DDGCN: A Dynamic Directed Graph Convolutional Network for Action Recognition

Matthew Korban and Xin Li*

Louisiana State University
{mzadgh1, xinli}@lsu.edu

Abstract. We propose a Dynamic Directed Graph Convolutional Network (DDGCN) to model spatial and temporal features of human actions from their skeletal representations. The DDGCN consists of three new feature modeling modules: (1) Dynamic Convolutional Sampling (DCS), (2) Dynamic Convolutional Weight (DCW) assignment, and (3) Directed Graph Spatial-Temporal (DGST) feature extraction. Comprehensive experiments show that the DDGCN outperforms existing state-of-the-art action recognition approaches in various testing datasets.

Keywords: Action Modeling and Recognition; Graph Convolutional Network; Dynamic Spatiotemporal Graph

1 Introduction

Human action recognition is an active research topic that attracted great attention in recent years [34]. It has broad applications in video analysis and annotation, content retrieval, human-computer interaction, virtual reality, and so on. However, action recognition remains challenging when the videos have noisy background with complex occlusion or illumination conditions, changing camera view angles, or inconsistency between individuals' motions and their semantics (e.g., different people could perform semantically similar motions differently). The majority of action recognition and analysis algorithms directly model action features on images using deep Convolutional Neural Networks (CNNs) [6]. But image-based approaches are usually sensitive to the aforementioned noisy background, occlusions, and different camera viewpoints. Another modality to model human actions is through human skeletons. The skeleton modality has some advantages over the image modality for its more compact representation, better robustness against occlusion and viewpoint change, and higher expressive power in capturing features in both temporal and spatial domains [18]. An appropriate way to represent human skeletons is using graphs where skeleton joints and bones are defined as graph nodes and edges respectively. Then to extract features from graphs one can use the Graph Convolutional Network (GCN), whose effectiveness is demonstrated in recent action recognition work [30].

* Corresponding author.

Many GCN-based action recognition approaches use multi-stream networks to process spatial and temporal information of skeleton graphs separately [33, 8], which are usually complex and computationally costly. Recently, the *Spatial-Temporal (ST) graph* is introduced [30] to represent the skeletal graph sequence. A key advantage of using the ST graph is its capability to build a single end-to-end network that comes with better efficiency. Nevertheless, GCN-based action recognition methods still have their own limitations. Our observations are that solving the following two issues could enhance the performance of GCN in action recognition. First, there are spatial-temporal correlations between different parts of a human skeleton. Exploring such correlation patterns helps improve the modeling and recognition of actions. But such correlations are dynamic and varied for different human actions in both spatial and temporal domains. Hence, extracting these correlations effectively is difficult. The standard convolutional operations commonly adopted in traditional GCN [30] are static and only describes spatial correlations between neighboring nodes, thus, cannot capture such dynamic spatial-temporal correlations properly. Second, the spatial hierarchical structure of skeletons and the temporal sequential properties of motions both encode order information that is important in action recognition. But most existing ST graph models [30] describe the actions using undirected graphs, which cannot capture such order information.

To tackle these issues, we propose an end-to-end *Dynamic Directed Graph Convolutional Network* (DDGCN), to recognize human actions on ST graphs. We develop three new modules that can adaptively learn the spatiotemporal correlations and model spatial and temporal order information in actions:

Dynamic Convolutional Sampling (DCS). In action ST graphs, the relationship between spatially or temporally correlated joints provides useful information. We call this relationship *ST correlations* and describe it using a feature vector $f_{ST}(v)$ on each node v. We compute $f_{ST}(v)$ using a convolution of shared kernel weights W on an ST graph node v and its neighboring node set $B(v)$. $B(v)$ includes v's spatiotemporal correlated nodes. We observe that *ST-correlations* among nodes, and hence $B(v)$, are varied for different actions. Hence, unlike existing approaches, we propose to dynamically model such ST-correlations and compute each node's neighboring node set from the data. We design a novel Dynamic Convolutional Sampling (DCS) module (Section 3.2) to define B adaptively using ST correlations explored in different actions.

Dynamic Convolutional Weights (DCW). To perform an element-wise ordered convolution within the neighbor $B(v)$ of a node v, we need to assign the learned weights W (of the convolution kernels) to v's neighboring nodes. However, the spatial order of neighboring nodes in a graph is often ambiguous. To make our proposed GCN order-invariant, we develop a Dynamic Convolutional Sampling (DCW) module (Section 3.3) to compute the order of weights W in an adaptive and dynamic procedure.

Directed Spatial-Temporal Graph (DSTG) Features. The inputs to DDGCN are ST graphs created by spatial and temporal connections in actions. However, existing ST graphs are usually designed as undirected graphs [30],

which cannot capture spatial and temporal order information effectively. However, such order information encodes important attributes of actions. Hence, we propose to use a Directed Spatial-Temporal Graph (DSTG), and develop a DSTG feature extraction module to capture such order information and make the action features more (spatially) structure-aware and (temporally) order-aware.

The main **contributions** of this work are as follows:

- We propose a new Dynamic Directed GCN (DDGCN) architecture to model the spatial and temporal correlations effectively in human actions. Our comprehensive experiments showed that DDGCN outperforms existing state-of-the-art approaches on various public benchmarks.
- We develop two new modules, DCS and DCW, to make DDGCN dynamic and action-adaptive. These new modules can effectively capture ST correlations exhibited among non-adjacent joints.
- We develop a new DSTG feature extraction module to enhance the action feature modeling by including spatial-temporal order information.

2 Related Work

Action recognition algorithms can be classified based on data modalities they run on. The majority of action recognition methods model actions on image sequence directly. Accordingly, they have developed various strategies based on handcrafted features [31, 26, 15], Convolutional Neural Network (CNN) [2, 9, 27], or Generative Adversarial Network (GAN) [14, 25, 29] to perform action recognition. However, using only appearance modality such as RGB images has its limitations including high inference of background, high dimensional inputs, sensitivity to image transformations, and low expressive capability.

To enhance the expressive capability of the appearance modality, some researchers add a new depth dimension to the modality to help extract features from actions. Kamel et al. [8] proposed an Action-Fusion network using both depth maps and posture data based on three CNNs defined on different modalities. Zhang et al. [33] used a multi-stream deep neural network to learn the motion attributes based on depth and joints inputs, and then represented the motions based on the combination of their attributes. However, depth images are sensitive to background inference and local transformations.

Compared with the depth images, motion flows are less sensitive to background inference and has a greater expressive capability. Some recent studies aimed to effectively compute motion flows. Piergiovanni et al. [16] proposed a method to reduce the computational cost of generating optical flows by capturing the flow within the model where the flow parameters are iteratively optimized by jointly learning other CNN model parameters. Sun et al. [22] designed a compact motion representation named Optical Flow guided Feature (OFF) that allows the CNN to extract the spatial-temporal information required for computing the flow between frames. Despite its better reliability, the motion flow is in general expensive to compute and still has limited capability in modeling moving background and dynamic motions.

The skeleton is a compact and expressive modality, and is insensitive to dynamic background and changing camera viewpoint. Li et al. [13] suggested an approach to capture richer motion-specific dependencies by using an encoder-decoder structure. The network automatically creates actional and structural links representing motions where each encoder-decoder block is called as actional-structural graph convolution network. Si et al. [20] proposed a method to extract high-level spatial-temporal features using a novel Attention Enhanced Graph Convolutional LSTM Network. The network captures co-occurrence relationship between spatial and temporal domains where the spatial features extracted by the GCN is fed to the LTSM to extract temporal dependencies.

3 Methodology

3.1 Overview

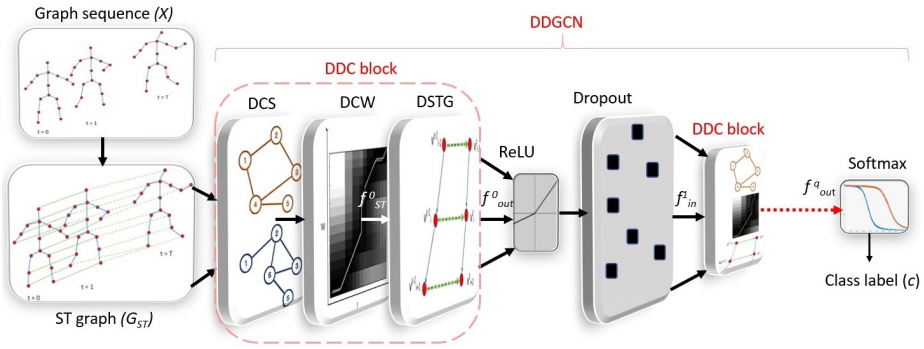


Fig. 1. An overview of the proposed action recognition pipeline.

The main pipeline of our proposed end-to-end action recognition pipeline DDGCN is illustrated in Fig. 1. Given an action sequence X, DDGCN outputs its action class label $c \in \{0, 1, ..., C\}$, C being the number of classes. Let action X be a sequence of skeletal graphs $X = \{G_t, t = 1, 2, ..., T\}$, where each skeletal graph in time t, $G_t = \{V_t, E_t^S\}$ consists of node (joint) set $V_t = \{v_k \in R^3, k = 0, 1, ..., m\}$ and spatial edge (bone) set $E_t^S = \{v_i^t v_j^t | (i, j) \in H\}$, where m is the skeletal node number. Here H is the set of bones (spatial edges) connecting joints (nodes) in a static human body skeleton template; subscripts i and j are the indices of joints, t is the index of the time frame, and T is the length of action sequence.

Spatial-Temporal Graph Construction. From a skeletal graph sequence, we construct an ST graph $G_{ST} = (V, E)$, following the notation given in [30], to store all the skeletal joints and their spatial and temporal relationship. Specifically, nodes $V = \{v\}$ correspond to skeletal joints and are connected by edges E

following two rules. (1) Spatially, joints on the same skeleton are connected by spatial edges E^S. (2) Temporally, each node and its counterparts in the previous and next frames are connected by temporal edges, i.e., $\{v_i^t v_i^{t+1}\} \subset E^T$.

Dynamic Directed Convolutional (DDC). The DGCNN takes the ST graph as its input, and builds the feature maps F_{out} using multiple DDC blocks (Fig. 1). Each DDC block consists of (1) two dynamic convolutional modules, namely, *Dynamic Convolutional Sampling (DCS)* and *Dynamic Convolutional Weight (DCW)* assignment, and (2) a *Directed Spatial-Temporal Graph (DSTG)* feature extraction module that captures the spatial-temporal order information. The last, say q-th, DDC block outputs a probability feature vector f_{out}^q, and it is finally converted to a resultant one-hot vector c through a softmax operator.

3.2 Dynamic Convolutional Sampling (DCS)

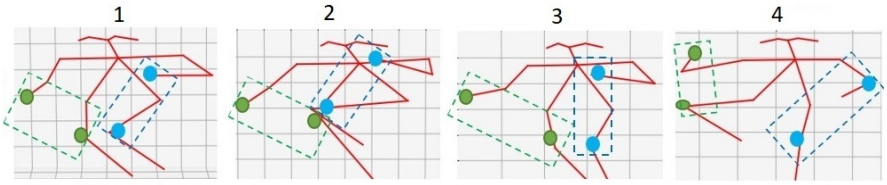
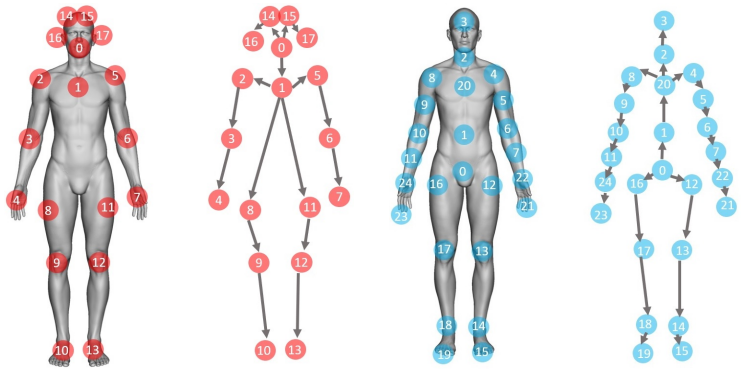


Fig. 2. Sample frames of a running motion, in which correlated patterns can be found on non-adjacent nodes

.



(a) Skeletal graph template from [4] (b) Skeletal graph template of [17]

Fig. 3. Our two (directed) skeletal graph representations, where the joint indexing follows (a) [4] and (b) [17].

Adjacent sub-parts of a human body are often correlated in human actions. Meanwhile, in many actions, some non-adjacent sub-parts are also correlated. Fig. 2 illustrates an example of running motion, where arm and leg joints exhibit similar spatial-temporal patterns and are correlated. But ST correlations of sub-parts are varied in different actions. So they need to be captured dynamically according to the data. Dai et al. [5] developed a procedure to find the non-local neighbors of a given pixel in an image. Inspired by this, we have generalized this idea to graphs to develop a Dynamic Convolutional Sampling (DCS) module to identify correlated non-adjacent nodes.

DCS runs on a static graph G_0 defined by the given skeleton template. In other words, G_0 is from a specific time frame of the ST-graph G_{ST}. For each node v in a G_0, we first define its *correlative nodes* on edges connecting v and its neighbors. Note that v's neighboring nodes could contain not only its adjacent nodes in G_0, but also other non-adjacent nodes that are correlated (i.e., exhibiting correlated motion patterns). We initiate v's *neighbor list* $B(v)$ with its adjacent nodes in the G_0, then the *dynamic convolutional sampling* (DCS) algorithm will update $B(v)$ to include those non-adjacent nodes according to correlated patterns exploited from action X. Specifically, the DCS computes the feature values on each node v_i in two steps. (1) First, we detect all the correlative node pairs $\{(v_i, v_j)\}, v_i, v_j \in G_0$, then we include those non-adjacent nodes to each node v_i's neighboring set $B(v_i)$ accordingly. We connect each v_i and its newly included non-adjacent neighbor v_j using a new edge. The update on $B(v_i)$ is done by performing a dynamic sampling procedure on v_i's non-adjacent correlative nodes. This dynamic procedure adaptively samples v_i's correlative nodes indexes and update their order using an index shift (offset). We use a function p_i to indicate the sampling for v_i, and Δp_i to denote an index shift. So $p_i(B(v_i)) + \Delta p_i(B(v_i))$ orders all the nodes $v_j \in B(v_i)$ and outputs a list of indexes (a permutation of these nodes). We iteratively update the order of v_i's neighboring nodes to find a better ordering, or offsets, such that under the new neighbor sampling, the recognition accuracy improves, namely, the recognition loss L decreases. (2) Second, on each edge connecting v_i and its neighbor $v_j \in B(v_i)$, we aggregate the correlative information indicating relative information of node v_i with respect to v_j to get the *correlative features*. Such correlative features can be defined as $f(v_i, p_i(B(v_i)) + \Delta p_i(B(v_i)))$ with respective to node v_i and its neighbor $v_j = p_i(B(v_i)) + \Delta p_i(B(v_i))$.

This **DCS Algorithm** can be summarized as follows.

1) initialize the static graph G_S following a skeleton template, and initialize indexes of all the nodes accordingly;
2) initialize neighbor sampling: for $\forall v_i \in G_S$, create its initial ordered neighboring set $p_i(B(v_i))$ in two steps:
   - create an ordered node set O_i including all the other nodes in the graph sorted by their graph distance to v_i. When two nodes v_j and v_r have the same graph distance (e.g., both are r-hop away from v_i), then sort them based on their initialized indexes;
   - given a kernel size r, pick the first r nodes from O_i, these nodes form the ordered neighboring set in this step $p_i(B(v_i))$;

3) update sampling neighbors: $\forall v_i$, update the index offsets and neighbor sampling by learning optimal offsets Δp_i that reduces recognition loss L.

Finally, on G_{ST}, the feature map f_{ST} is computed through a graph convolution following Equation (1):

$$f_{ST}(v_i) = \sum_{v_j \in B(v_i)} w(v_i) \cdot (p_i(v_j) + \Delta p_i(v_j)), \tag{1}$$

where i and j are indices for the central and neighboring sampling nodes respectively, B is the dynamic neighboring sampling nodes set, w is the dynamic weight function, p_i is the dynamic neighboring sampling function and Δp_i is the offset sampling function.

3.3 Dynamic Convolutional Weights

Both CNN and GCN extract features from input data by performing convolutions. On an image, each pixel's neighboring pixels are spatially ordered and the convolution kernel weights are learned following the same order. On a graph, however, each node's adjacent nodes are often unordered, and the number of neighbors could vary. To make our graph convolution order-invariant and valence-insensitive, we develop a DCW weight assignment module to order the computed convolution weights adaptively.

Inspired by [7] that aligns/re-orders weights for each pixel's neighboring pixels on images in CNN's convolutions, we propose a DCW assignment scheme to compute the order/assignment of weights in W on graph nodes. This makes the convolution order-insensitive; and also improves the GCN performance.

Specifically, given a node v and its neighboring nodes $B(v) = \{v_i, i = 1, \ldots, K\}$, where K is the size of neighboring nodes, our DCW assignment re-orders the kernel weights $W = \{w_i \in \mathcal{R}^3, i = 1, \ldots, r\}$ so that w_i is dynamically assigned to the corresponding node v_i. We compute this assignment as a $r \times 2$ matrix $P_v = DTW_{path}(W, B(v))$ that minimizes the distance between the two vectors $B(v)$ and the re-ordered W. The first column in P_v defines the ordered indices of elements in W, and the second column indicates the selected elements and their order in $B(v)$. We compute P_v, using the Dynamic Time Warping (DTW) algorithm [1].

With this dynamically computed P_v, W is adjusted according to $B(v)$, making the GCN convolution order-insensitive, and also, capturing the feature patterns in $B(v)$ more effectively.

The DCW assignment naturally handles the varied size of $B(v)$ through this assignment. While the kernel size r is fixed, the size of $B(v)$, denoted as K could change. Note that K is v's valence plus the size of its non-adjacent correlated vertex set. If K is larger than r, then the top-r significant nodes will be considered (using the DTW algorithm) while the others ignored. This flexibility allows us to use shared fixed-size kernels to have a fully-connected layer without over-adjusting the hyperparemeters [24].

3.4 Directed Spatial-Temporal Graph

We design a Directed Spatial-Temporal Graph (DSTG) feature extraction module to enhance the initial features f_{ST} obtained from the DCS and DCW modules. DSTG considers the spatial and temporal order information of actions which follow hierarchical (spatial) structure of human skeleton and sequential (temporal) properties of human motions respectively. Unlike existing approaches such as [19] that incorporate spatial and temporal order information using 3D joint/bone coordinates, the DSTG uses a high-dimension feature vector pointing from the feature vector of a joint v_i to the feature vector of its correlative joint v_j. The advantage of our method is that it can jointly learn bones and temporal high dimensional features using the spatial-temporal (ST) representation on the ST graph through an end-to-end network. For each node v_i we assign a feature vector $F_i = \{f_i^J, f_i^B, f_i^T\}$ which is the concatenation of three features vectors of joint features f_i^J, bone features f_i^B and temporal features f_i^T.

Directed Spatial Graph. We model the human body skeleton following standard templates. Two widely used skeletal graph templates are from [3] and [17]. Existing ST graphs are developed to be undirected. Based on observations we discussed previously, modeling spatial and temporal information in order is beneficial. Therefore, we change these undirected graphs to directed graphs. We define bone directions following a breadth-first search traversal from the root. The resultant edge directions are illustrated in Fig. 3. The hierarchical structure of skeleton joints is represented as directed bone vectors connecting adjacent joints. Each joint v_i is spatially correlated to its parent node v_{i-1} with a bone $b_i = \overrightarrow{v_{i-1}v_i}$. In this hierarchical structure, movement of a parent joint usually affects its children joints. In the opposite direction, the relationship may not be the same though. Following these bone vectors, we define the **bone features** as $f_i^B = \overrightarrow{f_{i-1}f_i} = f_{i-1} - f_i$, where f_i and f_{i-1} are feature vectors of node v_i and its parent v_{i-1}.

Directed Temporal Graph. Temporal sequence information in actions are important attributes, sometimes referred to as *motion trajectory* [32], in building action features. The temporal sequence order in an ST graph can be defined by directed edges connecting a joint and its counterpart in the next time frame. Such information has not been properly modeled in existing ST graph based approaches. To exploit such information, in ST graphs we calculate the **temporal features** f_i^T for each node v_i by $f_i^T = f_i^t - f_i^{t-1}$ where f_i^t and f_i^{t-1} are feature vectors of the node v_i in the current frame with respect to the previous frame.

3.5 Network Architecture

DDC Block. We compose the DCS, DCW, and DSTG modules to form a DDC block, and build the DDGCN pipeline by integrating multiple DDC blocks, as shown in Fig. 4. The first DDC block DDC_0 takes in an ST skeleton sequence, $f_0 \in \mathcal{R}^{T\times J\times 3}$, composed from an ST-graph, where T and J are frame and joint

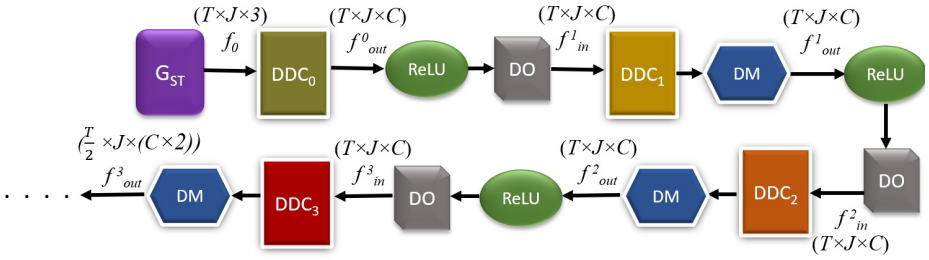


Fig. 4. The DDGCN architecture consisting of DDC blocks, Dropout (DO) layers, Dimension Modifier (DM) and ReLU.

numbers and 3 indicates the 3D coordinates. DDC_0 outputs the feature maps $f^0_{out} \in \mathcal{R}^{T\times J\times C}$ where C is the channel size. The ST-graph is processed by the DCS to produce dynamic neighboring sets for nodes, upon which the DCW is applied to create ST correlations f_{ST} and then the final features f_{out} are created by the DSTG module. The output features from a DDC block is processed by a ReLU layer then regularized by a DropOut (DO) layer to prevent over-fitting.

Composing Multiple DDC Blocks. We then build our DDGCN pipeline using multiple (9 in the current implementation) DDC blocks, as shown in Fig. 4. Except for the block DDC_0, a subsequent q-th DCC block DDC_q takes in a feature map $f^q_{in} \in \mathcal{R}^{T\times J\times C}$ and outputs a feature map $f^q_{out} \in \mathcal{R}^{T'\times J\times C'}$, where C and C' are the channel sizes. $T' = \frac{T}{2}$ if $q = 3k, k = \{1, 2\}$ and $T' = T$ for other q values, $C' = C \times 2$ if $q = 3k, k = 1, 2, \ldots$ and $C' = C$ for other q values. The dimension of the DCC output is controlled by a Dimension Modifier (DM).

Network Parameters. Our current implementation has 9 DDC blocks. The dimension for f^i_out for $i = 1, 2, 3$ is $(300, 18, 64)$; for $i = 4, 5, 6$ and $i = 7, 8, 9$, they are $(150, 18, 128)$ and $(75, 18, 256)$, respectively. In our network, the learning rate, dropout probability, and kernel size are 0.01, 0.5 and 3 respectively.

4 Experimental Results

4.1 Datasets and Evaluation Metrics

We evaluated our proposed action recognition algorithm on two public datasets.

1) **Kinect** [10] contains around $300,000$ video clips retrieved from YouTube. The videos cover 400 human action classes and each clip is 30 seconds. The joint locations were extracted from the video clips using the open-source algorithm OpenPose [4]. Following the commonly adopted evaluation in recent action recognition algorithms on this dataset, among these samples we used $240,000$ for training and $20,000$ for testing, and to evaluate the recognition accuracy we used *two metrics*: (1) *Top-1 accuracy* (i.e., how often the highest classification score, or our prediction, corresponds to the correct label) and (2) *Top-5 accuracy* (i.e., how often the correct label corresponds is in one of the top-five predictions).

2) **NTU-RGB+D** [17] contains 56,000 action clips in 60 action classes, with annotated joint locations. We used these provided skeletons to build our skeletal graph sequence and ST-graphs. Two evaluation metrics developed on this dataset are: (a) *Cross-Subject (CS) Evaluation*: 40 different persons are separated into training and testing groups, each containing 20 persons. The training and testing sets have 40,320 and 16,560 samples, respectively. *Cross-View (CV) Evaluation*: samples from two cameras-views are used for training, and samples from another camera-view are used for testing. The training and testing sets have 37,920 and 18,960 samples, respectively.

Table 1. Comparing action recognition accuracy of our approach with that of other state-of-the-art approaches on NTU-RGB+D [17] dataset.

Metric	DDGCN (ours)	[18]	[20]	[19]	[21]	[23]	[35]	[30]	[11]	[12]
CS	**91.05%**	89.90%	89.20%	88.5%	84.8%	83.5%	81.8%	81.50%	79.6%	83.1%
CV	**97.14%**	96.10%	95.00%	95.1%	92.4%	89.8%	89.0%	88.30%	84.8%	74.3%

Table 2. Comparing action recognition accuracy of our approach with that of other state-of-the-art approaches on Kinect [10] dataset .

Metric	DDGCN (ours)	[18]	[19]	[30]	[12]
top-1	**38.12%**	36.9%	36.1	30.7%	20.3%
top-5	**60.79%**	59.6%	58.7	52.8%	40.0%

4.2 Comparison with Existing Methods on Benchmarks

On the aforementioned benchmarks, we compared our method with existing state-of-the-art methods, including d-GCN (CVPR19) [18], LSTM-GCN (CVPR19) [20], 2S-GCN (CVPR19) [19] , ST-TSL (ECCV18) [21], DPRL (CVPR18) [23], Bayesian-GCN (ICCV19) [35], ST-GCN (AAAI18) [30], LSTM-CNN (CVPR17) [11], and T-CNN (CVPR17) [12]. The NTU-RGB+D results are shown in Table 1, and the Kinect results are shown in Table 2. Note that the performance of some approaches is only available on NTU-RGB+D dataset, so Table 2 has a shorter list of compared methods. In both benchmarks, our proposed DDGCN outperforms the existing methods in recognition accuracy.

For qualitative (visual) examination and comparison, we also randomly selected a set of testing action samples from the benchmark (the Kinect dataset) and illustrate the results from DDGCN (ours), ST-GCN [30], d-GCN [18], and 2S-GCN [19] (only these papers released their codes). For example, we picked the videos with indexes number $30 \times k, k = 1, \ldots$, so actions $\#30, \#60, \ldots$ from the Kinect dataset and perform the comparison. Some recognition results are

shown in Table 3. Some of these actions are and their recognition results are visualized in Fig. 5, and more are provided in the supplemental video.

Table 3. Comparing our DDGCN with other methods on randomly selected sample actions from the Kinect dataset. The actions with ground truth labels are listed on the left column; each method's results are reported in the table.

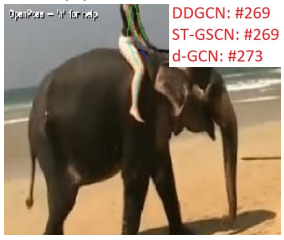
Action Sequencee	DDGCN	ST-GCN [30]	d-GCN [18]	2S-GCN [19]
class #30 (bookbinding)	**#30**	**#30**	**#30**	#137
class #60 (clean and jerk)	**#60**	**#60**	**#60**	**#60**
class #90 (decorating Christmas)	**#90**	#315	#15	**#90**
class #120 (exercising arm)	**#120**	#224	#217	#372
class #150 (headbanging)	#96	#353	**#150**	**#150**
class #180 (krumping)	**#180**	**#180**	#31	**#180**

(a) Action #30

(b) Action #60

(c) Action #90

(d) Action #150



(e) Action #270

(f) Action #300

Fig. 5. Recognition of action samples (a) #30 (bookbinding), (b) #60 clean and jerk, (c) #90 (decorating Christmas tree), (d) #150 (headbanging), (e) #270 (riding elephant), and (f) #300 (shoveling snow) from the Kinect dataset [10].

4.3 Real-time Experiments

We also conducted our real-time experiments by having a volunteer performing actions defined in the Kinect dataset. Some comparison results are illustrated in Fig. 6, and more results are provided in the supplementary video.

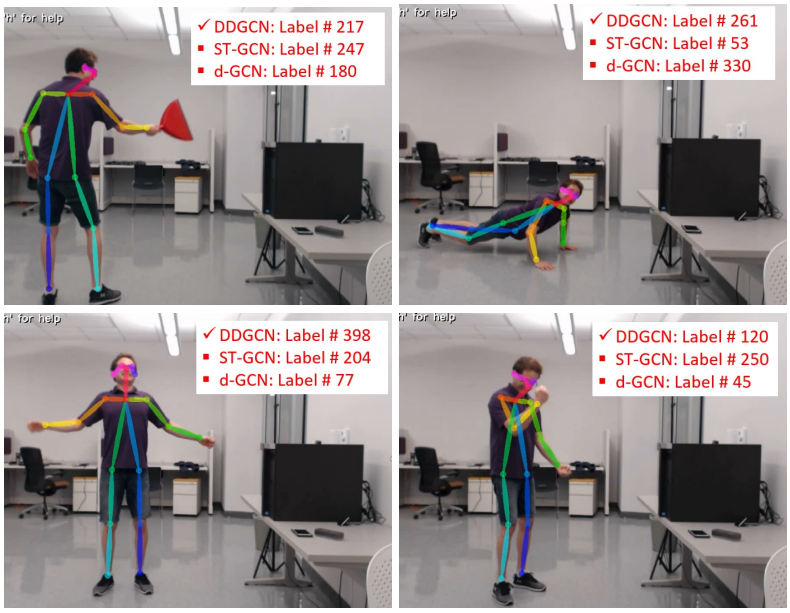


Fig. 6. Recognizing user-performed actions #217 (plastering), #261 (pushing up), #398 (yawning), and #120 (exercising arm), following actions defined in Kinect [10].

4.4 Ablation Study

Table 4. Comparing effectiveness of different modules in the DDC block.

Dataset/Models	Baseline	DCS	DCW	DSTG	DCS + DCW +DSTG
Kinect/top-1	30.7%	34.6%	32.1%	35.5%	**38.12%**
Kinect/top-5	52.8%	55.3%	54.6%	58.1%	**60.79%**
NTU-RGB+D/CS	81.5%	84.5%	81.9%	85.4%	**91.05%**
NTU-RGB+D/CV	88.3%	90.3%	89.4%	92.6%	**97.14%**

We designed an ablation study to evaluate the effectiveness of different components in the DDC block and also their combinations. We choose [30] as the Baseline, where standard convolutional layers and GCN blocks are used. Our proposed DDGCN is designed based on three new modules DCS, DCW, and DSTG. The recognition results are reported in Table 4. From these results, we can see that incorporating DCS, DCW, and DSTG brings 3.9% , 1.4% and 4.8% performance gains respectively for the Kinect/top-1. On other datasets/metrics, these components demonstrate similar performance improvement. Using the full DDC block yields the best accuracy.

4.5 Recognition of Incomplete Actions

Real-world videos often contain incomplete actions. Occlusion, fast motion, changing illumination, or hardware noise could result in blocked body parts, missing frames, or incomplete action sequences [28]. Many existing methods were not designed to address such incomplete action data specifically and cannot recognize them properly. Incomplete actions can sometimes be ambiguous, as similar motion subroutines could exist in different actions, and without a long enough sequence, sometimes they cannot be effectively differentiated. Our DGCNN can enhance the extraction of Spatial-Temporal dependencies and order information. Hence, it could tackle incomplete actions more reliably.

We intentionally removed some frames in the benchmark action videos and conduct experiments to compare our method's recognition performance. We simulated three types of incompleteness: (1) missing frames at the beginning of the motion, (2) missing frames at the end, and (3) random missing frames in the sequence. Different levels of data incompletion are tested on the NTU-RGB+D/CS dataset; each testing experiment run on $16.5k$ action samples.

We compared our algorithm with three other methods d-GCN (CVPR19) [18], 2S-GCN (CVPR19) [19] and ST-GCN (AAAI18) [30], whose source codes were released. The experimental results on the aforementioned three incompleteness types are reported in Tables 5 and 6. The results indicate the better robustness of the DDGCN over the existing approaches, especially when the missing rates are high. In most action sequences, we found a major portion of the features/patterns exists during the beginning of the action, hence, removing frames at the beginning of the sequence makes the recognition harder. This is also demonstrated in the experiments: when the first 30% of frames at the beginning of a sequence are removed, the recognition accuracy drops dramatically; but when a same amount of frames are removed at the end of a video, the recognition is still effective.

Table 5. Recognition of actions with missing frames at the beginning.

Missing Rate	DDGCN	ST-GCN [30]	d-GCN [18]	2S-GCN [19]
0%	**91.5%**	81.5%	89.8%	88.3%
10%	**82.3%**	65.2%	72.5%	71.7%
20%	**43.7%**	28.5%	31.9%	30.9%
30%	**21.1%**	5.9%	9.1%	10.3%

5 Conclusions

We have developed a Dynamic Directed Graph Convolutional Network (DDGCN) algorithm for action recognition based on skeleton graphs. The DDGCN consists of three new modules: Dynamic Convolutional Sampling (DCS), Dynamic Convolutional Weight (DCW) assignment, and Directed Graph Spatial-Temporal

Table 6. Recognition of actions with missing frames, where (1) missing frames are at the end (End), and (2) frames are randomly missing (Rand). The two numbers reported here reflect the recognition accuracy under these two cases, respectively.

Missing Rate	DDGCN	ST-GCN [30]	d-GCN [18]	2S-GCN [19]
0% (End / Rand)	**91.5% / 91.5%**	81.5% / 81.5%	89.8% / 89.8%	88.4% / 88.4%
30% (End / Rand)	**91.5% / 87.9%**	81.5% / 76.0%	89.8% / 85.2%	88.3% / 85.3%
60% (End / Rand)	**91.2% / 79.0%**	80.4%	82.3%	82.5%
70% (End / Rand)	**86.3% / 68.8%**	72.5%	76.9%	79.5%
80% (End / Rand)	**65.3% / 51.3%**	48.1%	51.1%	55.9%

(DGST) features extraction. These new modules help better capture the spatial-temporal dependencies as well as hierarchical and sequential structures for human action modeling and recognition. Experiments demonstrated that DDGCN has outperformed the state-of-the-art algorithms in action recognition accuracy on multiple public benchmarks.

Limitations and Future Work. The recognition accuracy on the Kinect dataset is significantly lower than the NTU dataset. A main reason is that the Kinect data include some videos that have severe occlusion and missing joints. Some failure examples are illustrated in Fig. 5 (d,e). In both examples, the skeletons are significantly truncated and incomplete, resulting in incorrect action identifications. Currently, the DDGCN (and other existing methods) has not designed specific module to tackle missing joints. We will explore this for action recognition in unconstrained videos.

References

1. Berndt, D.J., Clifford, J.: Using dynamic time warping to find patterns in time series. In: KDD workshop. vol. 10, pp. 359–370. Seattle, WA (1994)
2. Bilen, H., Fernando, B., Gavves, E., Vedaldi, A., Gould, S.: Dynamic image networks for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3034–3042 (2016)
3. Cao, J., Tagliasacchi, A., Olson, M., Zhang, H., Su, Z.: Point cloud skeletons via laplacian based contraction. In: Shape Modeling International (SMI 2010). pp. 187–197. IEEE (2010)
4. Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7291–7299 (2017)
5. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)
6. Herath, S., Harandi, M., Porikli, F.: Going deeper into action recognition: A survey. Image and vision computing **60**, 4–21 (2017)
7. Iwana, B.K., Uchida, S.: Dynamic weight alignment for temporal convolutional neural networks. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3827–3831. IEEE (2019)

8. Kamel, A., Sheng, B., Yang, P., Li, P., Shen, R., Feng, D.D.: Deep convolutional neural networks for human action recognition using depth maps and postures. IEEE Transactions on Systems, Man, and Cybernetics: Systems (2018)
9. Kar, A., Rai, N., Sikka, K., Sharma, G.: Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3376–3385 (2017)
10. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017)
11. Ke, Q., Bennamoun, M., An, S., Sohel, F., Boussaid, F.: A new representation of skeleton sequences for 3d action recognition. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 4570–4579 (2017)
12. Kim, T.S., Reiter, A.: Interpretable 3d human action analysis with temporal convolutional networks. In: 2017 IEEE conference on computer vision and pattern recognition workshops (CVPRW). pp. 1623–1631. IEEE (2017)
13. Li, M., Chen, S., Chen, X., Zhang, Y., Wang, Y., Tian, Q.: Actional-structural graph convolutional networks for skeleton-based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3595–3603 (2019)
14. Mandal, D., Narayan, S., Dwivedi, S.K., Gupta, V., Ahmed, S., Khan, F.S., Shao, L.: Out-of-distribution detection for generalized zero-shot action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9985–9993 (2019)
15. Peng, X., Zou, C., Qiao, Y., Peng, Q.: Action recognition with stacked fisher vectors. In: European Conference on Computer Vision. pp. 581–595. Springer (2014)
16. Piergiovanni, A., Ryoo, M.S.: Representation flow for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9945–9953 (2019)
17. Shahroudy, A., Liu, J., Ng, T.T., Wang, G.: Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1010–1019 (2016)
18. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Skeleton-based action recognition with directed graph neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7912–7921 (2019)
19. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12026–12035 (2019)
20. Si, C., Chen, W., Wang, W., Wang, L., Tan, T.: An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1227–1236 (2019)
21. Si, C., Jing, Y., Wang, W., Wang, L., Tan, T.: Skeleton-based action recognition with spatial reasoning and temporal stack learning. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 103–118 (2018)
22. Sun, S., Kuang, Z., Sheng, L., Ouyang, W., Zhang, W.: Optical flow guided feature: A fast and robust motion representation for video action recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1390–1399 (2018)

23. Tang, Y., Tian, Y., Lu, J., Li, P., Zhou, J.: Deep progressive reinforcement learning for skeleton-based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5323–5332 (2018)
24. Tran, D.V., Navarin, N., Sperduti, A.: On filter size in graph convolutional networks. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1534–1541. IEEE (2018)
25. Wang, D., Yuan, Y., Wang, Q.: Early action prediction with generative adversarial networks. IEEE Access **7**, 35795–35804 (2019)
26. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: Proceedings of the IEEE international conference on computer vision. pp. 3551–3558 (2013)
27. Wang, J., Jiao, J., Bao, L., He, S., Liu, Y., Liu, W.: Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4006–4015 (2019)
28. Wang, L., Gao, C., Yang, L., Zhao, Y., Zuo, W., Meng, D.: Pm-gans: Discriminative representation learning for action recognition using partial-modalities. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 384–401 (2018)
29. Wu, D., Chen, J., Sharma, N., Pan, S., Long, G., Blumenstein, M.: Adversarial action data augmentation for similar gesture action recognition. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2019)
30. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
31. Yao, B., Jiang, X., Khosla, A., Lin, A.L., Guibas, L., Fei-Fei, L.: Human action recognition by learning bases of action attributes and parts. In: 2011 International conference on computer vision. pp. 1331–1338. IEEE (2011)
32. Zadghorban, M., Nahvi, M.: An algorithm on sign words extraction and recognition of continuous persian sign language based on motion and shape features of hands. Pattern Analysis and Applications **21**(2), 323–335 (2018)
33. Zhang, C., Tian, Y., Guo, X., Liu, J.: Daal: Deep activation-based attribute learning for action recognition in depth videos. Computer Vision and Image Understanding **167**, 37–49 (2018)
34. Zhang, H.B., Zhang, Y.X., Zhong, B., Lei, Q., Yang, L., Du, J.X., Chen, D.S.: A comprehensive survey of vision-based human action recognition methods. Sensors **19**(5), 1005 (2019)
35. Zhao, R., Wang, K., Su, H., Ji, Q.: Bayesian graph convolution lstm for skeleton based action recognition. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 6882–6892 (2019)