Constructing Stronger and Faster Baselines for Skeleton-based Action Recognition

Yi-Fan Song, Zhang Zhang, *Member, IEEE,*
Caifeng Shan, *Senior Member, IEEE,* and Liang Wang, *Fellow, IEEE*

Abstract—One essential problem in skeleton-based action recognition is how to extract discriminative features over all skeleton joints. However, the complexity of the recent State-Of-The-Art (SOTA) models for this task tends to be exceedingly sophisticated and over-parameterized. The low efficiency in model training and inference has increased the validation costs of model architectures in large-scale datasets. To address the above issue, recent advanced separable convolutional layers are embedded into an early fused Multiple Input Branches (MIB) network, constructing an efficient Graph Convolutional Network (GCN) baseline for skeleton-based action recognition. In addition, based on such the baseline, we design a compound scaling strategy to expand the model's width and depth synchronously, and eventually obtain a family of efficient GCN baselines with high accuracies and small amounts of trainable parameters, termed EfficientGCN-Bx, where "x" denotes the scaling coefficient. On two large-scale datasets, *i.e.*, NTU RGB+D 60 and 120, the proposed EfficientGCN-B4 baseline outperforms other SOTA methods, *e.g.*, achieving **91.7%** accuracy on the cross-subject benchmark of NTU 60 dataset, while being $3.15\times$ smaller and $3.21\times$ faster than MS-G3D, which is one of the best SOTA methods. The source code in PyTorch version and the pretrained models are available at https://github.com/yfsong0709/EfficientGCNv1.

Index Terms—Action Recognition, Skeleton Sequence, Graph Convolutional Network, EfficientNet, Separable Convolution

1 Introduction

Human action recognition becomes increasingly crucial and achieves promising progress in various applications during the past decade, such as video surveillance, human-computer interaction, video retrieval and so on [1], [2], [3]. One essential problem in human action recognition is how to extract discriminative and rich features to fully describe the variations of spatial configurations and temporal dynamics in human actions.

Currently, skeleton-based representations are very popular for human action recognition, as human skeletons provide a compact data form to depict dynamic changes in human body movements [4]. Skeleton data is a time series of 3D coordinates of multiple skeleton joints, which can be either estimated from 2D images by pose estimation methods [5] or directly collected by multimodal sensors such as Kinect [6]. Moreover, compared to conventional RGB based action recognition methods, skeleton-based representations are more robust to the variations of illumination, camera viewpoints and other background changes. These merits inspire researchers to develop various methods to explore informative features from skeleton motion sequences for action recognition.

- *Yi-Fan Song, Zhang Zhang, and Liang Wang are with the School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing 100190, China, and also with the Center for Research on Intelligent Perception and Computing (CRIPAC), National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing 100190, China. (Email: yifan.song@cripac.ia.ac.cn, zzhang@nlpr.ia.ac.cn, wangliang@nlpr.ia.ac.cn)*
- *Caifeng Shan is with the College of Electrical Engineering and Automation, Shandong University of Science and Technology (SDUST), Qingdao 266590, China, and also with the Artificial Intelligence Research, Chinese Academy of Sciences (CAS-AIR), Beijing 100190, China. (Email: caifeng.shan@gmail.com)*

The development of skeleton-based action recognition can be divided mainly into two phases. In early years, conventional methods adopt Recurrent Neural Network (RNN)-based or Convolutional Neural Network (CNN)-based models to analyze skeleton sequences. For example, Du *et al.* [7] employ a hierarchical bidirectional RNN to capture rich dependencies between different body parts. Li *et al.* [8] design a simple yet effective CNN architecture for action classification from trimmed skeleton sequences. In recent years, due to the greatly expressive power for depicting structural data, graph-based models [9], [10] have been proposed for modeling dynamic skeleton sequences. Yan *et al.* [11] firstly propose the Spatial Temporal Graph Convolutional Networks (ST-GCN) for skeleton-based action recognition, after that increasing number of studies [12], [13], [14] are reported based on GCN models.

Nevertheless, for learning discriminative and rich features from skeleton sequences, current State-Of-The-Art (SOTA) models are often exceedingly sophisticated and over-parameterized, where the network often contains a multi-stream architecture with a large number of model parameters, which leads to a complicated training procedure and high computational cost (and thus low inference speed). For example, the 2s-AGCN in [13] contains about 6.94 million parameters, and it takes nearly 4 GPU-days for model training on the NTU RGB+D 60 dataset [15]. And the DGNN [16] contains more than 26 million parameters, which makes it very hard to do parameter tuning on large-scale datasets. The high model complexity has seriously limited the development of skeleton-based action recognition, while there are few literatures on this issue.

To tackle this problem, some efforts are made in this paper to extremely reduce the redundant trainable parameters while maintaining the model performance. Firstly,

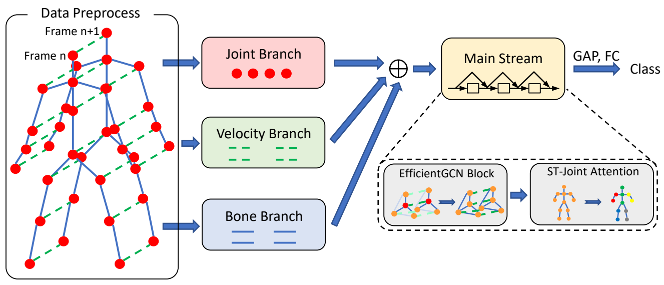

Fig. 1. The overall pipeline of our approach, where ⊕ represents concatenation operation, GAP and FC denote the Global Average Pooling operation and Fully Connected layer, respectively. (Best viewed in color.)

an early fused Multiple Input Branches (MIB) architecture is proposed to capture rich structure features from joints' spatial configurations and temporal dynamics in skeleton sequences, where the input three branches including joint positions (relative and absolute), motion velocities (one or two temporal steps), and bone features (lengths and angles) are fused in the early stage of the whole network. The early fused MIB architecture has proved to be a very effective way to reduce the model complexity without degrading the performance.

Secondly, besides the Basic Layer (BasicLayer) proposed in ST-GCN [11], we extend four kinds of convolutional layer in CNN, *i.e.*, Bottleneck Layer (BottleLayer) [17], Separable Layer (SepLayer) [18], Expanded Separable Layer (EpSepLayer) [19], and Sandglass Layer (SGLayer) [20], to the GCN network for extracting temporal dynamics and compressing the model size. These four layers can obviously reduce the amount of parameter tuning costs in training, and accelerate the model inference while running in real scenarios.

Thirdly, in order to determine the structural hyper-parameters for each block, we resort to the compound scaling method proposed by Tan *et al.* [21], which uniformly scales network width, depth and resolution with a set of fixed scaling coefficients. When applying this strategy to skeleton-based action recognition, the consideration of resolution is omitted because human skeleton is an artificially defined graph, hereby the modification of its resolution will destroy the graph convolutional operation. Consequently, we only scale the width and depth of our model to improve the modeling capability. This strategy improves the model performance in an efficient way, bringing a competitive model accuracy with significantly less trainable parameters and lower computational cost than other SOTA methods.

Finally, for more accurate recognition, inspired by Hou *et al.* [22], an attention module, named Spatial Temporal Joint Attention (ST-JointAtt), is proposed and inserted into each block of the model. This attention module aims to find the most essential joints from the whole skeleton sequence, and eventually enhances the model ability to extract discriminative features. Compared to other attention modules such as STC-attention (STCAtt) [23] and Part-wise Attention (PartAtt) [24], this new module jointly deals with the spatial and temporal attentions, while STCAtt is asynchronous and PartAtt ignores the temporal differences.

Combining these efforts mentioned above, a family of efficient GCN baselines with relatively small amounts of trainable parameters while keeping competitive performance to other SOTA methods is obtained and termed EfficientGCN-Bx, where "x" denotes the scaling coefficient in the model constructing strategy. The whole pipeline of EfficientGCN is shown in Fig. 1, where the three input sequences (Joint, Velocity and Bone) are initially extracted from the original skeleton sequence. Next, each input sequence is sent to a separate input branch consisting of several convolutional blocks. Then, the three branches will be fused and passed through the main stream. Similar to the input branches, the main stream is also composed of a number of convolutional blocks. Finally, the features of all frames and joints are globally averaged and fed into a Fully Connected (FC) layer for action classification. In this paper, three types of EfficientGCN with scaling coefficients of {0,2,4} are provided to verify the effectiveness of our approach. Compared to the most popular baseline, *i.e.*, 2s-AGCN [13], the EfficientGCN-B0 achieves over 1% relative performance improvement on both NTU RGB+D 60 [15] and 120 [25] datasets, while only needs **21.7**× smaller amount of model parameters and achieves **11.6**× faster inference speed. Besides, EfficientGCN-B4 obtains the SOTA performance on the two datasets, *e.g.*, **91.7%** on the cross-subject benchmark of NTU 60 dataset. Furthermore, when considering the model size and computational cost, EfficientGCN-B4 is **3.15**× smaller and **3.21**× faster than MS-G3D [26], which is one of the best SOTA methods in the field. For a clear illumination, Fig. 2 is drawn to demonstrate the accuracy-parameter performance of EfficientGCN, where the EfficientGCN is remarkably better than other SOTA methods.

This work is an extension of an earlier and preliminary version presented in [24], named ResGCN. Compared to our previous work, main modifications and contributions of this paper are summarized as follows:

- For convolutional layers, ResGCN proves that the BasicLayer and the BottleLayer are efficient for the GCN network. Besides, this work further introduces other three types of layers (SepLayer, EpSepLayer

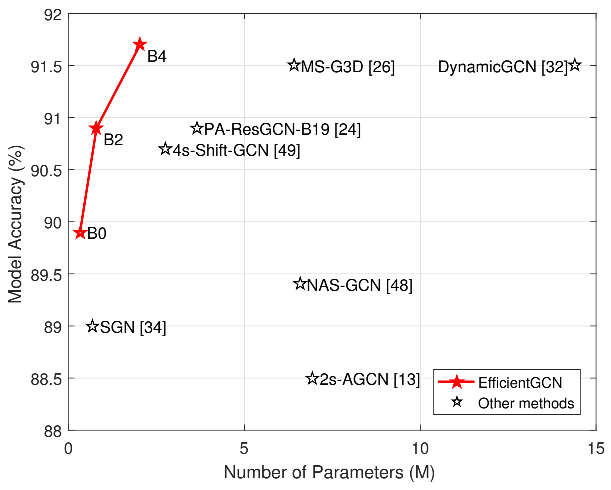

Fig. 2. Model size vs. model accuracy on the cross-subject benchmark of NTU 60 dataset. (Best viewed in color.)

and SGLayer) to skeleton-based action recognition, further improving the model efficiency.

- In previous work, each block in the model is constructed with manually selected hyper-parameters, where the number of layers and channels are fixed. In contrast, this study employs a compound scaling strategy to configure the model's width and depth with a scaling coefficient, which brings an effective and more flexible approach to design the model architecture.
- In ResGCN, an attention module named PartAtt is proposed to assign spatial attentions to body parts, while this paper offers a fine-grained module (ST-JointAtt), which not only considers the spatial attention, but also distinguishes the most essential temporal frames.
- Compared to the preliminary version, EfficientGCN achieves a better performance with a significantly lower number of parameters and calculations on two large-scale datasets, *i.e.*, NTU RGB+D 60 & 120. For example, EfficientGCN-B4 obtains a **91.7%** accuracy on the cross-subject benchmark with only 2.03 million parameters, significantly smaller and faster than ResGCN.

The remainder of this paper is organized as follows: Section 2 describes recent studies related to our work. Section 3 briefly introduces several crucial techniques of the proposed EfficientGCN. Section 4 discusses the details of our approach to EfficientGCN baselines. Extensive experiments on two large-scale datasets are reported in Section 5, and the conclusion is given in Section 6.

2 Related Work

2.1 Skeleton-based Action Recognition

Due to its compactness to the RGB-based representations, action recognition based on skeleton data has received increasing attentions. In an earlier work [27], a convolutional co-occurrence feature learning framework is proposed, where a hierarchical methodology is employed to gradually aggregate different levels of contextual information. The study in [28] designs a view adaptive model to automatically regulate observation viewpoints during the occurrence of an action, so as to obtain view invariant representations of human actions. However, due to the ignorance of spatial configurations, these CNN or RNN-based models gradually withdraw from the stage of frontier research.

Inspired by the booming graph-based methods, Yan *et al.* [11] firstly introduce GCN into the skeleton-based action recognition task, and propose the ST-GCN to model the spatial configurations and temporal dynamics of skeletons synchronously. Following this work, Song *et al.* [14], [29] aim to solve the occlusion problem in this task, and propose a multi-stream GCN to extract rich features from more activated skeleton joints. Liu *et al.* [26] explore the effects of multi-adjacency GCN and dilated temporal CNN, and design a sophisticated model named MS-G3D to disentangle multi-scale graph convolutions. Furthermore, the study in [30] provides a decoupling GCN to boost the graph modeling ability with no extra computation. To achieve global joint relationship modeling, Shi *et al.* [13] introduce the Non-local method [31] into a two-stream GCN model, named 2s-AGCN, which significantly improves the model accuracy. Similar as 2s-AGCN, Dynamic GCN proposed by Ye *et al.* [32] offer a novel method to model global dependency, by which the model achieves outstanding accuracy for skeleton-based action recognition. Although these methods achieve considerable performance, the increasing computational cost caused by the multi-stream structure becomes the obstacle to apply them in real tasks. Therefore, how to reduce the complexity of the GCN models is still a challenging problem.

2.2 Efficient Models

The model efficiency represented by the number of trainable parameters and Floating-point Operations Per Second (FLOPs), is always a non-negligible indicator in deep learning tasks. Extensive studies have made efforts to enhance the model efficiency, *i.e.*, reducing the amount of model parameters and FLOPs, such as MobileNetv1 [18], MobileNetv2 [19], MobileNeXt [20], and EfficientNet [21]. The model family of MobileNet mainly cuts the model size by separable convolutions, which factorizes standard convolutions into a depth-wise convolution applied to every channel individually and a 1×1 point-wise convolution to combine the outputs of the depth-wise convolution.

Some existing studies for skeleton-base action recognition have also been considering the model complexity problem. The study of [33] constructs a lightweight network with CNN-based blocks, which is not as accurate as GCN models. The work in [34] adopts a complex data preprocessing strategy, whose inputs include positions, velocities, frame indexes and joint types. This data preprocessing module enables the model to recognize actions with a shallow model, thereby achieves a very fast inference speed with 188 sequences/(second*GPU), yet its performance is obviously lower than other SOTA models.

2.3 Attention Models

Attention mechanisms have become an integral part of compelling sequence modeling in various tasks. Traditional attention modules for image processing can be divided into two categories: 1) channel-wise and 2) spatial-wise. Specifically, SENet [35] uses a bottleneck structure to obtain attention scores at channel dimension, providing a paradigm to build channel-wise attentions. Based on SENet, CBAM [36] not only focuses on channel-wise attention, but also utilizes convolutional layers to calculate attention maps at spatial dimension for adaptive feature refinement. Along with the popularity of Self Attention (SelfAtt) in Natural Language Processing (NLP), the Non-Local [31] method employs SelfAtt at spatial dimension, which globally explores attentions for the relationship between each pair of pixels.

With respect to action recognition, Baradel *et al.* [37] introduce the attention mechanism into an RGB-based action recognition model, which uses human poses to calculate spatial and temporal attentions. The study in [38] firstly introduces attention modules into skeleton-based action recognition, where a spatial-temporal attention Long Short-Term Memory (LSTM) is built to allocate different levels of

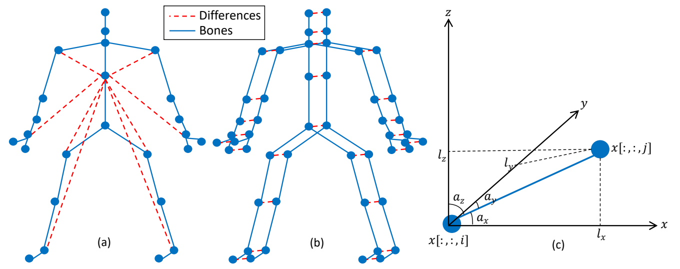

Fig. 3. The demonstration of input data. (a) is the relative positions, (b) is the motion velocities, and (c) demonstrates the 3D lengths and the 3D angles of a bone. (Best viewed in color.)

attention to the discriminative joints within each frame. Si *et al.* [39] also incorporate attention modules within LSTM units. Both of the two models apply attention modules for each frame individually, which may attend to some unstable noisy features. Besides, 2s-AGCN [13] offers a variant of attention model based on the Non-Local structure, and its improved version Dynamic-GCN [32] proposes another way to obtain the globally spatial attentions. In addition, Cheng *et al.* [30] embeds attention into its DropGraph module, leading to a significant accuracy increase.

3 Preliminary Techniques

In this section, we briefly discuss several crucial techniques used in the proposed EfficientGCN. Firstly, the data preprocessing module is introduced and formulated. Then, the GCN layer is reviewed. Finally, we compare the separable convolution to the standard convolution.

3.1 Data Preprocessing

Data preprocessing is very essential for skeleton-based action recognition, according to previous studies [13], [14], [40]. In this work, the input features after various preprocessing are mainly divided into three classes: **1)** joint positions, **2)** motion velocities and **3)** bone features.

Suppose that the original 3D coordinate set of an action sequence is $\mathcal{X} = \{x \in \mathbb{R}^{C_{in}\times T_{in}\times V_{in}}\}$, where C_{in}, T_{in}, V_{in} denote the coordinate, frame and joint, respectively. Then the relative position set is obtained as the normalized position features, *i.e.*, $\mathcal{R} = \{r_i | i = 1, 2, \cdots, V_{in}\}$, where

$$r_i = x[:,:,i] - x[:,:,c], \tag{1}$$

and c represents the index of the center spine joint. Next, the input of joint positions is formed by the concatenation of $\mathcal{X}$ and $\mathcal{R}$. Moreover, it is easy to obtain the two sets of motion velocities $\mathcal{F} = \{f_t | t = 1, 2, \cdots, T_{in}\}$ and $\mathcal{S} = \{s_t | t = 1, 2, \cdots, T_{in}\}$ with the following definitions

$$\begin{aligned} f_t &= x[:,t+2,:] - x[:,t,:],\\ s_t &= x[:,t+1,:] - x[:,t,:]. \end{aligned} \tag{2}$$

And the input of motion velocities is acquired by concatenating $\mathcal{F}$ and $\mathcal{S}$ for each joint to obtain a feature vector at each time. Finally, the input of bone features consists of the bone lengths $\mathcal{L} = \{l_i | i = 1, 2, \cdots, V_{in}\}$ and the bone angles

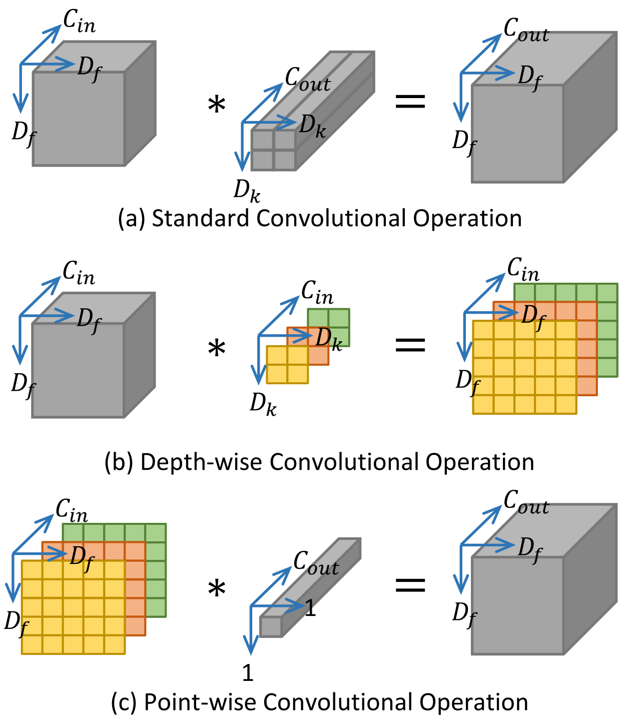

Fig. 4. Standard convolution vs. separable convolution for skeleton-based action recognition, where C_{in} and C_{out} denote the numbers of input and output channels, D_f and D_k denote the sizes of feature map and convolutional kernel, and $*$ represents convolutional operation.(Best viewed in color.)

$\mathcal{A} = \{a_i | i = 1, 2, \cdots, V_{in}\}$. To obtain these two sets, the lengths and angles of each bone are calculated by

$$\begin{aligned} l_i &= x[:,:,i] - x[:,:,i_{adj}],\\ a_{i,w} &= arccos(\frac{l_{i,w}}{\sqrt{l_{i,x}^2+l_{i,y}^2+l_{i,z}^2}}), \end{aligned} \tag{3}$$

where i_{adj} means the adjacent joint of the i-th joint, and $w \in \{x, y, z\}$ denotes the 3D coordinates. Fig. 3 displays the calculation method for these three inputs.

3.2 Graph Convolution

According to Yan *et al.* [11], graph convolutional operation for each frame t can be written as

$$f_{out}(v_{ti}) = \sum_{v_{tj}\in N(v_{ti})} \frac{1}{Z_{ti}(v_{tj})} f_{in}(v_{tj}) \cdot \mathbf{w}(l_{ti}(v_{tj})), \tag{4}$$

where v_{ti} denotes the i-th joint at the t-th frame, $f_{in}(\cdot)$ and $f_{out}(\cdot)$ are the input and output features of corresponding joints, $N(v_{ti})$ is the neighbor set of v_{ti}, the normalizing term Z_{ti} is set to balance the contributions of different neighbors, $\mathbf{w}(\cdot)$ is a weight function to allocate weights indexed by the label function $l_{ti}(\cdot)$, which is designed to construct several neighbor sets $N(v_{ti})$ by assigning different labels to each graph node. There are three label functions in [11], but we only choose the distance partitioning in our model, which defines $l_{ti}(v_{tj}) = d(v_{ti}, v_{tj})$, where $d(v_{ti}, v_{tj})$ denotes the graphic distance between v_{ti} and v_{tj}. The joints with the same distance will form a subset and share a learnable weight function $\mathbf{w}(\cdot)$. Generally, with the adjacency matrix $\mathbf{A}$, Eq.4 can be transformed into:

$$\mathbf{f}_{out} = \sum_{d=0}^{D} \mathbf{W}_d \mathbf{f}_{in} (\mathbf{\Lambda}_d^{-\frac{1}{2}} \mathbf{A}_d \mathbf{\Lambda}_d^{-\frac{1}{2}} \odot \mathbf{M}_d), \qquad (5)$$

where D is a predefined maximum graphic distance, $\mathbf{f}_{in}$ and $\mathbf{f}_{out}$ denote the input and output feature maps, $\odot$ means element-wise product, $\mathbf{A}_d$ represents the d-th order adjacency matrix that marks the pairs of joints with a graphic distance d, and $\mathbf{\Lambda}_d$ is used to normalize $\mathbf{A}_d$. $\mathbf{W}_d$ and $\mathbf{M}_d$ are both learnable parameters, which are utilized to implement the convolution operation and tune the importance of each edge, respectively.

3.3 Separable Convolution

Separable convolution is initially designed as the core layers on which MobileNet [18] is built, aiming at the deployment of deep learning models on computationally limited platforms such as robotics, self-driving car, augmented reality, *etc.*. As its name implies, separable convolution factorizes a standard convolution into a depth-wise convolution and a point-wise convolution. Concretely, for depth-wise convolution, a convolutional filter is only applied to one corresponding channel, while the point-wise convolution uses a 1×1 convolution layer to combine the output of depth-wise convolution and to adjust the number of output channels. The comparison of standard convolution and separable convolution is displayed in Fig. 4.

Concretely, suppose that the input feature size is $D_f \times D_f$, the kernel size is $D_k \times D_k$, the number of input and output channels are C_{in} and C_{out}, then the calculational process of standard convolution is illustrated in the top row of Fig. 4. This brings a batch of trainable parameters numbered $D_k \times D_k \times C_{in} \times C_{out} \times D_f \times D_f$. With respect to separable convolution shown in the bottom two rows of Fig. 4, the computational cost is changed to $D_k \times D_k \times C_{in} \times D_f \times D_f + C_{in} \times C_{out} \times D_f \times D_f$. Note that the most of trainable parameters are contained in the point-wise convolution [18], which is implemented by a 1×1 convolution. Thus, if the numbers of input and output channels are big enough, *e.g.*, > 256, then the computational cost of separable computation will decrease by nearly $D_k \times D_k$ times compared to that of standard convolution.

4 EfficientGCN

This part provides technical details to the proposed EfficientGCN for skeleton-based action recognition. Firstly, the MIB architecture is discussed and an example of EfficientGCN-B0 is constructed. Then, four kinds of convolutional layers popularly used in CNNs are extended to graph convolution to increase the efficiency of GCN blocks. Next, a compound scaling strategy is utilized to synchronously scale the width and depth of EfficientGCN-B0, generating a family of efficient baselines. Finally, an attention module is proposed to enhance the discrimination of skeleton features.

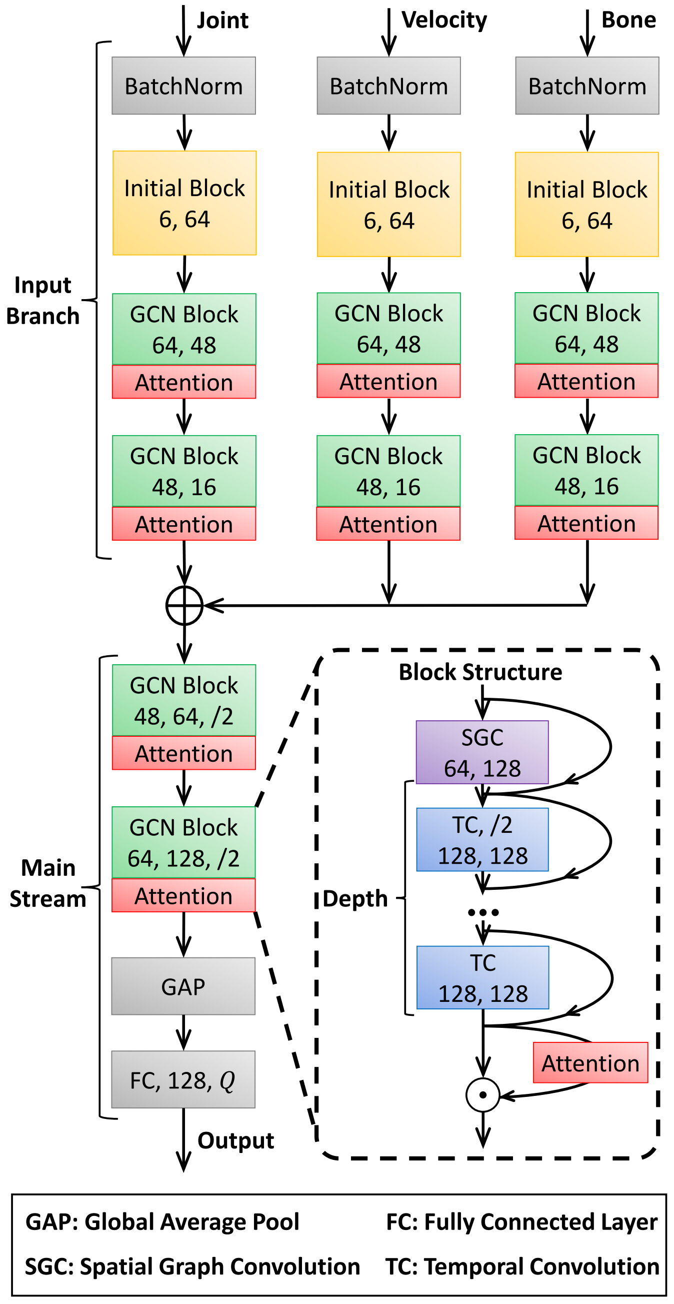

Fig. 5. The overview of the proposed EfficientGCN model, where the two numbers in each block denote input and output channels, Q is the number of action classes, $\oplus$ and $\odot$ represent concatenation and element-wise product, and $/2$ represents a stride of 2. (Best viewed in color.)

4.1 Model Architecture

After the data preprocessing module designed in Section 3.1, three types of input data are obtained, *i.e.*, Joint, Velocity and Bone. For current high-performance complex models, they usually apply a multi-stream architecture to handle these input data. For example, Shi *et al.* [13] take the joint data and bone data as inputs for feeding to two GCN branches with the same model structures separately, and eventually choose the fusion results of two streams as the final decision. This is an effective way to augment the input data and enhance the model performance. However, a multi-stream network often

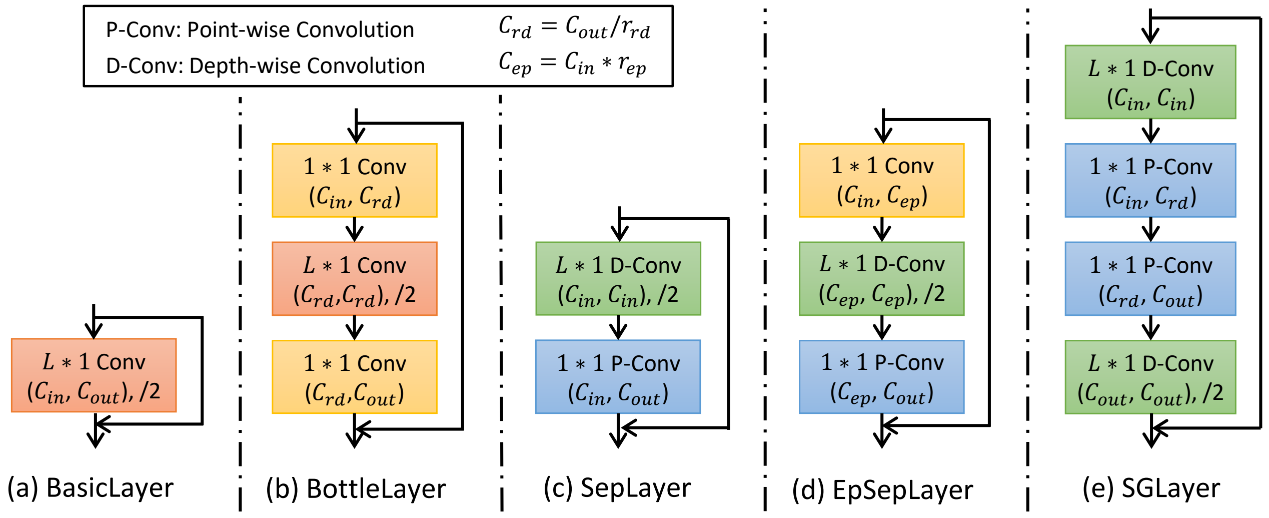
P-Conv: Point-wise Convolution
D-Conv: Depth-wise Convolution
$C_{rd} = C_{out}/r_{rd}$
$C_{ep} = C_{in} * r_{ep}$
$L * 1$ Conv (C_{in}, C_{out}), /2
(a) BasicLayer
$1 * 1$ Conv (C_{in}, C_{rd})
$L * 1$ Conv (C_{rd}, C_{rd}), /2
$1 * 1$ Conv (C_{rd}, C_{out})
(b) BottleLayer
$L * 1$ D-Conv (C_{in}, C_{in}), /2
$1 * 1$ P-Conv (C_{in}, C_{out})
(c) SepLayer
$1 * 1$ Conv (C_{in}, C_{ep})
$L * 1$ D-Conv (C_{ep}, C_{ep}), /2
$1 * 1$ P-Conv (C_{ep}, C_{out})
(d) EpSepLayer
$L * 1$ D-Conv (C_{in}, C_{in})
$1 * 1$ P-Conv (C_{in}, C_{rd})
$1 * 1$ P-Conv (C_{rd}, C_{out})
$L * 1$ D-Conv (C_{out}, C_{out}), /2
(e) SGLayer

Fig. 6. The details of various convolutional layers, where C_{in} and C_{out} denote the numbers of input and output channels, r_{rd} and r_{ep} are employed to reduce or expand the inner channels. (Best viewed in color.)

means high computational cost and difficulties of parameter turning on large-scale datasets. Thus, we devise the MIB architecture that fuses the three input branches at the early stage of our model, then apply one main stream to extract discriminative features. This architecture not only retains the rich input features, but also significantly suppresses the model complexity, and makes the training procedure easier to converge. An example of EfficientGCN with the MIB architecture is demonstrated in Fig. 5.

Concretely, the input branches are formed by orderly stacking a BatchNorm layer for fast convergence, an initial block implemented by ST-GCN layer [11] for data-to-feature transformation, and two GCN blocks with attentions for informative feature extraction. After the input branches, a concatenation operation is employed to fuse feature maps and then send them to the main stream, which is constructed by two GCN blocks. Finally, the output feature map of the main stream is globally averaged to a feature vector representing the input skeleton sequence, and an FC layer is used to determine the final action class.

4.2 Block Details

Inspired by MS-G3D [26] which achieves a considerable performance, as the subplot of Fig. 5 shows, the basic components of EfficientGCN (*i.e.*, GCN blocks) are implemented by orderly stacking a Spatial Graph Convolutional (SGC) layer, several Temporal Convolutional (TC) layers and an attention module. The depth for each GCN block is defined as the number of TC layers stacked in this block. Besides, for each layer, a residual link is utilized to make the model optimization performed earlier than the original unreferenced feature projection. It also should be noted that the first TC layer has a stride of 2 for each block in the main stream, which aims to compact the features and reduce the convolutional costs.

In detail, the SGC layer is implemented by a graph convolution mentioned in Section 3.2, and the attention module can be implemented by the proposed ST-JointAtt (see Section 4.4) or other traditional attention modules. For the implementation of TC layer, except for the basic $L \times 1$ convolution (**BasicLayer**) originally used in ST-GCN model [11], four types of convolutional architectures widely used in CNN literatures are adopted to further boost the efficiency of GCN models (see Fig. 6(a)-(e)). Specifically, within **BottleLayer**, a bottleneck structure [17] is utilized for temporal convolution, which is also used in our preliminary version of this paper, *i.e.*, ResGCN [24]. The other three layers, *i.e.*, **SepLayer**, **EpSepLayer** and **SGLayer**, are inspired by three versions of separable convolutions [18], [19], [20], all of which are composed of depth-wise convolutions and point-wise convolutions mentioned in Section 3.3. Note that the block with a certain layer, *e.g.*, BasicLayer, is denoted as **BasicBlock** for simplicity, and by analogy to **BottleBlock**, **SepBlock**, **EpSepBlock** and **SGBlock**.

4.3 Scaling Strategy

Empirically, expanding both the width (*e.g.*, WRN [42]) and the depth (*e.g.*, ResNet [17]) of networks will benefit the model capability and hence improve the performance. Commonly, the model's width and depth are defined as the numbers of channels and layers. These two factors are often considered independently, and scaled by handcrafted adjustment. However, it is intuitive that the model's width and depth are interrelated, hereby a compound scaling strategy is optimal for tuning network architectures. Accordingly, Tan *et al.* [21] construct a family of efficient models by jointly scaling the model width, depth and resolution from a baseline model. Based on this strategy, after removing the resolution factor, we propose a new scaling strategy for skeleton-based action recognition, by which a family of models are constructed in a principled way:

$$\begin{aligned} \text{width:} \quad & m_w = \alpha^{\phi} \\ \text{depth:} \quad & m_d = \beta^{\phi} \\ \text{s.t.} \quad & \alpha^2 \cdot \beta \approx 2 \\ & \alpha \geq 1, \beta \geq 1 \end{aligned} \tag{6}$$

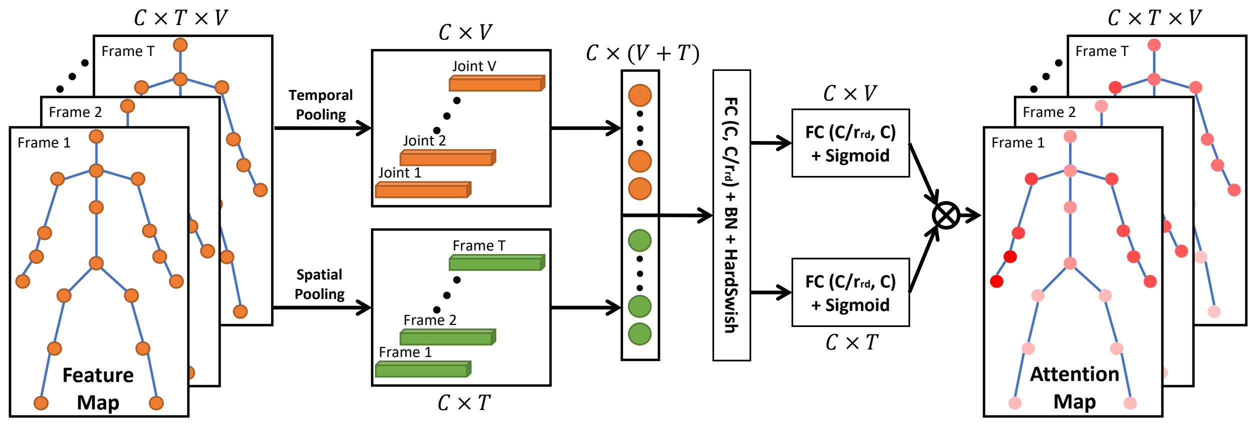

Fig. 7. The overview of the proposed ST-JointAtt module, where C, T, V denote the numbers of input channels, frames and joints respectively, $r_{rd} = 4$ is utilized to compact the features, $\otimes$ represents the channel-wise outer-product, BN denotes the batch normalization, HardSwish [41] and Sigmoid are both activated functions. (Best viewed in color.)

where m_w and m_d are width and depth multipliers, ϕ is a compound coefficient to state the available resources for model scaling, α and β are both hyper-parameters to control the resource assignments to the model's width and depth. In this paper, α and β are set to 1.2 and 1.35 by a small grid search (see Section 5.4). Here, the resolution factor is omitted due to the pre-defined skeleton structure. The modification of skeleton's resolution will destroy the graph convolutional operation.

The constraint conditions in Eq. 6 is utilized to constrain the increasing speed of the model size. When doubling the model depth or width, the FLOPs will approximately increase to 2 or 4 times. Thus, the FLOPs of the scaled model will be $(\alpha^2 \cdot \beta)^\phi \approx 2^\phi$ times than baseline. Note that this increase may differ from theoretical values due to the rounding function (see Appendix A).

4.4 Spatial Temporal Joint Attention

Previous attention modules for skeleton-based action recognition are mainly implemented by a Multi-layer Perception (MLP) like SENet structure [35], such as AGC-LSTM [39] and MS-AAGCN [23]. These modules are usually performed on each channel or spatial dimension independently, while other dimensions are globally averaged to a single unit. The preliminary version of this paper [24] proposes a PartAtt module which only works on the spatial dimension. However, intuitively, the spatial and temporal information could be relevant to each other. Thus, separately considering frames and joints is sub-optimal for weighting the importance of skeleton joints in different action phases. To address this issue, inspired by coordinate attention [22], we propose a novel attention module, named Spatial Temporal Joint Attention (ST-JointAtt), to jointly distinguish the most informative joints in certain frames from the whole skeleton sequence.

The overview of the proposed ST-JointAtt module is shown in Fig. 7, from which the input features are firstly averaged in frame- and joint-level respectively. Then, these pooled feature vectors are concatenated together and pass through an FC layer to compact information. Next, two independent FC layers are utilized to obtain two sets of attention scores for frame dimension and joint dimension respectively. Finally, the scores of frames and joints are multiplied by channel-wise outer-product, and the result can be seen as the attention scores for each joint in the whole action sequence. The proposed ST-JointAtt module can be formulated as

$$f_{inner} = \theta((pool_t(f_{in}) \oplus pool_v(f_{in})) \cdot W) \tag{7}$$

$$f_{out} = f_{in} \odot (\sigma(f_{inner} \cdot W_t) \otimes \sigma(f_{inner} \cdot W_v)) \tag{8}$$

where f_{in} and f_{out} denote input and output feature maps, $\oplus$ denotes concatenation operation, $\otimes$ and $\odot$ mean channel-wise outer-product and element-wise product, $pool_t(\cdot)$ and $pool_v(\cdot)$ are average pooling operations on frame- and joint-level respectively, $\sigma(\cdot)$ and $\theta(\cdot)$ represent Sigmoid and HardSwish activation functions, and $W \in \mathbb{R}^{C \times \frac{C}{r}}$, $W_t \in \mathbb{R}^{\frac{C}{r} \times C}$, $W_v \in \mathbb{R}^{\frac{C}{r} \times C}$ are trainable parameters. Note that W is shared by both frame- and joint-level pooled features.

5 Experimental Results

In this section, we evaluate the proposed EfficientGCN on two large-scale datasets, *i.e.*, NTU RGB+D 60 [15] and NTU RGB+D 120 [25]. Ablation studies are also performed to validate the contribution of each component in our model. For simplicity, all experiments in ablation studies choose EfficientGCN-B0 with EpSepBlocks ($r_{ep} = 2$) as the baseline model (details can be seen in Appendix A). Finally, result analysis and visualization are reported to prove the effectiveness of the proposed method.

5.1 Datasets

5.1.1 NTU RGB+D 60

This large-scale indoor captured dataset is provided in [15], which contains 56880 human action videos collected by three Kinect v2 cameras. These actions consist of 60 classes, where the last 10 classes are all interactions between two subjects. For simplicity, the input frame number is set to 300, and the sequences with less than 300 frames are padded by 0 at the end. Each frame contains no more than 2 skeletons, and each skeleton is composed of 25 joints. The authors of this dataset recommend two benchmarks: **1) cross-subject (X-sub)** contains 40320 training videos and 16560 evaluation videos divided by splitting the 40 subjects into two groups. **2) cross-view (X-view)** recognizes the videos collected by cameras 2 and 3 as training samples (37920 videos), while the videos collected by camera 1 are treated as evaluation samples (18960 videos). Note that there are 302 wrong samples selected by [25] that need to be ignored.

5.1.2 NTU RGB+D 120

This is currently the largest indoor action recognition dataset [25], which is an extended version of the NTU RGB+D 60. It totally contains 114480 videos performed by 106 subjects from 155 viewpoints. These videos consist of 120 classes, extended from the 60 classes of the previous dataset. Similarly, two benchmarks are suggested: **1) cross-subject (X-sub120)** divides subjects into two groups, to construct training and evaluation sets (63026 and 50922 videos respectively). **2) cross-setup (X-set120)** contains 54471 videos for training and 59477 videos for evaluation, which are separated based on the distance and height of their collectors. According to [25], 532 bad samples of this dataset should be ignored in all experiments.

5.2 Implementation Details

In our experiments, the maximum number of training epochs is set to 70. The initial learning rate is set to 0.1 and decays with a cosine schedule [43] after the 10th epoch. Moreover, a warmup strategy [17] is applied over the first 10 epochs, gradually increasing the learning rate from 0 to the initial value for a stable training procedure. The stochastic gradient descent (SGD) with the Nesterov momentum of 0.9 and the weight decay of 0.0001 is employed to tune the parameters. The hyper-parameters D and L defined in Section 3.2 and 4.2 are set to 2 and 5 respectively, which are determined by a grid search (see Appendix B). Other structural parameters will be discussed in ablation studies (Section 5.3). In addition, a dropout layer with 0.25 drop probability is added after the GAP layer and before the final FC layer to avoid overfitting. It also should be noticed that the activated function used in all convolutional blocks of our model is chosen as Swish function [44], which is similar with ReLU function but smooth and differentiable everywhere. In our experiments on X-view benchmark, a special data transformation [13] is performed for view alignment. All our experiments are performed on two TITAN RTX GPUs.

5.3 Ablation Studies

In this part, we mainly discuss the contributions of different components in the proposed EfficientGCN, including the selection of TC layers, the choice of the attention modules, the importance of data preprocessing module, and the necessity of the early fused architecture. This section explains why we use these settings to construct the baseline model EfficientGCN-B0.

TABLE 1
Comparisons of different TC layer types on X-sub benchmark in accuracy (%), FLOPs ($\times 10^9$) and parameter number ($\times 10^6$).

Layer	Acc.	FLOPs	# Param.
BasicLayer	89.6	2.96	0.34
BottleLayer ($r_{rd} = 4$)	88.8	2.62	0.26
SepLayer	89.2	2.62	0.26
EpSepLayer ($r_{ep} = 1$)	89.0	2.80	0.28
EpSepLayer ($r_{ep} = 2$) (Baseline)	**89.9**	3.08	0.32
EpSepLayer ($r_{ep} = 4$)	89.7	3.63	0.41
SGLayer ($r_{rd} = 2$)	89.5	2.73	0.29
SGLayer ($r_{rd} = 4$)	89.3	2.63	0.25

TABLE 2
Comparisons of different attention modules on X-sub benchmark in accuracy (%), FLOPs ($\times 10^9$) and parameter number ($\times 10^6$).

Attention	Acc.	FLOPs	# Param.
w/o Att	88.1	3.07	0.30
w/ ChannelAtt	89.1	3.07	0.32
w/ FrameAtt	88.1	3.07	0.30
w/ JointAtt	89.3	3.07	0.31
w/ STCAtt [23]	89.4	3.09	0.33
w/ PartAtt [24]	89.5	3.07	0.35
w/ ST-JointAtt (Baseline)	**89.9**	3.08	0.32

5.3.1 Comparisons of TC Layers

In Section 4.2 and Fig. 6, five types of TC layers are provided, namely BasicLayer, BottleLayer, SepLayer, EpSepLayer and SGLayer. To select the best TC layer for skeleton-based action recognition, we test them with the EfficientGCN-B0 model on X-sub benchmark, and the results are presented in Tab. 1. r_{ep} and r_{rd} denote the ratios of reducing and expanding channels in the corresponding layer. As shown in Tab. 1, the EpSepLayer with $r_{ep} = 2$ gets the best performance. Though its model complexity (FLOPs and number of parameters) is slightly higher than other types of layers, it receives a much higher accuracy. Accordingly, we choose the EpSepLayer with $r_{ep} = 2$ as the base layer to build the baseline model.

5.3.2 Comparisons of Attention Modules

To enhance the model ability to extract informative features, the ST-JointAtt module is designed and embedded into the convolutional blocks. We compare ST-JointAtt with other attention modules, *i.e.*, ChannelAtt, FrameAtt, JointAtt, STCAtt, and PartAtt, and the results are presented in Tab. 2. Note that the first three attention modules are designed by ourselves based on the SENet [35] structure, and the last two are proposed in [23], [24], respectively. It is observed that, after incorporating attention modules, the model achieves obvious accuracy improvement. And ST-JointAtt produces the best accuracy, while other five modules are slightly worse. This indicates the importance of inserting attention modules and the effectiveness of the ST-JointAtt module in finding the most informative joints and frames from the whole skeleton sequence.

TABLE 3
Comparisons of different inputs on X-sub benchmark in accuracy (%), FLOPs ($\times 10^9$) and parameter number ($\times 10^6$).

Inputs	Acc.	FLOPs	# Param.
Joint	87.6	1.63	0.22
Velocity	85.4	1.63	0.22
Bone	87.9	1.63	0.22
Joint + Velocity	88.6	2.29	0.27
Joint + Bone	88.2	2.29	0.27
Velocity + Bone	89.4	2.29	0.27
Joint + Velocity + Bone (Baseline)	**89.9**	3.08	0.32

TABLE 4
Comparisons of different fusion stages on X-sub benchmark in accuracy (%), FLOPs ($\times 10^9$) and parameter number ($\times 10^6$).

Fusion	Acc.	FLOPs	# Param.
Before 1st stage	88.6	2.64	0.29
After 1st stage	89.0	2.83	0.31
After 2nd stage (Baseline)	**89.9**	3.08	0.32
After 3rd stage	89.7	4.01	0.44
After 4th stage	89.0	4.89	0.66
At the score layer	89.4	7.92	0.87

5.3.3 Necessity of Data Preprocessing

Data preprocessing is essential to enhance the model performance, proven by several previous studies [13], [14], [40]. We summarize these preprocessing methods into three classes, *i.e.*, joint positions, motion velocities and bone features. To explore the necessity of each input branch, we present Tab. 3, from which the model with three inputs is clearly better than others. With the increase of branches, the model performance is improved. This phenomenon further confirms the effectiveness of the data preprocessing module.

5.3.4 Necessity of Early Fused Architecture

Originally proposed by 2s-AGCN [13], multi-stream models fused at the final score layer gradually dominate the model architectures for skeleton-based action recognition. Although fusing at the score layer brings an outstanding model accuracy, the model size and computational cost are meanwhile increased greatly, leading to an exceedingly sophisticated and over-parameterized model architecture. However, we find that many parameters in multi-stream models are redundant, with no contribution to model performance. Thus, we construct an early fused MIB architecture in this paper. Tab. 4 gives the results of models with different fusion stages. It is seen that fusing after the 2nd stage is the inflection point of the accuracy-parameter curve, which is the best model to balance accuracy and complexity. So, the MIB fused after the 2nd stage is chosen as the architecture of our baseline model.

5.4 Comparisons of Compound Scaling Strategies

In this section, we test several compound scaling strategies with different width and depth scaling hyper-parameters (α and β), which are mentioned in Section 4.3. To choose the best settings of α and β within the constraint conditions in Eq. 6, we set α to $\{1.0, 1.1, 1.2, 1.3, 1.4\}$, respectively, and the hyper-parameter β can be calculated by Eq. 6 with a precision of 0.05. Then, from the baseline model EfficientGCN-B0, we construct five EfficientGCN-B2 and five EfficientGCN-B4 with different scaling hyper-parameters. These models' accuracies and parameter numbers are shown in Fig. 8, from which the EfficientGCN-B4 obtains the best performance when $\alpha = 1.2$ and $\beta = 1.35$. It is also worth noting that, merely increasing either width or depth may hurt the model performance, whose accuracies with EfficientGCN-B4 are obviously lower than those of compound scaling strategies. This phenomenon is caused by the progressiveness of the compound scaling strategy.

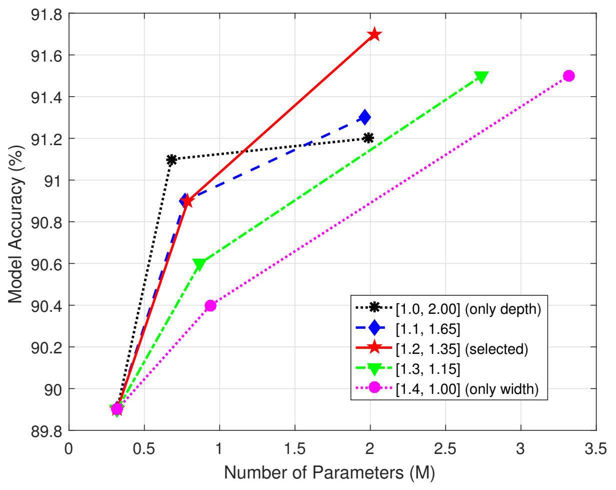

Fig. 8. Model size vs. model accuracy of different width and depth scaling hyper-parameters (α and β). (Best viewed in color.)

5.5 Comparisons with SOTA Methods

5.5.1 NTU RGB+D 60 Dataset

We compare the model accuracies of our EfficientGCN family against previous SOTA methods on both X-sub and X-view benchmarks of NTU 60 datasets. The results are displayed in Tab. 5. There are several typical comparisons shown as follows:

From Tab. 5, the baseline model EfficientGCN-B0 obtains a good performance, *i.e.*, 89.9% for X-sub benchmark and 94.7% for X-view benchmark. When applying the proposed scaling strategy with coefficients of 2 and 4, the EfficientGCN-B2 and EfficientGCN-B4 are built, and achieve excellent performances on the two benchmarks. Especially for EfficientGCN-B4, its accuracy on X-sub benchmark is 91.7%, outperforming other SOTA methods. Here, three typical methods should be noticed. **1)** The first one is ST-GCN [11], which is currently the most popular backbone model for skeleton-based action recognition. Compared with ST-GCN, our EfficientGCN-B4 leads over 10% on X-sub benchmark. **2)** 2s-AGCN [13] is another popular baseline for skeleton-based action recognition. The proposed baseline EfficientGCN-B0 outperforms 2s-AGCN in both accuracy and efficiency. **3)** The third one is MST-GCN [50], which is the current SOTA method with the GCN technique.

TABLE 5
Comparisons with SOTA methods on NTU 60 dataset in accuracy (%). The top part consists of several models without GCN techniques, while the other part contains some graph-based models.

Model	Conference	X-sub	X-view
HBRNN [7]	CVPR15	59.1	64.0
ST-LSTM [45]	ECCV16	69.2	77.7
STA-LSTM [38]	AAAI17	73.4	81.2
HCN [27]	IJCAI18	86.5	91.1
VA-fusion [28]	TPAMI19	89.4	95.0
ST-GCN [11]	AAAI18	81.5	88.3
SR-TSL [40]	ECCV18	84.8	92.4
RA-GCNv1 [14]	ICIP19	85.9	93.5
RA-GCNv2 [29]	TCSVT20	87.3	93.6
AS-GCN [46]	CVPR19	86.8	94.2
2s-AGCN [13]	CVPR19	88.5	95.1
AGC-LSTM [39]	CVPR19	89.2	95.0
DGNN [16]	CVPR19	89.9	96.1
PL-GCN [47]	AAAI20	89.2	95.0
NAS-GCN [48]	AAAI20	89.4	95.7
SGN [34]	CVPR20	89.0	94.5
4s-Shift-GCN [49]	CVPR20	90.7	96.5
MS-G3D [26]	CVPR20	91.5	96.2
DC-GCN+ADG [30]	ECCV20	90.8	96.6
PA-ResGCN-B19 [24]	ACMMM20	90.9	96.0
Dynamic-GCN [32]	ACMMM20	91.5	96.0
MST-GCN [50]	AAAI21	91.5	**96.6**
EfficientGCN-B0	–	89.9	94.7
EfficientGCN-B2	–	90.9	95.5
EfficientGCN-B4	–	**91.7**	95.7

The EfficientGCN-B4 is slightly better than MST-GCN in accuracy on X-sub benchmark. With respect to the comparisons within the EfficientGCN family, the accuracy shows a gradually improving trend with the increase of scaling coefficients.

STA-LSTM [38] and AGC-LSTM [39] are also enhanced by attention modules. However, there are obvious differences between EfficientGCN and these two models, *e.g.*, our attention module works jointly on frames and joints, while their models use the attention modules for frames and joints individually. The performance of EfficientGCN-B4 greatly exceeds STA-LSTM over 10% on the two benchmarks, and outperforms AGC-LSTM by 2.5% and 0.7% on X-sub and X-view benchmarks, respectively. Moreover, 2s-AGCN and its improved versions Dynamic-GCN [32] can also be considered as a variant of globally spatial attention (non-local structure), while they achieve comparable performances to our model. In addition, DC-GCN+ADG [30] utilizes attention mechanism to guide its DropGraph module. This model is better than EfficientGCN-B4 on X-view benchmark, but significantly worse on X-sub benchmark.

These results imply that the proposed EfficientGCN is a baseline with competitive performance compared to SOTA methods. We consider that this is caused by the strong capability of the compound scaling strategy to balance the model accuracy and complexity, hereby a wider and deeper model can be constructed and easily trained to achieve better performance. Moreover, the proposed ST-JointAtt module contributes to the model accuracy, which makes the model

TABLE 6
Comparisons with SOTA methods on NTU 120 dataset in accuracy (%). The top part consists of several models without GCN techniques, while the other part contains some graph-based models.

Model	Conference	X-sub120	X-set120
PA-LSTM [15]	CVPR16	25.5	26.3
ST-LSTM [45]	ECCV16	55.7	57.9
FSNet [51]	TPAMI19	59.9	62.4
ST-GCN [11]	AAAI18	70.7*	73.2*
SR-TSL [40]	ECCV18	74.1*	79.9*
RA-GCNv1 [14]	ICIP19	74.4	79.4
RA-GCNv2 [29]	TCSVT20	81.1	82.7
AS-GCN [46]	CVPR19	77.9*	78.5*
2s-AGCN [13]	CVPR19	82.5*	84.2*
SGN [34]	CVPR20	79.2	81.5
4s-Shift-GCN [49]	CVPR20	85.9	87.6
MS-G3D [26]	CVPR20	86.9	88.4
DC-GCN+ADG [30]	ECCV20	86.5	88.1
PA-ResGCN-B19 [24]	ACMMM20	87.3	88.3
Dynamic-GCN [32]	ACMMM20	87.3	88.6
MST-GCN [50]	AAAI21	87.5	88.8
EfficientGCN-B0	–	85.9	84.3
EfficientGCN-B2	–	87.9	88.0
EfficientGCN-B4	–	**88.3**	**89.1**

*: These results are implemented based on the released codes.

prone to discover the most informative joints.

5.5.2 NTU RGB+D 120 Dataset

As a newly released dataset, there are less papers reporting results on the NTU 120 dataset. For comprehensive comparisons, four popular models, *i.e.*, ST-GCN [11], SR-TSL [40], AS-GCN [46] and 2s-AGCN [13], are implemented by ourselves on the NTU 120 dataset, based on their released codes. Tab. 6 presents the experimental results, from which we can find an obvious gap between the proposed EfficientGCN and other models. For example, EfficientGCN-B4 outperforms the current SOTA method, MST-GCN [32], by 0.8% on X-sub120 benchmark and 0.3% on X-set120 benchmark. Similar to the results on the NTU 60 dataset, the performance of EfficientGCN is mainly attributed to the contribution of the compound scaling strategy.

5.5.3 Model Complexity

In order to verify the efficiency of our model, we compare our EfficientGCN family with other methods in accuracy and model complexity (FLOPs and number of parameters) on X-sub benchmark. The experimental results are demonstrated in Tab. 7. This table is divided into three parts, where the models with similar accuracies are grouped into one part. The ratios following FLOPs and parameter number denote the ratio between the model and its corresponding EfficientGCN. Due to the lack of reported FLOPs and parameter numbers for most models, we obtain the results by their released codes or directly asking their authors for helps. Note that the FLOPs of SGN [34] and Dynamic-GCN [32] are reported by their authors but not presented in this table, because these two models contain well-designed data transformation modules, which resize the original skeleton sequence to a very short sequence (*e.g.*, 20 frames), instead

TABLE 7
Comparisons with SOTA methods on X-sub benchmark in accuracy (%), FLOPs ($\times 10^9$) and parameter number ($\times 10^6$). The models in three parts are compared with EfficientGCN-B0, B2, B4, respectively.

Model	Acc.	FLOPs	Ratio	# Param.	Ratio
EfficientGCN-B0	**89.9**	**3.08**	1×	**0.32**	1×
ST-GCN [11]	81.5	16.32*	5.30×	3.10*	9.69×
SR-TSL [40]	84.8	4.20*	1.36×	19.07*	59.59×
RA-GCNv1 [14]	85.9	32.80*	10.65×	6.21*	19.40×
RA-GCNv2 [29]	87.3	32.80*	10.65×	6.21*	19.40×
AS-GCN [46]	86.8	26.76*	8.69×	9.50*	29.69×
2s-AGCN [13]	88.5	37.32*	12.12×	6.94*	21.69×
SGN [34]	89.0	–	–	0.69	2.16×
AGC-LSTM [39]	89.2	–	–	22.89†	71.53×
DGNN [16]	89.9	–	–	26.24†	82.00×
NAS-GCN [48]	89.4	–	–	6.57†	20.53×
PL-GCN [47]	89.2	–	–	20.70†	64.69×
EfficientGCN-B2	**90.9**	**6.12**	1×	**0.79**	1×
4s-Shift-GCN [49]	90.7	10.0	1.63×	2.76*	3.49×
DC-GCN+ADG [30]	90.8	25.72*	4.20×	4.96*	6.28×
PA-ResGCN-B19 [24]	90.9	18.52*	3.03×	3.64	4.61×
EfficientGCN-B4	**91.7**	**15.24**	1×	**2.03**	1×
MS-G3D [26]	91.5	48.88*	3.21×	6.40	3.15×
Dynamic-GCN [32]	91.5	–	–	14.40†	6.96×
MST-GCN [50]	91.5	–	–	12.00	5.91×

*: These results are implemented based on the released codes.
†: These results are provided by their authors.

of performing on the whole 300 frames. Thus, we ignore the FLOPs of these two models and only give the numbers of their parameters for fair comparisons.

In the top part of Tab. 7, it is observed that there is a huge gap between the efficiencies of EfficientGCN-B0 and previous models. Compared to the original baseline, *i.e.*, ST-GCN [11], EfficientGCN-B0 outperforms by 8.4% in accuracy, with a 5.30 times less FLOPs and a 9.69 times less parameters. DGNN [16] obtains the same accuracy as EfficientGCN-B0, but its amount of trainable parameters is exceedingly larger, about 82 times than EfficientGCN-B0. SGN [34] is a lightweight and efficient model for skeleton-based action recognition, which achieves 89.0% accuracy with only 0.69×10^6 parameters. However, it is still worse than our EfficientGCN-B0 in both model accuracy and model size.

As to the middle part, EfficientGCN-B2 achieves 90.9% accuracy with 6.12×10^9 FLOPs and 0.79×10^6 parameters, which are about 3 times less than other three models, including the preliminary version of this paper (PA-ResGCN-B19). Furthermore, the bottom part shows that EfficientGCN-B4 achieves a SOTA accuracy with a small amount of trainable parameters. Though it seems to have a large FLOPs, it is still less than the models with similar performance. These results clearly show that the proposed method brings a remarkable improvement in both model accuracy and model complexity, which will benefit to the real development of skeleton-based action recognition.

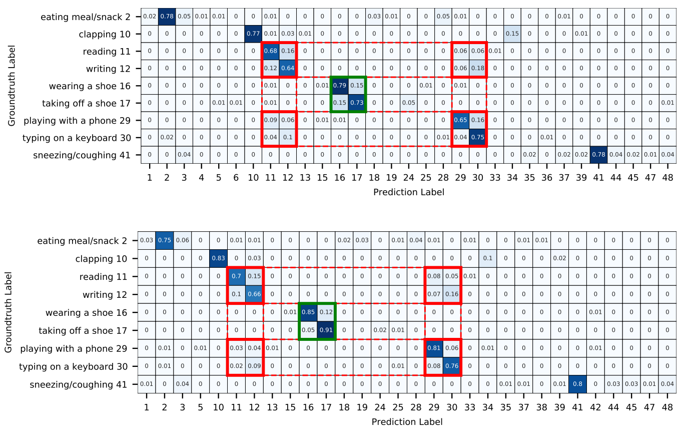

Fig. 9. Confusion matrices of EfficientGCN-B0 (top) and EfficientGCN-B4 (bottom) with failure actions (less than 80% accuracy on X-sub benchmark), where the numbers in coordinate axes represent the indexes of each action category, the red and green rectangles denote two groups of similar actions. (Best viewed in color.)

5.6 Discussion and Visualization

5.6.1 Confusion Matrices and Failure Cases

Although EfficientGCN receives promising results on the large-scale datasets, there are still some actions difficult to be well recognized. As Fig. 9 displays, we draw the confusion matrices of some actions for the proposed EfficientGCN-B0 and EfficientGCN-B4, respectively, where the selected actions are determined according to their insufficient accuracies (less than 80% on X-sub benchmark). From the top row of Fig. 9, two groups of similar actions should be noticed. The first one is marked by the red rectangles, including *reading*, *writing*, *playing with a phone*, and *typing on a keyboard*. All these actions are mainly performed by the slight shaking of two hands, which are extremely similar at spatial configurations and temporal dynamics. The second group, surrounded by green rectangles, consists of two similar actions, *i.e.*, *wearing a shoe* and *taking off a shoe*. These two actions have similar spatial configurations, but different temporal dynamics. With respect to EfficientGCN-B4 (bottom row of Fig. 9), the recognition accuracies of actions in the second group receive a significant improvement, while the actions in the first group are still hard to be distinguished.

This issue is mainly caused by the lack of joints to represent two hands, thus information of two hands is generally insufficient. Furthermore, the fringe joints of NTU 60 dataset often contain much noise (*e.g.*, the 4th, 7th, and 8th frames of *throwing* in Fig. 10), which makes a huge influence on capturing the discriminative features. However, our approach is not particularly designed for dealing with noise. Therefore, it is still challenging to recognize such subtle actions.

5.6.2 Class Activation Maps

To show how our model works, the activation maps of some skeleton sequences are calculated by class activation map [52], as presented in Fig. 10, in which the activated joints in several sampled frames are displayed. From this figure, we can find that the EfficientGCN-B4 model successfully concentrates on the most informative joints, *i.e.*, left arm for *drinking water*, *throwing* and *waving hand*, upper body

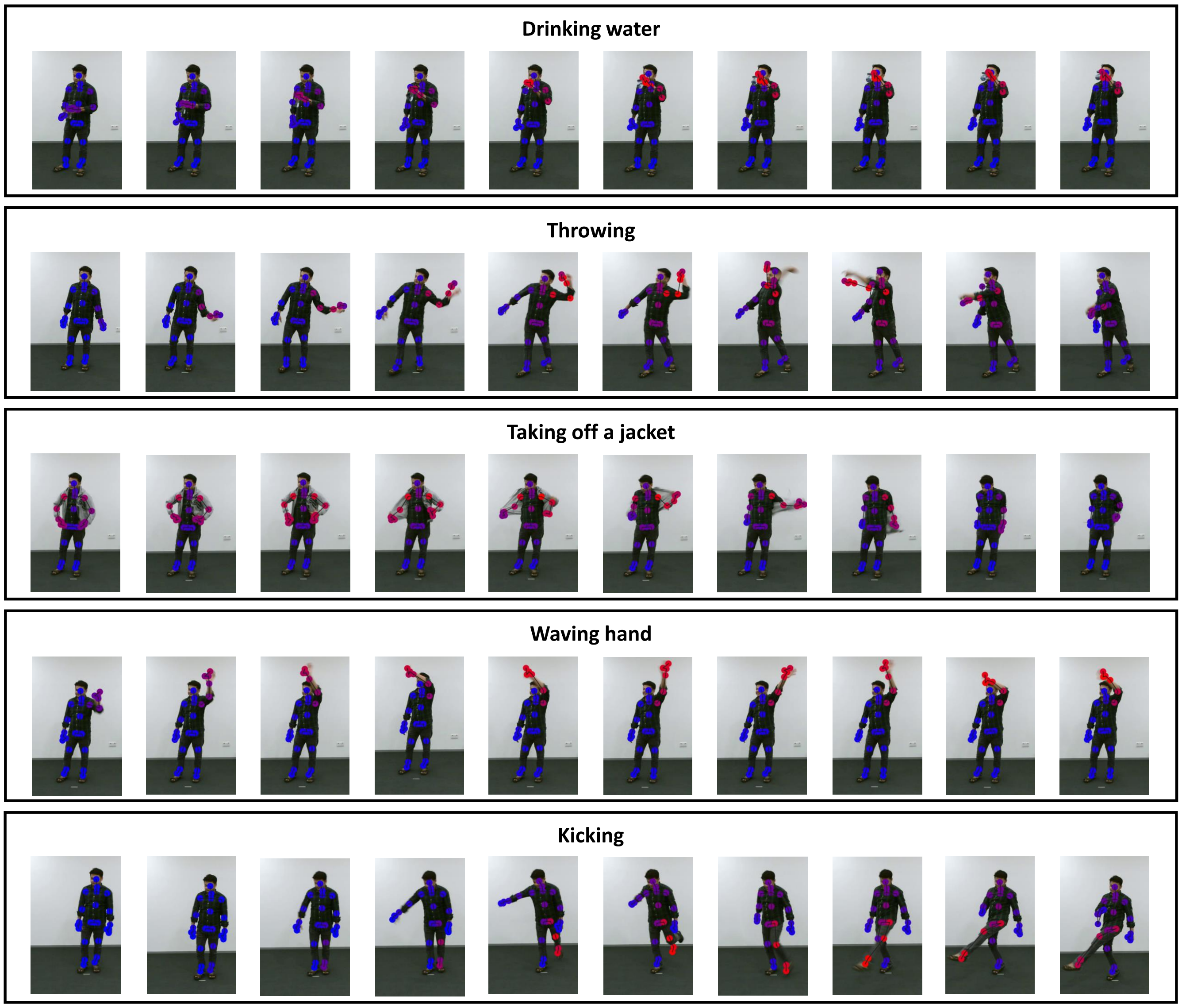

Fig. 10. Activated joints in 10 contextual frames of EfficientGCN-B4 for the sample actions, *i.e.*, *drinking water*, *throwing*, *taking off a jacket*, *waving hand*, and *kicking*. The red points denote the activated joints, while blue points represent non-activated joints. (Best viewed in color.)

for *taking off a jacket*, and left leg for *kicking*. This implies that the proposed ST-JointAtt module works well. Besides, compared with the preliminary PartAtt module proposed in [24], this new attention module results in more reasonable attention weights in temporal dimension, by which only informative moving joints in certain frames are captured (*e.g.*, all joints in the first two frames of *kicking* are not activated).

6 Conclusion

In this paper, we have constructed a family of efficient but strong baselines based on a compound scaling strategy. Different from other multi-stream models, the proposed EfficientGCN fuses three input branches at early stage, obviously eliminating the redundant parameters. In order to further reduce the model complexity, four convolutional layers are designed according to the bottleneck structure and separable convolution, which significantly saves the computational cost. Moreover, a global attention module named ST-JointAtt is proposed to find the most informative frames and joints from the whole skeleton sequence. On two large-scale datasets, NTU RGB+D 60 & 120, the proposed EfficientGCN-B4 achieves the SOTA performance, while its FLOPs and number of parameters are obviously less than other models. Thus, the new baselines will have a huge potential for developing more complicated models. In the future, we will extend the proposed baseline with the object appearance, which is likely responsible for the recognition of the extremely similar actions.

Acknowledgments

This work is sponsored by the National Key R&D Program of China under Grant 2016YFB1001002, the National Natural Science Foundation of China under Grant 61525306, Grant 61633021, and Grant 61721004, the Shandong Provincial Key Research and Development Program (Major Scientific and Technological Innovation Project) under Grant 2019JZZY010119, and CAS-AIR.

REFERENCES

[1] R. Poppe, "A survey on vision-based human action recognition," *Image Vis. Comput.*, vol. 28, no. 6, pp. 976–990, 2010.

[2] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Comput. Surv.*, vol. 43, no. 3, p. 16, 2011.

[3] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Comput. Vis. Image Understand*, vol. 115, no. 2, pp. 224–241, 2011.

[4] G. Johansson, "Visual perception of biological motion and a model for its analysis," *Percept. Psychophys.*, vol. 14, no. 2, pp. 201–211, 1973.

[5] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 7291–7299.

[6] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, 2012.

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

[8] C. Li, Q. Zhong, D. Xie, and S. Pu, "Skeleton-based action recognition with convolutional neural networks," in *IEEE Int. Conf. Multimedia Expo Workshop (ICMEW)*. IEEE, 2017, pp. 597–600.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.

[10] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *AAAI Conf. Artif. Intell.*, 2018.

[11] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *AAAI Conf. Artif. Intell.*, 2018.

[12] X. Zhang, C. Xu, X. Tian, and D. Tao, "Graph edge convolutional neural networks for skeleton-based action recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, 2019.

[13] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 12 026–12 035.

[14] Y.-F. Song, Z. Zhang, and L. Wang, "Richly activated graph convolutional network for action recognition with incomplete skeletons," in *IEEE Int. Conf. Image Process (ICIP)*. IEEE, 2019.

[15] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3d human activity analysis," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 1010–1019.

[16] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with directed graph neural networks," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 7912–7921.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.

[18] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 4510–4520.

[20] D. Zhou, Q. Hou, Y. Chen, J. Feng, and S. Yan, "Rethinking bottleneck structure for efficient mobile network design," in *Eur. Conf. Comput. Vis. (ECCV)*, 2020.

[21] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Internat. Conf. Mach. Learn. (ICML)*, 2019.

[22] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," *arXiv:2103.02907*, 2021.

[23] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with multi-stream adaptive graph convolutional networks," *IEEE Trans. Image Process.*, vol. 29, pp. 9532–9545, 2020.

[24] Y.-F. Song, Z. Zhang, C. Shan, and L. Wang, "Stronger, faster and more explainable: A graph convolutional baseline for skeleton-based action recognition," in *ACM Int. Conf. Multimedia (ACMMM)*, 2020, pp. 1625—1633.

[25] J. Liu, A. Shahroudy, M. L. Perez, G. Wang, L.-Y. Duan, and A. K. Chichung, "NTU RGB+D 120: A large-scale benchmark for 3d human activity understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[26] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, "Disentangling and unifying graph convolutions for skeleton-based action recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[27] C. Li, Q. Zhong, D. Xie, and S. Pu, "Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation," in *IJCAI Int. Joint Conf. Artif. Intell.*, 2018.

[28] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive neural networks for high performance skeleton-based human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[29] Y.-F. Song, Z. Zhang, C. Shan, and L. Wang, "Richly activated graph convolutional network for robust skeleton-based action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 5, pp. 1915–1925, 2021.

[30] K. Cheng, Y. Zhang, C. Cao, L. Shi, J. Cheng, and H. Lu, "Decoupling gcn with dropgraph module for skeleton-based action recognition," in *Eur. Conf. Comput. Vis. (ECCV)*, 2020.

[31] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7794–7803.

[32] F. Ye, S. Pu, Q. Zhong, C. Li, D. Xie, and H. Tang, "Dynamic gcn: Context-enriched topology learning for skeleton-based action recognition," in *ACM Int. Conf. Multimedia (ACMMM)*, 2020.

[33] F. Yang, S. Sakti, Y. Wu, and S. Nakamura, "Make skeleton-based action recognition model smaller, faster and better," in *ACM International Conference on Multimedia in Asia*, 2019.

[34] P. Zhang, C. Lan, W. Zeng, J. Xing, J. Xue, and N. Zheng, "Semantics-guided neural networks for efficient skeleton-based human action recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[35] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7132–7141.

[36] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "Cbam: Convolutional block attention module," in *Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[37] F. Baradel, C. Wolf, and J. Mille, "Human action recognition: Pose-based attention draws focus to hands," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 604–613.

[38] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *AAAI Conf. Artif. Intell.*, 2017.

[39] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, "An attention enhanced graph convolutional lstm network for skeleton-based action recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 1227–1236.

[40] C. Si, Y. Jing, W. Wang, L. Wang, and T. Tan, "Skeleton-based action recognition with spatial reasoning and temporal stack learning," in *Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 103–118.

[41] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *IEEE Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1314–1324.

[42] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Br. Mach. Vis. Conf. (BMVC)*, 2016.

[43] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv:1608.03983*, 2016.

[44] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv:1710.05941*, 2017.

[45] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal lstm with trust gates for 3d human action recognition," in *Eur. Conf. Comput. Vis. (ECCV)*. Springer, 2016, pp. 816–833.

[46] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 3595–3603.

[47] L. Huang, Y. Huang, W. Ouyang, and L. Wang, "Part-level graph convolutional network for skeleton-based action recognition," in *AAAI Conf. Artif. Intell.*, 2020.

[48] W. Peng, X. Hong, H. Chen, and G. Zhao, "Learning graph convolutional network for skeleton-based human action recognition by neural searching," in *AAAI Conf. Artif. Intell.*, 2020.

[49] K. Cheng, Y. Zhang, X. He, W. Chen, J. Cheng, and H. Lu, "Skeleton-based action recognition with shift graph convolutional network," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020.

[50] Z. Chen, S. Li, B. Yang, Q. Li, and H. Liu, "Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition," in *AAAI Conf. Artif. Intell.*, 2021.

[51] J. Liu, A. Shahroudy, G. Wang, L.-Y. Duan, and A. C. Kot, "Skeleton-based online action prediction using scale selection network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 6, pp. 1453–1467, 2019.

[52] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 2921–2929.

Yi-Fan Song received the M.Eng. degree from Zhengzhou University, Zhengzhou, China, in 2018. Currently, He is a Ph.D. candidate of the School of Artificial Intelligence, University of Chinese Academy and Sciences (UCAS). His research interests include computer vision, action recognition, action prediction, and neural architecture search.

Zhang Zhang received the B.S. degree in computer science and technology from Hebei University of Technology, Tianjin, China, in 2002, and the Ph.D. degree in pattern recognition and intelligent systems from the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China in 2009. Currently, he is an associate professor at the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences (CASIA). His research interests include activity recognition, video surveillance, and time series analysis. He has published 20s research papers on computer vision and pattern recognition, including IEEE TPAMI, CVPR, and ECCV etc.

Caifeng Shan received the B.Eng. degree from the University of Science and Technology of China (USTC), the M.Eng. degree from the Institute of Automation, Chinese Academy of Sciences, and the Ph.D. degree in computer vision from Queen Mary, University of London. His research interests include computer vision, pattern recognition, image and video analysis, machine learning, bio-medical imaging, and related applications. He has authored more than 100 papers and 60 patent applications. He has served as Associate Editor or Guest Editor for many scientific journals including IEEE Transactions on Circuits and Systems for Video Technology and IEEE Journal of Biomedical and Health Informatics. He is a Senior Member of IEEE.

Liang Wang received both the B.Eng. and M.Eng. degrees from Anhui University in 1997 and 2000, respectively, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences (CASIA) in 2004. From 2004 to 2010, he was a research assistant at Imperial College London, United Kingdom, and Monash University, Australia, a research fellow at the University of Melbourne, Australia, and a lecturer at the University of Bath, United Kingdom, respectively. Currently, he is a full professor of the Hundred Talents Program at the National Lab of Pattern Recognition, CASIA. His major research interests include machine learning, pattern recognition, and computer vision. He has widely published in highly ranked international journals such as IEEE Transactions on Pattern Analysis and Machine Intelligence and IEEE Transactions on Image Processing, and leading international conferences such as CVPR, ICCV, and ICDM. He is an IEEE Fellow, and an IAPR Fellow.

Appendix A Network Architecture

Here we present the details to calculate the scaled width and depth. Firstly, the initial channels and TC layers of four blocks are set to $\{48, 24, 64, 128\}$ and $\{0.5, 0.5, 1, 1\}$, respectively. Then, the rounding functions are given as:

$$C_\phi = \epsilon(C_0/16 \times \alpha^\phi) * 16 \tag{9}$$

$$L_\phi = \epsilon(L_0 \times \beta^\phi) \tag{10}$$

where C_ϕ and L_ϕ denote the numbers of scaled channels and TC layers with scaling coefficient ϕ, C_0 and L_0 denote the initial channels and TC layers, α and β are defined in Section 4.3, and $\epsilon(\cdot)$ represents the step function formulated as:

$$\epsilon(x) = \begin{cases} \lceil x \rceil, & if \quad x - \lfloor x \rfloor > 0.5 \\ \lfloor x \rfloor, & if \quad x - \lfloor x \rfloor \le 0.5 \end{cases} \tag{11}$$

where $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$ mean up and down rounding functions, respectively.

Finally, the network architectures of models with scaling coefficients $\{2, 4\}$, *i.e.*, EfficientGCN-B2 and EfficientGCN-B4, can be calculated by initial channels, initial TC layers, and rounding functions. The architectures of these three baselines are shown in Tab. 8, Tab. 9 and Tab. 10.

TABLE 8
The architecture of EfficientGCN-B0 network. Each row describes a block with the following settings and output shape, where $\times 3$ represents three input branches, $/2$ denotes a stride of 2, Q is the number of action classes.

Stage	Block	Depth	Channels	Shape
–	BN×3	–	$(6, 6) \times 3$	$(6 \times T_{in} \times V_{in}) \times 3$
–	*BasicBlock×3	1	$(6, 64) \times 3$	$(64 \times T_{in} \times V_{in}) \times 3$
1	EpSepBlock×3	$0^\dagger$	$(64, 48) \times 3$	$(48 \times T_{in} \times V_{in}) \times 3$
2	EpSepBlock×3	$0^\dagger$	$(48, 16) \times 3$	$(16 \times T_{in} \times V_{in}) \times 3$
–	Concat	–	$16 \times 3, 48$	$48 \times T_{in} \times V_{in}$
3	EpSepBlock	1	$48, 64, /2$	$64 \times T_{in}/2 \times V_{in}$
4	EpSepBlock	1	$64, 128, /2$	$128 \times T_{in}/4 \times V_{in}$
–	GAP	–	$128, 128$	128
–	FC	–	$128, Q$	Q

*: This BasicBlock is fixed and without attentions for stable training.
†: Actually, the initial depths of the two blocks in input branches are both 0.5. The depth of 0 is obtained after rounding.

TABLE 9
The architecture of EfficientGCN-B2 network. Each row describes a block with the following settings and output shape, where $\times 3$ represents three input branches, $/2$ denotes a stride of 2, Q is the number of action classes.

Stage	Block	Depth	Channels	Shape
–	BN×3	–	$(6, 6) \times 3$	$(6 \times T_{in} \times V_{in}) \times 3$
–	⋆BasicBlock×3	1	$(6, 64) \times 3$	$(64 \times T_{in} \times V_{in}) \times 3$
1	EpSepBlock×3	1	$(64, 64) \times 3$	$(64 \times T_{in} \times V_{in}) \times 3$
2	EpSepBlock×3	1	$(64, 32) \times 3$	$(32 \times T_{in} \times V_{in}) \times 3$
–	Concat	–	$32 \times 3, 96$	$96 \times T_{in} \times V_{in}$
3	EpSepBlock	2	$96, 96, /2$	$96 \times T_{in}/2 \times V_{in}$
4	EpSepBlock	2	$96, 192, /2$	$192 \times T_{in}/4 \times V_{in}$
–	GAP	–	$192, 192$	192
–	FC	–	$192, Q$	Q

⋆: This BasicBlock is fixed and without attentions for stable training.

TABLE 10
The architecture of EfficientGCN-B4 network. Each row describes a block with the following settings and output shape, where $\times 3$ represents three input branches, $/2$ denotes a stride of 2, Q is the number of action classes.

Stage	Block	Depth	Channels	Shape
–	BN×3	–	$(6, 6) \times 3$	$(6 \times T_{in} \times V_{in}) \times 3$
–	⋆BasicBlock×3	1	$(6, 64) \times 3$	$(64 \times T_{in} \times V_{in}) \times 3$
1	EpSepBlock×3	2	$(64, 96) \times 3$	$(96 \times T_{in} \times V_{in}) \times 3$
2	EpSepBlock×3	2	$(96, 48) \times 3$	$(48 \times T_{in} \times V_{in}) \times 3$
–	Concat	–	$48 \times 3, 144$	$144 \times T_{in} \times V_{in}$
3	EpSepBlock	3	$144, 128, /2$	$128 \times T_{in}/2 \times V_{in}$
4	EpSepBlock	3	$128, 272, /2$	$272 \times T_{in}/4 \times V_{in}$
–	GAP	–	$272, 272$	272
–	FC	–	$272, Q$	Q

⋆: This BasicBlock is fixed and without attentions for stable training.

Appendix B
Grid Search for Receptive Field

There are two hyper-parameters mentioned in Sections 3.2 and 4.2, *i.e.*, D for the maximum spatial graph distance and L for the temporal window size. They directly determine the receptive field of the base convolutional operation, thus have implication on the model performance. The effects of these two hyper-parameters are discussed in this section, and the experimental results are shown in Tab. 11 and Tab. 12. For the maximum graph distance D, we can find that there is an obvious decline when $D < 2$. However, $D > 3$ is not better than $D = 2$ or $D = 3$ because an oversized receptive field will make the skeleton graph over-smoothing. Similarly, the temporal window size L is also required to choose a suitable value by balancing the model accuracy and complexity. Thus, according to these two tables, we set $D = 2$ and $L = 5$ (**Bold** in tables) by choosing a high model accuracy and a low model complexity simultaneously. Note that the settings of $D = 2, L = 11$ and $D = 3, L = 9$ are slightly more accurate than $D = 2, L = 5$, but their model complexities are considerably higher than the selected setting.

TABLE 11
Comparisons with different receptive fields on X-sub benchmark in accuracy (%).

Acc.	$D = 1$	$D = 2$	$D = 3$	$D = 4$	$D = 5$
$L = 3$	87.7	88.9	89.2	88.8	88.5
$L = 5$	88.1	**89.9**	89.9	89.7	89.5
$L = 7$	88.6	89.8	89.9	89.5	89.7
$L = 9$	88.8	89.9	90.0	89.8	89.9
$L = 11$	88.7	90.0	89.9	89.7	89.7

TABLE 12
Comparisons with different receptive fields on X-sub benchmark in FLOPs ($\times 10^9$) and parameter numbers ($\times 10^6$).

F. / P.	$D = 1$	$D = 2$	$D = 3$	$D = 4$	$D = 5$
$L = 3$	2.35/0.27	2.72/0.30	3.09/0.33	3.46/0.36	3.83/0.39
$L = 5$	2.70/0.29	**3.08/0.32**	3.45/0.35	3.82/0.38	4.19/0.41
$L = 7$	3.06/0.32	3.43/0.35	3.80/0.38	4.18/0.41	4.55/0.44
$L = 9$	3.42/0.35	3.79/0.38	4.16/0.41	4.53/0.44	4.90/0.47
$L = 11$	3.78/0.37	4.15/0.40	4.52/0.43	4.89/0.46	5.26/0.49