, the motion information between consecutive frames is extracted to model the temporal information of a skeleton sequence, and both the spatial and motion information are fused in a two-stream framework. The final model exceeds current state-of-the-art performance on two large-scale datasets: NTU-RGBD and Skeleton-Kinetics. Future work might focus on how to exploit the skeleton data and RGB data together. In addition, exploration is recommended into how to combine the problem of pose estimation with skeleton-based action recognition in a unified architecture.

Acknowledgement This work was supported in part by the National Natural Science Foundation of China under Grant 61572500, 61876182 and 61872364, and in part by the State Grid Corporation Science and Technology Project.

References

[1] Peter Battaglia, Razvan Pascanu, Matthew Lai, and Danilo Jimenez Rezende. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016. 2

[2] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, and Ryan Faulkner. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 2

[3] C. Cao, C. Lan, Y. Zhang, W. Zeng, H. Lu, and Y. Zhang. Skeleton-Based Action Recognition with Gated Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems for Video Technology*, pages 1–1, 2018. 1, 2

[4] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 1, 7

[5] Joao Carreira and Andrew Zisserman. Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 6

[6] Y. Du, Y. Fu, and L. Wang. Skeleton based action recognition with convolutional neural network. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 579–583, 2015. 2, 4

[7] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1110–1118, 2015. 1, 8

[8] Basura Fernando, Efstratios Gavves, Jose M. Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387, 2015. 1, 2, 8

[9] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. *CoRR*, 2017. 2

[10] Kaiming He, Georgia Gkioxari, Piotr Dollr, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 1

[11] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation Networks for Object Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[12] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, and others. The Kinetics Human Action Video Dataset. *arXiv preprint arXiv:1705.06950*, 2017. 7

[13] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Ahmed Sohel, and Farid Boussad. A New Representation of Skeleton Sequences for 3d Action Recognition. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4570–4579, 2017. 1, 8

[14] Tae Soo Kim and Austin Reiter. Interpretable 3d human action analysis with temporal convolutional networks. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1623–1631, 2017. 1, 2, 8

[15] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning (ICML)*, 2018. 2

[16] Bo Li, Yuchao Dai, Xuelian Cheng, Huahui Chen, Yi Lin, and Mingyi He. Skeleton based action recognition using translation-scale invariant image mapping and multi-scale deep CNN. In *Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on*, pages 601–604. IEEE, 2017. 1, 2, 4, 8

[17] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Skeleton-based action recognition with convolutional neural networks. In *Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on*, pages 597–600. IEEE, 2017. 1

[18] Lin Li, Wu Zheng, Zhaoxiang Zhang, Yan Huang, and Liang Wang. Skeleton-Based Relational Modeling for Action Recognition. *arXiv:1805.02556 [cs]*, 2018. 1, 2, 3, 8

[19] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (indrnn): Building A longer and deeper RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2018. 1

[20] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-Temporal LSTM with Trust Gates for 3d Human Action Recognition. In *Computer Vision ECCV 2016*, volume 9907, pages 816–833. Springer International Publishing, Cham, 2016. 1, 2, 8

[21] Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition*, 68:346–362, 2017. 1, 2, 8

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS-W*, 2017. 7

[23] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning Spatio-Temporal Representation With Pseudo-3d Residual Networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017. 6

[24] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. A simple neural network module for relational reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4974–4983. Curran Associates, Inc., 2017. 2

[25] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A Large Scale Dataset for 3d Human Activity Analysis. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 2, 4, 6, 7, 8

[26] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Non-Local Graph Convolutional Networks for Skeleton-Based Action Recognition. *arXiv:1805.07694 [cs]*, May 2018. 1, 2, 3, 5

[27] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 1, 2, 6

[28] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An End-to-End Spatio-Temporal Attention Model for Human Action Recognition from Skeleton Data. In *AAAI*, volume 1, pages 4263–4270, 2017. 1, 2, 8

[29] Yansong Tang, Yi Tian, Jiwen Lu, Peiyang Li, and Jie Zhou. Deep Progressive Reinforcement Learning for Skeleton-Based Action Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 8

[30] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014. 1, 2, 8

[31] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European Conference on Computer Vision*, 2016. 1, 2, 6

[32] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-Local Neural Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[33] Xiaolong Wang and Abhinav Gupta. Videos as Space-Time Region Graphs. *arXiv preprint arXiv:1806.01810*, 2018. 2

[34] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In *AAAI*, 2018. 1, 2, 5, 6, 7, 8

[35] Yi Yang and Deva Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1385–1392. IEEE, 2011. 3

[36] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015. 1

[37] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View Adaptive Recurrent Neural Networks for High Performance Human Action Recognition From Skeleton Data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2126, 2017. 1, 2, 8

[38] Yifan Zhang, Congqi Cao, Jian Cheng, and Hanqing Lu. EgoGesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition. *IEEE Transactions on Multimedia*, pages 1–1, 2018. 1

[39] Beiji Zou, Shu Chen, Cao Shi, and Umugwaneza Marie Providence. Automatic reconstruction of 3d human motion pose from uncalibrated monocular video sequences based on markerless human motion tracking. *Pattern Recognition*, 42(7):1559–1571, 2009. 3