Part-Level Graph Convolutional Network for Skeleton-Based Action Recognition

Linjiang Huang,[1,3] **Yan Huang,**[1,3] **Wanli Ouyang,**[4] **Liang Wang**[1,2,3*]
[1]Center for Research on Intelligent Perception and Computing (CRIPAC)
National Laboratory of Pattern Recognition (NLPR)
[2]Center for Excellence in Brain Science and Intelligence Technology (CEBSIT)
Institute of Automation, Chinese Academy of Sciences (CASIA)
[3]University of Chinese Academy of Sciences (UCAS)
[4]University of Sydney
SenseTime Computer Vision Research Group, Australia
linjiang.huang@cripac.ia.ac.cn, {yhuang, wangliang}@nlpr.ia.ac.cn, wanli.ouyang@sydney.edu.au

Abstract

Recently, graph convolutional networks have achieved remarkable performance for skeleton-based action recognition. In this work, we identify a problem posed by the GCNs for skeleton-based action recognition, namely part-level action modeling. To address this problem, a novel Part-Level Graph Convolutional Network (PL-GCN) is proposed to capture part-level information of skeletons. Different from previous methods, the partition of body parts is learnable rather than manually defined. We propose two part-level blocks, namely Part Relation block (PR block) and Part Attention block (PA block), which are achieved by two differentiable operations, namely graph pooling operation and graph unpooling operation. The PR block aims at learning high-level relations between body parts while the PA block aims at highlighting the important body parts in the action. Integrating the original GCN with the two blocks, the PL-GCN can learn both part-level and joint-level information of the action. Extensive experiments on two benchmark datasets show the state-of-the-art performance on skeleton-based action recognition and demonstrate the effectiveness of the proposed method.

Introduction

Action recognition is a fundamental problem in computer vision with many applications (Poppe 2010). Due to the development of depth sensors and pose estimation (Shotton et al. 2011), skeleton-based action recognition has drawn much attention recently. Considering graph convolutional networks have achieved state-of-the-art performance in many tasks (Kipf and Welling 2016b; Schlichtkrull et al. 2018), several recent works (Yan, Xiong, and Lin 2018; Li et al. 2018b) propose to employ the GCNs to automatically learn the spatial-temporal patterns of skeletons.

Nevertheless, there are some challenges existing in these GCN-based methods. In this work, we mainly focus on one of these challenges, dubbed as *part-level action modeling*. This issue has not been well considered before and has notably limited the performance. Let's imagine some actions, such as *running* and *waving*. Most of these actions are performed by the co-movement of body parts, *e.g.*, arms and legs. Actually, in many cases, body parts can be considered as the smallest unit of action execution, which means these actions can be identified only by the movement of body parts. However, GCN only propagates information across the edges, even we can stack many layers to make the joints of different body parts inter-aware, it's hard to automatically learn such a high-level structure from data by GCN itself.

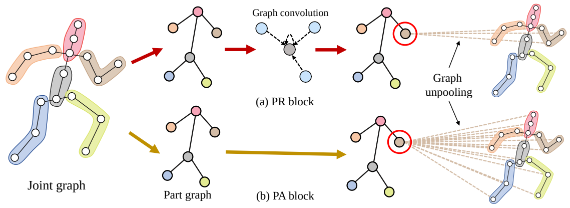

Figure 1: Illustration of our main idea. The PR block aims at capturing part-level relations. Besides, the PA block aims at highlighting important body parts. The partition of body parts is learned in a data-driven way.

Recently, Thakkar et al. (Thakkar and Narayanan 2018) propose a part-based graph convolutional network that manually divides the graph into several subgraphs with shared nodes. They first employ graph convolution operation within the subgraphs of body parts, and then propagate information between subgraphs via the shared nodes. Although they introduce the concept of body parts into graph convolutional networks, they still face the above problem because they lack the process of aggregating information from body joints to body parts. Moreover, different actions may have different definition of body parts (*e.g.,* hands for *clapping* and arms for *waving*), it may be sub-optimal for action recognition to manually partition body parts.

In light of these observations, we propose a novel Part-Level Graph Convolutional Network (PL-GCN). Our main target is to automatically aggregate body joints into body parts for capturing part-level information of actions. Thus, a differentiable graph pooling operation is devised to ensure the flexibility of this process, such that the body parts can be learned in a data-driven way. Besides, the connectivity of body parts can also be learned in this process, making it possible to capture relations between body parts. However,

[*]Corresponding author.

Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

since the graph of body joints is pooled into the one of body parts, the joint-level information may be missing. For some fine-grained actions, *e.g.*, *reading* and *writing*, it's necessary to additionally capture relations between body joints. To preserve the joint-level information of actions, we propose to keep the joint-level graph and regard the part-level graph as a parallel structure. We thus develop the graph unpooling operation, which can bridge the two level graphs, making our model be able to capture rich information of actions.

Based on the graph pooling and unpooling operation, we propose two blocks, namely Part Relation block (PR block) and Part Attention block (PA block) for modeling two properties of body parts. Specifically, the PR block aims at capturing relations between body parts. Therefore, a graph convolutional layer is employed after the graph pooling operation. Then the information of part-level relations is distributed back by the graph unpooling operation in an intra-part way, which is shown in Fig.1. As for PA part, it focuses on the importance of each body part. For fine-grained actions, such as *reading* and *writing*, which are performed by specific body parts, it is important to focus on the salient body parts for better identifying these actions. Thereupon, the graph unpooling operation of PA block is performed in a global way that allows the vertices of the joint-level graph to select important body parts as shown in Fig.1.

Integrating the original GCN with the two blocks, the PL-GCN can capture both joint-level and part-level information of actions. Our method outperforms the state-of-the-art methods on NTU RGB+D (Shahroudy et al. 2016) and SYSU (Hu et al. 2015). 1) We propose an effective graph pooling operation that can aggregate body joints into body parts and simultaneously learn the connectivity of body parts. 2) We propose two blocks, namely Part Relation block (PR block) and Part Attention block (PA block) for capturing part-level information of actions. Based on the two blocks, we present a Part-Level Graph Convolutional Network (PL-GCN) for skeleton-based action recognition. 3) The experimental results show that inserting the two blocks can consistently boost the performance of action recognition, and our PL-GCN achieves state-of-the-art performance on two benchmark datasets.

Related Work

Graph Convolutional Networks. GCNs can be classified into spatial domain methods and spectral domain methods (Bronstein et al. 2017). The **spatial domain methods** perform the convolution operation on graph vertices and their neighbors. Niepert et al. (Niepert, Ahmed, and Kutzkov 2016) propose a framework for performing convolution on arbitrary graphs by introducing graph labeling and nodes partition. Different from the spatial domain methods, the **spectral domain methods** utilize eigenvalues and eigenvectors of Laplacian matrix. The first formulation which extends CNN to graph is proposed by Bruna et al. (Bruna et al. 2014). This work is extended by Henaff et al. (Henaff, Bruna, and LeCun 2015) using frequency smoothing for spatial localization. Defferrard et al. (Defferrard, Bresson, and Vandergheynst 2016) use Chebyshev polynomials to reduce the computational cost of eigen decomposition. Kipf et al. (Kipf and Welling 2016a) propose to simplify the ChebNets (Defferrard, Bresson, and Vandergheynst 2016) by repeatedly performing one-hop graph convolution. Despite the impressive performance, these methods may be insufficient to capture high-level information of the graph due to the aforementioned problem.

Hierarchical Graph Representation. There are several recent works focusing on constructing hierarchical structures for the graph. Bruna et al. (Bruna et al. 2014) propose to reduce the sizes of nodes via multi-resolution clustering on graphs. Defferrard et al. (Defferrard, Bresson, and Vandergheynst 2016) use the binary tree for indexing nodes to simulate the pooling operation of 1D signals. However, these methods are performed in a two-stage fashion, which may be sub-optimal for learning on graphs. To meet the requirement of end-to-end learning, Ying et al. (Ying et al. 2018) use an assignment matrix to achieve pooling operation on the graph by assigning nodes to clusters. In addition, they learn a new coarsened adjacency matrix for the next layer.

Our proposed blocks are also hierarchical structures. Specifically, we propose a novel graph pooling operation to aggregate body joints into meaningful body parts. Moreover, we further propose the graph unpooling operations for distributing the part-level information to the original graph and preserving the low-level relations between body joints.

Skeleton-based Action Recognition. Recent advances in deep learning have made significant progress in this field. Various works use Recurrent Neural Networks (RNNs) (Du, Fu, and Wang 2016; Wang and Wang 2017; Zhang et al. 2017) and Convolutional Neural Networks (CNNs) (Du, Fu, and Wang 2015; Ke et al. 2017) to model spatial-temporal patterns of the skeleton sequence.

Inspired by the physical structure of human body, *e.g.*, body joints and body parts, some **hierarchical methods** (Du, Wang, and Wang 2015; Shahroudy et al. 2016) introduce the hierarchical structures into RNNs or CNNs for action recognition. However, the RNNs and CNNs may be not enough to handle the graph-shaped topology of skeleton. Naturally, the **graph-based methods** are introduced into skeleton-based action recognition. Yan et al. (Yan, Xiong, and Lin 2018) propose to treat the skeleton sequence as a spatial-temporal graph. Li et al. (Li et al. 2018b) design the graph convolutional filters which are simultaneously performed on spatial and temporal domains. Thakkar et al. (Thakkar and Narayanan 2018) propose a part-based graph convolutional network, which divides the graph into several subgraphs with shared nodes. However, this method needs to manually define body parts and still lacks the ability of modeling part-level information of actions.

Our method is based on GCNs. To the best of our knowledge, we are probably the first to explicitly model part-level information of actions based on GCNs for skeleton-based action recognition. Besides, unlike the above methods, it is unnecessary for our method to manually define body parts.

Proposed Method

In this section, we first briefly introduce some preliminaries. Then we outline two important operations on graph, *i.e.*, graph pooling operation and graph unpooling operation.

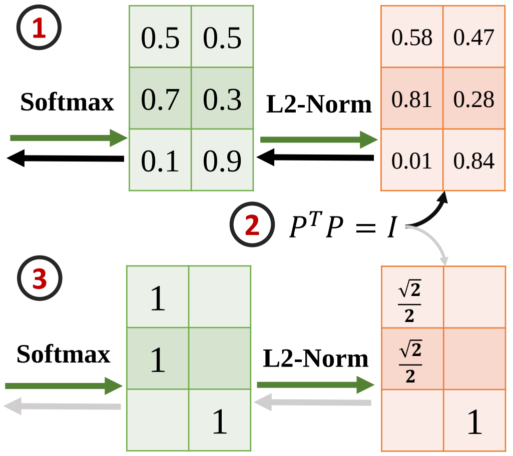

Figure 2: A simplified example for representing the process of learning $\boldsymbol{P}^T$. The green arrow and black arrow denote the forward process and the backward process respectively. The color depth of backward arrows represents the magnitude of gradient. The softmax operation enforces the elements of rows to be non-negative. Meanwhile, the orthogonal constraint is applied to the matrix after L2-normalization. Under the joint effect of softmax, L2-normalization and orthogonal constraint, the matrices in sign 1 would approximate to those in sign 3. (Best viewed in color)

The graph pooling operation is for automatically aggregating body joints into body parts and the graph unpooling operation is exactly the opposite. Based on the two operations, we describe the proposed two blocks, i.e., Part Relation block and Part Attention block. Finally, we introduce the Part-Level Graph Convolutional Network (PL-GCN).

Preliminaries

Notations. Let $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$ denote an undirected graph of skeleton, $\mathcal{V}$ is the set of body joints with $|\mathcal{V}|=N$, and $\mathcal{E}$ is the set of edges. Let $\boldsymbol{H}\in\mathbb{R}^{N\times F}$ denote the features of joints with F referring to the number of channels. Let $\boldsymbol{A}\in\mathbb{R}^{N\times N}$ denote the adjacency matrix, whose element a_{ij} is the weight assigned to the edge (i,j). We set $a_{ij}=1$ if joint i and j are connected and $a_{ij}=0$ otherwise. Let $\boldsymbol{D}\in\mathbb{R}^{N\times N}$ denote the degree matrix defined as $d_{ii}=\sum_j a_{ij}$.

Graph convolution. In this paper, we adopt the similar implementation of graph convolution as in (Kipf and Welling 2016a), which is formulated as:

$$\boldsymbol{H}'=\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}\boldsymbol{W} \tag{1}$$

where $\boldsymbol{W}\in\mathbb{R}^{F\times F'}$ is the weight matrix, $\boldsymbol{H}'\in\mathbb{R}^{N\times F'}$ is the feature after convolution. $\widetilde{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}_N$ and $\widetilde{\boldsymbol{D}}\in\mathbb{R}^{N\times N}$ is the degree matrix. The $\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$ can be viewed as a symmetric normalization. Different from (Yan, Xiong, and Lin 2018), we employ the distance partitioning, which is a more general way to achieve graph convolution operation.

Graph Pooling Operation

The graph pooling operation aims to aggregate the features of joints $\boldsymbol{H}$ into the features of body parts $\boldsymbol{H}_p\in\mathbb{R}^{N_p\times F}$, where N_p denotes the number of body parts. However, different from the pooling operation in CNN, the graph pooling operation has to consider the geometrical structure of graph.

Here, we regard the graph pooling operation as a process of matrix multiplication. But **what matrix is suitable for graph pooling?** We argue that it should be able to preserve joint-level information of the skeleton, such that we can capture information of actions as much as possible. Moreover, because of the requirement of capturing part-level relations, it ought to automatically learn the connectivity of body parts for down-stream GCN layers. We propose the following formulation to represent a graph:

$$T=\sum_{i,j=1}^{n}\|\boldsymbol{h}_i-\boldsymbol{h}_j\|^2a_{ij}=tr(\boldsymbol{H}^T(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{H}) \tag{2}$$

where T is a descriptor of the graph, $\boldsymbol{h}_i\in\mathbb{R}^F$ is the i-th row of $\boldsymbol{H}$. This formulation is widely used in manifold embedding (Belkin and Niyogi 2002) which can represent the local geometrical relations of the graph. Since the features are fed to the Batch Normalization (BN) layer before graph pooling, we remove the constraint for normalizing scale.

As stated in (Du et al. 2018), the pooling operation is similar to dimensionality reduction such as Principal Component Analysis (PCA). Suppose $\boldsymbol{S}^T\in\mathbb{R}^{N\times N_p}$ is a matrix whose columns are the principal components of the feature $\boldsymbol{H}$. It is expected that the reconstruction feature $\hat{\boldsymbol{H}}=\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{H}$ can keep the local geometrical relationships, *i.e.*, keep the descriptor T unchanged, as shown in Eq.(3):

$$T=tr(\boldsymbol{H}^T(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{H})\approx tr(\hat{\boldsymbol{H}}^T(\boldsymbol{D}-\boldsymbol{A})\hat{\boldsymbol{H}}) \tag{3}$$

If we substitute $\hat{\boldsymbol{H}}$ with $\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{H}$, we have: (some steps are omitted, refer to supplemental material for more details)

$$\begin{aligned} T&=tr(\boldsymbol{H}^T\boldsymbol{S}^T\boldsymbol{S}(\boldsymbol{D}-\boldsymbol{A})\boldsymbol{S}^T\boldsymbol{S}\boldsymbol{H})\\ &=tr((\boldsymbol{P}\boldsymbol{H})^T(\bar{\boldsymbol{D}}-\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T)(\boldsymbol{P}\boldsymbol{H}))\\ &=tr(\boldsymbol{H}_p^T(\bar{\boldsymbol{D}}-\hat{\boldsymbol{A}})\boldsymbol{H}_p)\end{aligned} \tag{4}$$

where $\boldsymbol{P}\in\mathbb{R}^{N_p\times N}$ is a rectangular matrix whose rows are mutually orthogonal, $\bar{\boldsymbol{D}}\in\mathbb{R}^{N_p\times N_p}$ is a diagonal matrix and $\hat{\boldsymbol{A}}\in\mathbb{R}^{N_p\times N_p}$ is a symmetric matrix. It is easy to see that the $\bar{\boldsymbol{D}}$ and $\hat{\boldsymbol{A}}$ have the similar characteristics with the degree matrix $\boldsymbol{D}$ and the adjacency matrix $\boldsymbol{A}$. Moreover, the descriptor T is just slightly changed, indicating this operation can preserve information of the graph. Consequently, it is reasonable that the graph pooling can be viewed as the multiplication of $\boldsymbol{P}$ and $\boldsymbol{H}$, and the matrix $\hat{\boldsymbol{A}}=\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T$ could demonstrate the connectivity of body parts.

Learning matrix for graph pooling. For the matrix $\boldsymbol{P}$, we can view it as a weighted assignment matrix, which aggregates body joints into body parts of size N_p. The element p_{ij} represents the proportion on how much of the j-th body joint belongs to the i-th body part. Base on this observation, we propose to generate it as:

$$\boldsymbol{P}^T=norm(softmax(\boldsymbol{H}\boldsymbol{W}_f),2) \tag{5}$$

where $\boldsymbol{W}_f \in \mathbb{R}^{F\times N_p}$, the softmax operation is performed along rows, and the L2-normalization is performed along columns. Besides, we impose an orthogonal constraint on $\boldsymbol{P}$, i.e., $\boldsymbol{P}\boldsymbol{P}^T = \boldsymbol{I}$. Under the joint effect of the function $f(\cdot,\cdot)$ and the orthogonal constraint, the matrix $\boldsymbol{P}$ would approximate to the one whose rows are weighted one-hot vectors, as shown in Fig.2. In this way, the graph pooling operation automatically divides the original graph into non-overlapped subgraphs, which meet the definition of body parts. For the sequence of skeletons, we learn different matrix $\boldsymbol{P}$ for different frames to represent the uniqueness of each frame.

Graph Unpooling Operation

After graph pooling, the graph of body joints is coarsened to a graph of body parts. To preserve joint-level information of actions, we argue that it is necessary to adopt the graph unpooling operation. The graph unpooling operation is for distributing the high-level features of body parts back to the original graph. In other words, it yields aligned features $\widetilde{\boldsymbol{H}} \in \mathbb{R}^{N\times F'}$ with regard to the features of body joints.

There are mainly two benefits of using the graph unpooling operation. (i) **Preserving joint-level information.** The graph unpooling operation can reconstruct the joint-level graph by the part-level information, which enables our model to be compatible with the original joint-level graph and thus preserves the joint-level information. (ii) **Temporal alignment.** For the sequence of skeletons, different frames have different matrices for graph pooling, which means that the pooled graph is misaligned along the temporal dimension. Thus, we need the graph unpooling operation for feature alignment. Here, two different graph unpooling operations are devised for the proposed two blocks, which are introduced in the following sections.

Part Relation Block

Our proposed Part Relation block (PR block) is inspired by the hierarchical structures in CNNs which can model high-level information of grid-shaped data. To obtain the part-level representations $\boldsymbol{H}_{rp}$, the graph is first forwarded to the proposed graph pooling layer. We denote the matrix for graph pooling of PR block as $\boldsymbol{P}_r$. After graph pooling, we employ a graph convolution operation with the learned adjacency matrix $\hat{\boldsymbol{A}}$ for reasoning high-level relations between body parts, which can be formulated as:

$$\boldsymbol{H}_r = \hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}_{rp}\boldsymbol{W}_c^r \tag{6}$$

where $\boldsymbol{H}_r$ is the updated feature, and $\boldsymbol{W}_c^r$ is the weight matrix. Given the updated feature $\boldsymbol{H}_r$, a corresponding graph unpooling operation is utilized to project the features of part-level relations back to the joint-level graph:

$$\widetilde{\boldsymbol{H}}_r = \boldsymbol{Q}\boldsymbol{H}_r, \quad \boldsymbol{Q} = g(\boldsymbol{H}, \boldsymbol{W}_g) \tag{7}$$

where $g(\cdot,\cdot)$ is the function for generating $\boldsymbol{Q} \in \mathbb{R}^{N\times N_p}$, and $\boldsymbol{W}_g$ is the corresponding weight matrix. For PR block, we find that the matrix $\boldsymbol{P}_r^T$ is enough to handle this task.

Discussion: The overall pipeline can be formulated as:

$$\widetilde{\boldsymbol{H}}_r = \boldsymbol{P}_r^T(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{P}_r\boldsymbol{A}\boldsymbol{P}_r^T\hat{\boldsymbol{D}}^{-\frac{1}{2}})(\boldsymbol{P}_r\boldsymbol{H})\boldsymbol{W}_c^r \tag{8}$$

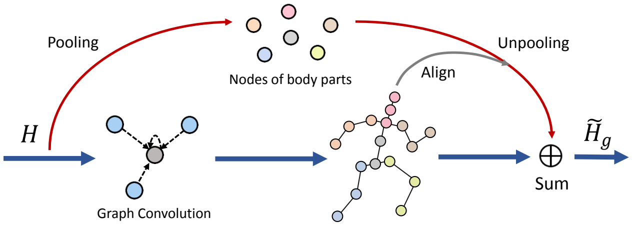

Figure 3: The framework of PA block. The graph pooling operation aggregates body joints into body parts, and the feature after graph convolution is utilized as a guidance for aligning the nodes of body parts to the joint-level graph.

if we regard the matrices before the matrix $\boldsymbol{H}$ as a matrix $\dot{\boldsymbol{A}}$, it is interesting to find that this matrix is also a symmetric matrix, and thus the above formulation can also be viewed as a graph convolution operation for the undirected graph.

However, the matrix $\dot{\boldsymbol{A}}$ is un-normalized, which means the scale of $\boldsymbol{H}$ will be changed. In order to avoid numerical instability, we reform Eq.(8) as:

$$\widetilde{\boldsymbol{H}}_r = \ddot{\boldsymbol{D}}^{-\frac{1}{2}}(\boldsymbol{P}_r^T\boldsymbol{P}_r\boldsymbol{A}\boldsymbol{P}_r^T\boldsymbol{P}_r)\ddot{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}\boldsymbol{W}_c^r \tag{9}$$

where $\ddot{\boldsymbol{D}}$ is the corresponding degree matrix. Since the major role of graph convolution is relation reasoning rather than normalization, it is reasonable to remove the matrix $\hat{\boldsymbol{D}}$.

Part Attention Block

The framework of Part Attention block (PA block) is shown in Fig.3. Here, we keep using the proposed graph pooling operation, and we denote the matrix of graph pooling as $\boldsymbol{P}_a \in \mathbb{R}^{N_a\times N}$. As stated above, the PA block aims to focus on the important body parts for better identifying actions, especially the fine-grained ones. To attend to salient body parts, an attention-based graph unpooling operation is developed. Formally, the graph unpooling operation can be viewed as an alignment operation. A good way for feature alignment would be to choose features that need to be aligned. Here, we utilize the softmax operation for graph unpooling operation because of its good capability of feature selection. The corresponding matrix $\boldsymbol{M} \in \mathbb{R}^{N\times N_a}$ can be generated as:

$$\boldsymbol{M} = softmax(\boldsymbol{H}_m\boldsymbol{W}_m) \tag{10}$$

where $\boldsymbol{H}_m \in \mathbb{R}^{N\times F'}$ is the feature in the original space, $\boldsymbol{W}_m \in \mathbb{R}^{F'\times N_a}$ is the parameters. The softmax operation is implemented in a row-wise fashion. Taking account of the topological structure of the graph, we set the matrix $\boldsymbol{H}_m$ as:

$$\boldsymbol{H}_m = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}\boldsymbol{W}_c^a \tag{11}$$

where $\boldsymbol{W}_c^g \in \mathbb{R}^{F\times F'}$ is the trainable parameters.

Discussion: Since the graph unpooling operation can be viewed as a process of feature selection guided by the feature $\boldsymbol{H}_m$, it is reasonable that the unpooled features are complementary with the feature $\boldsymbol{H}_m$. So we add the feature $\boldsymbol{H}_m$ to the unpooled features. The pipeline is as follows:

$$\widetilde{\boldsymbol{H}}_a = \boldsymbol{H}_m + \boldsymbol{M}\boldsymbol{P}_a\boldsymbol{H}\boldsymbol{W}_c^a \tag{12}$$

However, this formulation is numerical instable. We introduce an additional matrix to address this problem:

$$\widetilde{\boldsymbol{H}}_a = \boldsymbol{H}_m + \boldsymbol{M}\boldsymbol{D}_a^{-1}\boldsymbol{P}_a\boldsymbol{H}\boldsymbol{W}_c^a \tag{13}$$

where the matrix $\boldsymbol{D}_a \in \mathbb{R}^{N_a \times N_a}$ is similar to the degree matrix, whose element $d_{a(ii)} = \sum_j p_{a(ij)}$.

We can find that both of the matrices $\boldsymbol{M}$ and $\boldsymbol{D}_a^{-1}\boldsymbol{P}_a$ are generated from the input feature of body joints $\boldsymbol{H}$, and the multiplication of $\boldsymbol{M}$ and $\boldsymbol{D}_a^{-1}\boldsymbol{P}_a$ is the interaction between body joints. The matrix $\boldsymbol{M}\boldsymbol{D}_a^{-1}\boldsymbol{P}_a$ can be also viewed as an asymmetrical adjacency matrix, which corresponds to a densely connected graph. Consequently, the PA block captures joint-level relations and part-level information, *i.e.*, the importance of body parts, which makes it effective in identifying actions, especially the fine-grained ones.

Part-Level GCN

The proposed PR block and PA block can be easily plugged into the existing GCN architectures due to their residual nature. Here, we present a novel network, namely Part-Level Graph Convolutional Network (PL-GCN), which integrates the two blocks. The main block of PL-GCN is shown in Fig.4(a). After several blocks, a global average pooling layer is employed and the final output is fed to a softmax layer to get the prediction $\hat{\boldsymbol{y}}$. More details about the network are presented in the supplementary material.

The total loss is formulated as follows:

$$\begin{aligned}\mathcal{L} = &-\sum_{c=1}^{C} y_c log\hat{y}_c + \lambda_r \sum_{s=1}^{S} \mathcal{F}((\boldsymbol{P}_{rs}^T\boldsymbol{P}_{rs} \circ (\boldsymbol{1} - \boldsymbol{I})) \\ &+ \lambda_a \sum_{s=1}^{S} \mathcal{F}((\boldsymbol{P}_{as}^T\boldsymbol{P}_{as}) \circ (\boldsymbol{1} - \boldsymbol{I})) + \lambda_\omega \|\boldsymbol{W}\|\end{aligned} \tag{14}$$

where C denotes the number of classes, and S denotes the number of layers. $\boldsymbol{1} \in \mathbb{R}^{N_b \times N_b}$ and $\boldsymbol{I} \in \mathbb{R}^{N_b \times N_b}$ are a matrix of all ones and an identity matrix respectively, $\circ$ is element-wise multiplication. $\mathcal{F}(\cdot)$ is the operation of quadratic sum of elements. λ_r, λ_a and λ_ω are the coefficients for orthogonal loss of the PR block, orthogonal loss of the PA block and weight decay, respectively.

Experiments

Datasets

We evaluate our method on two challenging datasets, namely NTU RGB+D (Shahroudy et al. 2016) and SYSU 3D HOI (SYSU) (Hu et al. 2015).

NTU RGB+D. This dataset consists of 56880 actions with 60 classes. The benchmark evaluations include Cross-Subject (CS) and Cross-View (CV). In the CS evaluation, training samples come from one subset of actors and networks are evaluated on samples from remaining actors. In the CV evaluation, samples captured from cameras 2 and 3 are utilized for training, while samples from camera 1 are employed for testing.

SYSU. This dataset contains 12 actions performed by 40 subjects. We follow the protocols proposed by (Hu et al. 2015) to evaluate the performance. For setting 1, for each activity class, half of the samples are used for training and the rest for testing. For setting 2, half of the subjects are used for training and the rest for testing. For each setting, we employ the 30-fold cross-subject validation.

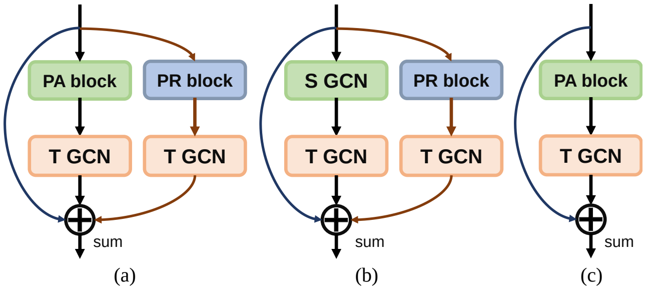

Figure 4: The overview of different variants of the block. *S-GCN* means spatial GCN and *T-GCN* means temporal GCN. (a) The used block in PL-GCN, which contains a PR block and a PA block. (b) A block contains a PR block and a spatial-temporal GCN. (c) A block only contains a PA block.

Implementation Details

The proposed PL-GCN is the stack of nine blocks. Following (Thakkar and Narayanan 2018), we concatenate joint locations J_{loc}, relative coordinates D_R and temporal displacement D_T as the input to improve the performance. We set the number of body parts as 6 for the two blocks. The proposed model is implemented by Pytorch (Paszke et al. 2017). We use SGD to optimize the model with a mini-batch size of 64. The learning rates for both datasets are 0.1 initially, multiplied by 0.1 after 20 epochs and 50 epochs. The training procedure stops at 80 epochs. Other details are provided in the supplemental material.

Comparison with The State-of-the-art

We evaluate our method by comparing it with the state-of-the-art approaches. Tab.1 and Tab.2 demonstrate the results on two datasets. For fair comparisons, we label the two-stream methods which take the joint locations J_{loc} and temporal displacement D_T as input. We first compare our method with a traditional method, *i.e.*, Lie Group (Vemulapalli, Arrate, and Chellappa 2014) that is based on hand-crafted features. As we can see, our method significantly outperforms this approach, which shows the superiority of deep learning methods over hand-crafted approaches. Then our method is compared with recent deep learning methods. We can see that our method outperforms all the state-of-the-art methods on CS setting. Even our method utilizes the same features as those two-stream methods (Si et al. 2018; Shi et al. 2019), we also achieve comparable performance with the newest method. Moreover, our method significantly outperforms the baseline model ST-GCN (Yan, Xiong, and Lin 2018) by 7.7% and 6.7% using CS and CV protocols on the NTU RGB+D, and improves the accuracy by 3.0% and 2.3% using setting-1 and setting-2 protocols on the SYSU. When only using the joint locations J_{loc}, the PL-GCN still outperforms the ST-GCN by 2.5% and 2.2% using CS and

Table 1: Comparison on NTU RGB+D dataset (%). T and C represent the two-stream methods and the methods which take the concatenation of signals as input.

Methods	CS	CV
Lie Group (Vemulapalli, Arrate, and Chellappa 2014)	50.1	82.8
HBRNN (Du, Wang, and Wang 2015)	59.1	64.0
Part-aware LSTM (Shahroudy et al. 2016)	62.9	70.3
Geo Features (Zhang, Liu, and Xiao 2017)	70.3	82.4
Two-Stream CNN (T) (Li et al. 2017)	83.2	89.3
Deep STGC$_K$ (Li et al. 2018b)	74.9	86.3
ST-GCN (Yan, Xiong, and Lin 2018)	81.5	88.3
SR-TSL (T) (Si et al. 2018)	84.8	92.4
HCN (T) (Li et al. 2018a)	86.5	91.1
PB-GCN (C) (Thakkar and Narayanan 2018)	87.5	93.2
2s-AGCN (T) (Shi et al. 2019)	88.5	**95.1**
PL-GCN	84.0	90.5
PL-GCN + $J_{loc}\|D_T$	88.6	94.2
PL-GCN (C)	**89.2**	95.0

Table 2: Comparison on SYSU dataset. (%)

Methods	Setting-1	Setting-2
Dynamic Skeleton (Hu et al. 2015)	75.5	76.9
LAFF (SKL) (Hu et al. 2016)	54.2	-
ST-LSTM (Liu et al. 2016)	76.5	-
VA-LSTM (Zhang et al. 2017)	76.9	77.5
DPRL (Tang et al. 2018)	76.9	-
SR-TSL (Si et al. 2018)	80.7	81.9
PL-GCN	**83.7**	**84.2**

CV protocols respectively, which means our method indeed can capture richer information of skeletons to improve the action recognition accuracy.

Discussion

To validate the effectiveness of the proposed blocks as well as the PL-GCN, extensive experiments are conducted on the NTU RGB+D dataset. More experimental results such as the number of parameters, FLOPs of the proposed blocks are reported in the supplemental material.

(a) Comparison with variants of PR block. The results of different variants of PR block are shown in Tab.3(a). For learning both part-level and joint-level relations, the PR block should be used with a spatial-temporal graph convolutional network, denoted as *ST-GCN + PR block*, which is shown in Fig.4(b). As we can see, the *ST-GCN + PR block* significantly outperforms the *ST-GCN* (distance partitioning) (Yan, Xiong, and Lin 2018) by 10.1% and 8.3%. Even the ST-GCN takes the same input as ours, the *ST-GCN + PR block* still surpasses it by 4.1% and 3.6%, demonstrating that the PR block can capture relations which cannot be easily captured by regular graph convolution layers. Then, we remove the ST-GCN and keep the PR block, denoted as *PR block*. After the removal of ST-GCN, the performance drops, indicating that joint-level information is also important for action recognition. Unless stated, we utilize the *ST-GCN + PR block* in the following experiments of PR block.

Table 3: Ablation study on NTU RGB+D dataset (%). (a) Evaluation of variants of PR block. (b) Evaluation of variants of PA block. (c) Evaluate the effectiveness of graph unpooling operation.

	Methods	CS	CV
	ST-GCN Distance Partition	78.0	85.1
	ST-GCN Distance + $J_{loc}\|D_R\|D_T$	84.0	89.8
(a)	**ST-GCN + PR block**	**88.1**	**93.4**
	PR block	84.2	90.2
	Double ST-GCN	86.5	91.8
	Learnable Matrix	84.9	91.0
	Pooling Matrix w/o Constraint	85.0	91.2
	Fixed Pooling Matrix	87.1	92.2
(b)	**PA block**	**88.3**	**93.5**
	PA block w/o Normalization	87.2	92.0
	Orthogonal Unpooling	87.0	91.0
(c)	Hierarchical Pooling	86.7	92.3
	Hierarchical Pooling (Misalignment)	84.7	90.3

To evaluate the effectiveness of PR block, two more questions must be considered. **The first question**: Dose the effectiveness is from the methodology or the increase of parameters? **The second question**: Does the improvement comes from the learnable property or the pooling property?

To answer the first question, we replace the PR block in *ST-GCN + PR block* with a ST-GCN, denoted as *Double ST-GCN*. The number of parameters of *Double ST-GCN* is almost the same as the one of *ST-GCN + PR block*. We can see that the accuracy of *Double ST-GCN* drops by 1.6% in CS and 1.6% in CV, indicating that the effectiveness of PR block indeed comes from part-level information learning.

As for the second question, three experiments are conducted as shown in Tab.3(a). In the first experiment, we replace the matrix $\boldsymbol{P}^T\boldsymbol{P}$ with a learnable matrix $\boldsymbol{M}$, indicating the PR block is learnable but not in a pooling way, denoted as *Learnable Matrix*. In the second experiment, we learn the pooling matrix without orthogonal constraint, which means the learned body parts may be overlapped, denoted as *Pooling Matrix w/o Constraint*. In the third experiment, we substitute the pooling matrix $\boldsymbol{P}$ for a fixed matrix, which aggregates body joints into two arms, upper torso, lower torso and two legs, denoted as *Fixed Pooling Matrix*. We find that the performance of *Learnable Matrix* drops by 3.2% and 2.4%, indicating the importance of pooling property. However, the performance of *Pooling Matrix w/o Constraint* presents a fact that an effective graph pooling operation should be based on an orthogonal pooling matrix. Besides, even the fixed matrix is consistent with common sense, the performance of *Fixed Pooling Matrix* drops obviously, demonstrating the importance of learnable property.

(b) Comparison with variants of PA block. Similarly, we also evaluate different variants of PA block, and the results are shown in Tab.3(b). The basic model is shown in Fig.4(c), denoted as *PA block*.

In the first experiment, we remove the normalization matrix $\boldsymbol{D}_b$, denoted as *PA block w/o Normalization*. We observe that the performance of *PA block w/o Normalization* drops by 0.9% in CS and 1.5% in CV. Besides, in our experiment, we find that the model is prone to collapse after removing $\boldsymbol{D}_b$, indicating the importance of normalization. Besides, we also design a combination of graph pooling and unpooling operation for the PA block, namely *Orthogonal Unpooling*. The *Orthogonal Unpooling* is the same as what used in the PR block. Since the *Orthogonal Unpooling* neither captures part-level relationships nor focuses on important body parts, the accuracy drops evidently by 1.3% in CS and 2.5% in CV.

(c) Is the graph unpooling operation necessary? The role of graph unpooling operation in two blocks is well demonstrated in the above experiments. To verify the role of graph unpooling operation in preserving joint-level information and temporal alignment, two extensive experiments are conducted in Tab.3(c).

Both of the experiments are based on a network with successive graph pooling operation and without graph unpooling. In the first experiment, we employ the same pooling matrix for all frames to avoid temporal misalignment, denoted as *Hierarchical Pooling*. By contrast, we learn different matrices for different frames in the second experiment, denoted as *Hierarchical Pooling (Misalignment)*. Compared with the second experiment, the *Hierarchical Pooling* achieves 2.0% gain in CS and 2.0% gain in CV, indicating the importance of temporal alignment, which is achieved by the graph unpooling operation. Moreover, we find that the *Hierarchical Pooling* is superior to the *ST-GCN* but still inferior to the *ST-GCN + PR block*, indicating that the joint-level information is important for action recognition, and the graph unpooling operation makes it possible to capture both part-level and joint-level information of actions for better performance.

Going Deeper with The Blocks

Visualization. To attain further insights of the two blocks, we present some visualizations of the PR block and the PA block. As stated above, we view the pipeline of the PR block as a process of learning a new adjacency matrix $\boldsymbol{P}^T\boldsymbol{P}\boldsymbol{A}\boldsymbol{P}^T\boldsymbol{P}$. The learned adjacency matrix of the action of *take off a shoe* is shown in Fig.5(a). As we can see, some joints are connected with other joints breaking the constraint of human body structure. For example, the node of the left hand is connected with the node of the right foot, which is reasonable for the action of *take off a shoe*.

Then we visualize the learned matrices of the PA block. As we can see, the graph pooling operation divides the original graph into six meaningful body parts. For example, the fifth body part contains most of the joints of two legs. Moreover, as shown in Fig.5(c), the learned matrix for graph unpooling mainly focuses on the fifth body part, which is quite reasonable for the action of *one foot jumping*.

Class-specific improvement. To deeper investigate the effectiveness of the two blocks, class-specific accuracy gains are computed. The baseline method is ST-GCN with the concatenation of three signals. We list the top-5 actions with the gains. We can see that the actions such as *cross hands in front*, which need co-operation of body parts have evident gains. Besides, the fine-grained actions such as *reading* and *writing* also have gains, demonstrating the effectiveness in modeling both part related actions and fine-grained actions.

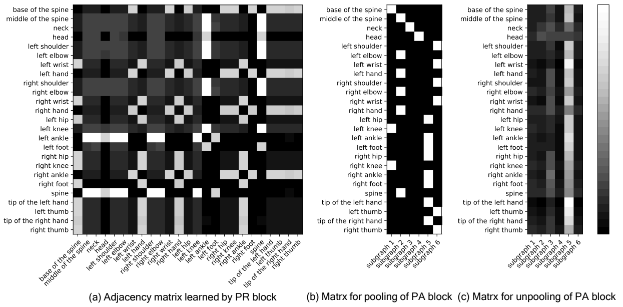

Figure 5: Visualize the matrices learned by two blocks. (a) Adjacency matrix learned by PR block. (b) Matrix $\boldsymbol{P}_a$ for graph pooling of PA block. (c) Matrix $\boldsymbol{M}$ for graph unpooling of PA block.

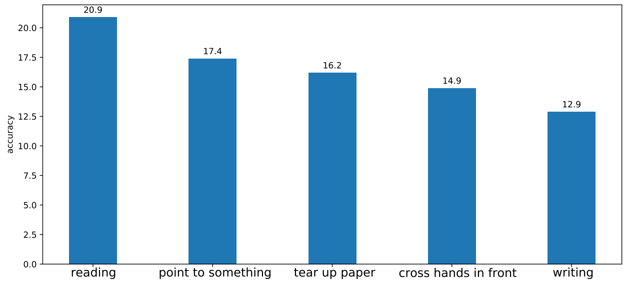

Figure 6: Class-specific gains on NTU RGB+D. The baseline model is ST-GCN with concatenation of three features.

Conclusion

In this paper, we have proposed a novel Part-Level Graph Convolutional Network (PL-GCN) for skeleton-based action recognition. The construction of PL-GCN is mainly based on two blocks, *i.e.*, Part Relation block (PR block) and Part Attention block (PA block). The PR block aims at learning part-level relations while the PA block aims at highlighting important body parts. We have evaluated our PL-GCN on two benchmark datasets and achieved state-of-the-art performance. In the future, we plan to further improve our method for automatically learning the number of body parts and focus more on modeling temporal dynamics of actions.

Acknowledgments

This work is jointly supported by National Key Research and Development Program of China (2016YFB1001000), National Natural Science Foundation of China (61525306, 61633021, 61721004, 61420106015, 61806194, U1803261), Capital Science and Technology Leading Talent Training Project (Z181100006318030), HW2019SOW01, and CASAIR. This work is also supported by grants from NVIDIA and the NVIDIA DGX-1 AI Supercomputer.

References

Belkin, M., and Niyogi, P. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 585–591.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34:18–42.

Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3844–3852.

Du, Y.; Yuan, C.; Li, B.; Zhao, L.; Li, Y.; and Hu, W. 2018. Interaction-aware spatio-temporal pyramid attention networks for action classification. In *ECCV*, 373–389. Springer.

Du, Y.; Fu, Y.; and Wang, L. 2015. Skeleton based action recognition with convolutional neural network. In *Asian Conference on Pattern Recognition*, 579–583. IEEE.

Du, Y.; Fu, Y.; and Wang, L. 2016. Representation learning of temporal dynamics for skeleton-based action recognition. *IEEE Transactions on Image Processing* 25:3010–3022.

Du, Y.; Wang, W.; and Wang, L. 2015. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 1110–1118. IEEE.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Hu, J.-F.; Zheng, W.-S.; Lai, J.; and Zhang, J. 2015. Jointly learning heterogeneous features for rgb-d activity recognition. In *CVPR*, 5344–5352. IEEE.

Hu, J.-F.; Zheng, W.-S.; Ma, L.; Wang, G.; and Lai, J. 2016. Real-time rgb-d activity prediction by soft regression. In *ECCV*, 280–296. Springer.

Ke, Q.; Bennamoun, M.; An, S.; Sohel, F.; and Boussaid, F. 2017. A new representation of skeleton sequences for 3d action recognition. In *CVPR*, 4570–4579. IEEE.

Kipf, T. N., and Welling, M. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kipf, T. N., and Welling, M. 2016b. Variational graph autoencoders. *arXiv preprint arXiv:1611.07308*.

Li, C.; Zhong, Q.; Xie, D.; and Pu, S. 2017. Skeleton-based action recognition with convolutional neural networks. In *IEEE International Conference on Multimedia & Expo Workshops*, 597–600. IEEE.

Li, C.; Zhong, Q.; Xie, D.; and Pu, S. 2018a. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. In *International Joint Conference on Artificial Intelligence*, 786–792.

Li, C.; Cui, Z.; Zheng, W.; Xu, C.; and Yang, J. 2018b. Spatio-temporal graph convolution for skeleton based action recognition. *arXiv preprint arXiv:1802.09834*.

Liu, J.; Shahroudy, A.; Xu, D.; and Wang, G. 2016. Spatio-temporal lstm with trust gates for 3d human action recognition. In *ECCV*, 816–833. Springer.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, 2014–2023.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Poppe, R. 2010. A survey on vision-based human action recognition. *Image and Vision Computing* 28:976–990.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; van den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Shahroudy, A.; Liu, J.; Ng, T.-T.; and Wang, G. 2016. Ntu rgb+ d: A large scale dataset for 3d human activity analysis. In *CVPR*, 1010–1019. IEEE.

Shi, L.; Zhang, Y.; Cheng, J.; and Lu, H. 2019. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *CVPR*, 12026–12035. IEEE.

Shotton, J.; Fitzgibbon, A.; Cook, M.; Sharp, T.; Finocchio, M.; Moore, R.; Kipman, A.; and Blake, A. 2011. Real-time human pose recognition in parts from single depth images. In *CVPR*, 1297–1304. IEEE.

Si, C.; Jing, Y.; Wang, W.; Wang, L.; and Tan, T. 2018. Skeleton-based action recognition with spatial reasoning and temporal stack learning. *arXiv preprint arXiv:1805.02335*.

Tang, Y.; Tian, Y.; Lu, J.; Li, P.; and Zhou, J. 2018. Deep progressive reinforcement learning for skeleton-based action recognition. In *CVPR*, 5323–5332.

Thakkar, K., and Narayanan, P. 2018. Part-based graph convolutional network for action recognition. *arXiv preprint arXiv:1809.04983*.

Vemulapalli, R.; Arrate, F.; and Chellappa, R. 2014. Human action recognition by representing 3d skeletons as points in a lie group. In *CVPR*, 588–595. IEEE.

Wang, H., and Wang, L. 2017. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In *CVPR*, 499–508. IEEE.

Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:1801.07455*.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. In *NIPS*, 4801–4811.

Zhang, P.; Lan, C.; Xing, J.; Zeng, W.; Xue, J.; and Zheng, N. 2017. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *ICCV*, 2117–2126. IEEE.

Zhang, S.; Liu, X.; and Xiao, J. 2017. On geometric features for skeleton-based action recognition using multilayer lstm networks. In *IEEE Winter Conference on Applications of Computer Vision*, 148–157. IEEE.