Long-Short Graph Memory Network for Skeleton-based Action Recognition

Junqin Huang[1], Zhenhuan Huang[1], Xiang Xiang[3]*, Xuan Gong[4]†,Baochang Zhang[1,2,5]†

[1]Beihang University [2]Hangzhou Innovation Institue, Beihang University, Hangzhou, 310051, China
[3]TuSimple, Inc. [4]University at Buffalo
[5]Shenzhen Academy of Aerospace Technology, Shenzhen, China
hjq18810013653@buaa.edu.cn, 16231192@buaa.edu.cn, xxiang@cs.jhu.edu, xuangong@buffalo.edu,
bczhang@buaa.edu.cn

Abstract

Current studies have shown the effectiveness of long short-term memory network (LSTM) for skeleton-based human action recognition in capturing temporal and spatial features of the skeleton sequence. Nevertheless, it still remains challenging for LSTM to extract the latent structural dependency among nodes. In this paper, we introduce a new long-short graph memory network (LSGM) to improve the capability of LSTM to model the skeleton sequence - a type of graph data. Our proposed LSGM can learn high-level temporal-spatial features end-to-end, enabling LSTM to extract the spatial information that is neglected but intrinsic to the skeleton graph data. To improve the discriminative ability of the temporal and spatial module, we use a calibration module termed as graph temporal-spatial calibration (GTSC) to calibrate the learned temporal-spatial features. By integrating the two modules into the same framework, we obtain a stronger generalization capability in processing dynamic graph data and achieve a significant performance improvement on the NTU and SYSU dataset. Experimental results have validated the effectiveness of our proposed LSGM+GTSC model in extracting temporal and spatial information from dynamic graph data. [1]

1. Introduction

Human action recognition has become an active research area in recent years due to its wide range of applications, such as video surveillance, patient monitoring, human-computer interaction, and virtual reality [1] [18] [21]. The conventional human action recognition methods are mainly based on processing of RGB images and videos, which focus on extracting spatial and temporal information from RGB frames and temporal optical flow [27] [7] [29]. However, there are still many limitations in RGB-based action recognition, mainly reflected in the influence of the background, such as background confusion, ambient light changes, low image resolution, camera viewpoint, and so on.

To address such limitations, skeleton-based action recognition was proposed. Skeleton sequence is a time series of skeleton joint nodes' coordinates, which can be easily captured by the depth sensor [34] and advanced human pose estimation algorithms [2]. As the skeleton sequence does not contain color information, it avoids the limitations of RGB videos or images. Previous works have successfully applied Recurrent Neural Network (RNN), Long Short-Term Memory Network (LSTM) [26] and Gated Recurrent Unit (GRU)[3] to skeleton-based action recognition. Moreover, Convolutional Neural Networks (CNNs) have also been successfully exploited to extract temporal-spatial information from skeleton sequence [8] [32] [10]. To handle the surrounding distractions problem and other variations, the attention mechanism [26][15][16] has been introduced to provide a robust recognition system. In recent years, graph-based models (*e.g.*, GNN) have become an active research area due to their effectiveness in extracting high-level representations of the graph structure data. Early studies learn a target nodes representation by multiple iterations of message passing between the node and their neighbors[6][22][4], which enables models to capture semantic relation and structural information of the entire graph. Since GNN is able to capture latent dependencies among nodes and edges, it has been successfully applied to the Traveling Salesman Problem(TSP) [19], chemical

*This work was done prior to Xiang joining TuSimple.
†Xuan Gong and Baochang Zhang are the corresponding authors.
[1]The code is made publicly available at https://github.com/bczhangbczhang/Long-Short-Graph-Memory-Network

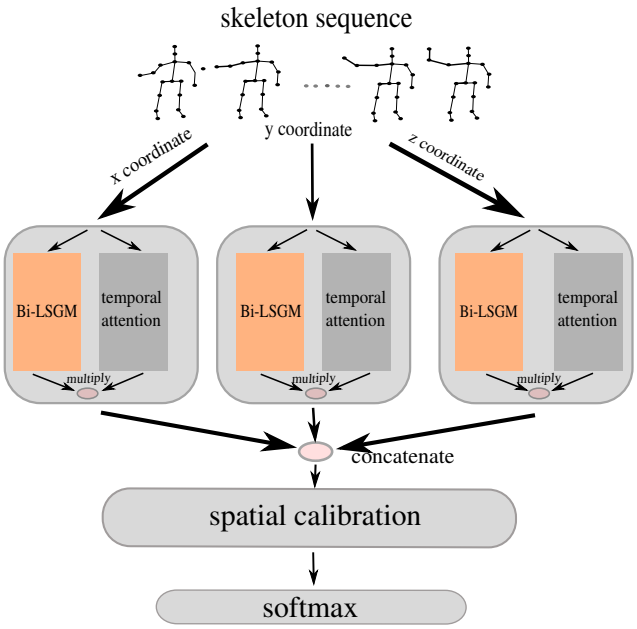

Figure 1. The pipeline of our proposed LSGM+GTSC model. Each input skeleton sequence is treated as a $T \times N \times 3$ matrix, and we use three identical LSGM to model it in each axis separately, cooperated with temporal attention module. Finally we concatenate three *feature maps* with the shape of $T \times N$ together and feed them to spatial calibration.

prediction problems [5], and detection and recognition of human-object interactions [20]. Moreover, as a branch of GNN, the graph convolutional network (GCN) has been widely applied to skeleton-based action recognition tasks in recent years due to its robustness in dealing with non-Euclidean data [31] [17] [24] [12]. Previous LSTM-based models have achieved great progress in skeleton-based action recognition due to its robustness in capturing temporal features. Nevertheless, LSTM works poorly on extracting valid spatial information. Motivated by this, exploiting the effectiveness of GCN in capturing the latent dependency among nodes, we deploy GCN on recurrent unit (*e.g.*, LSTM) to enhance its ability to extract spatial features, thus forming our LSGM. Furthermore, to calibrate the learned features, we propose a graph temporal-spatial calibration module (GTSC) to cooperate with LSGM.

In this paper, our goal is to combine these two powerful tools (LSTM and GCN) and exploit the advantages of LSTM in processing time series and the advantages of GCN in extracting spatial information. Based on this idea, we combine LSTM and GCN to propose a novel Long-Short Graph Memory network(LSGM), which can simultaneously learn temporal and spatial features. As illustrated in Fig. 3, each LSGM cell takes the current graph V^t with the shape of N x d as input, where N denotes the number of joints and d denotes the feature vector dimension of each joint of each frame. It is worth noticing that the hidden states and memory cells in the LSTM store data in the form of graph matrices instead of vectors. Thanks to the graph convolution, LSGM has a strong generalization capability on modeling dynamic graph data. Furthermore, we introduce a graph temporal-spatial calibration module (GTSC), which consist of a temporal attention module and a spatial calibration module. GTSC is believed to calibrate the features learned by LSGM, which can promote the ability of our model to extract high-level joint nodes representations. When initializing the joint nodes' features, most of the previous works connected three coordinates and initialized them directly as features. However, if there is no significant motion in one coordinate, the initialization of a single point cannot be effectively extracted. Instead, we choose to split it into three coordinates to obtain deeper and richer feature representations. As illustrated in Fig. 1, we use three Bidirectional LSGM to model the skeleton sequence on the X, Y and Z coordinates, respectively. Each bidirectional LSGM is accompanied by a temporal attention module. Finally, we stack the three output graph matrices that are treated as three feature maps and feed them to the spatial calibration module. In summary, the major contribution of this paper lies in two aspects:

1) Based on the graph convolution layer and LSTM, we propose a novel recurrent graph memory network LSGM to extract high-level temporal and spatial features simultaneously from the skeleton sequence.

2) We introduce a novel graph temporal-spatial calibration module which consists of a temporal attention module and a spatial calibration network to calibrate the features learned by LSGM, which enhances the ability of our network to capture discriminative features.

2. Related Works

2.1. Graph-based Neural Networks

Recently, there is an increasing interest on extending deep neural networks for graph-structure as graph-based models are robust in dealing with non-Euclidean data. Existing graph-based models mainly fall into two categories: recurrent graph neural network (GNN), graph convolutional neural network (GCN). Recurrent graph neural networks update nodes using the neighbouring information iteratively until a stable point is learned. Qi *et al.* [20] apply GNN to solve the task of detecting and recognizing human-to-object interactions in images and videos. Li *et al.* [14] exploit GNN to capture latent dependency between roles and predict a consistent structured output for situation. Different from GNN, GCN combines convolutional neural networks with graph. There are two mainstreams of GCN: spectral GCN and spatial GCN. Spectral GCN transforms

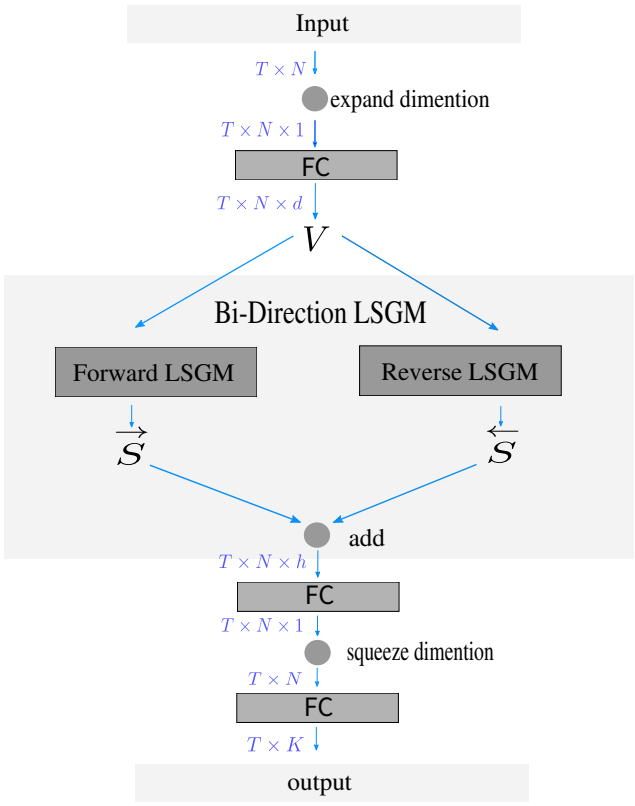

Figure 2. The flow chart of our Bi-Direction LSGM module, which is the part of the orange square in Fig. 1. Notice that here after expanding dimension we use one FC layer to act as a $F_{int}(\cdot)$ function to initialize the joint node features.

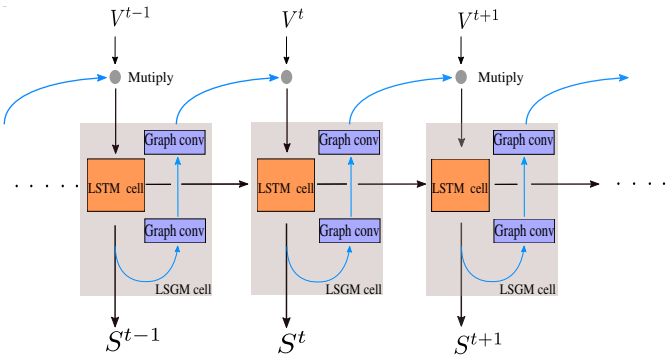

Figure 3. The internal structure of our proposed forward LSGM model. Each LSGM cell contains one LSTM cell and two graph convolution layer. We use the LSTM cell to update current hidden states, which are then fed into the GCN layer to capture the rich spatial information. Then the output of GCN layer is combined with the next graph input to feed into LSTM cell again.

graph signals on graph spectral domains and then apply spectral filters on spectral domains. Kipf *et al*. [9] introduce Spectral GCNs for semi-supervised classification on graph-structured data. For spatial GCNs, new features of each node were extracted using its neighboring information. Yan *et al*. [31] construct the CNN filters on the spatial domain, by limiting the application of each filter to the 1-neighbor of each node. Simonovsky *et al*. [25] formulate a convolution-like operation on graph signals performed in the spatial domain and apply it to point cloud classification..

2.2. Skeleton-based Action Recognition

By encoding joint positions in each frame to a feature vector and implementing graph convolution, many effective GCN-based frameworks are proposed to better model dynamic graph data (*e.g.*, skeleton sequence). Spatial-temporal GCN (ST-GCN) [31] is then developed to simultaneously learn spatial and temporal features of the skeleton sequence. Introducing the attention mechanism and combing it with the novel graph convolution LSTM, the Attention-enhanced Graph Convolution LSTM (AGC-LSTM) [24] also tries to learn discriminative temporal-spatial information. Furthermore, the Actional-Structural Graph Convolution Network (AS-GCN) [24] explores the latent dependency among joint nodes through Action-link and explores the high-order relationships of skeleton graph through Structural-link. With the guidance of semantics, Semantics-Guided Network (SGN) [17] is proposed to learn temporal-spatial features from the entire graph and exploit the global and local correlations among joints. Based on LSTM and GCN, most of the previous works aim to enhance the ability of the model to extract high-level spatial and temporal features by designing additional complex neural networks.

3. Methodology

Different from previous works, we aim to improve the ability of LSTM to extract spatial information by embedding the GCN layer in the LSTM cell. Furthermore, we introduce an attention module to extract keyframes and propose a spatial calibration module to calibrate the learned temporal-spatial features, which enhances the expressive ability of our network in capturing richer temporal-spatial features.

3.1. Graph Convolution Neural Network

Inspired by CNN, a new convolution mechanism has been proposed in recent years to better model the non-Euclidean data, which is termed as Graph Convolution Network (GCN). It is well known that the essence of convolution is weighted summed neighbor aggregation. By defining different approaches of weighted summation, GCN achieves great success in modeling skeleton sequence [31]. For skeleton-based action recognition, we carry out a similar implementation of graph convolution as in [9]. The natural connection between human skeleton joints is used to

construct the adjacency matrix *A*. In particular, the matrix *A* is a symmetric matrix as we treat the skeleton graph as an undirected graph:

$$A_{ij} = \begin{cases} 1 & \text{if joint } i \text{ connects to joint } j \text{ .} \\ 0 & \text{else} \end{cases}$$

Then the graph convolution can be implemented with the following formula:

$$f_{out} = \Lambda^{-\frac{1}{2}}(A+I)\Lambda^{-\frac{1}{2}} f_{in} W, \tag{1}$$

where $\Lambda^{ii} = \sum_j (A_{ij} + I_{ij})$ is used to implement the normalization of the feature vectors. I is the identity matrix that is introduced to add a self-loop for each node. W is a weight matrix and bias is omitted. f_{out} is the output of one GCN layer. After each graph convolution layer we add a *Relu* function to introduce nonlinear components. Then the Eq. 1 is transformed into

$$f_{out} = Relu\big(\Lambda^{-\frac{1}{2}}(A+I)\Lambda^{-\frac{1}{2}} f_{in} W\big), \tag{2}$$

which is also described as

$$f_{out} = G(f_{in}). \tag{3}$$

Practically, in our proposed LSGM we deploy two graph convolution layers, which we will describe the reason in Section 4.1. Our graph convolution implementation can be computed as

$$f_{out} = G^{(2)}(f_{in}), \tag{4}$$

By using Superscript 2 here we mean that we use 2 nested function $G(\cdot)$. Eq. 4 is also equivalent $f_{out} = G\big(G(f_{in})\big)$.

3.2. Long-Short Graph Memory network

As a variant of RNN, LSTM has been demonstrated to have a powerful ability to model long-term temporal dependencies. Various models based on LSTM are exploited to extract temporal features from the skeleton sequence. However, there is a limitation of LSTM on ignoring spatial dependencies for skeleton-based action recognition due to the fully connected operator. Therefore, we embed the graph convolution layer into the LSTM cell to capacitate it to extract spatial features, which is our LSGM cell. We use three LSGM to model the skeleton sequence at the X, Y and Z coordinate, separately. Here we take X-axis as an example to describe LSGM.

As illustrated in Fig. 2, for every input skeleton sequence with the shape of $T \times N$, we first use a function $F_{init} : \mathbb{R}^1 \rightarrow \mathbb{R}^d$ to initialize the joint vector $V \in \mathbb{R}^{T \times N \times d}$, which is then fed into Bidirectional LSGM (Bi-LSGM). As illustrated in Fig. 3, for a single direction LSGM, we divide V into T shares and feed them to LSGM in chronological order. At every time step t, we feed the current hidden state $S^t \in \mathbb{R}^{N \times h}$ into the graph convolution layer to capture the structural dependency among joints, where h denotes the hidden size of LSTM cell:

$$S_g^t = G^{(2)}(S^t), \tag{5}$$

where $G^{(2)}(\cdot)$ refers to the twice graph convolution function, which is equivalent to Eq. (4). Then we combine S_g^t with current graph to form a new graph matrix $V_g^{t+1} \in \mathbb{R}^{N \times d}$ by multiplying S_g^t by V^{t+1}:

$$V_g^{t+1} = S_g^t \odot V^t, \tag{6}$$

where $\odot$ denotes the element-wise multiplication.

After performing graph convolution twice, V_g^{t+1} contains rich structure information. Finally, we update graph hidden states by using ***lstm*** function:

$$(S^{t+1}, C^{t+1}) = lstm(V_g^t, S^t, C^t), \tag{7}$$

where C^t denotes the memory cell of LSTM at time step t. We then stack the hidden states of T frames to form $\overrightarrow{S} \in \mathbb{R}^{T \times N \times h}$. We build our model based on Bi-LSGM, then combine the forward hidden states $\overrightarrow{S}$ and the backward hidden states $\overleftarrow{S}$ together as

$$S = \overrightarrow{S} + \overleftarrow{S} \text{ .} \tag{8}$$

To calculate the feature map of LSGM, we compress the feature dimension and resize S via 2 FC layers as

$$O = FC\Big(FC\big(S\big)\Big) \in \mathbb{R}^{T \times K}. \tag{9}$$

3.3. Graph Temporal-Spatial Calibration

To calibrate the temporal-spatial features learned by LSGM, we introduce the GTSC module, which consists of a temporal attention module and a spatial calibration module. The attention module is very beneficial to extract temporal information and ignore the irrelative frames. For carrying out temporal attention, we use the same input as Bi-LSGM to calculate the attention weights F_a. As illustrated in Fig. 4, we add an FC layer and obtain $V_a \in \mathbb{R}^{T \times K}$. We first aggregate each row vector of V_a by the average pooling operation to produce $V_p \in \mathbb{R}^{T \times 1}$. Then, we duplicate it with K copies and get $V_d = (V_p, V_p, ..., V_p) \in \mathbb{R}^{T \times K}$:

$$V_d = duplicate\Big(pooling\Big(FC(V_{in})\Big)\Big), \tag{10}$$

where V_{in} denotes the input of the temporal attention module.

After pooling and duplication, we use a function $F_{encode} : \mathbb{R}^{T \times K} \rightarrow \mathbb{R}^{T \times \frac{K}{a}}$ to encode V_d. Then,

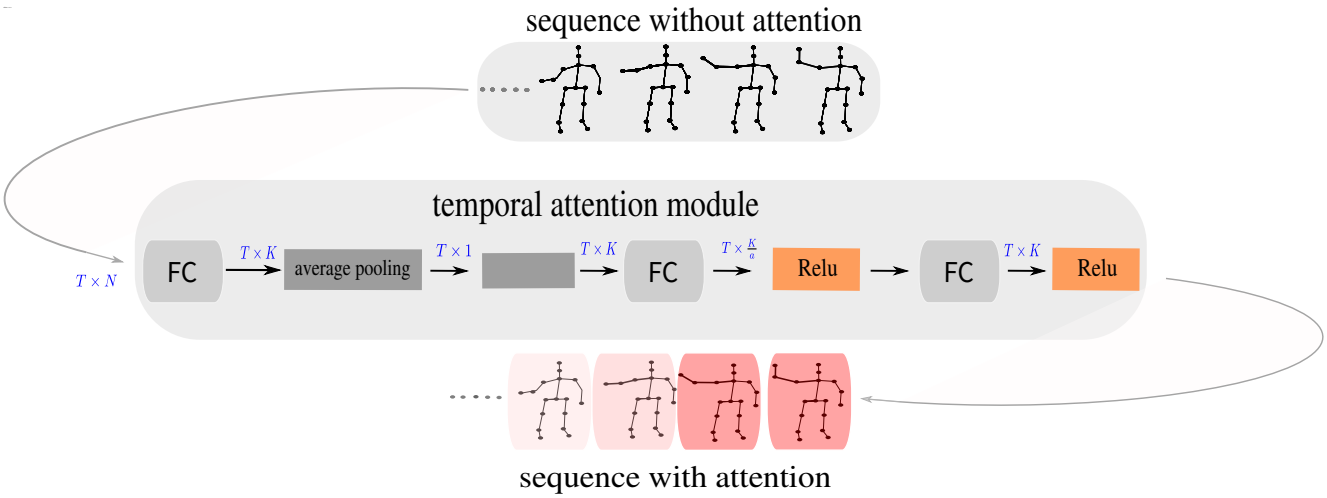

Figure 4. The temporal attention module. Darker frame indicates a higher level of attention after performing the attention mechanism. Here we use one FC layer and a Relu function to act as the F_{encode}, and use one FC layer and a sigmoid function to act as the F_{decode}.

$F_{decode} : \mathbb{R}^{T\times\frac{K}{a}} \rightarrow \mathbb{R}^{T\times K}$ is used to decode the features. Among them, F_{encode} is a dimension-reduction function and F_{decode} is a dimension-increasing function with a as a hyperparameter. This processing can be considered as denoising and excitation, respectively. Finally, the attention weight F_a can be computed as

$$F_a = F_{decode}\Big(F_{encode}(V_d)\Big) \tag{11}$$

where $F_{encode}(\cdot)$ can be replaced by any multilayer perceptron with Relu activation, and $F_{decode}(\cdot)$ can be replaced by any multilayer perceptron with Sigmoid activation in the last layer.

Using Eq. 9 and Eq. 11, we lead the features of temporal attention on the x-axis:

$$O_a = F_a \odot O. \tag{12}$$

As a result, O_a can effectively describe the temporal feature of the skeleton sequence on the x-axis. Similarly, we can use the same operation to calculate on the y-axis and the z-axis and stack them up to form the final $\widetilde{O}_a \in \mathbb{R}^{T\times K\times 3}$.

As illustrated in Fig. 3, $\widetilde{O}_a$ is the input of spatial calibration module of multiple residual blocks. CNNs have been demonstrated to be capable in capturing temporal and spatial information [8][32][10]. To further enhance the spatial and temporal features, we introduce spatial calibration to improve the feature representation. Actually, spatial calibration module can be replaced by any convolution networks (*e.g.*, DenseNet and ResNet) and in practice, only ResNet in use. The output of spatial calibration network is the input of a Softmax classifier for classification. The cross-entropy loss is adopted to measure the difference between the true class label and the prediction result.

4. Experiments

4.1. Experimental Settings

NTU RGB+D Dataset. NTU RGB+D data set is one of the largest data sets for skeleton-based action recognition, which includes 4 different modalities of data for each sample: RGB videos, depth map sequences, 3D skeleton data and infrared videos. It contains 56,880 action samples in total that are performed by 40 distinct subjects and categorized into 60 different classes. For evaluation, two protocols are recommended: Cross-Subject and Cross-View. we follow the preprocessing and the same cross-subject and cross-view protocol in [23]. In Cross-Subject, 40, 320 samples performed by 20 subjects are separated into the training set, and the rest belong to the test set. Cross-View assigns data according to camera views, where training and test sets have 37, 920 and 18, 960 samples,respectively.

SYSU-3D Dataset. The SYSU 3D Human-Object Interaction Set is captured by Kinect. It contains 12 actions performed by 40 subjects and each subject has 20 joints. In total, it has 480 sequences. This dataset is very challenging as there are lots of viewpoint variations. For evaluation, there are two standard protocols [33]. For setting-1, half of the samples are used for training and the rest for testing. For setting-2, half of the subjects are used for training and the rest for testing. We evaluate performance on setting-2, following the standard 30-fold cross-validation. Here we downsample the sequence in temporal as the maximum length of the sequence is high.

Implementation details. For all the datasets, we use the frames of the skeleton sequence of coordinate to generate the input matrices. For hyperparameter d, which denotes the length of nodes representation, we set it to 4. For LSGM

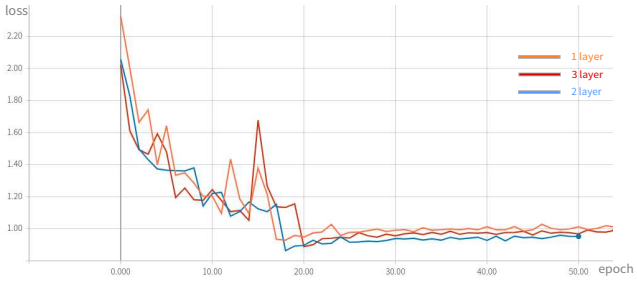

Figure 5. Test loss curve of our model when the number of GCN layers is 1, 2 and 3.

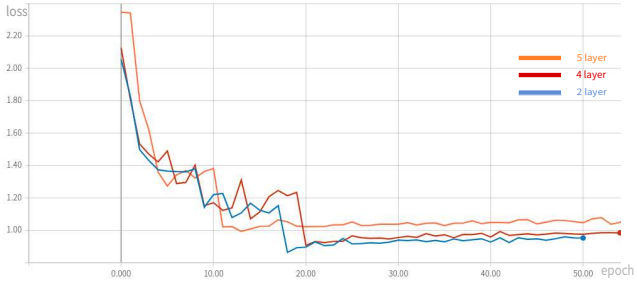

Figure 6. Test loss curve of our model when the number of GCN layers is 2, 4 and 5.

cell, we set the number of GCN layer to 2 and the number of hidden unites to 2×32 where 2 indicates Bidirectional LSGM. As illustrated in Fig. 4, for the introduced attention module, we set the hyperparameter a to 16 and use one FC layer and a nonlinear activation function to act as the F_{encode} and the F_{decode}. We replace the proposed spatial calibration nerwork with ResNet in practice. We trained our model using a stochastic gradient descent algorithm and set the learning rate, attenuation, and momentum to 0.1, 0, and 0.9, respectively. For the NTU and SYSU data sets, we set the batch size to 32 and 8, respectively. Training stop after 50 epochs for former data set and 100 epochs for latter. All experiments are performed based on the Keras2 with Tensorflow platform and use one NVIDIA Titans X Pascal card.

4.2. Experimental Results and Analysis

Model performance on different number of GCN layers. To investigate the performance of LSGM cell with different number of GCN layers, we conduct the experiments on the NTU RGB+D dataset. We examine the ability of our model in five cases based on Eq. 4.

$$f_{out} = G^{(n)}(f_{in}). \tag{13}$$

As shown in Table 1, LSGM with 2 GCN layers has achieved the best results of 84.71% on NTU CS dataset. We have plot the testing loss curves of the model in five cases, shown in Fig.5 and Fig.6. The result is consistent with recent research results which indicate that excessive GCN layer will cause over-smoothing and gradient vanishing problem[13]. As a result, most state-of-the-art GCN algorithms are no deeper than 3 or 4 layers. To settle this problem, [11] conducts an analysis and borrows concepts from CNNs to adapt residual or dense connections and dilated convolutions to GCN architecture, which boosts the performace of deep GCN. Considering that, we may embed deeper GCN with residual or dense connections in our LSGM cell to extract deeper structural features in the future work. In the following experiments, we set the layer of GCN to 2 to evaluate our model.

Table 1. Comparison of accuracy on the NTU datset when the number of GCN layer is 1, 2, 3, 4, 5 .

number of GCN layer	NTU.CS
1	83.9%
2	84.71%
3	84.4%
4	84.5%
5	83.83%

LSGM with different length of node representation. In addition to the number of GCN layer, the size of node representation is another significant factor in our model. As depicted in section 3.2, we use $F_{init} : \mathbb{R}^1 \rightarrow \mathbb{R}^d$ to initialize the joint vector $V \in \mathbb{R}^{T \times N \times d}$. Through experiments, we find that the parameter d, which denotes the size of node representation, matters in our method. As shown in Table 2, our model perform increasingly better as d decreases. The reason should be that, when transforming node representations into the input of spatial calibration module in Eq. 9, we use only two full connected layers to compress learned feature vector, which may result in information loss. With smaller d, two simple fully connected layers can effectively learn useful node representations and reduce information loss when compressing. In the following experiments, we evaluate our model by setting the hyperparameter d to 4.

Ablation study. To analyze the effectiveness of each individual component of our model, we conduct extensive experiments on Cross-Subject benchmark of the NTU data set.

LSGM vs. LSTM. To verify that LSGM has a greater advantage in dynamic graph data processing than LSTM, we use a baseline network architecture LSTM+STGC to model the skeleton sequence. In this case, we replace our LSGM with traditional LSTM, which means all joint nodes will not have a $\mathbb{R}^d$ representation vector and at every time step t, we will feed $V^t \in \mathbb{R}^{N \times 1}$ into LSTM cell where N is the number of joint nodes. Seen from Table 3, model based on LSGM outperform the LSTM-based model on NTU data set. Actually, the major difference between the baseline

Table 2. Comparison of accuracy on the NTU dataset when different hyperparameter d we choose.

d	NTU.CS
4	84.71%
8	84.12%
16	83.78%
32	83.73%

Table 3. Comparison of different methods on NTU data set

Methods	NTU.CS
LSTM + STGC	83.00%
LSGM + STGC	84.71%
LSGM + STGC(no attention)	83.2%

model and our LSGM+STGC is the joint nodes representation and the graph convolution layer between recurrent units. We believe that by introducing graph convolution, LSGM is better able to extract temporal and spatial information simultaneously than LSTM.

Attention vs. No Attention. To validate the necessity of using tempoaral attention operation, we build a LSGM+GTSC network without temporal attention, which means in Eq. 12 we directly let $O_a = O$ and will not calculate attention weight F_a. Seen from Table 3, the accuracy of LSGM+GTSC without temporal attention has dropped down a lot, which indicates that bringing temporal attention will make our model robustness to skeleton sequence.

4.3. Performance Comparison

NTU RGB+D dataset. On NTU dataset, we compare the performance of our proposed model against several previous state-of-the-art approaches in Table 4. The compared methods include ST-GCN[31], SR-TSL[17], VA-LSTM[33], Spatial Temporal LSTM with Trust Gates (ST-LSTM+TG)[15], Two-stream RNNs[28], GCA-LSTM[16], Memory Attention Network[30]. All of the above models

Table 4. Skeleton based action recognition performance on NTU-RGB+D datasets. We report the accuracies on both the cross-subject (X-Sub) and cross-view (X-View) benchmarks.

Methods	NTU.CS	NTU.CV
ST-LSTM + TG[15]	69.20%	77.70%
GCA-LSTM[16]	76.10%	84.00%
VA-LSTM[33]	79.40%	87.60%
ST-GCN[31]	81.5%	88.3%
MAN[30]	83.01%	90.66%
SR-TSL[17]	84.8%	92.4%
LSGM+GTSC(ours)	84.71%	91.74%

Table 5. Skeleton based action recognition performance on SYSU datasets. We report the accuracies on the recommended setting-2 protocal.

Methods	Setting-2
ST-LSTM + TG[15]	76.80%
VA-LSTM[33]	77.5%
SR-TSL[17]	81.9%
LSGM+GTSC(ours)	85.8%

can be divided into three categories: CNN-based model, LSTM-based model and GCN-based model. Our model belongs to the model of integrated GCN and LSTM. Seen from Table 4, our LSGM+GTSC is able to outperform previous state-of-the-art approaches on this dataset.

SYSU 3D dataset. On SYSU 3D dataset, following the standard 30-fold cross-validation protocal on setting-2, we compare our model with the state-of-the-art methods, including ST-LSTM + TG, VA-LSTM and SR-TSL. Experiments indicate that our model outperforms the state-of-the-art methods.

5. Conclusion

In this paper, an end-to-end framework named LSGM+GTSC is proposed to enhance the temporal-spatial features for skeleton-based action recognition. We construct a novel recurrent unit termed as LSGM, which has a stronger ability to simultaneously extract the temporal and spatial features, by embedding GCN layers to traditional LSTM cell. In addition, we propose GTSC to calibrate the output features of LSGM to capture high-level nodes representations. The extensive experiments demonstrate the effectiveness of our model, which consistently performs the best on two challenging datasets: NTU RGB+D dataset and SYSU 3D dataset. In our future work, we will try different variants of GCN layers embedded in LSGM, which may help to extract the latent dependency among joint nodes.

6. Acknowledgements

This work is supported by Shenzhen Science and Technology Program KQTD2016112515134654 and National Key R&D Plan (2017YFC0821102).

References

[1] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Comput. Surv.*, 43:16:1–16:43, 2011.

[2] Z. Cao, T. Simon, S. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1611.08050, 2016.

[3] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations

using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[4] C. Gallicchio and A. Micheli. Graph echo state networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2010.

[5] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *CoRR*, abs/1704.01212, 2017.

[6] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734 vol. 2, July 2005.

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li. Large-scale video classification with convolutional neural networks. pages 1725–1732, 2014.

[8] Q. Ke, M. Bennamoun, S. An, F. A. Sohel, and F. Boussaïd. A new representation of skeleton sequences for 3d action recognition. *CoRR*, abs/1703.03492, 2017.

[9] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[10] C. Li, Q. Zhong, D. Xie, and S. Pu. Co-occurrence feature learning from skeleton data for action recognition and detection with hierarchical aggregation. *CoRR*, abs/1804.06055, 2018.

[11] G. Li, M. Müller, A. K. Thabet, and B. Ghanem. Can gcns go as deep as cnns? *CoRR*, abs/1904.03751, 2019.

[12] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian. Actional-structural graph convolutional networks for skeleton-based action recognition. *CoRR*, abs/1904.12659, 2019.

[13] Q. Li, Z. Han, and X. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *CoRR*, abs/1801.07606, 2018.

[14] R. Li, M. Tapaswi, R. Liao, J. Jia, R. Urtasun, and S. Fidler. Situation recognition with graph neural networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 4183–4192, 2017.

[15] J. Liu, A. Shahroudy, D. Xu, and G. Wang. Spatio-temporal LSTM with trust gates for 3d human action recognition. *CoRR*, abs/1607.07043, 2016.

[16] J. Liu, G. Wang, P. Hu, L. Duan, and A. C. Kot. Global context-aware attention lstm networks for 3d action recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3671–3680, July 2017.

[17] Y. Liu, Y. Li, S. You, and F. Lu. Semantic guided single image reflection removal. *CoRR*, abs/1907.11912, 2019.

[18] R. Poppe. A survey on vision-based human action recognition. *Image Vision Comput.*, 28:976–990, 2010.

[19] M. O. R. Prates, P. H. C. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi. Learning to solve np-complete problems - A graph neural network for the decision TSP. *CoRR*, abs/1809.02721, 2018.

[20] S. Qi, W. Wang, B. Jia, J. Shen, and S.-C. Zhu. Learning human-object interactions by graph parsing neural networks. *ArXiv*, abs/1808.07962, 2018.

[21] S. M. R, S. K, S. A. S, S. G. Jacob, and M. S. Approaches and applications of virtual reality and gesture recognition: A review. 8(4):1–18, 2017.

[22] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, Jan 2009.

[23] A. Shahroudy, J. Liu, T. Ng, and G. Wang. NTU RGB+D: A large scale dataset for 3d human activity analysis. *CoRR*, abs/1604.02808, 2016.

[24] C. Si, W. Chen, W. Wang, L. Wang, and T. Tan. An attention enhanced graph convolutional LSTM network for skeleton-based action recognition. *CoRR*, abs/1902.09130, 2019.

[25] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 29–38, 2017.

[26] 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. 2016.

[27] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, Dec 2015.

[28] H. Wang and L. Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 3633–3642, 2017.

[29] P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera. Rgb-d-based human motion recognition with deep learning: A survey. *Computer Vision and Image Understanding*, 171:118 – 139, 2018.

[30] C. Xie, C. Li, B. Zhang, C. Chen, J. Han, and J. Liu. Memory attention networks for skeleton-based action recognition. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 1639–1645, 2018.

[31] S. Yan, Y. Xiong, and D. Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. *CoRR*, abs/1801.07455, 2018.

[32] Yong Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118, June 2015.

[33] 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 *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2136–2145, 2017.

[34] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, Feb 2012.