Temporal Shift and Attention Modules for Graphical Skeleton Action Recognition

Haidong Zhu, Zhaoheng Zheng and Ram Nevatia
Department of Computer Science
University of Southern California
Los Angeles, California 90089
{haidongz,zhaoheng.zheng,nevatia}@usc.edu

***Abstract*—Skeletons, consisting of joint positions and connections between them, are an important representation for modeling human bodies in image frames. Compared with understanding RGB videos, recognizing actions from the skeletons removes the biases of background and body shapes. Researchers use spatial-temporal graphs to model the skeleton sequences. These methods weigh all frames in the sequence equally even though many of the frames may not be useful for action and prediction and dilute the influence of important frames. Also, the temporal graph focuses on understanding only the low-level feature of the joints for the motion of the skeleton. In this paper, we introduce two modules, temporal shift module and temporal attention module that can be added to graph convolution networks for skeleton action recognition. Temporal attention module focuses on keyframes for making predictions, and temporal shift module helps to exchange the high-level features between different frames along the temporal dimension besides the local patterns. We evaluate the two modules with two existing skeleton action recognition networks, ST-GCN and MS-G3D, on three public datasets and show better results than the original methods.**

I. Introduction

To infer human actions in videos, it is common to take sequences of RGB frames as input for modeling human dynamics. While this approach has been highly successful, the action recognition methods may learn to rely on strong correlations with the environment, and focus on the surroundings instead of the action itself; for example, if the surrounding is a basketball court, the model may give the prediction of 'playing basketball', no matter what the human in the image is actually doing. To study action recognition using human body motion only, researchers [1], [2], [3], [4] have started to use human body *skeletons* for modeling human actions. Human skeletons record several key points (primarily joints) and their connection in the human shape for every person in the individual frame and capture the motion and movements of different parts, providing the essential signal for human actions. Unlike the use of RGB images as frames, skeletons remove the biases of background and human body shape. An example is shown in Figure 1, where Figure 1 (a) shows the presence of many background objects whereas Figure 1 (b) retains only the human body motion information.

To understand the actions in the sequences of skeletons of the human action sequence, early methods [5], [6] generated the prediction based on every individual frame and output the average prediction among all frames. They do not consider

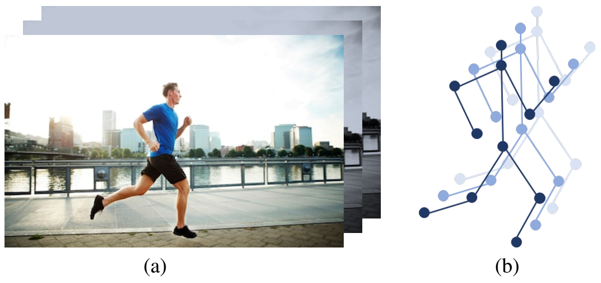


Fig. 1. Examples of the (a) RGB frame sequence and (b) Skeleton sequence of a running man.

the movement of the corresponding joints and discard the connection between the frame sequences. Recent approaches [1], [4], [7], [2] build temporal-spatial graph convolution structure networks for action understanding. Such networks extract the local features between different parts in the same frame as well as the movements between different frames of the specific joints or keypoints. Due to the computation complexity, these methods still cannot take the temporal and spatial features together into consideration. Moreover, these methods still use the average of all the frames for final predictions. This might make the keyframes covered by the frames that are not interesting.

In this paper, we propose two portable modules for skeleton action recognition, a graph temporal shift module and a temporal attention module. The graph temporal shift module, inspired by TSN [8], exchanges the information between different frames without extra computation resources for graph convolution network. This helps the graph convolution network exchange high-level information among the temporal dimension besides the local information. After generating the prediction of every individual frame, we calculate the relative weights for different frames in the sequences with the graph temporal attention module. This helps the network to focus on essential frames that make the final prediction of the action. We also spread the weight among the temporal dimension for the whole video instead of only focusing on a few frames to avoid overfitting the training examples. The two modules are abbreviated as G-TSM and G-TAM, respectively, while G-TSA when combined together. We apply our methods on two

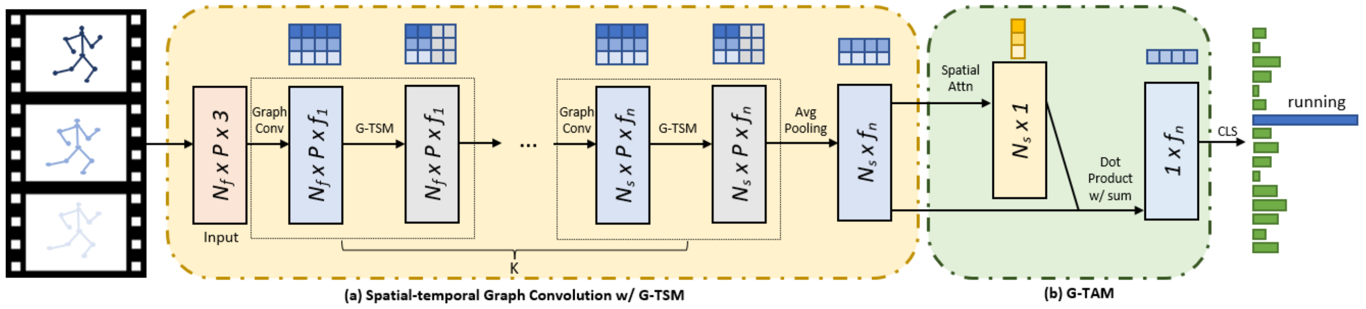


Fig. 2. Overall architecture of our proposed method. We use a spatial-temporal graph feature extractor with G-TSM to get the frame features, which are reweighted and summed up with the weights from G-TAM, followed by a classifier to generate final action prediction. N_f and N_s are the numbers of frames and video segments, and f_i is the dimensionality of the output graph convolution feature for layer n. K is the number of graph convolution with G-TSM operations, and P is the number of points. Graph conv, avg pooling and spatial attn are the abbreviations for graph convolution layer, average pooling, and spatial attention.

existing state-of-the-art methods, MS-G3D [2] and ST-GCN [1], and show better results on three public datasets, NTU-RGB+D 60, NTU-RGB+D 120, and Kinetic 400 skeleton dataset, where skeletons are provided in the datasets.

In summary, our contributions are as follows: 1) we introduce a temporal shift module for temporal information exchange for graph convolution network without extra computation, 2) we use the self-attention map for focusing on keyframes for making action prediction, and 3) we apply our two modules on two skeleton action recognition backbone, ST-GCN [1] and MS-G3D [2], and show better results on three public datasets compared with baseline methods.

II. Related Work

In this section, we introduce some recent developments in video action recognition and methods implemented in the task of skeleton action recognition.

Video Action Recognition: To understand the action happening in a video, a model should understand the movement between different frames. To understand the motions and movements between different frames, researchers build 3-D CNNs [9], [10] for the convolution kernels to take the motion information along the temporal axis. Tran *et al.* [9] introduce the C3D for convolution along with the third dimension, time, for understanding the video action. Carreira *et al.* [10] change all the 2-D convolution filters of the Inception [11] into 3-D convolutions. More recent works [8], [12] use 2-D CNNs for lighter computation for understanding the video. TSN [12] uses the shuffled temporal video segments to generate reliable predictions for all segments before the final combination. TSM [8] exchanges the features with nearby frames and fuses them for temporal understanding. Some methods [13], [14], [15], [16], [17] mix the 2-D with 3-D input, which do temporal fusion in 3-D space with 2-D convolutions as main operations for smaller networks and faster computation.

Skeleton Action Recognition: Unlike actions in an RGB video, the skeleton action recognition focuses on the body movement instead of the frame sequences. In this way, the model can focus more on the action than the environment and human body shape. Early research takes the 1-D feature vector sequence directly into the temporal network like LSTM for prediction [18], [1]. More recent research [19], [20], [21], [3], [7], [2] consider the joints map as a graph and applied the graph convolution methods for action prediction. ST-GCN [22] introduces the temporal and spatial graph convolutions for dealing with skeleton action recognition. MS-G3D [2] propose a disentangled multi-scale aggregation scheme that removes redundant dependencies between node features from different neighborhoods. Feedback graph convolution network [7] takes the LSTM for temporal fusion after the graph convolution network instead of average pooling on all the predictions.

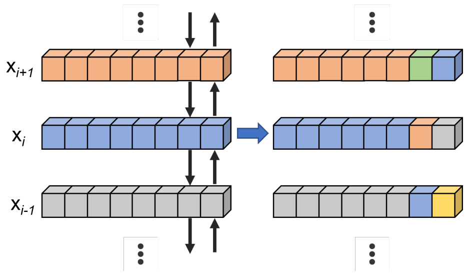
All these methods above simply take the local or global features into consideration, and do not focusing of the key frames that make the final predictions. This makes the model focusing on the whole video, where lots of frames do not contribute to the final predictions. Our method fuse the temporal feature of the local as well as global patterns for the skeletons between different frames and can extra the key frames for making the predictions automatically for more accurate predictions.

III. Methods

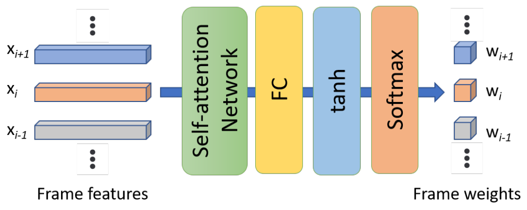
To fuse the feature for the temporal and spatial spaces together, we introduce two modules: graph temporal shift module (G-TSM) and temporal attention module (G-TAM). The overall proposed architecture is shown in Figure 2. We introduce these two components correspondingly in this section, followed by the discussions of our method.

A. *Graph Temporal Shift Module (G-TSM)*

The graph temporal shift module, abbreviated as G-TSM, shifts the skeleton features along the temporal dimension beyond the motion and movement between corresponding joints. Given a skeleton sequence $\{s_i\}_{i=1,..,N_f}$ in the form of a matrix with shape $N_f \times P \times 3$, where i represents the frame number in the sequence and N_f is the number of the frames, we extract the features $X = \{x_i\}$ in the shape of $N_s \times P \times 3$ for each frame or segment in the sequence, and exchange k percentage features between feature x_i and features x_{i+1} and x_{i-1} after every graph convolution operation. The details of



(a) G-TSM



(b) G-TAM

Fig. 3. Detailed structures for (a) G-TSM and (b) G-TAM. x_i represents the feature matrix at frame i and w_i represents the corresponding weight for this frame or video segment.

this module are shown in Figure 3 (a). After this operation, the new feature x'_i for the frame s_i consists of features from three parts, x_i from itself, along with x_{i-1} and x_{i+1} for frames s_{i-1} and s_{i+1}. For the first frame of the video, we copy the current feature in place of the feature for x_{i-1}, and for the last frame, we also make a copy of the feature in place of the feature for x_{i+1}. After such operation, the next convolution kernel will use k percentage features from the next frame, k percentage feature from the previous frame, and $(100-2k)$ percentage feature from the current frame. The graph convolution network will focus on the motion between the corresponding point along the temporal sequence and exchange the global information without extra computation expenses.

B. Graph Temporal Attention Module (G-TAM)

The graph temporal attention module (G-TAM) uses frame-wise self-attention to localize the essential frames in the sequence. We construct this module as Figure 3 (b). After extracting the feature $X=\{x_i\}$ for each segment or frame s_i, we generate the weight for each temporal segments following

$$w_i = Softmax(tanh(\sigma(Attn(x_i, X)))) \quad (1)$$

where x_i is the i-th input segments of the output X. $Attn(x_i, X)$ is the self-attention score between x_i and every segments X of the input video, which is generated with the self-attention network in Figure 3 (b). In our experiment, we use a Transformer [23] to generate the self-attention for every temporal segment in the same video. $\sigma(\cdot)$ is a projection layer to squeeze the dimension of the self-attention feature to 1. To avoid overfitting the data to a few bunches of frames, we follow [24] to set a *tanh* layer for all the generated weights, followed by a Softmax layer to normalize the scores along the temporal dimension of every segment in the video. G-TAM then outputs the generated weights w_i for every segment. With the generated weight w_i for the feature of every segment x_i, we use the following equation

$$P = \phi(\frac{1}{n}\sum_i w_i \cdot x_i) \quad (2)$$

as the probability distribution for action categories available in the dataset. $\phi(\cdot)$ is a fully connected layer as an action classifier for making the prediction based on the reweighted video feature $w_i \cdot x_i$.

C. Discussion

Our two modules can be applied to existing methods that use a graph convolution network as the backbone. In this part, we compare our modules with some other existing temporal feature fusion methods and discuss the benefits and functions of our two modules, as well as using the *tanh* function for generating the weight score w_i in G-TAM.

1) Difference between G-TSM and other temporal graph convolutions: Temporal operations for skeleton action recognition were introduced by ST-GCN [1]. It tries to find the temporal information flow between different frames. When doing the temporal convolution, it calculates fusion between corresponding nodes and does not consider any global spatial information. More recent networks, like MS-G3D [2], consider the local features from some nearby points as well as the temporal information, and also use the features from local patterns from a few points near to each other for both temporal and spatial information interaction. This indicates that they are only capable of 1-D operations on one of the features or local 2-D operations by using local patterns of the spatial space and temporal information.

The action of a person in the form of a skeleton should, ideally, be considered as interactions between the joints of the whole body and take the frame-level 2-D fused temporal and spatial information into consideration for temporal feature understanding. Our G-TSM is able to generate the frame-level 2-D understanding without extra computation expense for the graph convolution network. In this way, the graph convolution operation can not only do the 1-D and local 2-D operations on temporal and spatial information on the local area, but it can also do a good frame-level 2-D fused understanding for the whole temporal-spatial skeletons. Moreover, since we introduce temporal shifts at different feature levels after every graph convolution layer, we can exchange the feature from the local to global levels in different convolution operations.

2) Difference between G-TAM and other action recognition methods: Existing methods [12], [8], [2] commonly use the average features or action probabilities for the final prediction among all the frames. This makes the model focusing on frames that do not contribute to the final prediction. G-TAM focuses on the key frames that give the most crucial response for making the prediction of the action, which helps to give the highest confidence for accurate predictions. In addition,

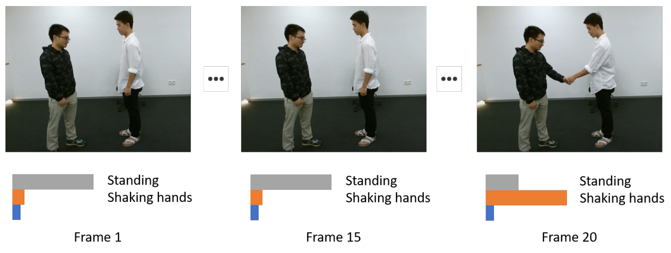


Fig. 4. An example of the framewise action prediction result. This sequence has 20 frames and is labeled as 'shaking hands'. Up to the 15^{th} frame, the highest confident action prediction of from the model is 'standing', while the highest confident prediction between the 15^{th} and 20^{th} frame of the video is 'shaking hands'. After averaging along the prediction for all frames, the category with highest response is 'standing' instead of 'shaking hand'.

if a video has more than one of action label, it can create ambiguities for the network.

Consider we have two action categories as the potential predictions: standing and shaking hands, and we also have a video labeled as 'shaking hands' as Figure 4. Men in the video stand still at the beginning, and shake hands with each other at the end. From human perception, the focus of the whole video should be the last few frames, and the prediction would be shaking hands instead of standing. However, if we do the average of the prediction on every individual frame, the model might give an extremely high score for "standing" at the beginning of the video. Even if the model can provide a perfect prediction for every individual frame, after averaging, the standing score might still be higher than shaking hands since more frames show this action.

3) Usage of tanh function: Instead of directly using the score from the self-attention network to reweight every temporal segment, we use a *tanh* function to rescale the score before applying the Softmax function. This ensures that the sum of the weight is one for normalization. When we are training the network, the Softmax function tends to focus on very few frames of the sequence, and it prefers to give a considerable weight for one or two examples while depressing the score for others. This does not help reweight the temporal segments for the skeleton sequence since an action may take several frames or temporal segments for the model to understand. To avoid this, we create a function to distribute the score to other frames that were first incorrectly treated as 'non-relevant'. The *tanh* function controls the gap between the highest and lowest weights and makes it possible for the model to be pulled back when it was optimized to an incorrect direction. Before the *tanh* function, the range for the possible weight from the self-attention is $-\infty$ to ∞, while after the *tanh*, the range will be reduced to -1 to 1. In this way, after the Softmax is applied, the score difference between the highest and lowest weights can only be $e^2 = 7.39$ times.

The introduction of non-linearity benefits the equation in two ways. If the network focuses on the wrong temporal segments, the relatively small difference introduced by the *tanh* function will help it pull the score back faster when the points are far from the zero points. Also, the network can find the global similarity between different temporal segments and increase the score for some others that are relevant to the action instead of only focusing on one of them and making its weight as large as possible. In addition, the non-linearity of the *tanh* function changes more frequently near the 0 points, while the change is more flattered when it goes further from the zero point. This can let the points change less frequently when the weight is further from 0 points, making it possible to be pulled back along with the whole training process.

IV. Experiments and Results

In this section, we will first describe our experiment settings and then show some quantitative results and visualizations.

A. Experiment setups

We discuss how we build our model with some implementation details for the experiment setups, followed by describing the datasets and metrics that we used in the experiments.

1) Baseline Methods : Since our model can be applied to the backbone of spatial-temporal graph convolution networks, we choose to use ST-GCN [1] and MS-G3D [2] as two of the backbone networks and baseline methods. Both of these two networks are state-of-the-art action recognition networks for skeleton action recognition. ST-GCN [1] is faster while MS-G3D [2] yields a higher accuracy. We also make some comparisons for other state-of-the-art methods [25], [7], [19], [20] for comparisons.

2) Implementation details: We build our two modules on ST-GCN and MS-G3D as ST-G-TSA and MS-G-TSA. The information module is implemented after every spatial convolution operation and the ratio of feature exchange percentage k is set to 25 according to our ablation experiments, which can be found in the supplementary document. The G-TAM is applied before the final prediction. The number of the head of the Transformer model is set to be 4, and the number of layers for the encoder and the decoder is set to 3. We follow [1], [2] to set the learning rate as 0.1 for ST-GCN for 80 epochs and 0.05 for MS-G3D for 70 epochs. We pad all the videos to 300 frames with zero vectors after the skeleton sequences shorter than 300 frames.

3) Datasets: We use three datasets for evaluation, NTU RGB+D 60 [26], NTU RGB+D 120 [27], and Kinetic 400 skeleton [28] datasets.

(i) **NTU RGB+D 60** [26] is a dataset with videos that consist at most two different entities. It consists of 56,578 videos over 60 categories captured from 40 subjects from 3 different camera view angles. Each full skeleton includes the 3-D location and the number of joints for each skeleton is 25. It consists of two setting: cross-subject (*xsub*) and cross-view (*xview*). In the *xsub* setting, the action in the test set is not on the same subject as the training set, while for the *xview* setting, the view position for the subjects in the training and test is different.

(ii) **NTU RGB+D 120 dataset** [27] is the extension version of the NTU RGB+D 60 dataset with videos that consist at most two different entities. It consists of 114,480 videos

TABLE I
RESULTS FOR ACTION PREDICTION ON NTU-RGB+D 60 AND 120 DATASETS WITH JOINTS ONLY.

Method	NTU-RGB+D 60		NTU-RGB+D 120	
	xsub	*xview*	*xsub*	*xview*
STA-LSTM [29]	73.4	81.2	-	-
VA-LSTM [30]	79.4	87.6	-	-
FGCN [7]	87.1	93.6	-	-
AS-GCN [19]	86.8	94.2	-	-
ST-GR [20]	86.9	92.3	-	-
AGC-LSTM [31]	89.2	95.0	-	-
SAN [32]	87.2	92.7	-	-
ST-TR [33]	88.7	95.6	81.9	84.1
1s-AGCN [3]	87.8	95.1	80.9	83.2
ST-TR [33]	88.7	95.6	81.9	84.1
ST-GCN [1]	81.8	88.4	76.0	79.4
ST-G-TSA	83.3	90.2	77.6	80.4
MS-G3D [2]	89.4	95.0	-	-
MS-G3D*	89.0	94.6	84.3	86.2
MS-G-TSA	89.9	95.2	85.0	86.8

TABLE II
RESULTS FOR ACTION PREDICTION ON NTU-RGB+D 60 AND 120 DATASETS WITH BOTH BONES AND JOINTS.

Method	NTU-RGB+D 60		NTU-RGB+D 120	
	xsub	*xview*	*xsub*	*xview*
2s-AGCN [3]	88.5	95.1	82.9	84.9
DGNN [4]	89.9	96.1	-	-
2s Shift-GCN [34]	89.7	96.0	85.3	86.6
4s Shift-GCN [34]	90.7	96.5	85.9	87.6
ST-TR [33]	88.7	95.6	81.9	84.1
MS-G3D [2]	91.5	96.2	86.9	88.4
MS-G3D*	90.6	95.8	86.0	88.1
MS-G-TSA	91.7	96.4	87.6	88.7

over 120 categories captured from 106 subjects from 155 different camera view angles. The number of joints for each skeleton is 25 and it also consists of two setting: cross-subject (*xsub*) and cross-view (*xview*).

(iii) **Kinetic 400 skeleton dataset** [28] is a dataset with the skeleton sequences extracted from Kinetic 400 action recognition video dataset. It contains 240,436 training and 19,796 testing sequences for 400 categories. Each video consists of 2-D locations for each individual body skeleton. The number of nodes for each skeleton is 18.

4) Metrics: In our experiment, we evaluate top-1 action prediction accuracy for NTU-RGB+D 60 and 120 datasets, as well as the top-1 and top-5 accuracies for Kinetics 400 skeletons dataset. The final accuracy for action prediction is calculated as the ratio of the correct predictions in the predictions.

B. Quantitative results

In this subsection, we show the results compared with existing state-of-the-art methods on NTU-RGB+D 60 and 120 datasets in Table I and Table II, as well as the performance on Kinetic 400 skeleton dataset in Table III and Table IV respectively. We will also discuss the choice of different modules and hyperparameters as well as some ablations for our experiments in the supplemental material.

TABLE III
RESULTS FOR ACTION PREDICTION ON KINETICS 400 SKELETON DATASET WITH JOINTS ONLY.

Method	Top 1	Top5
AS-GCN [19]	34.8	56.5
ST-GR [20]	33.6	56.1
S-TR [33]	32.4	55.3
T-TR [33]	32.4	55.2
ST-TR [33]	34.4	57.1
ST-GCN [1]	30.7	52.8
ST-G-TSA	32.3	54.6
MS-G3D [2]	34.2	-
MS-G3D*	34.0	57.3
MS-G-TSA	34.8	57.6

TABLE IV
RESULTS FOR ACTION PREDICTION ON KINETICS 400 SKELETON DATASET WITH BOTH BONES AND JOINTS.

Method	Top 1	Top5
2s-AGCN [3]	36.1	58.7
DGNN [4]	34.6	57.6
S-TR [33]	35.4	57.9
T-TR [33]	33.1	55.9
ST-TR [33]	37.0	59.7
MS-G3D [2]	38.0	60.9
MS-G3D*	37.4	60.6
MS-G-TSA	37.9	60.9

1) Performance on NTU-RGB+D datasets: We show results compared with other supervised methods for the NTU-RGB+D 60 and 120 datasets in Table I and II compared with some other state-of-the-art skelection action recognition methods. We named our model ST-G-TSA and MS-G-TSA after applying these two modules to ST-GCN and MS-G3D, two state-of-the-art methods. Note that our hardware cannot achieve the precision of the original implemented MS-G3D [2], we half their precision and report the results as MS-G3D* in Table I, and MS-G-TSA use the same precision as MS-G3D* for fair comparisons. To make comparable results, for ST-GCN, we compare with the original setting, where we only use the joints for training, and for MS-G3D, we use the fused results for joints and bones for comparison.

Compared with the original ST-GCN and MS-G3D, ST-G-TSA and MS-G-TSA outperform them on most of the splits of the dataset. For ST-G-TSA, it has an average of 1.5 accuracy improvement compared with the original implementation of ST-GCN, while for MS-G-TSA, it has an average of 1.0 increase on action prediction accuracy for both of the splits for two datasets. Also, the improvement on ST-GCN is greater than with MS-G3D, indicating much more potential for improving the spatial-temporal understanding for ST-GCN.

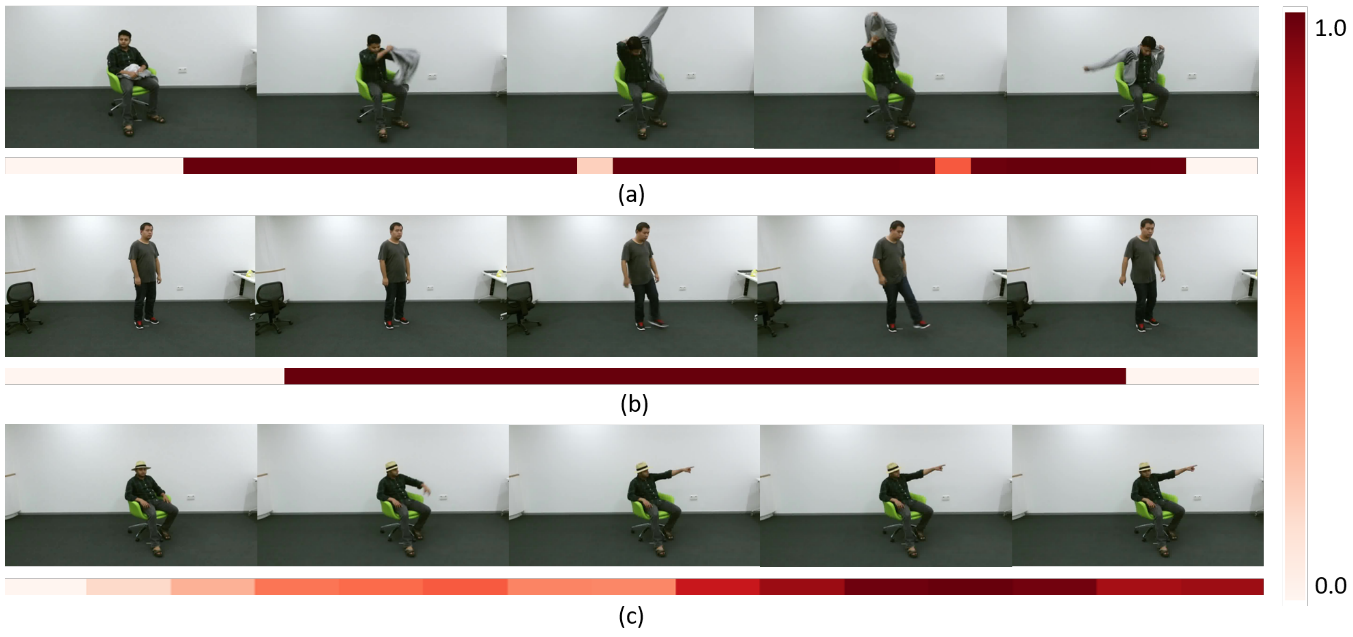


Fig. 5. Weights from the G-TAM for three videos in the NTU-RGB+D *xview* test set for ST-G-TSA. The more saturated the color is, the higher weight is. The annotated actions for the video are (a) 'put on a jacket', (b) 'kicking something', and (c) 'point to something'. We use RGB frames instead of skeletons for better visualization. Weights are divided by the highest weight of the same video for relative comparison.

2) Performance on Kinetics 400 skeleton dataset: In addition to the NTU-RGB+D datasets, we also evaluate on Kinetics 400 skeleton dataset in Table III and IV. We show the top-1 and top-5 accuracy for the performance on this dataset. We notice that we have better results when applying the modules to both baseline methods for most of the splits. For results reported with joints only, we show better results compared with the full precision of MS-G3D and ST-GCN and achieve the best performance for a single model on both top-1 and top-5 accuracies. After adding bones into the network, although the gap between the half-precision implementation for MS-G3D* and MS-G3D, using G-TSM and G-TAM helps narrow the gap show state-of-the-art performances.

C. Qualitative results

In addition, to see if the network is able to focus on the correct frames or segments of the video, we show three visualization results for the final frame or segment-wise weights generated from the G-TAM of ST-G-TSA in Figure 5. We choose the three examples: i) action occurs in almost all frames, ii) action occurs in the middle of the sequence, and iii) action occurs at the end of the video. We plot the weight as the color bar, where more saturated color in the same temporal sequence represents bigger weight values than other frames in the video sequence.

From the visualization results for the weights generated for three different examples, we notice that G-TAM can focus on the important frames and include all the frames that help make the prediction instead of overfitting only a few frames. For example, when the duration of an action is almost the same length as the video in Figure 5 (a), G-TAM distributes the attention for all related frames for the whole video sequence instead of overfitting on a few frames. When an action is happening only in the middle of the video like Figure 5 (b) or not happening at the beginning as Figure 5 (c), the model can focus on the correct segments of the video instead of focusing on the frames that do not help for making the action predictions. For actions that gradually happen along the temporal dimension, the generated weights gradually increase the weight along with the occurrence of the action. Although the action happens at different positions of the video sequence, G-TAM can generate the corresponding weight focusing on the segments or frames that help predict the final action happening in the video.

V. Conclusion

In this paper, we propose two portable modules for skeleton action recognition for graph convolution networks, graph temporal shift module (G-TSM) and graph temporal attention module (G-TAM). G-TAM locates the important frames of the video to focus on making action predictions based on these frames, while G-TSM is a temporal shift network for exchanging the information along the spatial and temporal dimension for graph convolution network without extra computation efforts. Both modules improve the performance of the action recognition on temporal sequences and help the network focus on crucial frames to make final predictions. We show our results on three settings of the three public datasets, NTU-RGB+D 60, NTU-RGB+D 120, and Kinetics 400 skeleton action recognition datasets, and show better results after applying our modules on baseline methods.

References

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

[2] Z. Liu, H. Zhang, Z. Chen, Z. Wang, and W. Ouyang, "Disentangling and unifying graph convolutions for skeleton-based action recognition," in *CVPR*, 2020, pp. 143–152.

[3] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Two-stream adaptive graph convolutional networks for skeleton-based action recognition," in *CVPR*, 2019, pp. 12 026–12 035.

[4] ——, "Skeleton-based action recognition with directed graph neural networks," in *CVPR*, June 2019, pp. 7912–7921.

[5] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015, pp. 2625–2634.

[6] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *CVPR*, 2016, pp. 1933–1941.

[7] H. Yang, D. Yan, L. Zhang, D. Li, Y. Sun, S. You, and S. J. Maybank, "Feedback graph convolutional network for skeleton-based action recognition," *arXiv preprint arXiv:2003.07564*, 2020.

[8] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *ICCV*, 2019, pp. 7083–7093.

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015, pp. 4489–4497.

[10] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, 2017, pp. 6299–6308.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015, pp. 1–9.

[12] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *ECCV*. Springer, 2016, pp. 20–36.

[13] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *CVPR*, 2018, pp. 6450–6459.

[14] Z. Qiu, T. Yao, and T. Mei, "Learning spatio-temporal representation with pseudo-3d residual networks," in *ICCV*, 2017, pp. 5533–5541.

[15] M. Lee, S. Lee, S. Son, G. Park, and N. Kwak, "Motion feature network: Fixed motion filter for action recognition," in *ECCV*, 2018, pp. 387–403.

[16] M. Zolfaghari, K. Singh, and T. Brox, "Eco: Efficient convolutional network for online video understanding," in *ECCV*, 2018, pp. 695–712.

[17] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *ECCV*, 2018, pp. 305–321.

[18] J. Liu, G. Wang, L.-Y. Duan, K. Abdiyeva, and A. C. Kot, "Skeleton-based human action recognition with global context-aware attention lstm networks," *TIP*, vol. 27, no. 4, pp. 1586–1599, 2017.

[19] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-structural graph convolutional networks for skeleton-based action recognition," in *CVPR*, 2019, pp. 3595–3603.

[20] B. Li, X. Li, Z. Zhang, and F. Wu, "Spatio-temporal graph routing for skeleton-based action recognition," in *AAAI*, vol. 33, 2019, pp. 8561–8568.

[21] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-based action recognition with directed graph neural networks," in *CVPR*, 2019, pp. 7912–7921.

[22] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *AAAI*, vol. 32, 2018.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[24] H. Zhu, J. Shi, and J. Wu, "Pick-and-learn: Automatic quality evaluation for noisy-labeled image segmentation," in *MICCAI*. Springer, 2019, pp. 576–584.

[25] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *CVPR*, 2018, pp. 5457–5466.

[26] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "Ntu rgb+ d: A large scale dataset for 3d human activity analysis," in *CVPR*, 2016, pp. 1010–1019.

[27] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "Ntu rgb+ d 120: A large-scale benchmark for 3d human activity understanding," *PAMI*, vol. 42, no. 10, pp. 2684–2701, 2019.

[28] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[29] 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*, vol. 31, no. 1, 2017.

[30] P. Zhang, C. Lan, J. Xing, W. Zeng, J. Xue, and N. Zheng, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," in *ICCV*, 2017, pp. 2117–2126.

[31] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan, "An attention enhanced graph convolutional lstm network for skeleton-based action recognition," in *CVPR*, 2019, pp. 1227–1236.

[32] S. Cho, M. Maqbool, F. Liu, and H. Foroosh, "Self-attention network for skeleton-based human action recognition," in *WACV*, 2020, pp. 635–644.

[33] C. Plizzari, M. Cannici, and M. Matteucci, "Skeleton-based action recognition via spatial and temporal transformer networks," *Computer Vision and Image Understanding*, vol. 208, p. 103219, 2021.

[34] K. Cheng, Y. Zhang, X. He, W. Chen, J. Cheng, and H. Lu, "Skeleton-based action recognition with shift graph convolutional network," in *CVPR*, 2020, pp. 183–192.