This WACV 2023 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Efficient Skeleton-Based Action Recognition via Joint-Mapping strategies

Min-Seok Kang, Dongoh Kang, HanSaem Kim
Kakao Enterprise
Pangyo, Gyeonggi-do, South Korea
{ahstarwab, kito9021, kensaem}@gmail.com

Abstract

Graph convolutional networks (GCNs) have brought remarkable progress in skeleton-based action recognition. However, high computational cost and large model size make models difficult to be applied in real-world embedded system. Specifically, GCN that is applied in automated surveillance system pre-require models such as pedestrian detection and human pose estimation. Therefore, each model should be computationally lightweight and whole process should be operated in real-time. In this paper, we propose two different joint-mapping modules to reduce the number of joint representations, alleviating a total computational cost and model size. Our models achieve better accuracy-latency trade-off compared to previous state-of-the-arts on two datasets, namely NTU RGB+D and NTU RGB+D 120, demonstrating the suitability for practical applications. Furthermore, we measure the latency of the models by using TensorRT framework to compare the models from a practical perspective.

1. Introduction

Recently, reliable automatic surveillance systems attract much interest where emergent situations such as swooning, fighting, and kidnapping situations take place occasionally at any time. The system should notice surveillance personnel immediately and reliably at the moment when emergent situations take place. Video-based action recognition has achieved remarkable performance improvements recently due to large datasets [1, 2, 3] and large models [4, 5]. In particular, transformer models break records of CNN-based models in performance [6, 7, 8], but they are too heavy for practical applications. Furthermore, video-based action recognition methods require large amount of training data to deal with a variety of scenes and cluttered backgrounds, but a lot of efforts are required to collect refined video dataset. On the other hand, skeleton-based action recognition can shed light on fast and reliable surveillance system in real-world for many superior points compared to the video-based action recognition methods. First, it can reduce an impact of noises such as complicated backgrounds in dynamic scenes with the aid of pedestrian detector and human pose estimator, which are trained on large-scale datasets. Second, the inputs of model consume much less memory footprint than the video counterpart. Third, the skeleton data can be diversified easily with data augmentations (such as rotating the arms, adjusting human height, giving some perturbation on coordinates, etc). As a result, skeleton based action recognition can yield more reliable and generalized results in a variety of scenes.

Skeleton-based action recognition use skeleton data, which can be represented as a vector sequence of 2D or 3D coordinates. Early approaches for skeleton-based action recognition use RNNs or CNNs [9, 10, 11, 12] for processing joint representations. Afterwards, Yan *et al.* first introduce a Graph Convolutional Network (GCN) for skeleton-based action recognition called ST-GCN and it has been a strong baseline for most of the skeleton-based action recognitions up to the present. They utilize graph structure of skeleton where the joints and the connections of human body are represented as vertexes and edges, respectively. In order to learn spatio-temporal representations of human joints, they add temporal edges to a spatial graph to connect the same joints across consecutive frames. Early GCN-based approaches use heuristically initialized topology and a same untrainable topology throughout layers. However, they show suboptimal performance in action recognition task, since it is difficult to model the dependencies between distant joints and the semantics of joint features are different between layers. To mitigate the issues, Shi *et al.* [13, 14] first introduce a flexible design of graph topology by dividing the fixed graph into two trainable graphs, i.e., global graph and and individual graph. The global graph, which is shared by all the data samples, is initialized with the topology reflecting the physical connections. The individual graph, which represents a unique graph for each data sample, is calculated through self-attention mechanism. Recently, most of the works adopt the similar form of adaptive topologies to enhance the adaptability and flexibility.

GCNs basically employ the coordinates of human joints, but there are some studies introducing additional features to improve the model performance. Shi *et al.* [13] firstly introduce a bone feature that is a vector between two joints. The bone can be informative to capture the meaningful joint representations, since it represents the lengths and directions of joints. Zhang *et al.* [15] exploit some additional features such as joint type, frame index, and velocity to incorporate the semantics of spatial, temporal and motion information to the GCN. Song *et al.* [16, 17] introduce multi-branch structure that fuses three different input features such as joint, bone, and velocity to capture richer information of the complicated joint sequences. Some recent works employ ensemble learning of different modalities such as joint, bone, velocity, and velocity of bone to boost performance [18, 19, 20, 21]. However, it can significantly increase training and inference time that can make models impractical in real-world applications.

For real world applications, pedestrian detection and human pose estimation are pre-requisites of GCN and the whole pipeline should be operated in real-time with making reliable decision. The recent developments of lightweight pedestrian detection [22, 23, 24] and human pose estimation [25, 26, 27] models shed positive light on real-time and embedded surveillance system, while many GCNs still introduce non-negligible latency overhead in the pipeline. Especially in crowd scenes, as the number of people increases in a scene, additional inference steps by the GCNs are required. Therefore, the smaller model size for ensuring sufficient batch-size and the faster inference speed are necessary. Recent GCN works have made efforts to lighten the models considering the number of parameters and floating-point operations (FLOPs). However, we empirically found that the latency can be more crucial factor to action recognition models since minimum throughput should be guaranteed to recognize some action classes, especially the fast actions such as *punching*, *standing*, etc. We also found that larger free memory is available for GCNs, since they use 2D or 3D coordinates instead of RGB frames, which are the inputs of pre-requisite models such as pedestrian detector and human pose estimator. Therefore, we focus on minimizing the latency to apply skeleton-based action recognition in real-world system.

Many neural networks have hierarchical structure where the semantics of features in different layers are different. That is, the abstract features are processed in higher layers. To reflect the hierarchical traits, most of the CNNs for classification tasks typically compress representations into low spatial dimension, but GCNs generally keep the number of joints throughout layers. Accordingly, the models can suffer from computational burden and the performance can be suboptimal. Our work has arisen from the question of whether whole joints are necessary for recognizing human action. Moreover, GCNs have to capture the joints that are located apart, since they can be strongly correlated in distinguishing actions. For example, strong correlation between hand and toe, which are distant in spatial domain, should be captured to distinguish *"cross toe touch"*. GCNs generally enlarge the joint receptive field with powering the topologies throughout layers and make topology adaptive. Consequently, more layers can be required to capture effective receptive field. In this work, we solve this problem by decreasing the dimension of joint space manually and adaptively to make distant joints reachable.

Overall, our work focus on building a lightweight and low latency GCN that can run on resource-constrained devices. Although recent GCNs are becoming smaller and faster, they still do not satisfy the low-latency requirements with limited computational resources. As a step towards efficient skeleton-based action recognition, our contributions are summarized as follows:

1. Inspired by the hierarchical structure of CNNs that reduce the spatial dimension of feature maps throughout layers, we explore two types of joint-mapping modules (manual and adaptive mappings) for GCN to reduce the number of nodes in the middle of the layers. Together with multiple-branch structure, our models achieve competitive performance with lower computational cost compared to state-of-the-art models.

2. The ablations of the proposed joint-mappings and extensive experiments across two public datasets, NTU RGB+D [28] and NTU RGB+D 120 [29] demonstrate the effectiveness of our work. In particular, we measure latency of the state-of-the-art models and ours with TensorRT framework to show the efficiency of the proposed joint-mapping modules.

3. To demonstrate real-world deployment and applications, we measure the latency of our models on some mobile devices such as Jetson AGX Xavier and iPhone XR. The experimental results demonstrate that our model can provide inspiring insight of GCN-based action recognition in practical application.

2. Related Works

2.1. GCN-based Action Recognition

GCN-based methods have been proved to successfully capture motion of human skeleton data. They can process dynamically structured graphs and it is straightforward to design a graph of human joints as an adjacency matrix. ST-GCN [30] is a strong baseline for many recent works, which is the first to adopt the GCN in action recognition with untrainable topology of skeleton. They make efforts to design a graph topology heuristically to have physical structure of human body and use the same graphs over all the

layers, keeping it fixed throughout layers. However, using a same graph topology throughout layers of GCN shows limited performance, since the semantics of different layers are different. Furthermore, it is difficult to model the complicated dependencies between joints for recognizing various actions with untrainable topology. To mitigate the issues, subsequent works introduce adaptive topology that is not shared with different layers to improve the flexibility of the models. Lei *et al.* [13] introduce 2S-AGCN with an adaptive topology that can deal with variants of data samples and the distant joint modeling. Specifically, they divide the fixed graph into two trainable graphs, *i.e.* global and individual graphs. The global graph is initialized as in ST-GCN, but it is updated during training process. The individual graph is the output of self-attention module to capture the correlation between two joint features so that it reflects an unique topology for each data sample. Shi *et al.* [14] introduce a directed acyclic graph to model the dependencies between joints and bones. Liu *et al.* [18] design a model called MS-G3D, which disentangle the scales of graph convolutions to remove redundant dependencies of neighbor joints and capture the relations of distant joints effectively. They also decouple temporal convolution into multiple branches to aggregate the multiple temporal contexts and introduce the motion modality to boost the performance. Cheng *et al.* [31] divide channels of joint features into several groups and allocate different topology for each group to learn a rich representation. They also introduce new dropout skill for GCN, which drop random nodes and their neighbor nodes together. Ye *et al.* [32] introduce static and dynamic graph topologies that are the pre-defined topology of physical connections and the unique topology varying on data samples, respectively. They combine the two topologies throughout layers to add complementary information to physical information of static graph. Qin *et al.* [21] introduce angular modality to provide complementary information to model for differentiating actions. Overall, recent designs of graph topology become trainable and dynamically changed depending on data samples. Also, they are optimized individually across different layers to consider the hierarchical structure of GCN models.

There are some previous works that are similar to our proposed manual joint-mapping module in the sense that they partition full joints into several body parts, but their roles are totally different. Some of the LSTM-based methods divide joints into several parts to process each part with different LSTM encoders [33, 28]. Some GCN-based works employ joint-partitioning to complement part-wise features to joint-wise features with feature concatenation [34] or summation [35]. Song *et al.* [16] employ joint partitioning for an attention module that is located at the classification step to give attention to the critical body parts for classifying actions. Yang *et al.* [36] introduce joint-partitioning by averaging heatmaps corresponding to each part for hand gesture recognition. This work, on the other hand, tries to map the joints into smaller number of nodes manually or adaptively to reflect the hierarchical structure of deep learning models and decrease the model complexity. To our best knowledge, our work is the first to decrease the number of nodes in the middle of the layers of GCN for the sake of model efficiency and practical applications.

2.2. Lightweight GCN models

Early GCNs tend to be over-parameterized without considering of applications, but recent works increasingly focus on developing computationally efficient networks [37, 15, 17, 20]. Peng *et al.* [37] introduce Neural Architecture Search (NAS) to build a memory-efficient GCN architecture consuming low cost of computational resources. Zhang *et al.* [15] introduce a lightweight network and leverage semantics of joint type with frame index as inputs to improve model performance. Cheng *et al.* [38] introduce shift operations along the joint-axis to mingle information across joint and channel dimensions efficiently. Song *et al.* [17] aims to construct an efficient GCN baseline, so that they employ compound scaling strategy to regulate hyper-parameters, depth, and width of the models. Shi *et al.* [39] introduce policy network for GCN to adaptively select joints and channels to adjust the accuracy-efficiency trade-off. Chen *et al.* [20] introduce channel-specific topology calculated from the correlation between joint representations to enhance the flexibility of feature extraction. Though those works successfully lighten the memory footprint and reduce the computational cost, they lack the consideration of latency. In this work, we convert GCNs with TensorRT, which is a SDK for high-performance deep learning inference, to measure the inference speed and discover hardware-friendly models. With the aid of TensorRT, we demonstrate the efficiency of our proposed joint-mapping modules and our network design on Nvidia GPU.

3. Preliminaries

In this section, we define notations of our work and briefly describe about ST-GCN, which is the most popular baseline model.

3.1. Notations

A skeleton graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_J\}$ is a representation set of J joints, and $\mathcal{E}$ is the edge set. The graph can be represented by a graph topology $\mathbf{A} \in \mathbb{R}^{N \times N}$, which is initialized by an adjacency matrix where $\mathbf{A}(i, j) = 1$ if v_i and v_j are connected and 0 otherwise. Since $\mathcal{G}$ is undirected graph, $\mathbf{A}$ is symmetric and we set $\mathbf{A}(i, i) = 1$ to keep identity feature throughout layers. $\hat{\mathbf{A}}_p = \mathbf{D}_p^{-\frac{1}{2}} \mathbf{A}_p \mathbf{D}_p^{\frac{1}{2}}$ is a normalized adjacency matrix, where

$\mathbf{D}_p(i,i) = \sum_{i=1}^{S} + \epsilon$ and p is the channel index of topology matrix. ϵ is set to an arbitrary small number to avoid divide-by-zero. GCN process joint representations $\mathbf{X}_l \in \mathbb{R}^{T \times J \times C}$, where T, J, C are the number of frames, joints, and channels of joint feature at layer l, respectively.

3.2. Graph Convolutional Network (GCN)

Most of the works of skeleton-based action recognition built upon a base block of ST-GCN, which is alternating between graph convolution and temporal convolution. We explain the basic concept of module used in ST-GCN in this subsection.

3.2.1 Graph Convolution

Graph convolution (GC) can be divided into two steps : feature embedding and application of joint connections. Feature embedding transforms joint representation $\mathbf{X}$ into higher-dimensional feature $\dot{\mathbf{X}}$ as follows:

$$\dot{\mathbf{X}}_l = Conv2D[1 \times 1](\mathbf{X}_l), \tag{1}$$

where l denotes the index of layer. Then the topologies are applied to aggregate joint features from neighbors as follows:

$$\ddot{\mathbf{X}}_l = \sum_{s=1}^{S} \hat{\mathbf{A}}_p \dot{\mathbf{X}}_l, \tag{2}$$

where S is the number of topologies following a spatial configuration introduced in ST-GCN [30], and $\hat{\mathbf{A}}$ reflects a connection of joints in a feature space of $\mathbf{X}$. Recent works make topologies trainable to extract useful information for specific actions and capture the relations between distant joints effectively. Likewise, we employ adaptive topology, so that $\hat{\mathbf{A}}_{i,j}$ represents correlation between v_i and v_j.

3.2.2 Temporal Convolution

Temporal convolution (TC) is a convolution with a 1D kernel striding along time axis as follows:

$$\mathbf{X}' = Conv2D[K_t \times 1](\ddot{\mathbf{X}}), \tag{3}$$

where K_t denotes a kernel size along time axis. We adopt multi-scale temporal convolution that decouples TC branches to learn multiple temporal context. The multi-scale temporal convolution is firstly introduced in [18], and has been a common practice to employ multi-scale temporal convolutions for temporal modeling in GCNs, since it is computationally efficient but helps improve performance. Specifically, we follow the design of multi-scale temporal convolution introduced in [20], which decreases the number of branches for efficiency's sake.

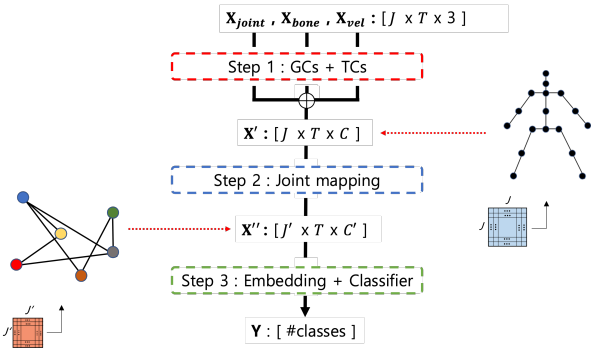


Figure 1. Overview of our skeleton-based action recognition approach. GC and TC denote graph convolution and temporal convolution, respectively.

4. Proposed Methods

We refer to a multi-branch design introduced in [17] and employ CTR-GC block as a base module. In particular, we calculate two additional modalities such as *bone* and *velocity* with the methods introduced in [15] to make three input branches. CTR-GC contains GC with channel-wise topologies for spatial modeling with increased flexibility and multi-scale TC for temporal modeling. Based on the structure and the module, we introduce mapping strategies to decrease the number of joints in the middle of the layers, inspired by the hierarchical structures of CNN models that decrease the spatial size of feature maps throughout layers. There are several benefits with the proposed mapping modules. First, the impact of redundant joints and model complexity decrease with the number of joints. Second, it can make distant joints easily reachable, so the model can effectively capture long-range spatial dependency.

Overall, our proposed pipeline can be divided into three steps: (1) GCs and TCs for joint embedding, (2) joint mapping to decrease the number of joints, (3) performing cross-correlation between the decreased joints and classifying actions. Overview of our method is illustrated in Fig. 1.

4.1. First step : Joint embedding in original joint space

First, we need to encode joint inputs in original joint space with GC-TC modules. We employ CTR-GC [20] as a base module and multi-branch structure used in [17] that process three different modalities. CTR-GC employs channel-wise topologies that are unique to data samples resulting in increased flexibility with wider channels of topologies. We employ a few CTR-GCs before the proposed joint-mapping modules to process joint representations in an original joint space.

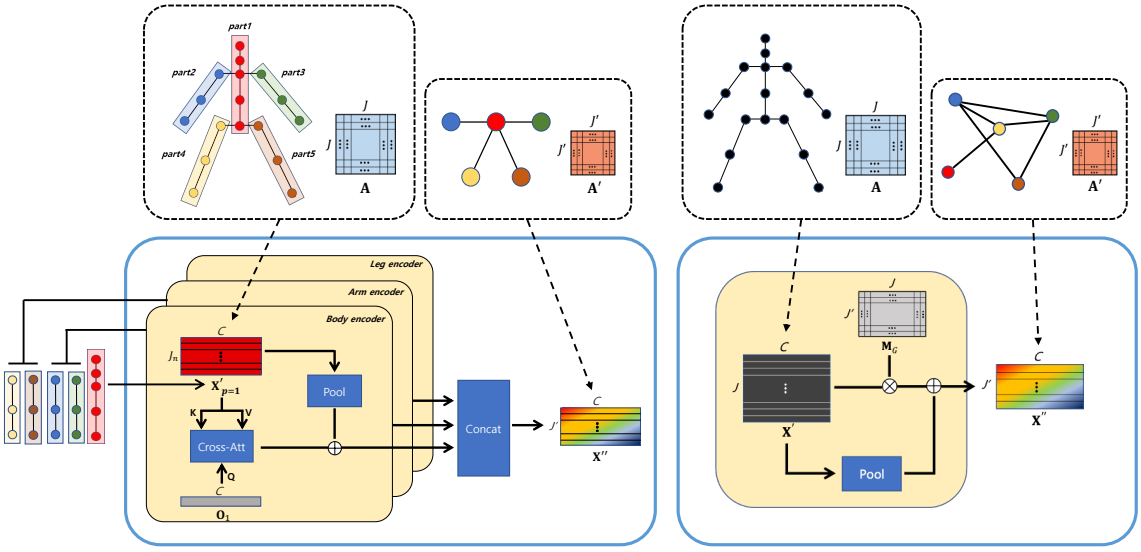


Figure 2. Overview of our two different joint-mapping strategies : (a) Manual Mapping (MM) and (b) Adaptive Mapping (AM). Q, K, V, and Cross-Att are referred to as query, key, value, and cross attention, respectively. Pool and Concat are referred to a joint-wise average pooling and joint-wise concatenation, respectively. After mapping module, the number of joints decrease to J', resulting in computational and model complexity reduction.

4.2. Second step : Joint mapping modules

We propose manual and adaptive joint-mapping strategies to decrease the number of joints from J to J'. The manual mapping module maps the J joints to 5 nodes ($J' = 5$) based on the prior knowledge of the physical connections. The adaptive mapping, on the other hand, learn to map the J joints to J' nodes (J' can be varied) without any prior knowledge. The corresponding figures of two mapping modules are illustrated in Fig. 2.

4.2.1 Manual Mapping

We divide joint representations in original joint space into five parts based on the prior knowledge of human body: *two arms (left and right)*, *two legs (left and right)*, and *main body*. A key motivation of this method is to learn discriminative part with removing redundant joint dependencies for classifying action classes.

Manual Mapping (MM) employs bottleneck fusion used in [40, 41, 8] to transfer information from original joint space to the decreased one using a simple cross-attention module. Specifically, a bottleneck fusion f_b maps input feature $\mathbf{X}'_p \in \mathbb{R}^{J_p \times C}$ to a randomly initialized vector $\mathbf{O} \in \mathbb{R}^{1 \times C}$, which is a joint representation in decreased joint manifold. Note that J_p is the number of joints of part p, and C is the number of channels. The equation can be formulated as:

$$f_b(\mathbf{X}'_p, \mathbf{O}) = Softmax\left(\frac{\mathbf{W}_q\mathbf{O}\mathbf{W}_k{\mathbf{X}'_p}^T}{\sqrt{d_k}}\right)\mathbf{W}_v\mathbf{X}'_p, \quad (4)$$

where $\mathbf{W}_q$, $\mathbf{W}_k$, and $\mathbf{W}_v$ are the query, key, and value projection matrices, respectively. d_k is the dimension of query and key. MM module consists of three bottleneck fusion blocks, *i.e.* arm, leg, and body encoders. Each encoder processes corresponding joints to yield a single joint representation. Specifically, arm and leg encoders process each side (left or right) independently with the same weights. We also add residual connection with joint-wise pooling layer to keep channel-wise activations from previous layer. To sum up, we divide $\mathbf{X}'$ into five parts $\mathbf{X}'_p (1 \le p \le 5)$ and process each part independently as follows:

$$\mathbf{X}''_p = f_b^p(\mathbf{X}'[i_p], \mathbf{O}) + Pool(\mathbf{X}'[i_p]), \quad (5)$$

where $Pool$ denotes joint-wise pooling. f_b^p is a bottleneck fusion for body part p, and i_p denotes the set of joint indices in original joint space that is belong to part p. $\mathbf{X}''_p$ are then concatenated along joint dimension for $1 \le p \le 5$ into $\mathbf{X}''$.

4.2.2 Adaptive Mapping

We try to map the joint representations to a lower dimensional space without using any prior knowledge. That is, we try to train a mapping matrix to map the joints to the decreased number of nodes that can not be represented in a human sense, but can be trained to classify action classes well.

Adaptive Mapping (AM) module uses an adaptive mapping matrix $\mathbf{M}_G \in \mathbb{R}^{J \times J'}$ that can be trained to map J joints to J' nodes. It has a residual connection with joint-wise

pooling layer. The process in AM module can be formulated as follows:

$$\mathbf{X}'' = \mathbf{M}_G \times \mathbf{X}' + Pool(\mathbf{X}'), \tag{6}$$

where $\times$ denotes matrix multiplication.

4.3. Third step : 2D Convolution through time with joint-sized kernel

We employ a single 2D convolutional layer with J'-sized kernel to model the relations of decreased nodes as follows:

$$\mathbf{X}''' = Conv2D[J' \times K_t'](\mathbf{X}''). \tag{7}$$

As above, $J' \times K_t'$ sized kernel stride along the time-axis to perform a cross-correlation between J' nodes. In this work, we set $K_t' = J'$ throughout experiments for convenience.

5. Experiments

5.1. Datasets

NTU RGB+D. (NTU60) [28] contains 56880 video clips captured by three Kinect V2 cameras concurrently. It also provides the estimated 3D skeletons with 25 joints of maximum two people from the clips. There are two benchmarks for this dataset: **1) cross-subject (X-sub)** that the training and evaluation data are divided by subjects, and **2) cross-view (X-view)** that the training and evaluation data are divided by camera views.

NTU RGB+D 120. (NTU120) [29] is an extended version of the NTU RGB+D 60, containing 114480 video clips with 120 action classes. There are also two benchmarks for this dataset: **1) cross-subject (X-sub)** that the training and evaluation data are divided by subjects, and **2) cross-setup (X-set)** that the training and evaluation data are separated depending on camera setups.

5.2. Implementation Details

Implementation of our networks is based on PyTorch. We train our models on a single Nvidia-Tesla V100 throughout experiments. We follow the same hyperparameter settings of CTR-GCN [20] for fair comparison. Concretely, we employ the SGD with momentum 0.9 and a weight decay of 0.0004. An initial learning rate, total epoch, batch size, and the number of frames are set to 0.1, 65, 64, and 64, respectively. A warmup strategy is utilized at the first 5 epochs and learning rate decays with a factor 0.1 at 35th and 55th epochs. Since the NTU60 and NTU120 datasets have 25 joints per person, J is 25. J' is set to 5 for both MM and AM modules, and the ablations on the different settings of J' are provided in supplementary. We follow the data preprocessing introduced in [15]. Furthermore, we use TensorRT 8.2.1.8 on Nvidia T4 GPU and Coremltools 5.2.0 for measuring latency.

Type	Pos.	Acc. (%)	#Params (M)	FLOPs (G)
MM	5	90.4	0.79	0.69
	6	**90.8**	**0.89**	**0.84**
	7	90.6	0.98	0.99
AM	5	89.8	0.70	0.66
	6	**90.3**	**0.74**	**0.78**
	7	90.1	0.84	0.93

Table 1. Ablation studies on different positions of MM and AM, conducted on NTU60 (X-sub). The best results are highlighted in bold, whose position will be set by default throughout experiments.

Type	Inputs			Acc.	FLOPs	Latency
	J	B	V	(%)	(G)	(ms)
MM	✓			89.4	0.64	8.5
	✓	✓		89.2	0.73	10.7
	✓	✓	✓	90.8	0.84	13.2
AM	✓			89.4	0.58	7.6
	✓	✓		89.1	0.67	9.8
	✓	✓	✓	90.3	0.78	12.5

Table 2. Ablation studies on the number of input branches. J, B, and V denote joint, bone, and velocity, respectively. Latency for processing 8 batches is measured using a Nvidia T4 GPU.

Type	NTU60 (%)		NTU120 (%)		FLOPs	Lat.
	X-sub	X-view	X-sub	X-set	(G)	(ms)
MM	90.8	95.2	86.3	87.5	0.84	13.2
AM	90.3	95.2	86.4	88.2	0.78	12.5
X-6	89.7	94.6	85.7	87.0	0.66	11.2
X-7	89.8	94.7	85.9	87.2	0.81	12.5

Table 3. Ablation studies on with (MM, AM) and without (X-6, X-7) the proposed joint-mapping modules. Lat. denotes the latency that is measured using a Nvidia T4 GPU.

5.3. Ablation Studies

5.3.1 Position of two mapping modules

We experiment with varying position of the MM and AM, while the number of channels at the joint-mapping modules remain fixed. As in Table 1, we observe that placing the joint-mapping modules at 6th layer yields the best results, so we fix the position of the modules at layer 6 throughout experiments. The detailed parameter settings are given in the supplementary.

5.3.2 Ablations on the number of inputs

Considering the fact that recent GCNs consume less GPU memory than pre-requisite models (pedestrian detector and

Model	Ens.	NTU60 (%)		NTU120 (%)		#Params	FLOPs	Latency
		X-sub	X-view	X-sub	X-set	(M)	(G)	(ms)
ST-GCN [30]		81.5	88.3	-	-	3.08	16.32	46.4
MS-G3D [18]		89.4	-	-	-	3.20	24.44	147.5
MS-G3D [18]	$\checkmark_2$	91.5	96.2	86.9	88.4	6.40	48.88	-
MST-GCN [19]		89.0	95.1	82.8	84.5	2.82	16.03	82.8
MST-GCN [19]	$\checkmark_4$	91.5	96.6	87.5	88.8	11.29	64.14	-
Eff-GCN-B0 [17]		90.2	94.9	86.6	85.0	0.29	2.73	30.4
CTR-GCN [20]		90.2*	95.2*	84.9*	86.6*	1.43	1.79	14.1
CTR-GCN [20]	$\checkmark_4$	92.4	96.8	88.9	90.6	5.72	7.16	-
MM-GCN		**90.8**	**95.2**	**86.3**	**87.5**	**0.89**	**0.84**	**13.2**
AM-GCN		**90.3**	**95.2**	**86.4**	**88.2**	**0.74**	**0.78**	**12.5**

Table 4. Comparisons against state-of-the-art methods evaluated on the NTU RGB+D 60 and 120 datasets, abbreviated as NTU60 and NTU120, respectively. Latency for processing 8 batches is measured using a Nvidia T4 GPU. Ens denotes whether the model employ ensemble learning, and the suffix of the check marks denotes the number of input modalities. The results with star marks are obtained from the author's released code. The results of our proposed models are highlighted in bold.

human pose estimator) regarding their size of the model and input, we mainly focus on discovering good accuracy-latency trade-off models. To find a better model, we perform ablations of the number of input branches. As in Table 2, the models with three inputs provide the best accuracy with some latency overhead. We use the models with three branches by default, since they still provide lower latency with competitive performance compared to the recent GCNs. The comparisons against recent GCNs will be given in 5.3.4.

5.3.3 Effectiveness of the joint-mapping modules

We conduct ablation study on the joint-mapping modules to demonstrate the effectiveness. Specifically, the models without the joint-mapping modules are not plausible to apply joint-sized kernels (since the size of the kernels are too large to be well optimized), so we replace them with a single (X-6) or two CTR-GC blocks (X-7) just before the final classier to make comparisons fair in computational cost. As in the experimental results in Table 3, the proposed joint-mapping modules bring performance improvement with about 1∼2 ms latency overhead compared to the model without any mapping modules. Further details of the parameter settings are given in the supplementary.

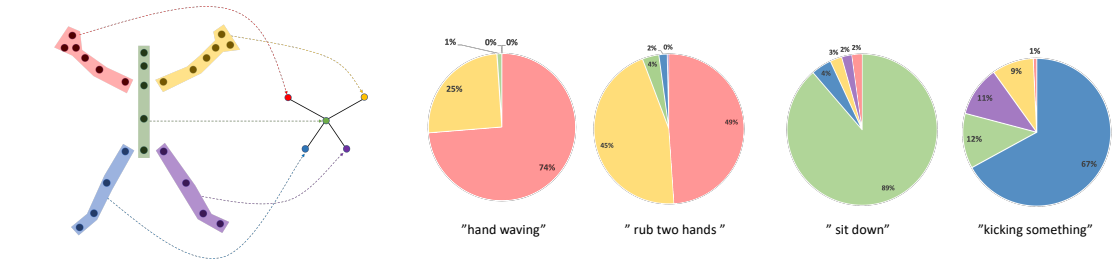
5.3.4 Comparison with SOTA methods

As shown in Table 4, we compare the performance of our two models with others. Some models employ ensemble learning of some different input modalities such as joint, bone, motion, and motion of bone, since they can benefit the model performance in terms of accuracy. However, the ensemble learning can significantly increases training and inference time that can make models impractical in applications. Our models achieve competitive or superior performance on four benchmarks with lower latency compared to the SOTA methods that are not using ensemble learning. We obtain the results in Table 4 from the authors' source codes (#Params, FLOPs, Latency, and Accuracy with $\star$) or papers.

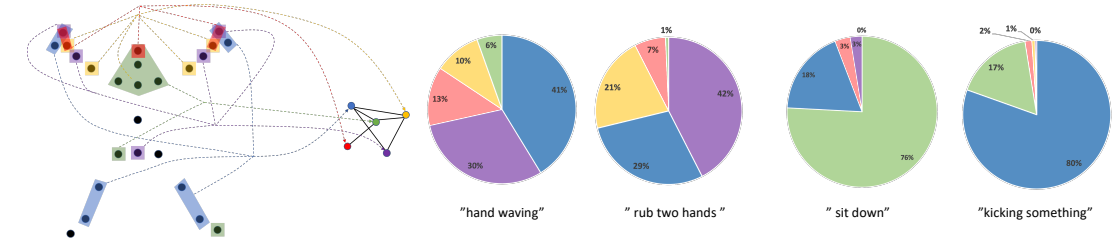
6. Discussion

6.1. Discussion on Model Interpretability

We illustrate the joint-mapping results of the two modules and pie charts indicating which nodes are more activated depending on action classes in Figure 3. The pie charts are derived from Gradient-weighted Class Activation Mapping (Grad-CAM) [42]. Specifically, we calculate Grad-CAM of every data sample on *time-node* space and find the index of the maximum value (*e.g.* second *node* at *frame* 13). Then, we count the maximum *node* (*e.g.* second *node*) for every data sample on each class across NTU RGB+D dataset. The mapping result of MM illustrated in Figure 3-(a) shows straightforward results, since we divide five parts manually with our prior knowledge. For example, a pie chart of *"hand waving"* explain that the trained model determine *"hand waving"* by concentrating on right or left arm. Also, we can guess that there are more right-handed subjects than left-handed in NTU RGB+D. For another example, the model determine *"rub two hands"* with similar contribution of left and right hands. On the other hand, the mapping result of the adaptive mapping illustrated in Figure 3-(b) is the approximations for clarity. In detail, the elements of $\mathbf{M}_G \in \mathbb{R}^{J \times J'}$ are highlighted if they are greater than the mean value of $\mathbf{M}_G$. Since every joint affects every decreased node in AM module, the mapping result and the



(a) Manual Mapping (MM)



(b) Adaptive Mapping (AM)

Figure 3. Illustration of mapping results and the pie charts derived from Grad-CAM results across NTU RGB+D dataset. The figures on the left demonstrate what the mapping results of the two joint-mappings look like. Besides, the pie charts on the right indicate which part is more activated for recognizing different action categories. (Best viewed in color and zoomed images).

pie charts are more complicated. For instance, pie charts in Figure 3-(b) explains that the model determine *"sit down"* by concentrating on the movements of body and legs, but other nodes corresponding to legs, hands, and head are activated the most for some data samples. Further discussion on the visualization results are available in the supplementary.

6.2. Discussion on Applications

We measure the latency of our models and compare the results with CTR-GCN [20]. Careful adjustment of batch size is important for both low latency and high throughput, by taking into account environmental conditions of the camera (*e.g.* crowded or uncrowded) and the hardware for model inference. If the number of detected people present more than the batch size, more inference steps are needed for the current frame. In this experiment, we calculate the latency for processing 64 joint sequences (frames) with different batch size depending on hardware capacity. The batch size is set to 4 and 16 for iPhone XR and Jetson AGX Xavier, respectively. We convert the trained models based on Pytorch into TensorRT model for Jetson AGX Xavier and into CoreML for iPhone XR. As reported in Table 4 and Table 5, our models surpass the CTR-GCN in terms of both accuracy and latency.

Model	Latency (ms)	
	XR (4 batches)	AGX (16 batches)
CTR-GCN	170.2	54.9
MM-GCN	94.1	52.1
AM-GCN	82.6	50.0

Table 5. Latency for processing 64 sequences of 16 people for iPhone XR) with 25 joints on different devices.

7. Conclusion

In this work, we have proposed two different joint-mapping modules to decrease the number of joints for efficiency's sake. Extensive experiments have shown that the proposed modules with multi-branch structure show notable results with regards to accuracy-latency trade-off. Besides, we have demonstrated how our joint-mapping modules work by visualizing the mapping results and analyzing the Grad-CAM results across NTU-RGB+D. We hope that our work can be extended to future research on skeleton-based action recognition and be applied in real-world applications.

Acknowledgement

This research is supported by a grant from R&D program (Development of core technology for evacuation control in the accident case and passenger safety, PK2202A2) of the Korea Railroad Research Institute.

References

[1] Ang Li, Meghana Thotakuri, David A. Ross, João Carreira, Alexander Vostrikov, and Andrew Zisserman. The AVA-Kinetics localized human actions video dataset. *CoRR*, abs/2005.00214, 2020.

[2] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The Kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

[3] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "Something Something" video database for learning and evaluating visual common sense. In *IEEE International Conference on Computer Vision, ICCV*, pages 5843–5851, 2017.

[4] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the Kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 4724–4733, 2017.

[5] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision, ICCV*, pages 4489–4497, 2015.

[6] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. ViViT: A video vision transformer. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 6816–6826, 2021.

[7] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. *CoRR*, abs/2106.13230, 2021.

[8] Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. *CoRR*, abs/2201.04288, 2022.

[9] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1110–1118, 2015.

[10] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-Temporal LSTM with trust gates for 3D human action recognition. In *Proceedings of the European conference on computer vision ECCV*, pages 816–833, 2016.

[11] Tae Soo Kim and Austin Reiter. Interpretable 3D human action analysis with temporal convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, pages 1623–1631, 2017.

[12] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Ahmed Sohel, and Farid Boussaïd. A new representation of skeleton sequences for 3d action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 4570–4579, 2017.

[13] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 12026–12035, 2019.

[14] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with directed graph neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 7912–7921, 2019.

[15] Pengfei Zhang, Cuiling Lan, Wenjun Zeng, Junliang Xing, Jianru Xue, and Nanning Zheng. Semantics-guided neural networks for efficient skeleton-based human action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1109–1118, 2020.

[16] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Stronger, faster and more explainable: A graph convolutional baseline for skeleton-based action recognition. In *MM '20: The 28th ACM International Conference on Multimedia*, pages 1625–1633, 2020.

[17] Yi-Fan Song, Zhang Zhang, Caifeng Shan, and Liang Wang. Constructing stronger and faster baselines for skeleton-based action recognition. *CoRR*, abs/2106.15125, 2021.

[18] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. Disentangling and unifying graph convolutions for skeleton-based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 140–149, 2020.

[19] Zhan Chen, Sicheng Li, Bing Yang, Qinghan Li, and Hong Liu. Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 1113–1122, 2021.

[20] Yuxin Chen, Ziqi Zhang, Chunfeng Yuan, Bing Li, Ying Deng, and Weiming Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 13339–13348, 2021.

[21] Zhenyue Qin, Yang Liu, Pan Ji, Dongwoo Kim, Lei Wang, Bob McKay, Saeed Anwar, and Tom Gedeon. Fusing higher-order features in graph neural networks for skeleton-based action recognition. *arXiv preprint arXiv:2105.01563*, 2021.

[22] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 9626–9635, 2019.

[23] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021.

[24] Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 13039–13048, 2021.

[25] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A

lightweight high-resolution network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 10440–10450, 2021.

[26] Yihan Wang, Muyang Li, Han Cai, Wei-Ming Chen, and Song Han. Lite pose: Efficient architecture design for 2d human pose estimation. *CoRR*, abs/2205.01271, 2022.

[27] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision ECCV*, pages 472–487, 2018.

[28] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. NTU RGB+D: A large scale dataset for 3d human activity analysis. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1010–1019, 2016.

[29] Jun Liu, Amir Shahroudy, Mauricio Perez, Gang Wang, Ling-Yu Duan, and Alex C. Kot. NTU RGB+D 120: A large-scale benchmark for 3d human activity understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 2684–2701, 2020.

[30] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 7444–7452, 2018.

[31] Ke Cheng, Yifan Zhang, Congqi Cao, Lei Shi, Jian Cheng, and Hanqing Lu. Decoupling GCN with dropgraph module for skeleton-based action recognition. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proceedings of the European conference on computer vision ECCV*, pages 536–553, 2020.

[32] Fanfan Ye, Shiliang Pu, Qiaoyong Zhong, Chao Li, Di Xie, and Huiming Tang. Dynamic GCN: context-enriched topology learning for skeleton-based action recognition. In *MM '20: The 28th ACM International Conference on Multimedia*, pages 55–63, 2020.

[33] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 1110–1118, 2015.

[34] Maosen Li, Siheng Chen, Xu Chen, Ya Zhang, Yanfeng Wang, and Qi Tian. Symbiotic graph neural networks for 3d skeleton-based human action recognition and motion prediction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44:3316–3333, 2022.

[35] Linjiang Huang, Yan Huang, Wanli Ouyang, and Liang Wang. Part-level graph convolutional network for skeleton-based action recognition. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 11045–11052, 2020.

[36] Siyuan Yang, Jun Liu, Shijian Lu, Meng Hwa Er, and Alex C. Kot. Collaborative learning of gesture recognition and 3d hand pose estimation with multi-order feature analysis. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Proceedings of the European conference on computer vision ECCV*, pages 769–786, 2020.

[37] Wei Peng, Xiaopeng Hong, Haoyu Chen, and Guoying Zhao. Learning graph convolutional network for skeleton-based human action recognition by neural searching. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 2669–2676, 2020.

[38] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. Skeleton-based action recognition with shift graph convolutional network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 180–189, 2020.

[39] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Adasgn: Adapting joint number and model size for efficient skeleton-based action recognition. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 13393–13402, 2021.

[40] Arsha Nagrani, Shan Yang, Anurag Arnab, Aren Jansen, Cordelia Schmid, and Chen Sun. Attention bottlenecks for multimodal fusion. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems*, pages 14200–14213, 2021.

[41] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. *CoRR*, abs/2107.14795, 2021.

[42] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, ICCV*, pages 618–626, 2017.