The Visual Computer (2020) 36:2433–2444
https://doi.org/10.1007/s00371-020-01955-w

ORIGINAL ARTICLE

STA-GCN: two-stream graph convolutional network with spatial–temporal attention for hand gesture recognition

Wei Zhang[1,2] · Zeyi Lin[1,2] · Jian Cheng[1,2] · Cuixia Ma[1,2] · Xiaoming Deng[1,2] · Hongan Wang[1,2]

Published online: 28 August 2020
© Springer-Verlag GmbH Germany, part of Springer Nature 2020

Abstract

Skeleton-based hand gesture recognition is an active research topic in computer graphics and computer vision and has a wide range of applications in VR/AR and robotics. Although the spatial–temporal graph convolutional network has been successfully used in skeleton-based hand gesture recognition, these works often use a fixed spatial graph according to the hand skeleton tree or use a fixed graph on the temporal dimension, which may not be optimal for hand gesture recognition. In this paper, we propose a two-stream graph attention convolutional network with spatial–temporal attention for hand gesture recognition. We adopt pose stream and motion stream as the two input streams for our network. In pose stream, we use the joint in each frame as the input; In motion stream, we use the joint offsets between neighboring frames as the input. We propose a new temporal graph attention module to model the temporal dependency and also use a spatial graph attention module to construct dynamic skeleton graph. For each stream, we adopt graph convolutional network with spatial–temporal attention to extract the features. Then, we concatenate the feature of the pose stream and motion stream for gesture recognition. We achieve the competitive performance on the main hand gesture recognition benchmark datasets, which demonstrates the effectiveness of our method.

Keywords Hand gesture recognition · Graph convolutional network · Spatial-temporal attention

1 Introduction

Vision-based hand gesture recognition that aims to predict gesture types from videos is an active research area in computer graphics and human–computer interaction, and it has a wide range of applications in VR/AR, health care, and robotics.

Existing hand gesture recognition methods can be classified into two categories according to the type of input: image-based methods [12–14] and hand skeleton pose-based methods [1,2,5,6,15,16]. Image-based methods adopt RGB or RGB-D image sequences as input, and skeletons-based method use 2D/3D hand joints sequences as input. Compared to image-based hand gesture recognition methods, hand skeleton-based methods relieve the difficulty caused by the cluttered background and also has lower computation cost, thus can enable real-time hand gesture interactions on mobile devices. With the boom of low-cost depth cameras and the rapid progress of hand pose estimation research [18–20], the 2D/3D hand joints can be recovered from RGB or RGB-D images easily. For example, Leap Motion and Intel Realsense camera provide hand skeleton pose in real time. Thus, hand skeleton-based methods have been the major hand gesture recognition pipeline.

Skeleton-based hand gesture recognition is still challenging. Existing methods [1,3,21] extract handcrafted feature from skeleton sequences and feed the feature to classifiers for hand gesture recognition. However, these handcrafted features cannot model the spatial and temporal information effectively. Recently, several deep learning-based works [4,15,16] are proposed, which feed hand skeleton sequence into LSTM or CNN networks for hand gesture recognition.

Zeyi Lin and Jian Cheng have contributed equally to this paper. Xiaoming Deng and Hongan Wang are the corresponding authors.

✉ Xiaoming Deng
xiaoming@iscas.ac.cn

✉ Hongan Wang
hongan@iscas.ac.cn

[1] Beijing Key Laboratory of Human-Computer Interaction, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

[2] School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

However, these methods do not exploit the spatial connections among joints in each frame and temporal connections between joints in neighboring frames well.

Recently, the spatial–temporal graph convolutional neural network (ST-GCN) [22] has been successfully used in skeleton-based human action recognition, and several works [5,6] apply it to hand gesture recognition. They construct the hand skeletons into a spatial–temporal graph and use ST-GCN network to extract features. Compared to the other deep learning-based methods, these ST-GCN-based methods can effectively exploit the connections between the joints in both the spatial and temporal domains. However, these works often use a fixed graph according to the kinematic tree of hand skeleton or use a fixed graph on the temporal dimension, which may not be optimal for gesture recognition task as found in [41]. In addition, these ST-GCN methods often use the raw joint coordinates as input and could not capture the motion feature such trajectory effectively.

In this paper, we propose a two-stream graph attention convolutional network with spatial–temporal attention for hand gesture recognition (STA-GCN). Figure 1 illustrates an overview of our approach. We adopt two-stream architecture for skeleton-based hand gesture recognition, which uses pose stream and motion stream as input. For both streams, we use the same network structure. We first initialize the skeleton graph, then we use spatial–temporal graph convolutional network with both spatial attention and temporal attention to extract the feature. In order to encode hand gestures with multi–scale temporal features, we adopt the temporal pyramid pooling layer. Then, we feed the feature of pose stream and motion stream into fully connected layer, fuse the features, and feed to softmax layer for hand gesture prediction.

Compared to the existing ST-GCN methods [5,6,22], we construct a dynamic graph using a spatial–temporal graph attention which can be well adapted to the input data with diversity. Second, we not only use the original hand joint coordinates as input for pose stream and use the hand skeleton joint offsets between different frames as input for motion stream. Third, we use the temporal pyramid pooling [42] to encode multi-scale temporal features.

Our method is also related to [41], which adopts the non-local attention mechanism on the GCN-based action recognition methods for constructing dynamic skeleton graph. In this paper, motivated by the correlation of different frames in [40], we extend the spatial attention mechanism proposed in [41] to temporal domain and use more effective motion stream input than the bone stream used in [41].

Our contributions can be summarized as follows:

1. We propose a graph convolutional network with spatial–temporal attention for hand gesture recognition. Especially, we design an effective dynamic graph on the temporal dimension using a new temporal graph attention module. In order to encode hand gestures with the multi-scale temporal features, we use the temporal pyramid pooling layer [42];
2. We use a two-stream hand gesture network that uses both the raw hand skeleton joints and the joint offsets between neighboring frames as input. The proposed two-stream network can model the spatial–temporal context of hand gestures effectively;
3. We achieve competitive performance on the main hand gesture benchmark datasets.

2 Related work

2.1 Skeleton-based human action recognition

Conventional skeleton-based action recognition methods [23,24] use the handcrafted feature to model the human action, but the performance of these methods are often not satisfied. Recently, several RNN-based methods [25–30] and CNN-based methods [31–35] for human action recognition are proposed. RNN-based methods feed the skeleton coordinate sequences into RNN for feature extraction and recognition. CNN-based methods usually encode the skeleton coordinates with a matrix [43] and feed the matrix to CNN for action recognition.

However, CNN- and RNN-based methods cannot model the connections of the joints effectively. In order to address this problem, Yan et al. [22] represent the human joints with a graph and use the spatial–temporal graph convolutional network (ST-GCN) for feature extraction. Inspired by Yan et al. [22], several GCN-based methods [36,37] are designed for action recognition. They construct the human joint graph using rules in a heuristic manner. Shi et al. [41] adopt the non-local attention mechanism on the GCN-based action recognition method by constructing dynamic skeleton graph. However, they only design the attention mechanism on the spatial graph and use a fixed graph on the temporal dimension; thus, it could not model the temporal connections between frames effectively.

2.2 Skeleton-based hand gesture recognition

Existing hand gesture recognition methods can be divided into two categories: handcrafted feature-based methods and deep learning-based methods. Conventional handcrafted feature-based methods [1,3,21] often encode the hand skeleton joint sequences into handcrafted features. Deep learning-based methods [4,15,16] encode the skeletons sequences into feature vectors and feed them into RNN or CNN to extract spatial–temporal features. However, these methods do not effectively exploit the spatial–temporal context of joints such

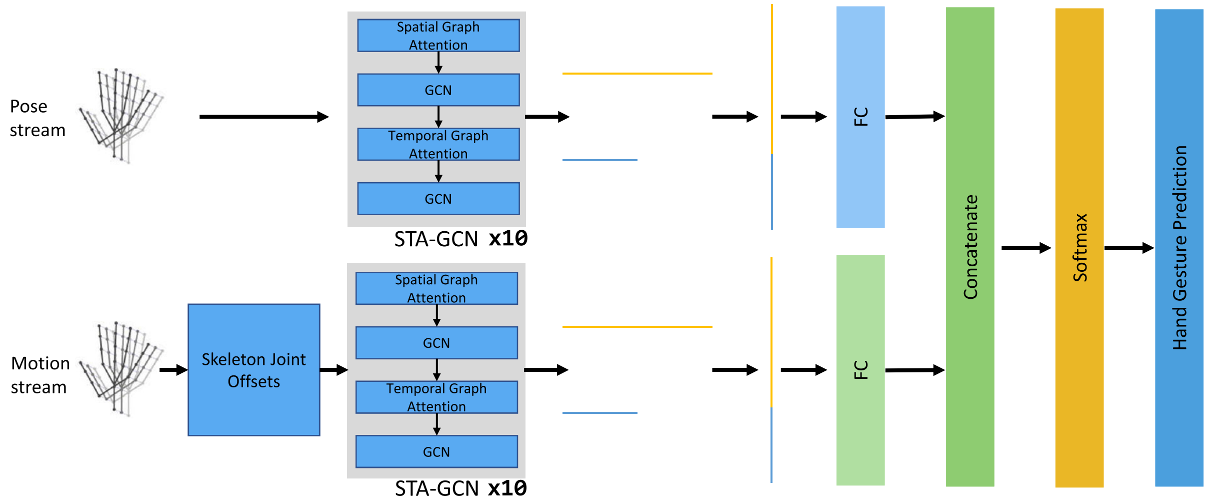


Fig. 1 Illustration of our two-stream graph convolutional network with spatial–temporal attention (STA-GCN) for hand gesture recognition. For pose stream, we use the original joint coordinates as input; For the motion stream, we use the skeleton joint offsets as input. During training stage, we train pose stream and motion stream separately. During the testing stage, the features of pose stream and motion stream are concatenated for hand gesture recognition

as the connections between hand joints of each frame and the connections between hand joints between different frames.

Inspired by the spatial–temporal graph convolutional network (ST-GCN) [22], several works [5,6] also explore ST-GCN for hand gesture recognition. They encode joint sequences into skeleton graph and use GCN to extract features for hand gesture recognition. However, these works do not model the motion feature explicitly, or do not use dynamic graph to enhance the representation capacity of the network.

2.3 Attention models in hand gesture recognition

Several works [4,6] explore to apply attention mechanism for hand gesture recognition. Chen et al. [6] construct dynamic graph using spatial–temporal attention. Hou et al. [4] use the mask branch for applying spatial–temporal attention on each feature extracted by the convolutional network. Compared to the existing works that adopt attention mechanism on spatial dimension or temporal dimension separately, our work uses both spatial attention and temporal attention in GCN for hand gesture recognition.

3 Methodology

3.1 Overview

Motivated by the two-stream network architecture [38] in action recognition, we propose two-stream architecture for skeleton-based hand gesture recognition. Figure 1 gives an overview of our approach. We adopt pose stream and motion stream as our two streams for hand gesture recognition. The two streams use the same network structure, but use different input data. In pose stream, we use the joints in each frame as the input; In motion stream, we use the joint offsets between neighboring frames as the input. We first initialize the skeleton graph, then use spatial–temporal graph convolutional network with the spatial graph attention and temporal graph attention to extract the spatial–temporal features. The output feature of GCN with spatial graph attention will be fed to the GCN with temporal graph attention. Then, we use the temporal pyramid pooling layer (TPP) to extract multi-scale temporal features from input. Finally, we feed the features into fully connected layer and softmax for hand gesture recognition. To enhance the performance of a single stream, the extracted features of pose stream and motion stream are concatenated for hand gesture recognition.

3.2 Graph initialization

For a video of T frames with annotated N joints for each frame, we define a skeleton graph with the node set $V = \{v_{i,j} | i = 1, 2, ..., T, j = 1, 2, ..., N\}$ using joints, where $v_{i,j}$ denotes the j-th node (joint) in the i-th frame, and the feature set $F = \{f_{i,j} | i = 1, 2, ..., T, j = 1, 2, ..., N\}$, where $f_{i,j}$ denotes the feature vector of node $v_{i,j}$ encoded with the position of the joint $v_{i,j}$.

Figure 2 illustrates an overview of the graph initialization. Inspired by [6], we initialize spatial edge, temporal edge and self-connection edge of the graph as follows

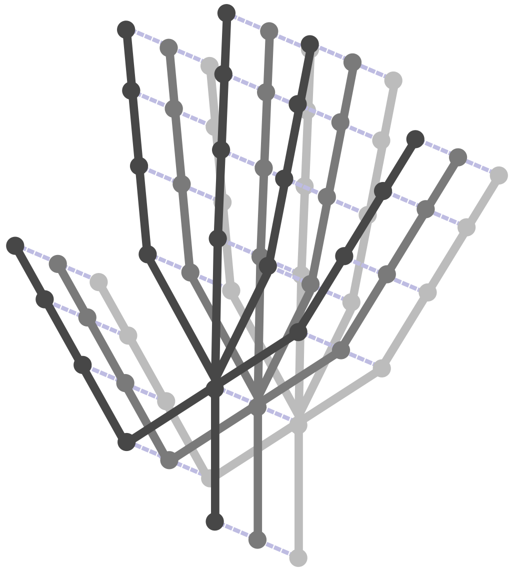


Fig. 2 The spatial–temporal joint connections of the initial graph. The black line denotes the spatial connections and the blue line denotes the temporal connections

1. The spatial edge connects the node v_{i,j_1} and node v_{i,j_2} in the same frame, where node j_1 and node j_2 have the connected joints in the hand kinematic skeleton tree.
2. The temporal edge connects the node $v_{i_1,j}$ and node $v_{i_2,j}$ of the same node in different frames for $|i_1 - i_2| \leq K$. In our experiment, we set $K = 4$.
3. The self-connection edge connects the node $v_{i,j}$ itself.

3.3 Graph convolutional network

After initialing the skeleton graph, we adopt multiple layers of spatial–temporal graph convolution to extract high-level features. Denote the input of the spatial–temporal convolution layer to be a $N \times T \times C$ tensor, where N denotes the number of nodes in each frame, T denotes the number of frames, and C is the number of channels. The skeleton graph consists of spatial graph and temporal graph.

Each spatial graph consists of the hand joints of a frame and the spatial edges (See Sect. 3.2). To implement the graph convolution in the spatial dimension, the propagation rule can be formulated as follows:

$$\mathbf{F}^{(l+1)} = \sigma(\mathbf{A}^s\mathbf{W}^{(l)}\mathbf{F}^{(l)}), \mathbf{A}^s = \tilde{\mathbf{D}}_s^{-\frac{1}{2}}\tilde{\mathbf{A}}^s\tilde{\mathbf{D}}_s^{-\frac{1}{2}} \tag{1}$$

where $\mathbf{A}^s$ is the normalized adjacent matrix of the spatial graph, $\mathbf{W}^{(l)}$ of the size $C_{out} \times C_{in} \times 1 \times 1$ is a weight matrix to be learned (C_{out} and C_{in} are the numbers of input and output channels, respectively), $\mathbf{F}^{(l)}$ and $\mathbf{F}^{(l+1)}$ are the input and output features of the l-th graph convolutional layer, and $\sigma(\cdot)$ is the ReLU activation function. The matrix $\tilde{\mathbf{A}}^s = \{\tilde{\mathbf{A}}^s(i, j)\}$ of the size $N \times N$ is an adjacent matrix of the skeleton graph, and its element $\tilde{\mathbf{A}}^s(i, j)$ denotes whether there is a connection between the nodes i and j, $\tilde{\mathbf{D}}_s$ is the diagonal node degree matrix of $\tilde{\mathbf{A}}^s$.

Each temporal graph consists of the sequence of the same hand joint and the temporal edges (See Sect. 3.2), and the input feature of temporal graph comes from the output of the spatial graph. The graph convolution of the temporal dimension is similar to that of the spatial dimension. In the temporal dimension, the input tensor is transformed to $T \times N \times C$, the matrix $\tilde{\mathbf{A}}^t = \{\tilde{\mathbf{A}}^t(i, j)\}$ of the size $T \times T$ is an adjacent matrix of the temporal graph, and its element $\tilde{\mathbf{A}}^t(i, j)$ denotes whether there is a connection between frame i and frame j. $\tilde{\mathbf{D}}_t$ is the diagonal node degree matrix of $\tilde{\mathbf{A}}^t$

3.4 Graph attention layer

Existing graph convolutional networks often use a fixed graph in both training and testing stages and it is difficult to model the spatial–temporal context effectively from input skeleton sequences with great diversity. Inspired by 2S-AGCN [41], we adopt the spatial attention module in [41] and propose a new temporal attention module to construct a dynamic skeleton graph using the hand joint sequences. For each attention module, we adopt three graph structures as [41], a fixed graph structure, a global graph that denotes the common pattern learned from the training data, and a specific graph for each data that denotes the unique pattern for each data. During the feature propagation, we use the sum of the adjacent matrices of these three graph structures as the adjacent matrix. In order to enforce the stability of the original model, residual connections for spatial and temporal attention modules are used.

3.4.1 Spatial attention module

For the spatial attention module, we follow [41] to reformulate the propagation rule in Eq. (1) as follows:

$$\mathbf{F}^{(l+1)} = \sigma(\mathbf{A}^s\mathbf{W}^{(l)}\mathbf{F}^{(l)}) \tag{2}$$

where the spatial adjacent matrix $\mathbf{A}^s = \mathbf{A}_1^s + \mathbf{A}_2^s + \mathbf{A}_3^s$ is built upon the sum of three matrices: $\mathbf{A}_1^s$, $\mathbf{A}_2^s$, and $\mathbf{A}_3^s$ (the superscript 's' means 'spatial'). Compared to Eq. (1), the adjacent matrix $\mathbf{A}^s$ in Eq. (2) has two more matrices $\mathbf{A}_2^s$ and $\mathbf{A}_3^s$. The matrix $\mathbf{A}_2^s$ represents the global spatial graph that denotes the common pattern for the training data, and the matrix $\mathbf{A}_3^s$ represents the specific spatial graph for each data. Then, we elaborate on the details to compute $\mathbf{A}_1^s$, $\mathbf{A}_2^s$, and $\mathbf{A}_3^s$ [41].

(1) The matrix $\mathbf{A}_1^s$ is the normalized adjacent matrix $\mathbf{A}^s$ in Eq. (1).

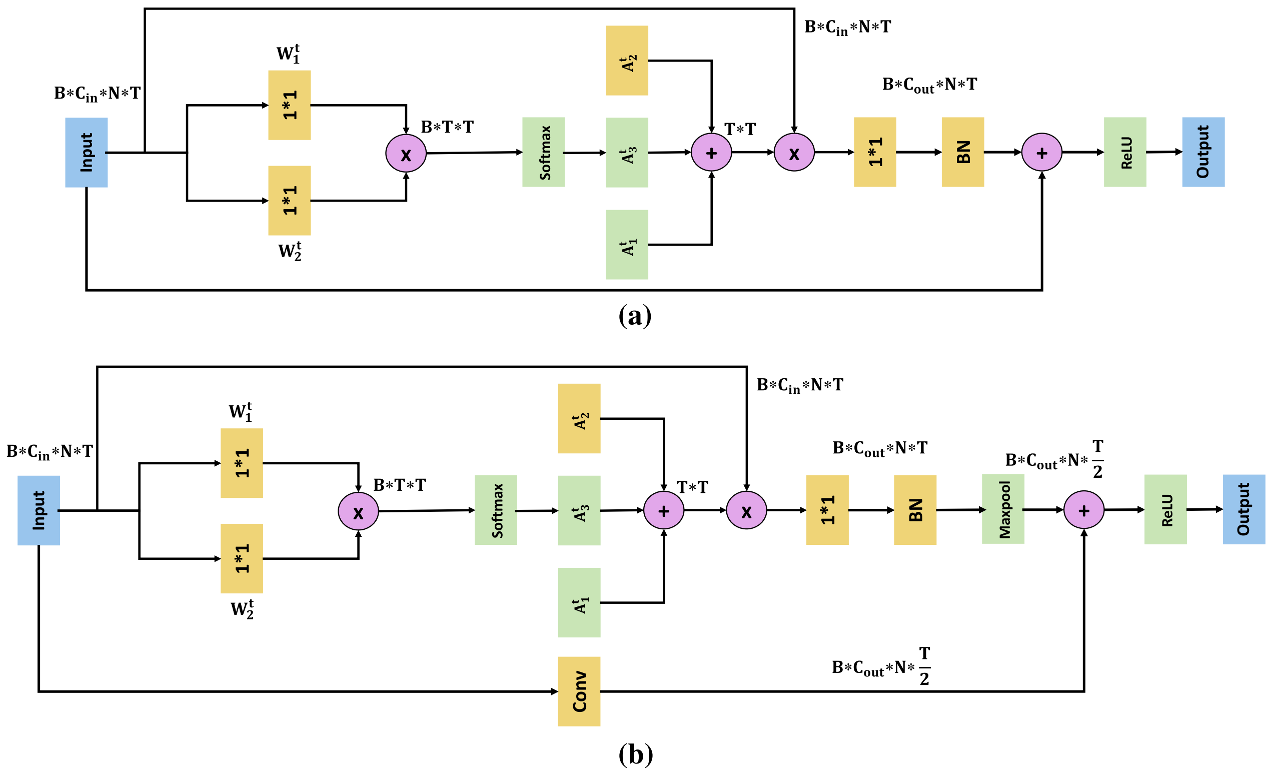


Fig. 3 Illustration of our temporal attention layer. **a** is the vanilla temporal attention layer; **b** is the temporal attention layer with dimension reduction in time dimension. The '1*1' block denotes the 1*1 convolution operation, the orange block indicates that the parameters are learnable, the 'Maxpool' block denotes the 1*2 max pool operation, the 'Conv' block denotes the convolution with size 1*2 and step (1, 2), $\oplus$ denotes the element-wise summation, and $\otimes$ denotes the matrix multiplication

(2) The matrix $\mathbf{A}_2^s$ is a trainable adjacent matrix. Compared to $\mathbf{A}_1^s$ that is fixed, the matrix $\mathbf{A}_2^s$ is learned by the training process, which means that the adjacency of the graph is fully learned from the training dataset. In this data-driven way, the model can learn a specific graph adjacent matrix that can match the hand gesture recognition task. Compared to the conventional adjacent matrix $\mathbf{A}_1^s$, the value of $\mathbf{A}_2^s$ can be arbitrary, which indicates the existence of connections and the correlation strength for the hand joints.

(3) The matrix $\mathbf{A}_3^s$, named as the spatial attention matrix, can be learned based on the input of the each layer, and it can learn the connection strength of the joint connections based on the input data. As a comparison, both $\mathbf{A}_1^s$ and $\mathbf{A}_2^s$ are both fixed for different input data.
Then, we follow [41] to get the matrix $\mathbf{A}_3^s$. Given the input feature $\mathbf{F}^{(l)}$ of the size $B \times C_{in} \times N \times T$ (B is the batch size of input, C_{in} is the number of input channels), we feed it into two different '1*1' convolution layers with the parameters $\mathbf{W}_1^s$ and $\mathbf{W}_2^s$, reshape the output matrices $(\mathbf{W}_1^s\mathbf{F}^{(l)})^T$ and $\mathbf{W}_2^s\mathbf{F}^{(l)}$ to the sizes of $B \times N \times C^sT$ and $B \times C^sT \times N$ (C^s is the number of the '1*1' convolution layers with the parameters $\mathbf{W}_1^s$ and $\mathbf{W}_2^s$), then we multiply them to get the attention matrix $\mathbf{A}_3^s = \{\mathbf{A}_3^s(i, j)\}$ of the size $B \times N \times N$, in which each element $\mathbf{A}_3^s(i, j)$ represents the connection strength between node i and node j. In order to enforce $\mathbf{A}_3^s(i, j)$ within the range of [0, 1], we adopt softmax function to normalize the matrix $\mathbf{A}_3^s$ as follows

$$\mathbf{A}_3^s = softmax((\mathbf{W}_1^s\mathbf{F}^{(l)})^T\mathbf{W}_2^s\mathbf{F}^{(l)}). \tag{3}$$

3.4.2 Temporal attention module

Motivated by the spatial adjacency matrices $\mathbf{A}_1^s$, $\mathbf{A}_2^s$, $\mathbf{A}_3^s$ in the spatial attention module [41], we extend them to the temporal dimension and also use three matrices $\mathbf{A}_1^t$, $\mathbf{A}_2^t$, and $\mathbf{A}_3^t$ to compute the temporal adjacent matrices (the superscript 't' means 'temporal'). Figure 3 illustrates our two temporal graph attention modules. Figure 3a shows the vanilla temporal attention layer, and Fig. 3b shows the temporal attention layer with dimension reduction in time dimension. The vanilla temporal attention layer contains more parameters than the temporal attention layer with dimension reduction. In order to achieve good balance of network capacity and total parameters of the network, we use both the two temporal graph attention modules in our network (see Sect. 3.6).

With the temporal attention module, we can model the temporal dependency of feature maps as follows:

$$\mathbf{F}^{(l+1)} = \sigma(\mathbf{A}^t\mathbf{W}^{(l)}\mathbf{F}^{(l)}) \tag{4}$$

where the temporal adjacent matrix $\mathbf{A}^t = \mathbf{A}_1^t + \mathbf{A}_2^t + \mathbf{A}_3^t$ consists of $\mathbf{A}_1^t$, $\mathbf{A}_2^t$, and $\mathbf{A}_3^t$ (the superscript 't' means 'temporal'), and $\mathbf{W}^{(l)}$ is a weight matrix to be learned. Similar to $\mathbf{A}_2^s$ and $\mathbf{A}_3^s$, $\mathbf{A}_2^t$ and $\mathbf{A}_3^t$ represent a global temporal graph with common pattern of the training data and a specific temporal graph for each data, respectively. Next, we elaborate on how to compute $\mathbf{A}_1^t$, $\mathbf{A}_2^t$, and $\mathbf{A}_3^t$.

(1) The matrix $\mathbf{A}_1^t$ of the size $T \times T$ represents the temporal connections between each node. The matrix $\mathbf{A}_1^t$ is the normalized temporal adjacent matrix of $\tilde{\mathbf{A}}_1^t = \{\tilde{\mathbf{A}}_1^t(i,j)\}$, and the value of $\tilde{\mathbf{A}}_1^t(i,j)$ means whether frame i and frame j are connected in the temporal dimension. We set the values of $\tilde{\mathbf{A}}_1^t(i,j)$ to 1, if $|i-j| \leq K$; otherwise, we set the values of $\tilde{\mathbf{A}}_1^t(i,j)$ to 0. Then, we normalize the temporal adjacency matrix $\tilde{\mathbf{A}}_1^t$ with $\mathbf{A}_1^t = \tilde{\mathbf{D}}_t^{-\frac{1}{2}}\tilde{\mathbf{A}}_1^t\tilde{\mathbf{D}}_t^{-\frac{1}{2}}$.

(2) The matrix $\mathbf{A}_2^t$ is a trainable adjacent matrix. Compared to $\mathbf{A}_1^t$ that is fixed, the matrix $\mathbf{A}_2^t$ is learned from the training dataset.

(3) The matrix $\mathbf{A}_3^t$, named as the temporal attention matrix, is a learned adjacent matrix from its input. Given the input feature $F^{(l)}$ of the size $B \times C_{in} \times N \times T$ (B is the batch size of input, C_{in} is the number of input channels), we feed it into two different '1*1' convolution layers $\mathbf{W}_1^t$ and $\mathbf{W}_2^t$, reshape the output matrices $(\mathbf{W}_1^t\mathbf{F}^{(l)})^T$ and $\mathbf{W}_2^t\mathbf{F}^{(l)}$ to the sizes of $B \times T \times NC^t$ and $B \times NC^t \times T$ (C^t is the number of the '1*1' convolution layers with the parameters $\mathbf{W}_1^t$ and $\mathbf{W}_2^t$), then we multiply these two matrices to get the attention matrix $\mathbf{A}_3^t = \{\mathbf{A}_3^t(i,j)\}$ of the size $B \times T \times T$, in which each element $\mathbf{A}_3^t(i,j)$ represents the connection strength between frame i and frame j. In order to enforce $\mathbf{A}_3^t(i,j)$ within the range [0, 1], we adopt the softmax function to normalize the matrix $\mathbf{A}_3^t$ as follows

$$\mathbf{A}_3^t = softmax((\mathbf{W}_1^t\mathbf{F}^{(l)})^T\mathbf{W}_2^t\mathbf{F}^{(l)}). \tag{5}$$

3.5 Temporal pyramid pooling layer

When dealing with the skeleton-based hand gesture recognition problem, one key issue is how to extract effective features from the input hand skeleton joint sequences.

Inspired by the temporal pyramid pooling (TPP) [42] that has been successfully used in video-based action recognition, we apply TPP for our skeleton-based hand gesture recognition. Benefiting from the advantage of TPP, we can model hand gesture effectively with fusing multiple scale features in the temporal dimension. Figure 4 illustrates the temporal pyramid pooling layer. We adopt two levels of feature pooling. At the first level, we apply average pooling to the feature after STA-GCN. At the second level, we divide the feature after the STA-GCN into several segments of the same length and conduct average pooling on each segment. Then, we get a fixed length feature by concatenating pooling features of the two levels. Compared to global average pooling, temporal pyramid pooling extracts more temporal features, which are helpful to enhance the hand gesture recognition performance.

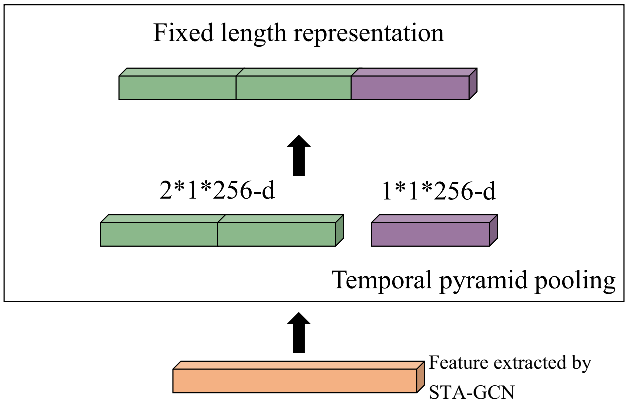


Fig. 4 Illustration of the temporal pyramid pooling layer (TPP). Here, 256 is the channel number of the last graph convolution layer

3.6 Two-stream STA-GCN for hand gesture recognition

We propose a two-stream graph convolutional network with spatial–temporal attention for hand gesture recognition (STA-GCN). Our network adopts pose stream and motion stream as the two input streams, and for each stream, we adopt the same network architecture. For the motion stream, we adopt the skeleton joint offsets $\delta\mathbf{J} = \mathbf{J}_i(t_f+\tau) - \mathbf{J}_i(t_f)$ for as input, where $\mathbf{J}_i(t_f)$ and $\mathbf{J}_i(t_f+\tau)$ to be the positions of the i-th joint at the frame t_f and frame $t_f+\tau$ (τ is the temporal gap for sampled neighboring frames). Conceptually, motion stream is a trajectory-related input.

Figure 5 gives an overview of network architecture for single stream. The input layer has a size of $B \times N \times T \times C$. We use 10 blocks of spatial–temporal graph convolutional network with attention (STA-GCN) for feature extraction, each block includes a spatial graph convolution layer with attention and a temporal graph convolution layer with attention. The number of convolution channels of the first five blocks is 64, the channel number of the next three layers is 128, and

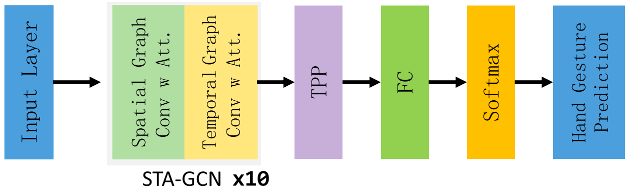


Fig. 5 Our hand gesture recognition network architecture for single stream. We use 10 blocks of spatial–temporal graph convolutional network with attentions (STA-GCN) for feature extraction, then use the temporal pyramid pooling (TPP) layer, a fully connected layer (FC) and a softmax function for hand gesture recognition

the channel number of the last two layers is 256. The temporal graph convolution of the 5th block and the 8th block use the temporal attention layer with dimension reduction in time dimension (see Fig. 3b), and the rest blocks use the vanilla temporal attention layer (see Fig. 3a).

After the 10 blocks of STA-GCN, the extracted feature is fed to a temporal pyramid pooling layer mentioned in Sect. 3.5 and a fully connected layer with a softmax function for hand gesture recognition. The size of the fully connected layer is the number of hand gesture categories.

3.7 Implementation details

We use the PyTorch to implement our network architecture. We use Kaiming initialization to initialize the parameters of weights. During training, we use the dropout with a probability of 0.5 after temporal pyramid pooling layer. We use Adam [39] optimizer to optimize our network, and the batch size is set to 60. In order to conduct data augmentation, we slightly disturb the hand joint coordinates with random translations and scales in each dataset. Because different videos have different frame lengths, data need to be filled and sampled. We processed the frame lengths of all videos to W (W is set to 64 in our experiment). For videos containing less than W frames, we randomly fill in blank frames at the beginning and end of the video in the training set, and fill in blank frames at the end of the video in the testing set. For videos containing more than W frames, we randomly select continuous W frames from the video in the training set and select continuous W frames from the video in the testing set.

For the two-stream network architecture, we train each stream separately during the training stage, and during the testing stage, we sum the feature with the same weight (0.5) before the softmax layer, and feed to softmax for hand gesture recognition.

Because different subjects perform hand gestures at different speeds, the motion stream with a fixed temporal gap may not be optimal for all gestures. The motion stream with the small gap is suitable for fast gestures, but it may introduce abundant frames for slow gestures. The motion stream using a big gap is suitable for slow gestures, but it may overlook details for fast gestures. To solve this problem, we adopt motion stream consisting of three streams with temporal gap of 5, 10, and 15 in the experiment. Therefore, our network actually has four streams, one pose stream and three motion streams with different temporal gaps.

4 Experiments

4.1 Datasets and evaluation metrics

4.1.1 DHG14/28 dataset [1]

The DHG dataset contains 2800 videos and 14 gestures categories, which are performed 5 times by 20 participants in two finger configurations. The key frames of each video are labeled. The gestures are performed in two ways: using one single finger or the whole hand. The 3D positions of 22 hand joints in each frame are provided.

4.1.2 SHREC2017 dataset [2]

The SHREC2017 dataset contains 2800 videos and 14 gestures categories, performed between 1 and 10 times by 28 participants in two finger configurations. The 22 joint coordinates are also provided. Compared to DHG14/28 dataset, SHREC2017 dataset does not provide the key frame of each video, which makes it more difficult.

4.1.3 Evaluation metrics

For DHG14/28, models are evaluated by using the leave-one-subject-out cross-validation strategy. In each experiment, 19 subjects are used for training and 1 subject is used for testing. For SHREC2017, the videos have been divided to 1960 training videos (70% of the dataset) and 840 testing videos (30% of the dataset). We report the accuracy of 14 gestures and the accuracy of 28 gestures for both datasets.

4.2 Ablation study

To further understand the effect of each component of our network, we conduct several ablation studies.

4.2.1 Effect of temporal pyramid pooling layer

In order to evaluate the effect of our temporal pyramid pooling layer, we conduct comparison experiments with pose steam on SHREC2017 dataset using the temporal pyramid pooling layer (TPP) and an alternative global average pooling layer (GAP). In the comparison, we simply replace the temporal pyramid pooling in our network with global average pooling. Table 1 shows the hand gesture recognition

Table 1 The hand gesture recognition accuracy on SHREC2017 dataset using global average pooling layer (GAP) and temporal pyramid pooling layer (TPP)

Layer	14 gestures
GAP Layer	91.5
TPP Layer	**93.2**

Table 2 The accuracy of different streams on SHREC2017 dataset

Stream	14 gestures
Pose stream	93.2
Motion stream	94.4
Two streams	**94.5**

accuracy. We outperform the approach using global average pooling by 1.7%. It demonstrates that the temporal pyramid pooling layer can model the temporal feature more effectively.

4.2.2 Effect of two-stream fusion

Table 2 shows the recognition accuracy using different streams on SHREC 2017 dataset. We get 93.4% accuracy using the pose stream with hand joint coordinates as input and 94.4% accuracy using the motion stream with the offset of the joint coordinates between different frames with temporal gap of 5 as input. Our network using two-stream fusion outperforms the results with single pose stream and single motion stream by 1.1% and 0.1%, respectively. This experiment demonstrates that the two-stream architecture can exploit both pose and movement information and enhance the performance of hand gesture recognition effectively.

4.2.3 Effect of different adjacency matrices in spatial attention layer

Table 3 shows the recognition accuracy using different spatial adjacent matrices on SHREC-2017 dataset. We observe that our method using the adjacent matrix $A_1^s + A_2^s + A_3^s$ gains 2.4% and 7.0% over our method using the original adjacent matrix A_1^s for 14 gestures and 28 gestures. This demonstrates that the graph attention layer is effective. We evaluate the performance using other adjacent matrices $A_1^s + A_2^s$, $A_1^s + A_3^s$, and $A_2^s + A_3^s$ and find that they all perform worse than that with the adjacent matrix $A_1^s + A_2^s + A_3^s$. Therefore, we choose to use $A_1^s + A_2^s + A_3^s$ as the spatial adjacent matrix for graph convolution.

Table 3 The accuracy of our method using different spatial adjacent matrices on SHREC2017 dataset

Spatial adjacent matrix	14 gestures	28 gestures
A_1^s	93.0	84.8
A_2^s	95.1	91.7
A_3^s	94.3	88.7
$A_1^s + A_2^s$	94.6	92.1
$A_1^s + A_3^s$	93.2	87.4
$A_2^s + A_3^s$	95.1	91.1
$A_1^s + A_2^s + A_3^s$	**95.4**	**91.8**

Table 4 The accuracy of our method using different temporal adjacent matrices on SHREC2017 dataset

Temporal adjacent matrix	14 gestures	28 gestures
A_1^t	94.8	90.5
A_2^t	92.9	88.8
A_3^t	94.3	89.6
$A_1^t + A_2^t$	94.1	90.2
$A_1^t + A_3^t$	95.0	90.7
$A_2^t + A_3^t$	94.2	89.8
$A_1^t + A_2^t + A_3^t$	**95.4**	**91.8**

4.2.4 Effect of different adjacency matrices in temporal attention layer

In order to investigate the effect of the proposed temporal attention layer, we conduct comparison experiments with the spatial adjacency matrix $A^s = A_1^s + A_2^s + A_3^s$ and different temporal adjacency matrices. Table 4 shows the recognition accuracy using different temporal adjacent matrices on SHREC2017 dataset. We observe that our method using the adjacent matrix $A_1^t + A_2^t + A_3^t$ gains 1.4% and 7.0% over our method using the original adjacent matrix A_1^t for 14 gestures and 28 gestures. This demonstrates that our temporal graph attention layer is effective. We evaluate the performance using A_2^t, A_3^t, and $A_2^t + A_3^t$ and observe that they all perform worse than using $A_1^t + A_2^t + A_3^t$, because they lack the original temporal connections between the adjacent frames. We also evaluate the performance using other adjacent matrices $A_1^t + A_2^t$ and $A_1^t + A_3^t$, and find that they also perform worse than the adjacent matrix $A_1^t + A_2^t + A_3^t$. Therefore, we choose to use $A_1^t + A_2^t + A_3^t$ as the temporal adjacent matrix for graph convolution.

4.2.5 Effect of spatial and temporal attention layers

In order to verify whether the spatial and temporal attention layers are useful, we conduct experiments with different attention settings. Table 5 shows the recognition accuracy

Table 5 The accuracy of our method using different attention modules on SHREC2017 dataset

Method	14 gestures	28 gestures
Baseline	92.0	82.9
Spatial Attention	94.8	90.5
Spatial–Temporal Attention	**95.4**	**91.8**

Table 6 Performance comparison of our motion stream and bone stream on SHREC2017 dataset

Method	14 gestures	28 gestures
Ours (bone stream [41])	78.6	74.5
Ours (motion stream)	**94.4**	**89.6**

using different attention modules on SHREC2017 dataset. ‘Baseline’ means that we use the same network architecture of our full model and use the original propagation rule (Eq. 1) for spatial and temporal dimensions. ‘Spatial Attention’ means that we use the spatial adjacency matrix $A^s = A^s_1 + A^s_2 + A^s_3$ and the temporal adjacency matrix $A^t = A^t_1$. ‘Spatial–Temporal Attention’ means that we use the spatial adjacency matrix $A^s = A^s_1 + A^s_2 + A^s_3$ and the temporal adjacency matrix $A^t = A^t_1 + A^t_2 + A^t_3$. We observe that: 1) For 14 gesture classification task, the performance of our method with spatial–temporal attention gains 0.6% and 3.4% over our method with spatial attention and our baseline method; 2) For 28 gesture classification task, the performance of our method with spatial–temporal attention gains 1.3% and 8.9% over our method with spatial attention and our baseline method. Therefore, both the spatial attention and our temporal attention are effective to improve the gesture recognition accuracy.

4.2.6 Our motion stream versus bone stream in [41]

In order to investigate the effect of other types of the second stream, we compare the performance of our method (single stream) with our motion stream and the bone stream (B-stream) used in [41]. For bone stream, each bone is represented as a vector from a joint from its parent joint. Table 6 shows the performance of our single-stream network with motion stream and bone stream. We can see that our motion stream can achieve better performance. The possible reason that our motion stream is better could be that bone stream overlooks the temporal movement of joints, while most types of hand gesture are related to such movement.

4.3 Visualization of the temporal attention

Figure 6 shows temporal attention of the gesture ‘Shake’ in SHREC2017 dataset. Figure 6a, d shows temporal attention matrices of the first and fifth STA-GCN block. The gray value in the matrices means the connection strength of each pair of frames. Since we adopt sampled 64 frames from all videos by data preprocessing, temporal attention matrices are of the size 64 × 64. From Fig. 6a, we can see that frame 12 and frame 40 have strong connection with the other frames. Figure 6b, c is depth images of frame 12 and frame 40, respectively. From Fig. 6d, we can see that frame 1, 24 and frame 60 have strong connection with the other frames. Figure 6e–g is the depth images of frame 1, 24, and 60, respectively. The images in Fig. 6b, c are both hands moving to the right, and the images in Fig. 6e–g are all hand moving to the left. We notice that hand movements to left and right are the key frames related to the gesture ‘Shake’. Therefore, our temporal attention could extract key frames related to the hand gesture.

4.4 Comparisons with state-of-the-art methods

We compare our method with state-of-the-art methods on two standard benchmark datasets: DHG14/28 [1] dataset and SHREC2017 [2] dataset. We compare with the existing approaches using handcrafted features [7,10] [1,3], deep learning-based approaches [4,8,11,17], and graph-based approaches [5,6,22]. Tables 7 and 8 show the comparison results.

4.4.1 Comparisons with state-of-the-art method on SHREC2017 dataset

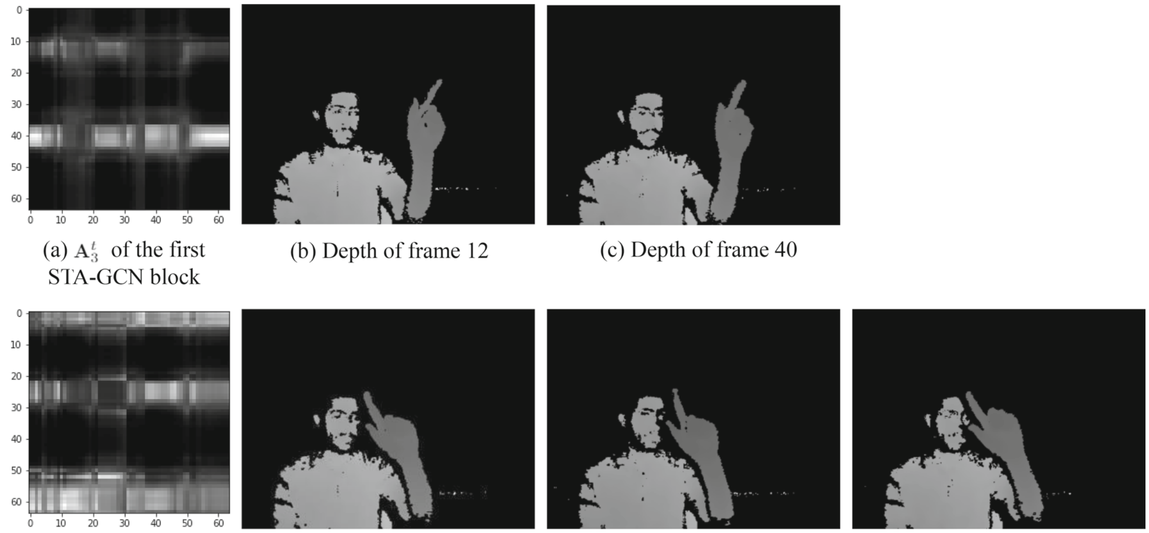
Compared to DHG14/28 dataset that contains the labeled key frames of videos, SHREC 2017 dataset only contains the raw videos captured by the cameras that makes it more difficult for hand gesture recognition. Table 7 compares our method with state-of-the-art methods on SHREC2017 dataset. We outperform state-of-the-art method DeepGRU on 14 gestures and 28 gestures recognition by 0.9% and 0.4%.

4.4.2 Comparisons with state-of-the-art methods on DHG14/28 dataset

Table 8 compares our method with state-of-the-art hand gesture recognition methods on DHG14/28 dataset. Our method outperforms most of state-of-the-art methods except DG-STA, and our method is superior to 2S-AGCN and ST-GCN by 1.6% and 0.3% for 14 gestures recognition, and 1.2% and 0.6% for 14 gestures recognition, respectively. We can see that our method can achieve competitive results.

4.4.3 Comparisons with 2S-AGCN [41]

We also compare our method with 2S-AGCN [41]. The method 2S-AGCN is proposed for skeleton-based action recognition. In order to conduct fair comparison, we train



(a) $\mathbf{A}_3^t$ of the first STA-GCN block
(b) Depth of frame 12
(c) Depth of frame 40
(d) $\mathbf{A}_3^t$ of the fifth STA-GCN block
(e) Depth of frame 1
(f) Depth of frame 24
(g) Depth of frame 60

Fig. 6 Visualization of a temporal attention matrix A_3^t of gesture 'Shake.' First row: visualization of the temporal attention of the first STA-GCN block. **a** Temporal attention matrix A_3^t. **b** Depth image of frame 12. **c** Depth image of frame 40. Second row: visualization of the temporal attention of the fifth STA-GCN block. **a** Temporal attention matrix A_3^t. **e** is a depth image of frame 1. **f** is a depth image of frame 24. **g** is a depth image of frame 60

Table 7 Comparisons with state-of-the-art methods on SHREC2017 dataset

Method	14 gestures	28 gestures
HIF3D [10]	90.4	80.4
De Smedt et al. [7]	88.2	81.9
Devineau et al. [11]	91.2	84.3
ST-GCN [22]	92.7	87.7
STA-Res-TCN [4]	93.6	90.7
ST-TS-HGR-NET [5]	94.29	89.4
DG-STA [6]	94.4	90.7
2S-AGCN [41]	93.3	91.1
DeepGRU [17]	94.5	91.4
Our method	**95.4**	**91.8**

Table 8 Comparisons with state-of-the-art methods on DHG14/28 dataset

Method	14 gestures	28 gestures
SoCJ+HoHD+HoWR [1]	83.1	80.0
De Smedt et al. [7]	82.5	68.1
CNN + LSTM [8]	85.6	81.1
Chen et al. [15]	84.6	80.3
DPTC [9]	85.8	80.2
STA-Res-TCN [4]	89.2	85.0
2S-AGCN [41]	89.9	86.5
ST-GCN [22]	91.2	87.1
DG-STA [6]	**91.9**	**88.0**
Our method	91.5	87.7

2S-AGCN on DHG14/28 and SHREC2017 dataset and use the learned models for evaluation. In Tables 7 and 8, we show the performance of 2S-AGCN models on SHREC2017 and DHG14/28 dataset. We can observe that: (1) For SHREC2017, our method outperforms 2S-AGCN by 2.1% on 14 gestures recognition and 0.7% on 28 gestures recognition; (2) For DHG14/28, our method outperforms 2S-AGCN by 1.6% on 14 gestures recognition and 1.2% on 28 gestures recognition.

5 Conclusions

In this paper, we propose a new neural network architecture for hand gesture recognition, a two-stream graph convolutional network with spatial–temporal attention. We adopt a two-stream architecture to model the pose feature and motion feature of hand gesture. For each stream, we encode the input as a graph, design a new temporal graph attention module to model the temporal dependency, and also use a spatial graph attention module to construct dynamic skeleton graph. In order to extract multiple scale temporal features of hand gesture, we use an effective temporal pyramid pooling layer. Experiments on two main benchmark datasets demonstrate that our method can achieve competitive performance. Although our method is designed for hand gesture recognition, it can also inspire related researches (such as action recognition) that can be modeled as spatial–temporal graph.

Acknowledgements This work was supported by the National Key Research and Development Plan (2016YFB1001200, 2018YFC0809300), Natural Science Foundation of China (61473276, 61872346), and Natural Science Foundation of Beijing (L182052).

References

1. De Smedt, Q., Wannous, H., Vandeborre, J.-P.: Skeleton-based dynamic hand gesture recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2016)
2. De Smedt, Q., et al.: 3D hand gesture recognition using a depth and skeletal dataset: SHREC'17 track. In: Proceedings of the Workshop on 3D Object Retrieval. Eurographics Association (2017)
3. De Smedt, Q., Wannous, H., Vandeborre, J.-P.: Heterogeneous hand gesture recognition using 3D dynamic skeletal data. In: Computer Vision and Image Understanding (2019)
4. Hou, J., et al.: Spatial-temporal attention res-TCN for skeleton-based dynamic hand gesture recognition. In: Proceedings of the European Conference on Computer Vision (2018)
5. Nguyen, X.S., et al.: A neural network based on SPD manifold learning for skeleton-based hand gesture recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
6. Chen, Y., Zhao, L., Peng, X., Yuan, J., Metaxas, D.N.: Construct dynamic graphs for hand gesture recognition via spatial-temporal attention. In: Proceedings of the British Machine Vision Conference (2019)
7. De Smedt, Q., Wannous, H., Vandeborre, J.-P.: 3d hand gesture recognition by analysing set-of-joints trajectories. In: International Workshop on Understanding Human Activities through 3D Sensors (2017)
8. Nunez, J.C, Cabido, R., Pantrigo, J.J., Montemayor, A.S., Velez, J.F.: Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. Pattern Recognition (2018)
9. Weng, J., Liu, M., Jiang, X., Yuan, J.: Deformable pose traversal convolution for 3d action and gesture recognition. In: In Proceedings of the European Conference on Computer Vision (2018)
10. Boulahia, S.Y., Anquetil, E., Multon, F., Kulpa, R.: Dynamic hand gesture recognition based on 3D pattern assembled trajectories. In: International Conference on Image Processing Theory, Tools and Applications (2017)
11. Devineau, G., Moutarde, F., Xi, W., Yang, J.: Deep learning for hand gesture recognition on skeletal data. In: Proceedings of IEEE International Conference on Automatic Face Gesture Recognition (2018)
12. Freeman, W.T, Roth, M.: Orientation histograms for hand gesture recognition. In: Proceedings of International Workshop on Automatic Face and Gesture Recognition (1995)
13. Molchanov, P., Gupta, S., Kim, K., Kautz, J.: Hand gesture recognition with 3D convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2015)
14. Wang, C., Liu, Z., Chan, S.-C.: Superpixel-based hand gesture recognition with kinect depth camera. In: IEEE Transactions on Multimedia (2015)
15. Chen, X., Guo, H., Wang, G., Zhang, L.: Motion feature augmented recurrent neural network for skeleton-based dynamic hand gesture recognition. In: In Proceedings of the IEEE International Conference on Image Processing (2017)
16. Núñez, J.C., Cabido, R., Pantrigo, J.J., Montemayor, A.S, Vélez, J. F.: Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. In: Pattern Recognition (2018)
17. Mehran, M., LaViola Jr, J.J.: DeepGRU: deep gesture recognition utility. In: In Proceedings of International Symposium on Visual Computing (2018)
18. Oberweger, M., Lepetit, V.: Deepprior++: improving fast and accurate 3D hand pose estimation. In: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
19. Oberweger, M., Wohlhart, P., Lepetit, V.: Hands deep in deep learning for hand pose estimation. *arXiv preprint* (2015)
20. Oberweger, M., Wohlhart, P., Lepetit, V.: Training a feedback loop for hand pose estimation. In: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
21. Ohn-Bar, E., Trivedi, M.: Joint angles similarities and HOG2 for action recognition. In: In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (2013)
22. Yan, S., Xiong, Y., Lin, D.: Spatial temporal graph convolutional networks for skeleton-based action recognition. In: In Proceedings of the AAAI Conference on Artificial Intelligence (2018)
23. Vemulapalli, R., Arrate, F., Chellappa, R.: Human action recognition by representing 3d skeletons as points in a lie group. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2014)
24. Fernando, B., Gavves, E., Oramas, J.M., Ghodrati, A., Tuytelaars, T.: Modeling video evolution for action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
25. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
26. Shahroudy, A., Liu, J., Ng, T.-T., Wang, G.: NTU RGB+D: a large scale dataset for 3d human activity analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
27. Liu, J., Shahroudy, A., Xu, D., Wang, G.: Spatio-temporal LSTM with trust gates for 3d human action recognition. In: Proceedings of the European Conference on Computer Vision (2016)
28. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In: Proceedings of the AAAI Conference on Artificial Intelligence (2017)
29. Zhang, P., Lan, C., Xing, J., Zeng, W., Xue, J., Zheng, N.: View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
30. Cao, C., Lan, C., Zhang, Y., Zeng, W., Lu, H., Zhang, Y.: Skeleton-based action recognition with gated convolutional neural networks. In: IEEE Transactions on Circuits and Systems for Video Technology (2018)
31. Liu, H., Tu, J., Liu, M.: Two-stream 3d convolutional neural network for skeleton-based action recognition. *arXiv preprint* (2017)
32. Kim, T.S., Reiter, A.: Interpretable 3d human action analysis with temporal convolutional networks. In: Proceedings of Computer Vision and Pattern Recognition Workshops (2017)
33. Ke, Q., Bennamoun, M., An, S., Sohel, F.A., Boussad, F.: A new representation of skeleton sequences for 3d action recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2017)
34. Liu, M., Liu, H., Chen, C.: Enhanced skeleton visualization for view invariant human action recognition. In: Pattern Recognition (2017)
35. Li, C., Zhong, Q., Xie, D., Pu, S.: Skeleton-based action recognition with convolutional neural networks. In: In Proceedings of International Conference on Multimedia & Expo Workshops (2017)
36. Zhang, X., Xu, C., Tian, X., Tao, D.: Graph edge convolutional neural networks for skeleton-based action recognition. In: IEEE Transactions on Neural Networks and Learning Systems (2018)

37. Gao, X., Hu, W., Tang, J., Liu, J., Guo, Z.: Optimized skeleton-based action recognition via sparsified graph regression. In: Proceedings of the 27th ACM International Conference on Multimedia (2019)
38. Simonya, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems (2014)
39. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization (2014). arXiv:1412.6980
40. Wen, Y., Gao, L., Fu, H., Zhang, F.-L., Xia, S.: Graph CNNs with motif and variable temporal block for skeleton-based action recognition. *AAAI* (2019)
41. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (2019)
42. Wang, P., Cao, Y., Shen, C., Liu, L., Shen, H.T.: Temporal pyramid pooling-based convolutional neural network for action recognition. In: IEEE Transactions on Circuits and Systems for Video Technology (2017)
43. Lin, Z., Zhang, W., Deng, X., Ma, C., Wang, H.: Image-based pose representation for action recognition and hand gesture recognition. In: Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (2020)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Wei Zhang is a Ph.D. student at Institute of Software, Chinese Academy of Sciences. His main research topics are hand gesture recognition, human–computer interaction.

Zeyi Lin a Master student at Institute of Software, Chinese Academy of Sciences. His main research topics are hand gesture recognition, human–computer interaction.

Jian Cheng is a Master student at Institute of Software, Chinese Academy of Sciences. His main research topics are hand gesture recognition, human-computer interaction.

Cuixia Ma received the B.S. and M.S. degrees from Shandong University, China, in 1997 and 2000, respectively, and the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 2003. She was a Research Associate in the Department of Computer Science, Naval Postgraduate School in Monterey, CA, USA, from 2005 to 2006. She is a Professor with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, and Beijing Key Laboratory of Human-Computer Interaction. Her research interests include human -computer interaction and multimedia computing.

Xiaoming Deng received the BS and MS degrees from Wuhan University, in 2001 and 2004, respectively, and the Ph.D. degree from Institute of Automation, Chinese Academy of Sciences (CAS), in 2008. After a two-year postdoctoral at Institute of Computing Technology, CAS, he joined Institute of Software, CAS, in 2010, and he is currently an Associate Professor. He has been a Research Fellow at National University of Singapore from 2012 to 2013. His main research topics are in computer vision, and specifically related to 3D reconstruction, human motion tracking and synthesis, and deep learning.

Hongan Wang received the PhD degree in computer science from Institute of Software, Chinese Academy of Sciences. He is a Professor and the Director of Beijing Key Laboratory of Human-Computer Interactions Laboratory at Institute of Software, Chinese Academy of Sciences. His research interests are real-time reasoning and intelligent user interaction.