Spatio-Temporal Inception Graph Convolutional Networks for Skeleton-Based Action Recognition

Zhen Huang*
University of Science and Technology of China
hz13@mail.ustc.edu.cn

Xu Shen
Alibaba Group
shenxu.sx@alibaba-inc.com

Xinmei Tian†
University of Science and Technology of China
xinmei@ustc.edu.cn

Houqiang Li
University of Science and Technology of China
lihq@ustc.edu.cn

Jianqiang Huang
Alibaba Group
jianqiang.hjq@alibaba-inc.com

Xian-Sheng Hua†
Alibaba Group
xiansheng.hxs@alibaba-inc.com

ABSTRACT

Skeleton-based human action recognition has attracted much attention with the prevalence of accessible depth sensors. Recently, graph convolutional networks (GCNs) have been widely used for this task due to their powerful capability to model graph data. The topology of the adjacency graph is a key factor for modeling the correlations of the input skeletons. Thus, previous methods mainly focus on the design/learning of the graph topology. But once the topology is learned, only a single-scale feature and one transformation exist in each layer of the networks. Many insights, such as multi-scale information and multiple sets of transformations, that have been proven to be very effective in convolutional neural networks (CNNs), have not been investigated in GCNs. The reason is that, due to the gap between graph-structured skeleton data and conventional image/video data, it is very challenging to embed these insights into GCNs. To overcome this gap, we reinvent the split-transform-merge strategy in GCNs for skeleton sequence processing. Specifically, we design a simple and highly modularized graph convolutional network architecture for skeleton-based action recognition. Our network is constructed by repeating a building block that aggregates multi-granularity information from both the spatial and temporal paths. Extensive experiments demonstrate that our network outperforms state-of-the-art methods by a significant margin with only 1/5 of the parameters and 1/10 of the FLOPs.

CCS CONCEPTS

• **Information systems** → Information systems applications.

*This work was done when the author was visiting Alibaba as a research intern.
†Corresponding authors.

MM '20, October 12–16, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413666

KEYWORDS

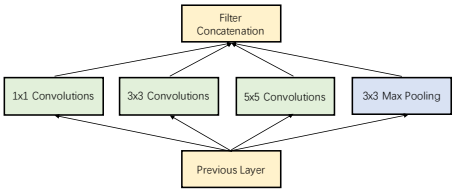
graph convolutional networks, skeleton-based classification

ACM Reference Format:
Zhen Huang, Xu Shen, Xinmei Tian, Houqiang Li, Jianqiang Huang, and Xian-Sheng Hua. 2020. Spatio-Temporal Inception Graph Convolutional Networks for Skeleton-Based Action Recognition. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413666

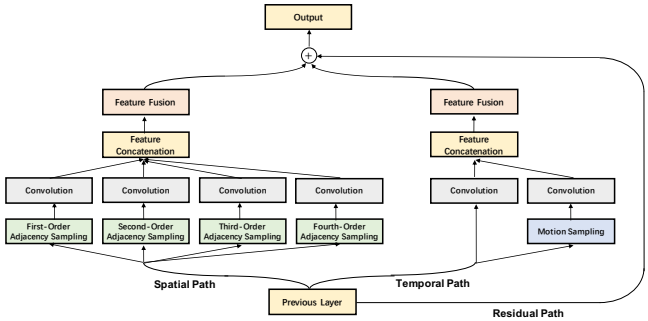
1 INTRODUCTION

Human action recognition attracts considerable attention due to their potential for many applications. Recently, skeleton-based action recognition has been widely studied because skeleton data convey compact information of body movement and have strong adaptability to dynamic circumstances, e.g., variations in viewpoints, occlusions and complicated background [13]. Previous works formulate skeleton data as a sequence of grid-shaped joint-coordinate vectors and use CNNs [4][41][33][44][54][30][1] or recurrent neural networks (RNNs) [31][21][20][34][24][28][27] to learn the actions. As skeleton data naturally lies in a non-Euclidean space with joints as vertexes and their connections in the human body as edges, CNN- and RNN-based methods cannot fully utilize the rich information conveyed in the graph structure of skeleton data.

Recently, graph convolutional networks, with their superior capability in dealing with graph data, have been introduced to skeleton-based action recognition and have achieved state-of-the-art performance [26][52][46][43][25][29][9][42][39]. Most of these methods focus on the design/learning of the graph topology. Yan et al. [52] introduced GCNs to model skeleton data and constructed a predefined graph with a fixed topology constraint. Shi et al. [42] proposed to learn an adaptive graph by parameterizing the graphs, and then updated the graph jointly with convolutional parameters. Gao et al. [9] introduced a high-order approximation for a larger receptive field of the graph. Peng et al. [39] tried to search for different graphs at different layers via automatic neural architecture searching (NAS) [55]. However, once the graph is generated, only a single scale and one transformation exist in each layer of the networks. As a consequence, the backbone of these GCN-based



(a) GoogLeNet Inception for CNNs



(b) Our Spatio-Temporal Inception for GCNs

Figure 1: GoogLeNet inception block for CNNs and our spatio-temporal inception block for GCNs. Both blocks follow the split-transform-merge strategy. In our spatio-temporal (ST) inception block, the inputs are split into three paths: a spatial path for spatial features, a temporal path for sequential features, and a residual path for the reuse of the input features. In the spatial path, graph convolutions with $1\times$ to $4\times$ hop connections are applied. In the temporal path, graph convolutions are applied to position features and motion features.

methods has intrinsic limitations on extracting and synthesizing information from different scales and transformations from different paths at different levels.

Intuitively, many insights from the design philosophy of CNN can be integrated into GCN-based backbone networks. Specifically, a) inputs can be split into different paths with a few lower-dimensional embeddings or identity mappings (GoogLeNet [45], ResNeXt [51], ResNet [14]); b) different sets of transformations can be applied to each path (GoogLeNet [45], ResNeXt [51]); c) the outputs of all the paths can be aggregated with concatenation or summation (ResNet [14], ResNeXt [51], GoogLeNet [45], DenseNet [16]). However, due to the graph structure of skeleton data, it is very challenging to embed these insights in GCNs. For example, for multi-scale spatial processing, instead of using multiple kernel sizes in CNNs, a customized operation for different orders of hop connections is needed for GCNs. For multi-view temporal processing (position and motion), motion features specified for graph convolution of skeleton sequences are still not touched. In fact, many biological studies [17][35][3][7][48] have shown that, 20% of cells in primate visual systems are responsive to dynamic motion changes, but are not sensitive to spatial details [6].

In this paper, we adopt the strategy of repeating layers in CNNs and reinvent the split-transform-merge strategy in GCNs for spatial and temporal skeleton sequence processing in each layer. For each layer, the inputs are split into three paths: a spatial path for spatial features, a temporal path for sequential features, and a residual path for the reuse of the input features, as shown in Fig. 1. The spatial path (named spatial inception) is further split into four branches. We apply 1st to 4th order adjacency sampling as four sets of graph transformations with $1\times$ to $4\times$ hop connections. The following are the specified graph transformations with 1×1 convolution, batch normalization and ReLU. The temporal path (named temporal inception) consists of two sets of transformations. One set is a direct graph convolution on position features of the same joints across consecutive frames, and the other set is a graph convolution on motion features of the same joints across consecutive frames. Notably, this is the first time that the motion features of joints have been used in skeleton-based action recognition. Finally, in the merging stage, the outputs of both spatial path and temporal path are first concatenated and fused with 1×1 convolution. Then, features of the three paths are aggregated by summation. The whole block is named the spatio-temporal inception for its analogy to inception modules in CNNs.

To verify the superiority of our proposed spatio-temporal inception graph convolutional network (STIGCN) for skeleton-based action recognition, extensive experiments are conducted on two large-scale datasets. Our network outperforms state-of-the-art methods by a significant margin with only 1/5 of the parameters and 1/10 of the FLOPS. Furthermore, while other methods rely on a two-stream pipeline that requires skeleton data and crafted bone data as inputs, our method only requires raw skeleton data as input.

The contributions of this paper are summarized as follows:

- We propose a graph convolution backbone architecture, termed spatio-temporal inception graph convolutional network, for skeleton-based action recognition. This network overcomes the limitations of state-of-the-art methods in extracting and synthesizing information of different scales and transformations from different paths at different levels.
- To overcome the gap of the convolution operation between CNNs and GCNs, we reinvent the split-transform-merge strategy in GCNs for skeleton sequence processing.
- Our method indicates that increasing the number of transformation sets is a more effective way of gaining accuracy than simply creating wider GCNs. We hope this insight will facilitate the iteration of GCN-based backbones for spatio-temporal sequence analyses.
- On two large-scale datasets for skeleton-based action recognition, the proposed network outperforms state-of-the-art methods by a significant margin with surprisingly fewer parameters and FLOPs. The code and pretrained models will be released to facilitate related future research.

2 RELATED WORK

2.1 Skeleton-Based Action Recognition

In human action recognition, skeleton data have attracted increasing attention due to their robustness against body scales, viewpoints and backgrounds. Conventional methods in skeleton-data-based human action recognition utilize handcrafted feature descriptors to model the human body [49][8][18] [50]. However, these methods either ignore the information of interactions between specific sets of body parts or suffer from complicated design processes.

CNN-based and RNN-based methods have been well investigated for skeleton-based action recognition. CNN-based methods [31][21][20][34][24][28][27] formulate skeleton data as a pseudo-image based on manually designed transformation rules. RNN-based methods [4][41][33][44] [54][30][1] focus on modeling the temporal dependency of the inputs, where joint data of the human body are rearranged by grid-shaped structure. However, both CNN-based and RNN-based models neglect the co-occurrence pattern between spatial and temporal features since the skeleton data are naturally embedded in the form of graphs rather than a vector sequence or 2D grid.

Recently, GCNs have been introduced to skeleton-based action recognition and have achieved state-of-the-art performance. Most of these methods focus on the design/learning of the graph topology. Yan et al. [52] first introduced GCNs to model skeleton data and constructed a predefined graph with a fixed topology constraint. Shi et al. [42] proposed learning an adaptive graph by parameterizing the graphs and then updated the graph jointly with convolutional parameters. Gao et al. [9] introduced a high-order approximation for a larger receptive field of the graph and learned it by solving a sparsified regression problem. Peng et al. [39] tried to search for different graphs at different layers via NAS [55].

2.2 Backbone Convolutional Neural Networks

Many works have shown that synthesizing the outputs of different information paths in a building block is helpful. Deep neural decision forests [23] are tree-patterned multi-branch networks with learned splitting functions. GoogLeNet [45] uses an inception module to introduce multi-scale processing in different paths of the building block. The generated multi-scale features are merged by concatenation. ResNet [14] uses a residual learning framework in which the identity mapping of the inputs and the convolutional outputs are merged through elementwise addition. ResNeXt [51] designs a building block that aggregates a set of transformations. In DenseNet [16], the feature maps of all the preceding layers are fed into the current layer, and the feature maps of this layer are used as inputs to all the subsequent layers. A transition layer is designed to synthesize the feature maps of all the layers in one dense block. Qiu et al. [40] split 3×3×3 convolutions into 1×3×3 convolutional filters on spatial domain and 3×1×1 convolutions on temporal connections between adjacent feature maps. LocalCNN [53] uses a local operation as genetic building blocks for synthesizing global and local information in any layer. In the local path, Yang et al. [53] used a sampling module to extract local regions from the inputs, and the feature extraction module and feature fusion module were designed to transform and merge features.

2.3 Graph Convolutional Networks

GCNs are widely used on irregular data, e.g., social networks and biological data. The key challenge is to define convolutions over graphs, which is difficult due to the unordered graph data. The principle of constructing GCNs mainly follows the spatial perspective or the spectral perspective. Spatial perspective methods [5][37][10][36][22] directly perform convolutions on the graph vertexes and their neighbors, then normalize the outputs based on manually designed rules. Spectral GCNs transform graph signals into spectral domains by graph Laplacian methods [5][15], and then apply spectral filters on the spectral domains. In [11], Chebyshev expansions are used to approximate the graph Fourier transform, and the graph convolution is well approximated by a weighted summation of Chebyshev transformations over the skeleton data.

3 APPROACH

3.1 Motivation

The topology of the adjacency graph is the key factor for modeling correlations of the input skeletons. Therefore, state-of-the-art methods, including NAS [39], adaptive graph learning [42] and sparsified graph regression [9], mainly focus on the design/learning of the graph topology. However, once the graph is generated, only a single scale and one transformation exist in each layer of the networks. As a consequence, the backbone of these methods has intrinsic limitations on extracting and synthesizing information of different scales and transformations from different paths at different levels.

Intuitively, the success of the split-transform-merge strategy in recently developed backbone convolutional neural networks could be adopted for GCN-based backbone networks. However, due to the gap between graph skeleton data and traditional images/videos, it is not trivial to apply the split-transform-merge strategy in CNNs to GCNs. Specified modules are required to extract and synthesize features from multiple scales and transformations on graph data. To solve this problem, we design such modules, including a multi-scale spatial graph convolution module and a motion graph convolution module, and propose a simple graph convolution backbone architecture for skeleton-based action recognition.

3.2 Instantiation

In this section, we describe our design of spatio-temporal inception block for skeleton-based action recognition. First, we briefly show how to construct a multi-scale spatial graph convolution.

Consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, A\}$ composed of $n = |\mathcal{V}|$ nodes. The nodes are connected by $|\mathcal{E}|$ edges and the connections are encoded in the adjacency matrix $A \in \mathcal{R}^{n\times n}$. $F_{in} \in \mathcal{R}^n$ is the input representation of $\mathcal{G}$. After a graph Fourier transform, the convolutional filtering in spatial domain could be formulated as an inner-product operation in spectral domain [12]. Specifically, the graph Laplacian L, of which the normalized definition is $L = I_n - D^{-1/2}AD^{-1/2}$ and $D_{ij} = \sum_j A_{ij}$, is used for Fourier transform. Then a graph filtered by operator g_θ, parameterized by θ, can be formulated as

$$F_{out} = g_\theta(L)F_{in} = Ug_\theta(\Lambda)U^T F_{in}, \tag{1}$$

where F_{out} is the output feature of the input graph, U is the Fourier basis, $L = U\Lambda U^T$, and Λ is the corresponding eigenvalue of L. Hammond et al. [12] proved that the filter g_θ could be well approximated by Rth order Chebyshev polynomials,

$$F_{out} = \sum_{r=0}^{R} \theta'_r T_r(\hat{L})F_{in}, \tag{2}$$

where θ'_r denote Chebyshev coefficients. The Chebyshev polynomial is recursively defined as

$$T_r(\hat{L}) = 2\hat{L}T_{r-1}(\hat{L}) - T_{r-2}(\hat{L}') \tag{3}$$

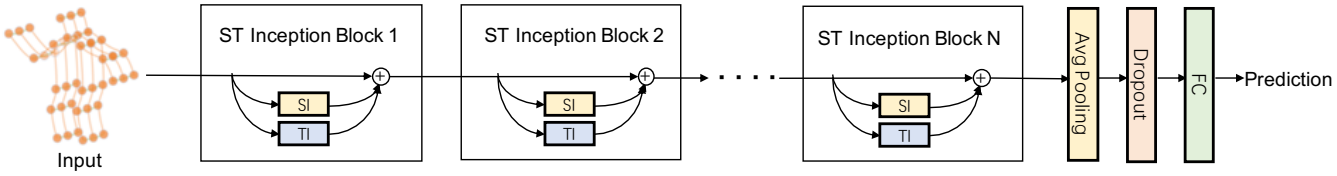


Figure 2: The overall architecture of our spatio-temporal inception graph convolutional network. It consists of a stack of ST inception blocks shown in Fig. 1 (b). This network takes raw skeleton data as inputs and is trained in an end-to-end manner.

with $T_0 = 1$ and $T_1 = \hat{L}$. $\hat{L} = 2L/\lambda_{max} - I_n$ is normalized to $[-1, 1]$.

In general, the graph L filtered by g_θ can be approximated as a linear combination of input representation transformed by Chebyshev polynomials. *As a consequence, a spatial graph convolution with a receptive field of k can be formulated as a linear transformation of a kth order Chebyshev polynomial matrix.*

3.2.1 Spatial Inception. An overall instantiation of our ST inception building block is shown in Fig. 1 (b). It consists of a spatial-inception (SI) path, a temporal inception (TI) path and a residual path. There are three components in each inception path: sampling, convolution and fusion. We now present the details of spatial inception first.

Adjacency Sampling. Inspired by the spectral formulation of graph convolutions, we reformulate the feature sampling module as a matrix multiplication operation between skeleton representations and a graph transformation defined as Chebyshev polynomials. The rth order Chebyshev polynomial $T_r(\hat{L})$, as defined in Eq. (3), represents the rth order hop connections between skeleton joints. Fig. 3 shows that most joints in the graph can be reached by 4 hop connections from the center joint (labeled as 1). Thus we choose $R = 4$ to approximate the multi-scale graph filtering operation. A higher order approximation may bring larger performance gains with additional computational costs, but this direction is not the priority of this paper. In detail, the 0th to 4th order Chebyshev polynomials are defined as follows:

$$\begin{aligned} T_0 &= I, \\ T_1 &= \hat{L}, \\ T_2 &= 2\hat{L}^2 - I, \\ T_3 &= 4\hat{L}^3 - 3\hat{L}, \\ T_4 &= 8\hat{L}^4 - 8\hat{L}^2 + I. \end{aligned} \tag{4}$$

As illustrated in Fig. 1, there are 4 branches in the SI path, corresponding to the 1st to 4th order graph transformations, respectively. T_0 represents the identity transformation, which is identical to the residual connection. Therefore, the 0th order sampling module is already included in the residual path. Following the adaptive graph topology introduced in [42], we apply layer-dependent bias and data-dependent bias to the transformation matrix for more flexible hop connections. Other parameters of the predefined transformation matrix are fixed during training except for the adaptive bias.

Convolution Module. The graph convolution module is used to extract graph features of every scale. It consists of a 1×1 convolutional layer, a batch normalization layer and a ReLU layer. The number of output feature maps is set to 1/4 of the total width of the spatial path for computational efficiency. Bottleneck-like architecture will be investigated in our future work.

Fusion Module. The feature fusion module is introduced to generate more robust and discriminative representations by synthesizing outputs of all paths. In this paper, the feature fusion module is formed as a concatenation layer of all the outputs, followed by a 1×1 convolutional layer with batch normalization and ReLU. The number of output channels of the 1×1 convolutional layer is set to the number of input channels to maintain the cardinality.

3.2.2 Temporal Inception. As shown in Fig. 1 (b), there are two branches in TI path. One branch directly takes features of the same joints in consecutive frames as inputs for position feature processing. The other branch feeds inputs into the motion sampling module for motion feature processing. This is the first time that motion features of joints are used in skeleton-based action recognition.

Motion Sampling. The second-order spatial information, i.e., the bone information, was first introduced in [42] and then widely used in later works [39][9]. However, the second-order temporal information is still ignored in previous works. In this paper, we design a motion sampling module to explicitly model the second-order temporal information, termed the motion information. In particular, the motion information is defined as the difference between consecutive frames. For example, given a frame of skeleton data at time t,

$$v_t = \{(x_1^{(t)}, y_1^{(t)}, z_1^{(t)}), \cdots, (x_n^{(t)}, y_n^{(t)}, z_n^{(t)})\}$$

where $(x_i^{(t)}, y_i^{(t)}, z_i^{(t)})$ are the 3D coordinates of the ith joint at time t and its next frame at time $t+1$ is

$$v_{t+1} = \{(x_1^{(t+1)}, y_1^{(t+1)}, z_1^{(t+1)}), \cdots, (x_n^{(t+1)}, y_n^{(t+1)}, z_n^{(t+1)})\}.$$

The vector of the motion is calculated as

$$\begin{aligned} m_t = v_{t+1} - v_t = \{&(x_1^{(t+1)} - x_1^{(t)}, y_1^{(t+1)} - y_1^{(t)}, z_1^{(t+1)} - z_1^{(t)}), \cdots, \\ &(x_n^{(t+1)} - x_n^{(t)}, y_n^{(t+1)} - y_n^{(t)}, z_n^{(t+1)} - z_n^{(t)})\}. \end{aligned}$$

The motion information can also be considered the optical flow of skeleton sequence. The joints of the skeleton data are similar to observed objects in RGB videos, and the optical flow is calculated as the relative motion of objects between consecutive frames. Therefore, it is natural to utilize the aforementioned motion sampling operation for motion feature processing.

Convolution and Fusion. The feature extraction module is designed to extract features from the frame sequence and motion sequence. Different from the convolution in the SI path, we use a 3×1 kernel for temporal convolution, where kernel size 3 corresponds to the temporal span, to construct temporal connections on adjacent

layer name	output size	components
	$3 \times 300 \times N_j$	data batch normalization
Stage 1	$64 \times 300 \times N_j$	$\begin{bmatrix} S=4 & T=2 \\ 1\times1, 16 & 1\times3, 32 \\ 1\times1, 64 & 1\times1, 64 \end{bmatrix} \times 1$
Stage 2	$64 \times 150 \times N_j$	1×2 max pooling, stride 1×2
		$\begin{bmatrix} S=4 & T=2 \\ 1\times1, 16 & 1\times3, 32 \\ 1\times1, 64 & 1\times1, 64 \end{bmatrix} \times 3$
Stage 3	$128 \times 75 \times N_j$	1×2 max pooling, stride 1×2
		$\begin{bmatrix} S=4 & T=2 \\ 1\times1, 32 & 1\times3, 64 \\ 1\times1, 128 & 1\times1, 128 \end{bmatrix} \times 3$
Stage 4	$256 \times 37 \times N_j$	1×2 max pooling, stride 1×2
		$\begin{bmatrix} S=4 & T=2 \\ 1\times1, 64 & 1\times3, 128 \\ 1\times1, 256 & 1\times1, 256 \end{bmatrix} \times 3$
	256	avg pooling, dropout
classifier	N_c	fc

Table 1: Architecture of spatio-temporal inception graph convolutional networks. "S=4" and "T=2" denote the numbers of branches in spatial inception and temporal inception, respectively, followed by the size of kernels in convolution modules and fusion modules. N_j is the number of joints in the graph. N_c is the number of action classes.

feature maps in the input sequence. The feature fusion module concatenates the outputs of the two temporal branches, followed by a 1×1 convolution, batch normalization and ReLU.

3.2.3 Spatio-Temporal Fusion. In the final merging stage, the outputs of spatial path, temporal path and residual path are aggregated by summation.

3.3 Network Architecture

To maintain consistent with state-of-the-art GCNs [52][42][39], we introduce ten ST inception blocks into our STIGCN. The overall architecture is illustrated in Fig. 2 and Table 1. It is a stack of basic building blocks shown in Fig. 1 (b). There are four stages in STIGCN, consisting of 1, 3, 3 and 3 building blocks, respectively. The numbers of output channels for these blocks are 64, 64, 64, 64, 128, 128, 128, 256, 256 and 256, respectively. Inside each block, "S=4" and "T=2" denote the numbers of branches in SI and TI, respectively, followed by the size of kernels in convolution modules and fusion modules. A batch normalization layer is added to the beginning to normalize the input data. Max pooling is applied after the first three stages, to construct a temporal hierarchical structure. The extracted features of the last block are fed into a global average pooling layer to pool feature maps of different samples to the same size. After a dropout layer, a softmax classifier is used to generate the final prediction.

4 EXPERIMENTS

4.1 Datasets and Evaluation Protocol

NTU RGB+D [41] is the most widely used and the largest multi-modality indoor-captured action recognition dataset. It contains

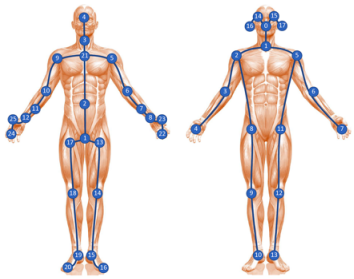


Figure 3: Left: the 25 joint labels in NTU RGB+D. Right: the 18 joint labels in Kinetics-Skeleton.

56,880 action clips (samples) from 60 action classes. For classification task, we follow the benchmark evaluations in the original work [41], which are cross-subject (X-Sub) and cross-view (X-View) evaluations. In X-Sub evaluation, 40,320 samples performed by 20 subjects are used as the training set, while the rest belong to the testing set. X-View evaluation divides the dataset according to camera views, where training and testing sets have 37,920 and 18,960 samples, respectively. For retrieval task, we follow the settings in [32] and split the dataset into two parts: a training set containing 47,180 samples from 50 action classes and a testing set containing 9,700 samples from the remaining 10 action classes. No data augmentation is performed in either task, and the data processing procedure is the same as which in [52].

Kinetics-Skeleton [19] is a large-scale human action dataset that contains 260,000 video clips from 400 action classes. Yan et al. [52] employed the open source toolbox OpenPose [2] to estimate coordinates of 18 joints in each frame. For classification task, the dataset is divided into a training set (240,000 samples) and a testing set (20,000 samples). For retrieval task, the dataset is randomly divided into a training set containing 228,273 samples from 350 classes and a testing set containing 31,727 samples from the remaining 50 action classes. We use the same data augmentation as in [52]. The definitions of the joints and their natural connections in these two datasets are shown in Fig. 3.

Evaluation Protocol. We calculate the top-1 accuracy on NTU RGB+D and the top-1/top-5 accuracy on Kinetics-Skeleton to evaluate the performance. And in retrieval task, we calculate the mean average precision (mAP) and cumulative matching characteristics (CMC) at rank-1 on both datasets to evaluate the performance.

4.2 Implementation Details

Our framework is implemented on PyTorch [38] and the code will be released later. Following [42], all experiments use stochastic gradient descent with a Nesterov momentum of 0.9. For NTU RGB+D, the batch size is 64, the weight decay is $5e-4$ and the initial learning rate is 0.1. The learning rate is divided by 10 at the 30th and 40th epochs. The training process ends at the 50th epoch. For Kinetics-Skeleton, the batch size is 128 and the training lasts 60 epochs. The learning rate is set to 0.1 at the beginning and is divided by 10 at the 45th and 50th epochs. The weight decay is $1.5e-4$.

4.3 Comparison with State-of-the-Art Methods

4.3.1 Action Classification. Our method is compared to state-of-the-art methods, including handcrafted-feature-based methods [49],

Input	Method	X-Sub(%)	X-View(%)
Joint	Lie Group [49]	50.1	82.8
	HBRNN [4]	59.1	64.0
	Deep LSTM [41]	60.7	67.3
	P-LSTM [41]	62.9	70.3
	ST-LSTM [33]	69.2	77.7
	STA-LSTM [44]	73.4	81.2
	VA-LSTM [54]	79.2	87.7
	TCN [21]	74.3	83.1
	SynCNN [34]	80.0	87.2
	Deep STGCK [26]	74.9	86.3
	ST-GCN [52]	81.5	88.3
	DPRL [46]	83.5	89.8
	SR-TSL [43]	84.8	92.4
	STGR-GCN [25]	86.9	92.3
	AS-GCN [29]	86.8	94.2
	GR-GCN [9]	87.5	94.3
	2S-AGCN [42]	86.6	93.7
	NAS-GCN [39]	87.6	94.5
	STIGCN (ours)	**90.1**	**96.1**
Joint+Bone	2S-AGCN [42]	88.5	95.1
	NAS-GCN [39]	89.4	95.7

Table 2: Comparison of classification accuracy on NTU RGB+D.

RNN-based methods [4][41][33][44][54], CNN-based methods [21][34], and GCN-based methods [26][52][46][43][25][29][9][42][39]. The results on NTU RGB+D and Kinetics-Skeleton are summarized in Table 2 and Table 3, respectively. We can see that STIGCN outperforms other methods on both datasets by a notable margin .

It is worth noting that both 2S-AGCN and NAS-GCN use extra bone data. They first train two independent models with joint data and bone data respectively, then ensemble the outputs of them during testing. STIGCN is trained in an end-to-end manner and outperforms aforementioned methods without any ensemble or extra bone data. When only joint data are used in 2S-AGCN and NAS-GCN, STIGCN outperforms them by 3.5% and 2.5% on NTU RGB+D X-Sub. We can conclude that STIGCN is much better at leveraging the multi-scale and multi-view knowledge from the joint sequence data, which significantly boosts the performance.

Moreover, Table 4 illustrates that STIGCN needs much fewer parameters and FLOPs than state-of-the-art methods. It is 1/8 parameters and 1/18 FLOPs compared with NAS-GCN [39]. This finding shows that STIGCN is more efficient at extracting representations from the skeleton sequence, which is important in practical scenarios. More importantly, as the width and depth of our network is the same as the network for single-stream inputs in 2S-GAN [42] and NAS-GAN [39], the superior performance indicates that increasing the number of transformation sets is a more effective way of gaining accuracy than simply creating wider GCNs.

Fig. 4 shows the training and testing curves of STIGCN and NAS-GCN (joint) on NTU RGB+D X-Sub. STIGCN exhibits lower training accuracy but higher testing accuracy, which indicates that STIGCN is more generalizable to the testing data. With much fewer parameters and FLOPs, STIGCN achieves a higher generalization capacity by alleviating the overfitting problem.

Input	Method	Top-1(%)	Top-5(%)
Joint	Feature [8]	14.9	25.8
	P-LSTM [41]	16.4	35.3
	TCN [21]	20.3	40.0
	ST-GCN [52]	30.7	52.8
	AS-GCN [29]	34.8	56.5
	2S-AGCN [42]	35.1	57.1
	NAS-GCN [39]	35.5	57.9
	STIGCN (ours)	**37.9**	**60.8**
Joint+Bone	2S-AGCN [42]	36.1	58.7
	NAS-GCN [39]	37.1	60.1

Table 3: Comparison of classification accuracy on Kinetics-Skeleton.

Method	Params(M)	GFLOPs
2S-AGCN [42]	7.0	37.3
NAS-GCN [39]	13.0	73.2
STIGCN (ours)	**1.6**	**4.0**

Table 4: Comparison of number of parameters and FLOPs.

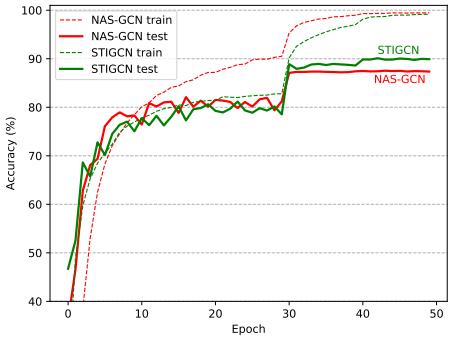


Figure 4: Training and testing curves on NTU RGB+D X-Sub.

4.3.2 Action Retrieval. To further validate the representation learning capability of STIGCN, we validate its performance on action retrieval tasks. Following the configurations in [32], we split the dataset into training and testing sets without action class overlap. The model is trained on training set with only the cross entropy loss, and tested by single query among all testing samples.

We choose 2S-AGCN [42] and NAS-AGCN [39] as baseline methods and train them by replicating the same training hyperparameters and architectures as those in the original papers. Features extracted from the trained models are used for retrieval. Since both methods are trained with a two-stream pipeline, we train two models using joint-skeleton data and bone-skeleton data separately and concatenate the output features for similarity calculation during retrieval. Our STIGCN uses only single-stream joint-skeleton data. For all three methods, outputs of the final average pooling layer are used as features for retrieval.

Table 5 and Table 6 demonstrate that STIGCN achieves the best performance. Specifically, compared with NAS-GCN [39], our model achieves a 2.04% mAP gain on NTU RGB+D and a 1.04% gain on Kinetics-Skeleton with only 1/8 of the parameters, 1/18 of the FLOPs and 1/2 of the output features. When the feature dimension is the same, the gains become 6.04% and 2.05%. The superior

Input	Method	Feat dim	mAP(%)	CMC(%)
Joint	2S-AGCN [42]	256	73.83	93.97
	NAS-GCN [39]	256	74.04	94.12
	STIGCN (ours)	**256**	**80.08**	**96.17**
Joint+Bone	2S-AGCN [42]	512	77.18	95.36
	NAS-GCN [39]	512	78.04	95.78

Table 5: Comparison of action retrieval results on NTU RGB+D. "Feat dim": the feature dimension.

Input	Method	Feat dim	mAP(%)	CMC(%)
Joint	2S-AGCN [42]	256	15.51	41.01
	NAS-GCN [39]	256	16.13	41.88
	STIGCN (ours)	**256**	**18.18**	**44.35**
Joint+Bone	2S-AGCN [42]	512	16.23	42.30
	NAS-GCN [39]	512	17.04	43.07

Table 6: Comparison of action retrieval result on Kinetics-Skeleton.

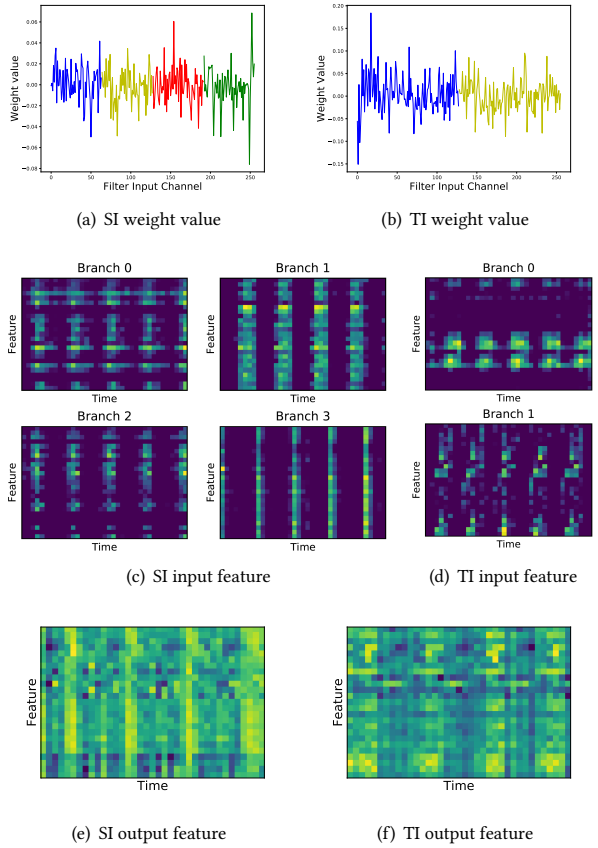
performance of STIGCN in the retrieval task reveals that the proposed GCN backbone is better at general representation learning for skeleton sequence data.

4.4 Ablation Analysis

4.4.1 Architecture. To validate the effectiveness of the proposed transformations in spatio-temporal inception block, including adjacency sampling of multiple orders, motion sampling and feature fusion, we present the performance of models with and without these components on NTU RGB+D X-Sub. To ensure a fair comparison, the number of channels inside the block is fixed across all settings. For example, if the number of channels of spatial path in (a) is n, then the number of channels in each of the two branches in spatial path in (b) is $n/2$. If the number of channels of temporal path in $(a-e)$ is m, then the number of channels in each of the two branches of temporal path in (f) is $m/2$.

Table 7 shows that the proposed network consistently benefits from the introduced transformations. Moreover, settings (e) and (f) show that the motion sampling module and feature fusion module provide much help in action recognition due to the synthesis of motion information and multiple scale information.

4.4.2 Fusion of Multiple Order Information. One key insight of our spatio-temporal inception design is the fusion of features from different branches, and each branch processes specific-order information. To inspect how the feature fusion module synthesizes features of different branches, we visualize the input feature maps, output feature maps and the weights of 1×1 convolutional layer in the fusion module of SI and TI paths. For convenience, we visualize the weights of one randomly selected filter, as well as its input and output feature maps with the maximum coefficient, as shown in Fig. 5. In this figure, (a) and (b) demonstrate that every branch contributes to the output; (c)-(f) indicate that different branches learn discriminative features. Thus, we can obtain more informative outputs by synthesizing these features. In general, STIGCN merges multiple sets of transformed information and benefits greatly from the aggregated representations.



(a) SI weight value (b) TI weight value

(c) SI input feature (d) TI input feature

(e) SI output feature (f) TI output feature

Figure 5: The weight values of one random filter in the fusion modules of SI and TI paths, as well as the corresponding input/output features in the feature fusion module. (a) and (b) are the values of the 256-dimensional weights. The horizontal axis denotes the input channels and each color denotes one branch. (c)-(f) are input and output feature maps with the maximum coefficient of different branches in SI and TI. The horizontal axis denotes the temporal dimension and the vertical axis denotes the feature channel dimension. These figures show that STIGCN merges multiple order information and generates more informative representations.

4.4.3 The Effect of the Representation Dimension. We reduce the dimension of output features of STIGCN, NAS-GCN and 2S-AGCN via principal component analysis (PCA) and test their performance on the NTU RGB+D retrieval task. The results are shown in Fig. 7. The representation learned by STIGCN consistently outperforms the others at varying dimensions from 200 to 3. An interesting observation is that the performance of 2S-GCN and NAS-GCN tends to be similar when the feature dimension is very small. STIGCN, in contrast, outperforms these two models by a wide margin. This finding shows that with the synthesis of multiple sets of transformed information, the representations from STIGCN are more robust to the change of feature dimension.

Setting	Order=1	Order=2	Order=3	Order=4	Motion	Fusion	Acc (%)
a	✓						86.53
b	✓	✓					87.67
c	✓	✓	✓				88.22
d	✓	✓	✓	✓			88.49
e	✓	✓	✓	✓	✓		89.45
f	✓	✓	✓	✓	✓	✓	90.10

Table 7: Performance of different settings in STIGCN. "Order=i" means that there are i adjacency sampling modules in spatial path. "Motion" denotes the motion sampling module, and "Fusion" represents the feature fusion module.

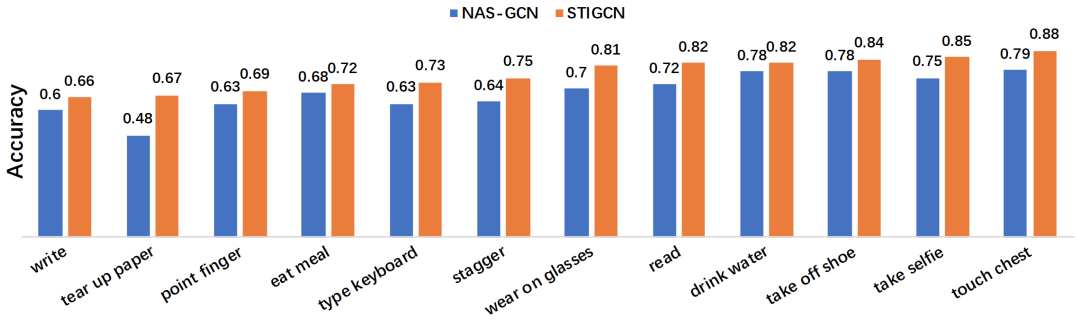


Figure 6: Comparison of classification accuracy of 12 difficult action classes on NTU RGB+D X-Sub.

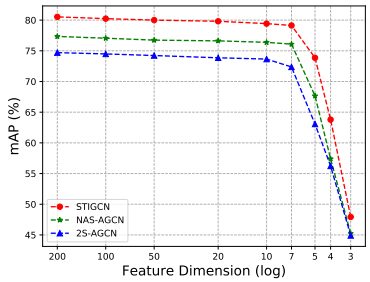
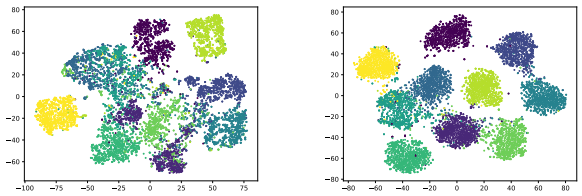


Figure 7: The evaluation of representations learned by different architectures with different dimensions.



(a) NAS-GCN (b) STIGCN

Figure 8: Visualizations of skeleton-sequence representation embeddings. Each sequence is visualized as one point, and the colors denote different action classes.

4.4.4 Embedding Representations. Fig. 8 further shows the t-SNE [47] visualization of the embedding of skeleton-sequence representations learned from NAS-GCN and STIGCN. We use the testing set of NTU RGB+D and the output representations are projected into a 2-dimensional space using t-SNE. This figure clearly shows that representations generated by STIGCN are semantically better grouped than those of NAS-GCN.

4.4.5 Classification Accuracy on Difficult Actions. We further analyze the performance of NAS-GCN and STIGCN on difficult action classes, i.e., actions whose classification accuracy is less than 80% in either NAS-GCN or STIGCN. As shown in Fig. 6, there are 12 difficult actions for NAS-GCN and 6 difficult actions for STIGCN. STIGCN outperforms 2S-AGCN on all these difficult actions. This finding shows that STIGCN has better ability to handle challenging actions.

5 CONCLUSION

In this paper, we propose a simple graph convolution backbone architecture called spatial temporal inception graph convolutional networks for skeleton-based action recognition. It overcomes the limitations of previous methods in extracting and synthesizing information of different scales and transformations from different paths at different levels. On two large-scale datasets, the proposed network outperforms state-of-the-art methods by a significant margin with surprisingly fewer parameters and FLOPs. Our method indicates that increasing the number of sets of transformations is a more effective way of gaining accuracy than simply creating wider GCNs. We hope this insight will facilitate the iteration of GCN-based backbones for spatio-temporal sequence analyses. In the future, we will explore more types of transformations for the design of graph convolution building blocks.

ACKNOWLEDGMENTS

This work was partially supported by Major Scientific Research Project of Zhejiang Lab (No. 2019DB0ZX01), National Key Research and Development Program of China under Grant 2017YFB1002203 and the National Natural Science Foundation of China under Grant 61872329.

REFERENCES

[1] Congqi Cao, Cuiling Lan, Yifan Zhang, Wenjun Zeng, Hanqing Lu, and Yanning Zhang. 2018. Skeleton-Based Action Recognition with Goated Convolutional Neural Networks. *IEEE Transactions on Circuits and Systems for Video Technology* 29, 11 (2018), 3247–3257.

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime multi-person 2d pose estimation using part affinity fields. *CVPR*, 7291–7299.

[3] AM Derrington and P Lennie. 1984. Spatial and temporal contrast sensitivities of neurones in lateral geniculate nucleus of macaque. *The Journal of physiology* 357, 1 (1984), 219–240.

[4] Yong Du, Wei Wang, and Liang Wang. 2015. Hierarchical recurrent neural network for skeleton based action recognition. *CVPR*, 1110–1118.

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. *NeurIPS*, 2224–2232.

[6] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. 2019. Slowfast networks for video recognition. In *CVPR*. 6202–6211.

[7] Daniel J Felleman and DC Essen Van. 1991. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)* 1, 1 (1991), 1–47.

[8] Basura Fernando, Efstratios Gavves, Jose M Oramas, Amir Ghodrati, and Tinne Tuytelaars. 2015. Modeling video evolution for action recognition. *CVPR*, 5378–5387.

[9] Xiang Gao, Wei Hu, Jiaxiang Tang, Jiaying Liu, and Zongming Guo. 2019. Optimized skeleton-based action recognition via sparsified graph regression. In *ACM Multimedia*. 601–610.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *NeurIPS*, 1024–1034.

[11] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.

[12] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.

[13] Fei Han, Brian Reily, William Hoff, and Hao Zhang. 2017. Space-time representation of people based on 3D skeletal data: A review. *Computer Vision and Image Understanding* 158 (2017), 85–105.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *CVPR*, 770–778.

[15] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *Arxiv* abs/1506.05163 (2015).

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. *CVPR*, 4700–4708.

[17] David H Hubel and Torsten N Wiesel. 1965. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of neurophysiology* 28, 2 (1965), 229–289.

[18] Mohamed E Hussein, Marwan Torki, Mohammad A Gowayyed, and Motaz El-Saban. 2013. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. *IJCAI*.

[19] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. [n.d.]. The kinetics human action video dataset. *Arxiv* abs/1705.06950 ([n. d.]).

[20] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. 2017. A new representation of skeleton sequences for 3d action recognition. *CVPR*, 3288–3297.

[21] Tae Soo Kim and Austin Reiter. 2017. Interpretable 3d human action analysis with temporal convolutional networks. In *CVPR workshops*. IEEE, 1623–1631.

[22] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. [n.d.]. Neural relational inference for interacting systems. *Arxiv* abs/1802.04687 ([n. d.]).

[23] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. 2015. Deep neural decision forests. *ICCV*, 1467–1475.

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

[25] Bin Li, Xi Li, Zhongfei Zhang, and Fei Wu. 2019. Spatio-temporal graph routing for skeleton-based action recognition. *AAAI* 33, 8561–8568.

[26] Chaolong Li, Zhen Cui, Wenming Zheng, Chunyan Xu, and Jian Yang. 2018. Spatio-temporal graph convolution for skeleton based action recognition. *AAAI*.

[27] Chuankun Li, Pichao Wang, Shuang Wang, Yonghong Hou, and Wanqing Li. 2017. Skeleton-based action recognition using LSTM and CNN. In *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 585–590.

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

[29] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. 2019. Actional-structural graph convolutional networks for skeleton-based action recognition. *CVPR*, 3595–3603.

[30] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. 2018. Independently recurrent neural network (indrnn): Building a longer and deeper rnn. *CVPR*, 5457–5466.

[31] Hong Liu, Juanhui Tu, and Mengyuan Liu. 2017. Two-stream 3d convolutional neural network for skeleton-based action recognition. *Arxiv* abs/1705.08106 (2017).

[32] Jun Liu, Amir Shahroudy, Mauricio Lisboa Perez, Gang Wang, Ling-Yu Duan, and Alex Kot Chichung. 2019. Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding. *IEEE transactions on pattern analysis and machine intelligence* (2019).

[33] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. 2016. Spatio-temporal lstm with trust gates for 3d human action recognition. *ECCV*, 816–833.

[34] Mengyuan Liu, Hong Liu, and Chen Chen. 2017. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition* 68 (2017), 346–362.

[35] Margaret Livingstone and David Hubel. 1988. Segregation of form, color, movement, and depth: anatomy, physiology, and perception. *Science* 240, 4853 (1988), 740–749.

[36] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. *CVPR*, 5115–5124.

[37] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. *ICML*, 2014–2023.

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[39] Wei Peng, Xiaopeng Hong, Haoyu Chen, and Guoying Zhao. 2019. Learning Graph Convolutional Network for Skeleton-based Human Action Recognition by Neural Searching. *Arxiv* abs/1911.04131 (2019).

[40] Zhaofan Qiu, Ting Yao, and Tao Mei. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. *ICCV*, 5533–5541.

[41] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. 2016. NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis. *CVPR* (2016), 1010–1019.

[42] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. 2019. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. *CVPR*, 12026–12035.

[43] Chenyang Si, Ya Jing, Wei Wang, Liang Wang, and Tieniu Tan. 2018. Skeleton-based action recognition with spatial reasoning and temporal stack learning. *ECCV*, 103–118.

[44] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. 2017. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. *AAAI*.

[45] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. *CVPR*, 1–9.

[46] Yansong Tang, Yi Tian, Jiwen Lu, Peiyang Li, and Jie Zhou. 2018. Deep progressive reinforcement learning for skeleton-based action recognition. *CVPR*, 5323–5332.

[47] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

[48] David C Van Essen, Jack L Gallant, et al. 1994. Neural mechanisms of form and motion processing in the primate visual system. *Neuron* 13, 1 (1994), 1–10.

[49] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. 2014. Human action recognition by representing 3d skeletons as points in a lie group. *CVPR*, 588–595.

[50] Junwu Weng, Chaoqun Weng, and Junsong Yuan. 2017. Spatio-temporal naive-bayes nearest-neighbor (st-nbnn) for skeleton-based action recognition. *CVPR*, 4171–4180.

[51] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. *CVPR*, 1492–1500.

[52] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. *AAAI*.

[53] Jiwei Yang, Xu Shen, Xinmei Tian, Houqiang Li, Jianqiang Huang, and Xian-Sheng Hua. 2018. Local convolutional neural networks for person re-identification. In *ACM Multimedia*. 1074–1082.

[54] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. 2017. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. *ICCV*, 2117–2126.

[55] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. https://arxiv.org/abs/1611.01578