Skeleton-based human activity understanding

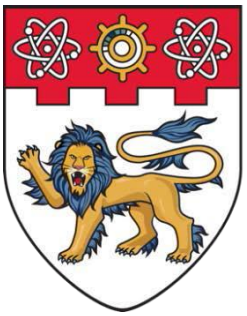
Liu, Jun

2019

Liu, J. (2019). Skeleton-based human activity understanding. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/104427

https://doi.org/10.32657/10220/49510



NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Skeleton-Based Human Activity Understanding

Liu Jun

SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING

2019

Skeleton-Based Human Activity Understanding

Liu Jun

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

21, Jan, 2019

. . . . . .

Date

Liu Jun

. . . . . .

Liu Jun

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

21 Jan 2019
. . . . . .
Date

. . . . . .
Alex Kot

Authorship Attribution Statement

This thesis contains material from 5 paper(s) published in the following peer-reviewed journals (or conferences) where I was the first and/or corresponding author.

Chapter 3 is published as:
[a] Jun Liu, Amir Shahroudy, Dong Xu, Gang Wang, “Spatio-temporal LSTM with trust gates for 3D human action recognition,” European Conference on Computer Vision (ECCV), 2016.
[b] Jun Liu, Amir Shahroudy, Dong Xu, Alex C. Kot, Gang Wang, “Skeleton-Based Action Recognition Using Spatio-Temporal LSTM Network with Trust Gates,” IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2018.

The contributions of the co-authors are as follows:

- I proposed the initial method. The method was improved by discussions with Prof. Alex C. Kot, Dr. Amir Shahroudy, Dr. Gang Wang, and Prof. Dong Xu.
- I conducted the experiments. Dr. Amir Shahroudy also helped in the experiments.
- I prepared the manuscript drafts. The manuscripts were revised by Dr. Amir Shahroudy, Dr. Gang Wang, Prof. Dong Xu, and Prof. Alex C. Kot.

Chapter 4 is published as:
[c] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, Alex C. Kot, “Global context-aware attention LSTM networks for 3D action recognition,” IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[d] Jun Liu, Gang Wang, Ling-Yu Duan, Kamila Abdiyeva, Alex C. Kot, “Skeleton-Based Human Action Recognition with Global Context-Aware Attention LSTM Networks,” IEEE Transactions on Image Processing (TIP), 2018.

The contributions of the co-authors are as follows:

- I designed the initial method.
- Prof. Alex C. Kot and I had regular discussions, and improved the design.
- Dr. Gang Wang, Prof. Ling-Yu Duan, Mr. Ping Hu, and Ms. Kamila Abdiyeva also provided suggestions.
- I conducted the experiments.
- I prepared the manuscript drafts. Prof. Alex C. Kot, Dr. Gang Wang, Prof. Ling-Yu Duan, Mr. Ping Hu, Ms. Kamila Abdiyeva revised the manuscript.

Chapter 5 is published as:

[e] Jun Liu, Amir Shahroudy, Gang Wang, Ling-Yu Duan, Alex C. Kot, "SSNet: Scale selection network for online 3D action prediction," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

The contributions of the co-authors are as follows:

- I proposed the initial idea, and had regular discussions with Prof. Alex C. Kot on the design.
- Dr. Amir Shahroudy, Dr. Gang Wang, and Prof. Ling-Yu Duan also gave suggestions on this work.
- I implemented the method.
- I prepared the manuscript drafts. The manuscripts were revised by Prof. Alex C. Kot, Dr. Amir Shahroudy, Dr. Gang Wang, and Prof. Ling-Yu Duan.

21, Jan, 2019 Liu Jun

. . . . . .

Date

. . . . . .

Liu Jun

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Alex C. Kot, for the support of my Ph.D study. This research was not possible without his direction, advice, and encouragement. Second, I would like to acknowledge Dr. Gang Wang, my former supervisor, for his advice on my work. Third, I appreciate the help from the staffs of Rapid-Rich Object Search (ROSE) Lab, who supported my research by providing the excellent research environment and facilities. Lastly, I would like to thank the lab-mates for the collaboration and friendship.

Abstract

Human activity understanding is an important research problem due to its relevance to a wide range of applications. Recently, 3D skeleton-based activity analysis becomes popular due to its succinctness, robustness, and view-invariant representation. In this thesis, we focus on human activity understanding in 3D skeleton sequences.

Recent works attempted to utilize recurrent neural networks (RNNs) and long short-term memory (LSTM) networks to model the temporal dependencies between the 3D positional configurations of human body joints for better analysis of human activities in the 3D skeletal data. As the first work of this thesis, we apply recurrent analysis to spatial domain as well as temporal domain to better analyze the hidden sources of action-related information within the human skeleton sequences in both of these domains simultaneously. Based on the pictorial structure of Kinect's skeletal data, an effective tree-structure based traversal framework is also proposed. In order to deal with the noise in the skeletal data, a new gating mechanism within LSTM module is introduced, with which the network can learn the reliability of the sequential data and accordingly adjust the effect of the input data on the updating procedure of the long-term context representation stored in the unit's memory cell. Moreover, we introduce a novel multi-modal feature fusion strategy within the LSTM unit in this thesis. The comprehensive experimental results on seven challenging benchmark datasets for human action recognition demonstrate the effectiveness of the proposed method.

In skeleton-based action recognition, not all skeletal joints are informative for activity analysis, and the irrelevant joints often bring noise which can degrade the performance. Therefore, we need to pay more attention to the informative ones. However, the original LSTM network does not have explicit attention ability. In our second piece of work, we propose a new class of LSTM network, global context-aware attention LSTM, for skeleton-based action recognition, which is capable of selectively focusing on the informative joints in each frame by using a global context memory cell. To further improve the attention capability, we also introduce a recurrent attention mechanism, with which the attention performance of our network can be enhanced progressively. Besides, a two-stream framework, which leverages coarse-grained attention and fine-grained attention, is also introduced. The proposed method achieves state-of-the-art performance on five challenging datasets for skeleton-based action recognition.

The aforementioned two works focus on action recognition in well-segmented skeleton sequences, in which each sequence includes one action sample and we need to recognize its class. In the third work, we focus on online action prediction in untrimmed streaming skeleton data, in which each sequence contains multiple action samples and we need to recognize the class label of the current ongoing activity when only a part of it is observed. A dilated convolutional network is introduced to model the motion dynamics in temporal dimension via a sliding window over the temporal axis for online action prediction. As there are significant temporal scale variations in the observed part of the ongoing action at different time steps, a novel window scale selection method is proposed, which makes our network focus on the performed part of the ongoing action and suppress the possible incoming interference from the previous actions. An activation sharing scheme is also proposed to handle the overlapping computations among the adjacent time steps, which enables our framework to run more efficiently. Moreover, to enhance the performance of our framework for action

prediction with the skeletal input data, a hierarchy of dilated tree convolutions are also designed to learn the multi-level structured semantic representations over the skeleton joints at each frame. The proposed approach is evaluated on four challenging datasets. The extensive experiments demonstrate the effectiveness of the proposed method for skeleton-based online action prediction.

Table of contents

List of figures **xiii**

List of tables **xxi**

1 Introduction **1**

1.1 Background 1

1.2 Challenges and Motivations 3

1.3 Proposed Activity Analysis Models 6

1.4 Thesis Contributions 7

1.5 Organization of Thesis 8

2 Literature Review **9**

2.1 Related Methods 9

2.1.1 Skeleton-Based Action Recognition with Hand-crafted Features 9

2.1.2 Skeleton-Based Action Recognition with Deep Learning Models 12

2.1.3 Online Action Prediction in Untrimmed Sequences 16

2.2 Related Datasets 18

3 Skeleton-Based Action Recognition Using Spatio-Temporal LSTM Network with Trust Gates **21**

3.1 Introduction 21

3.2 Spatio-Temporal Recurrent Networks 23

3.2.1 Temporal Modeling with LSTM 24
3.2.2 Spatio-Temporal LSTM 24
3.2.3 Tree-Structure Based Traversal 27
3.2.4 Spatio-Temporal LSTM with Trust Gates 29
3.2.5 Feature Fusion within ST-LSTM Unit 31
3.2.6 Learning the Classifier 35
3.3 Experiments 37
3.3.1 Implementation Details 38
3.3.2 Experiments on the NTU RGB+D Dataset 39
3.3.3 Experiments on the UT-Kinect Dataset 42
3.3.4 Experiments on the SBU Interaction Dataset 44
3.3.5 Experiments on the SYSU-3D Dataset 45
3.3.6 Experiments on the ChaLearn Gesture Dataset 46
3.3.7 Experiments on the MSR Action3D Dataset 47
3.3.8 Experiments on the Berkeley MHAD Dataset 48
3.3.9 Visualization of Trust Gates 48
3.3.10 Evaluation of Different Spatial Joint Sequence Models 50
3.3.11 Evaluation of Temporal Average, LSTM and ST-LSTM 52
3.3.12 Evaluation of the Last-to-first Link Scheme 53
3.4 Chapter Summary 54

4 Skeleton-Based Action Recognition with Global Context-Aware Attention LSTM Networks 57

4.1 Introduction 57
4.2 GCA-LSTM Network 61
4.2.1 Spatio-Temporal LSTM 61
4.2.2 Global Context-Aware Attention LSTM 63

4.2.3 Training the Network 67
4.3 Two-stream GCA-LSTM Network 69
4.4 Experiments 72
4.4.1 Experiments on the NTU RGB+D Dataset 73
4.4.2 Experiments on the SYSU-3D Dataset 75
4.4.3 Experiments on the UT-Kinect Dataset 76
4.4.4 Experiments on the SBU-Kinect Interaction Dataset 77
4.4.5 Experiments on the Berkeley MHAD Dataset 77
4.4.6 Evaluation of Attention Iteration Numbers 78
4.4.7 Evaluation of Parameter Sharing Schemes 79
4.4.8 Evaluation of Training Methods 81
4.4.9 Evaluation of Initialization Methods and Attention Designs . . 82
4.4.10 Visualizations 84
4.5 Chapter Summary 85

5 Skeleton-Based Online Action Prediction Using Scale Selection Network 87
5.1 Introduction 87
5.2 The Proposed Method 91
5.2.1 Temporal Modeling with Convolutional Layers 93
5.2.2 Scale Selection 96
5.2.3 Details of the Main Structure 99
5.2.4 Activation Sharing Scheme 100
5.2.5 Multi-level Structured Skeleton Representations 101
5.2.6 Objective Function 105
5.3 Experiments 106
5.3.1 Implementation Details 108
5.3.2 Experiments on the OAD Dataset 109

5.3.3 Experiments on the ChaLearn Gesture Dataset 112
5.3.4 Experiments on the PKUMMD Dataset 113
5.3.5 Experiments on the G3D Dataset 114
5.3.6 Evaluation of Skeleton Representations 114
5.3.7 Evaluation of Distance Regression 115
5.3.8 Evaluation of Network Configurations 117
5.3.9 Frame-level Classification Accuracies 119
5.4 Chapter Summary 119

6 Conclusion and Future Work **121**
6.1 Conclusion 121
6.2 Future Work 122
6.2.1 Multi-Modal Feature Fusion for Activity Analysis 122
6.2.2 One-Shot Action Recognition in Skeleton Data 123
6.2.3 Online Action Detection in Untrimmed Skeleton Sequences . 123

Author's Publications **125**

References **127**

List of figures

1.1 A sequence of human skeleton data. 2

1.2 Illustration of the depth sensors. (a) Microsoft Kinect v1. (b) Microsoft Kinect v2. (c) Asus Xtion. 2

1.3 Illustration of temporal context dependence over frames (denoted as green lines) and spatial context dependence among different joints in each single frame (denoted as red lines). 3

1.4 Examples of the noisy skeletons from the NTU RGB+D dataset [1]. . 4

1.5 In the actions "kicking" and "shaking hands", the most informative joints are the foot joints and hand joints, respectively. The most informative joints are labeled as blue circles. Image samples are from the NTU RGB+D dataset [1]. 5

1.6 Online activity analysis over a streaming skeleton sequence that contains multiple action samples. 5

2.1 Illustration of the actionlet ensemble framework [2] ©2012 IEEE. . . 11

2.2 Illustration of the Hierarchical RNN network [3] ©2015 IEEE. 13

2.3 Illustration of the LSTM network with a differential gating scheme [4] ©2015 IEEE. 13

2.4 Illustration of the deep LSTM network proposed by Zhu *et al.* [5] ©2015 AAAI. 14

2.5 Illustration of the Part-aware LSTM network [1] ©2016 IEEE. 15
2.6 Illustration of the soft regression model for action prediction [6] ©2016 Springer. 17
2.7 First row: sample frames from the NTU RGB+D dataset [1]. Second row: the RGB, RGB+skeleton, depth, depth+skeleton, and IR modalities of a sample frame. ©2016 IEEE. 19

3.1 Illustration of the spatio-temporal LSTM network. In temporal dimension, the corresponding body joints are fed over the frames. In spatial dimension, the skeletal joints in each frame are fed as a sequence. Each unit receives the hidden representation of the previous joints and the same joint from previous frames. 25
3.2 Illustration of the proposed ST-LSTM with one unit. 26
3.3 (a) The skeleton of the human body. In the simple joint chain model, the joint visiting order is 1-2-3-...-16. (b) The skeleton is transformed to a tree structure. (c) The tree traversal scheme. The tree structure can be unfolded to a chain with the traversal scheme, and the joint visiting order is 1-2-3-2-4-5-6-5-4-2-7-8-9-8-7-2-1-10-11-12-13-12-11-10-14-15-16-15-14-10-1. 27
3.4 Illustration of the deep tree-structured ST-LSTM network. For clarity, some arrows are omitted in this figure. The hidden representation of the first ST-LSTM layer is fed to the second ST-LSTM layer as its input. The second ST-LSTM layer's hidden representation is fed to the softmax layer for classification. 29
3.5 Illustration of the proposed ST-LSTM with trust gate. 32
3.6 Illustration of the proposed structure for feature fusion inside the ST-LSTM unit. 32

3.7 Recognition accuracy per class on the NTU RGB+D dataset 40

3.8 (a) Performance comparison of our approach using different values of neuron size (d) on the NTU RGB+D dataset (X-subject). (b) Performance comparison of our method using different λ values on the NTU RGB+D dataset (X-subject). The blue line represents our results when different λ values are used for trust gate, while the red dashed line indicates the performance of our method when trust gate is not added. 42

3.9 Experimental results of our method by early stopping the network evolution at different time steps. 43

3.10 Visualization of the trust gate's behavior when inputting noisy data. (a) $j_{3'}$ is a noisy joint position, and j_3 is the corresponding rectified joint location. In the histogram, the blue bar indicates the magnitude of trust gate when inputting the noisy joint $j_{3'}$. The red bar indicates the magnitude of the corresponding trust gate when $j_{3'}$ is rectified to j_3. (b) Visualization of the difference between the trust gate calculated when the noise is imposed at the step (j_N, t_N) and that calculated when inputting the original data. 49

4.1 Skeleton-based human action recognition with the Global Context-Aware Attention LSTM network. The first LSTM layer encodes the skeleton sequence and generates an initial global context representation for the action sequence. The second layer performs attention over the inputs by using the global context memory cell to achieve an attention representation for the sequence. Then the attention representation is used back to refine the global context. Multiple attention iterations are performed to refine the global context memory progressively. Finally, the refined global context information is utilized for classification. 60

4.2 Illustration of our GCA-LSTM network. Some arrows are omitted for clarity. 63

4.3 Illustration of the two network training methods. (a) Directly train the whole network. (b) Stepwise optimize the network parameters. In this figure, the global context memory cell $\mathbb{F}^{(n)}$ is unfolded over the attention iterations. The training step #n corresponds to the n-th attention iteration. The black and red arrows denote the forward and backward passes, respectively. Some passes, such as those between the two ST-LSTM layers, are omitted for clarity. Better viewed in colour. 68

4.4 Illustration of the two-stream GCA-LSTM network, which incorporates fine-grained (joint-level) attention and coarse-grained (body part-level) attention. To perform coarse-grained attention, the joints in a skeleton are divided into five body parts, and all the joints from the same body part share a same informative score. In the second ST-LSTM layer for coarse-grained attention, we only show two body parts at each frame, and other body parts are omitted for clarity. . . . 70

4.5 Convergence curves of the GCA-LSTM network with two attention iterations by respectively using *stepwise training* (in red) and *direct training* (in green) on the NTU RGB+D dataset. Better viewed in colour. 82

4.6 Examples of qualitative results on the NTU RGB+D dataset. Three actions (*taking a selfie*, *pointing to something*, and *kicking other person*) are illustrated. The informativeness scores of two attention iterations are visualized. Four frames are shown for each iteration. The circle size indicates the magnitude of the informativeness score for the corresponding joint in a frame. For clarity, the joints with tiny informativeness scores are not shown. 83

4.7 Visualization of the average informativeness gates for all testing samples. The size of the circle around each joint indicates the magnitude of the corresponding informativeness score. 85

5.1 Figure (a) illustrates an untrimmed streaming sequence that contains multiple action instances. We need to recognize the current ongoing action at each time step when only a part (*e.g.*, 10%) of it is performed. Figure (b) depicts our SSNet for online action prediction. At time t, only a part of the action *waving hand* is observed. Our SSNet selects the convolutional Layer #2 rather than #3 for prediction, as the perception window of #2 mainly covers the performed part of current action, while #3 involves too many frames from the previous action which can interfere the prediction at time step t. 88

5.2 Illustration of the proposed SSNet for action prediction over the temporal axis. The solid lines denote the SSNet links activated at current step t, and the dashed lines indicate the links activated at other time steps. Our SSNet has 14 1-D convolutional layers. Here we only show 3 layers for clarity. At each time step, SSNet predicts the class ($\hat{c}_t$) of the ongoing action, and also estimates the temporal distance ($\hat{s}_t$) to current action's start point. Calculation details of $\hat{c}_t$ and $\hat{s}_t$ are shown in Fig. 5.3. 92

5.3 Details of our SSNet that jointly predicts the class label $\hat{c}_t$ and regresses the start point's distance $\hat{s}_t$ for the current ongoing action **at time *t***. If the regressed result $\hat{s}_{t-1}$ at the previous time step $(t-1)$ indicates that Layer #3 corresponds to the most *proper* window scale (*i.e.*, $l_t^p = 3$), then our network will use Layers #1-3 for class prediction, while the activations from the layers above #3 are dropped (marked with *cross* in the figure). In this figure, we only show a subset of convolutional nodes of our SSNet, and other ones in the hierarchical structure (depicted as the solid lines in Fig. 5.2) are omitted for clarity. 94

5.4 Illustration of the dilated convolution layer used in our network. At each position, the 1-D convolutional filter covers a time range (labeled as a red box), and only the two boundary nodes (corresponding to two time steps) in the covered range are used, while the other nodes between these two nodes are not used by the dilated convolutional operation for this position. 94

5.5 (a) The skeleton joints of the human body form a tree structure. We set the head joint (joint 1) as the root node, and the height of the tree in this figure is 8. (b) Illustration of the convolution with triangular filters sliding over the tree structure. The green and the blue triangles indicate the convolutions with two different filter sizes. 101

5.6 Illustration of the hierarchy of dilated tree convolutions that learns the multi-level structured representations over the input skeleton joints (labeled in red) at each frame. The solid arrows denote the dilated tree convolutions with triangular filters. In our method, 3 dilated tree convolutional layers are used to cover the input skeleton tree with height 8, while in this figure, we only show 2 layers that cover the tree with height 4 for clarity. Note that the bottom of this figure shows a full binary tree, while the human skeleton only has a subset of the nodes of a full binary tree. Therefore, in implementation, the convolutional operations only need to be performed on a subset of the nodes. The channel number of the input skeleton is 3, namely, the 3D coordinates (x, y, z) of each joint. (Best viewed in color) 103

5.7 Action prediction results on the OAD dataset. 110

5.8 Action prediction results on the ChaLearn Gesutre dataset. 112

5.9 Action prediction results on the PKUMMD dataset. 114

5.10 Action prediction results on the G3D dataset. 115

5.11 Examples of the start point regression results on the four datasets. The leftmost point of the green bar is the ground truth start point position of the current ongoing action. The leftmost point of each red bar (ending at $p\%$) is the regressed start point position when $p\%$ of the action instance is observed. 117

5.12 Action prediction results with different γ values on the OAD dataset. . 118

5.13 Detailed network architecture configurations of SSNet (for action prediction at the time step t). The distance regression is performed based on the top convolutional layer (together with the layers below it with the skip connections), which has a large perception window. The class prediction is performed based on the selected *proper* layer (together with the layers below it), which is selected based on the estimated window scale. 120

List of tables

3.1 Experimental results on the NTU RGB+D Dataset. "X-S" and "X-V" denote X-Subject and X-View, respectively. 39

3.2 Evaluation of different feature fusion strategies on the NTU RGB+D dataset. "Geometric + Visual (1)" indicates the early fusion scheme. "Geometric + Visual (2)" indicates the late fusion scheme. "Geometric $\bigoplus$ Visual" means our newly proposed feature fusion scheme within the ST-LSTM unit. 42

3.3 Experimental results on the UT-Kinect dataset (LOOCV protocol [7]) 43

3.4 Results on the UT-Kinect dataset (half-vs-half protocol [8]) 43

3.5 Evaluation of our approach for feature fusion on the UT-Kinect dataset (LOOCV protocol [7]). 44

3.6 Experimental results on the SBU Interaction dataset 45

3.7 Experimental results on the SYSU-3D dataset 46

3.8 Experimental results on the ChaLearn Gesture dataset 47

3.9 Experimental results on the MSR Action3D dataset 48

3.10 Experimental results on the Berkeley MHAD dataset 48

3.11 Performance comparison of different spatial sequence models 51

3.12 Comparison of different start joints (root nodes) of the tree traversal on the NTU RGB+D dataset. "S.D." denotes standard deviation. . . . 52

3.13 Performance comparison of Temporal Average, LSTM, and our proposed ST-LSTM 53
3.14 Evaluation of the last-to-first link in our proposed network 54

4.1 Experimental results on the NTU RGB+D dataset. 74
4.2 Performance of the two-stream GCA-LSTM network on the NTU RGB+D dataset. 75
4.3 Experimental results on the SYSU-3D dataset. 75
4.4 Performance of the two-stream GCA-LSTM network on the SYSU-3D dataset. 76
4.5 Experimental results on the UT-Kinect dataset. 76
4.6 Experimental results on the SBU-Kinect Interaction dataset. 77
4.7 Experimental results on the Berkeley MHAD dataset 78
4.8 Evaluation of robustness against the input noise. Gaussian noise $\mathcal{N}(0, \sigma^2)$ is added to the 3D coordinates of the skeletal joints. 78
4.9 Performance comparison of different attention iteration numbers (N). 79
4.10 Performance comparison of different parameter sharing schemes. . . 80
4.11 Performance comparison of different methods of initializing the global context memory cell. 82

5.1 Details of the main structure of SSNet. Refer to Fig. 5.13 for the detailed architecture configurations of SSNet. 99
5.2 Details of the hierarchy of dilated tree convolutions (corresponding to Fig. 5.6). 105
5.3 Number of parameters and computational efficiency of our SSNet when using different skeleton representations within it. 109

5.4 Action prediction accuracies on the OAD dataset. Note that in the last row, *SSNet-GT* is an "ideal" baseline, in which the *ground truth (GT)* scales are used for action prediction. Our SSNet, which performs prediction with the regressed scales, is even comparable to *SSNet-GT*. Refer to Fig. 5.7 for more results. 110

5.5 Action prediction accuracies on the ChaLearn Gesture dataset. Refer to Fig. 5.8 for more results. 112

5.6 Action prediction accuracies on the PKUMMD dataset. Refer to Fig. 5.9 for more results. 113

5.7 Action prediction accuracies on the G3D dataset. Refer to Fig. 5.10 for more results. 114

5.8 Action prediction accuracies (%) of SSNet with different skeleton representations. 116

5.9 Start point regression performance (*SL-Score*). SSNet^C indicates that we use the coordinate concatenation representation for the network. . 116

5.10 Start point regression errors. 117

5.11 Evaluation of different configurations of the proposed network on the OAD dataset. 118

5.12 Frame-level classification accuracies. FSNet(best) denotes the FSNet that gives the best results among all FSNets. 119

Chapter 1

Introduction

1.1 Background

Human activity understanding is amongst the most challenging problems in computer vision. It has attracted a lot of research attention due to its wide range of application in security surveillance, human-machine interaction, robotics, patient monitoring, and so on.

Many existing works in this domain focus on using RGB videos as input [9, 10]. However, it is often challenging to achieve effective and efficient human activity understanding when we use RGB sequences as input. The first challenge comes from the large amount of data to be processed, as each video frame generally consists of millions of pixel values. The second challenge is that the 3D structural information of the observed activities, which is often important for reliable activity analysis, is not provided in the 2D frame-based input. Besides, the illumination variations, viewpoint changes, and cluttered backgrounds are also challenging factors of this task.

To mitigate the aforementioned issues, some recent works [11, 12, 4, 13–16] started to investigate human activity understanding based on a high-level succinct

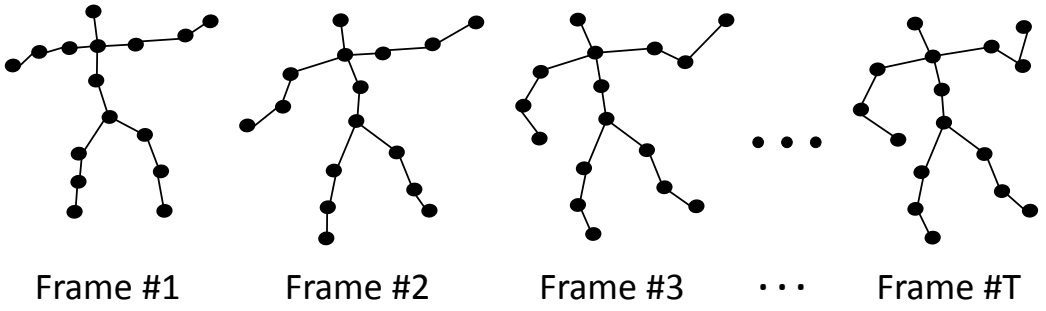


Fig. 1.1 A sequence of human skeleton data.

representation: 3D skeleton data, i.e., performing activity analysis by using the 3D coordinates of the major body joints in each frame as input, as illustrated in Fig. 1.1.

The research on skeleton-based activity analysis is motivated by the studies and observations in biology [17] that demonstrate that the skeleton data is informative enough for representing human behaviors, even without appearance information [18].

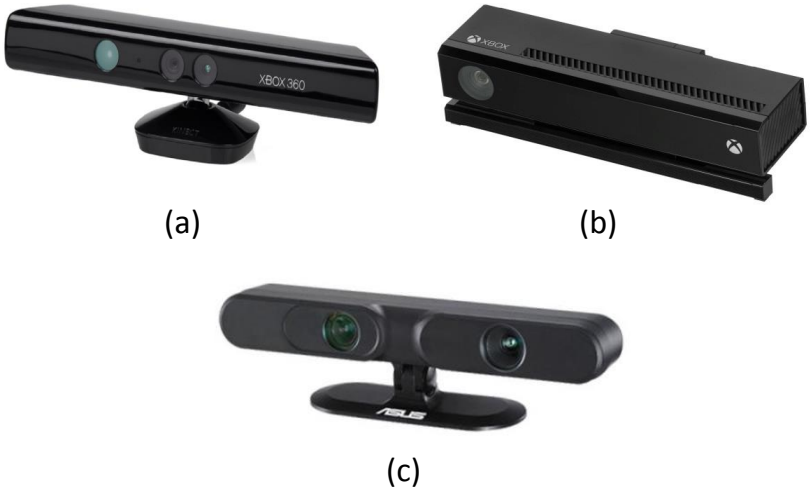


Fig. 1.2 Illustration of the depth sensors. (a) Microsoft Kinect v1. (b) Microsoft Kinect v2. (c) Asus Xtion.

Human activities are naturally performed in 3D space, thus 3D skeleton data is suitable for representing human actions [19]. Specifically, the 3D skeleton data can be easily and effectively acquired in real-time with the low-cost depth sensors [20], such as the Microsoft Kinect and the Asus Xtion (see Fig. 1.2). Moreover, such a

representation is robust against variations in viewpoints, illumination, clothing textures, and cluttered backgrounds [3, 21, 6]. As a result, human activity analysis with 3D skeleton data becomes popular.[1]

1.2 Challenges and Motivations

In this thesis, we mainly focus on the task of 3D skeleton-based human activity analysis. Below we introduce some of the important aspects that need to be considered when handling this task and the motivations of the proposed methods in this thesis.

(1) Spatio-temporal context dependence.

As illustrated in Fig. 1.3, in the sequence of 3D skeleton data, there are strong dependency relations among different frames, which reveal the temporal dynamics of the human motions. Besides, in each singe frame, there is also high dependence among different body joints, which shows the spatial posture pattern.

Intuitively, to understand the human behaviors in the 3D skeleton sequence well, the analysis of the context dependency information in both spatial dimension and temporal dimension is important.

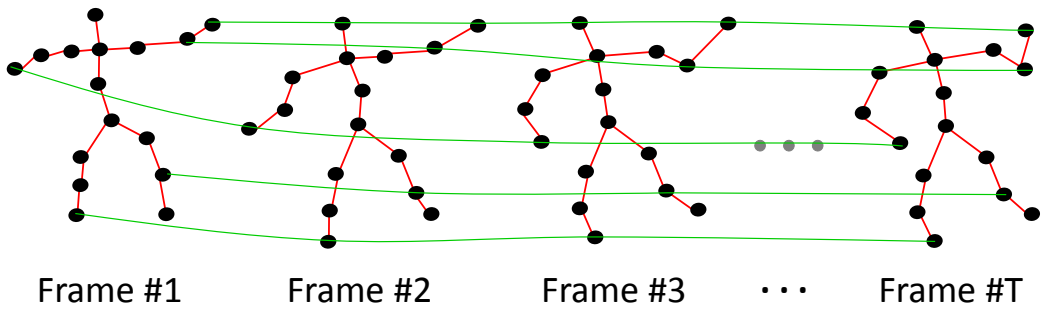


Fig. 1.3 Illustration of temporal context dependence over frames (denoted as green lines) and spatial context dependence among different joints in each single frame (denoted as red lines).

[1] The details of deriving the RGB-D data and the skeleton data can be found in these papers: [22, 23].

This motivates us to design network models to model the context dependencies in both the spatial and temporal dimensions for human activity analysis (Chapter 3 and Chapter 5).

(2) Noisy skeleton data.

In 3D skeleton-based activity analysis, the input is a sequence of skeleton data consisting of the 3D locations of the human body joints. However, these joint locations that are provided by the depth sensors using a pose estimation algorithm [23] are not always accurate, as shown in Fig. 1.4. The noisy input data will affect the activity analysis performance.

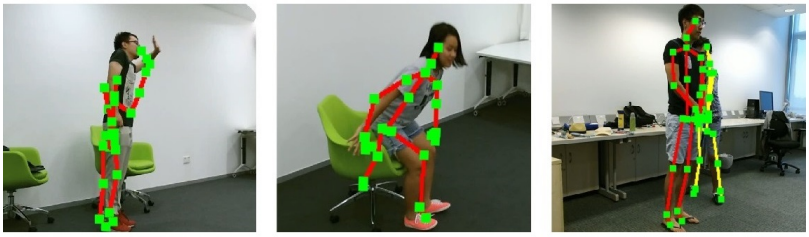


Fig. 1.4 Examples of the noisy skeletons from the NTU RGB+D dataset [1].

This motivates us to design a gating method to handle the noisy skeleton data by learning the reliability of the sequential data and accordingly adjusting the effect of the input (Chapter 3).

(3) Identifying informative joints.

Not all joints in the 3D skeleton sequence are informative for activity analysis. For example, the movements of the foot joints are very informative for the action "kicking", while the hand joints' motions are not. As illustrated in Fig.1.5, different activity sequences often have different informative joints. Besides, in the same sequence, the informativeness degree of a joint may also vary over the frames.

Motivated by this observation, we investigate to identify the informative joints in each frame of the skeleton sequence and emphasize their features, and meanwhile suppress the features of the irrelevant joints, because the irrelevant joints often con-

Fig. 1.5 In the actions "kicking" and "shaking hands", the most informative joints are the foot joints and hand joints, respectively. The most informative joints are labeled as blue circles. Image samples are from the NTU RGB+D dataset [1].

tribute little for activity analysis, and even bring in noise that corrupts the performance (Chapter 4).

(4) Temporal scale selection.

In activity analysis, specifically in online activity analysis over a streaming skeleton sequence that contains multiple action samples, a common approach is to use a sliding window over the temporal dimension to segment a sub-sequence for analysis at each temporal step, as shown in Fig. 1.6. In this setting, the used temporal window scale (size) will affect the effectiveness and efficiency of the activity analysis at each time step. This indicates choosing a "proper" window scale is important.

Fig. 1.6 Online activity analysis over a streaming skeleton sequence that contains multiple action samples.

However, it is challenging to determine the temporal window scale, because at different progress levels of the actions, the "proper" scale may change. For example, at the early stages of an action sample, we need to use a smaller scale because the larger ones will cover irrelevant information from the previous action samples, while at the

later stages, it is better to use a larger window scale to cover more of the observed parts of the current action.

This motivates us to propose an approach to estimate the proper window scale at each step, and adopt the estimated proper scale for analysis at each step (Chapter 5).

1.3 Proposed Activity Analysis Models

In this thesis, we present three network models for skeleton-based activity analysis, each of which addresses one or several important aspects that are discussed in Section 1.2. Below we briefly introduce these three network models.

The first model (Chapter 3) is based on the recurrent networks for sequential data analysis. We propose a spatio-temporal recurrent network to model the context dependencies in both the spatial and temporal dimensions of the skeleton sequence for action recognition. Inspired by the graphical structure of the human skeleton, we also propose a powerful traversal method to represent the spatial dependency information within the skeleton. Due to the unreliability of the 3D input data, a new gating mechanism, called trust gate, is also proposed to improve the robustness of the network against noise and occlusion. This model achieves state-of-the-art performance on seven challenging benchmark datasets for 3D human action recognition.

The second model (Chapter 4) extends the original LSTM network to achieve a Global Context-Aware Attention LSTM (GCA-LSTM) network for skeleton-based action recognition, which has strong capability in selectively focusing on the informative joints in each frame of the skeleton sequence with the assistance of global contextual information. A recurrent attention mechanism is also designed to strengthen the attention capability of the GCA-LSTM network. Besides, a two-stream framework, which leverages coarse-grained attention and fine-grained attention, is also introduced.

This network model yields state-of-the-art performance on five challenging datasets for 3D action recognition.

The third model (Chapter 5) is designed for handling online action prediction in untrimmed streaming sequences. Dilated convolutions are applied to model the motion dynamics in temporal dimension via a sliding window over the time axis. A hierarchy of dilated tree convolutions are also designed to learn the multi-level spatial dependency structure over the skeleton joints at each frame. As there are significant temporal scale variations of the observed part of the ongoing action at different progress levels, a window scale selection scheme is also designed to make the network focus on the performed part of the ongoing action and try to suppress the noise from the previous actions at each time step. This model is evaluated on four challenging datasets and achieves state-of-the-art performance of online activity analysis.

1.4 Thesis Contributions

The main contributions of this thesis are summarized as follows:

(1) We introduce two different methods to model the spatio-temporal context dependency information for the skeleton sequence. First, we extend the LSTM network that models the context dependency in the temporal domain to a spatio-temporal LSTM network that analyzes the hidden sources of action-related information within the input data over both domains concurrently (Chapter 3). Second, we propose a hierarchy of dilated tree convolutions to model the spatial dependency structure of the skeleton joints and apply a stack of dilated convolution layers to model the motion dynamics in the temporal dimension (Chapter 5). These two methods are shown to be effective in representing the spatio-temporal dependency information in the skeleton data, and both achieve superior performance for human activity analysis in our experiments.

(2) We propose a trust gating mechanism to handle the noisy skeleton data that improves the performance of the network model when dealing with the noisy input data (Chapter 3). The extensive experiments also demonstrate the efficacy of the trust gating mechanism for skeleton-based activity analysis.

(3) We design a method to selectively focus on the informative joints in each frame of the skeleton sequence with the assistance of global contextual information. We also design a recurrent attention mechanism to improve the attention performance progressively (Chapter 4). Our experiments show that by selectively focusing on the informative joints, the activity analysis performance can be improved significantly.

(4) A temporal scale selection scheme is also proposed to handle the temporal scale variations for online activity analysis. This scheme enables the network model to focus on the performed part of the ongoing action and try to suppress the interference from the previous actions at each time step (Chapter 5). The effectiveness of this scheme is also demonstrated in our experiments.

1.5 Organization of Thesis

The remainder of this thesis is organized as follows. In Chapter 2, we review the related works in the literature. In Chapter 3, we introduce the proposed spatio-temporal LSTM network with trust gates for skeleton-based action recognition. In Chapter 4, we introduce the Global Context-aware Attention LSTM network for skeleton-based action recognition. In Chapter 5, the scale selection network is proposed for skeleton-based online action prediction. Finally, in Chapter 6, we conclude this thesis and show our future research directions.

Chapter 2

Literature Review

In this chapter, we review the existing methods and benchmark datasets that are relevant to our proposed models for skeleton-based activity analysis.

2.1 Related Methods

We first review the methods that extract hand-crafted features for skeleton-based action recognition. We then introduce the existing deep learning-based approaches for skeleton-based action recognition. Finally, we review the methods that are related to online action prediction in untrimmed sequences.

2.1.1 Skeleton-Based Action Recognition with Hand-crafted Features

With the advent of cheap and easy-to-use depth sensors, such as Kinect [20], 3D skeleton-based human action recognition becomes very popular [24, 25], and different methods that extract hand-crafted features have been proposed for skeleton-based action recognition [26–31, 16, 32–40].

Xia *et al.* [7] proposed to model the temporal dynamics in action sequences with the Hidden Markov models (HMMs). They also introduced a compact representation of postures by calculating the histograms of 3D joint locations for action recognition. The static postures and dynamics of the motion patterns were represented via eigenjoints by Yang and Tian [29]. A Naive-Bayes-Nearest-Neighbor (NBNN) classifier learning approach was also used in [29]. Chaudhry *et al.* [41] encoded the skeleton sequences to spatio-temporal hierarchical models and used a set of linear dynamical systems (LDSs) to learn the dynamic structures. Vemulapalli *et al.* [42] represented the skeleton configurations and action patterns as points and curves in a Lie group, and a support vector machine (SVM) classifier was used to classify the actions.

Evangelidis *et al.* [43] introduced a local skeleton representation to encode the relative position of the joint quadruples. They also learned a Gaussian mixture model (GMM) over the Fisher kernel representation of the skeletal quads features. An angular skeletal representation over the tree-structured set of joints was proposed in [44], which calculated the similarity of these features over temporal dimension to build the global representation of the action samples and fed them to a linear SVM for final classification. A skeleton-based dictionary learning method using geometry constraint and group sparsity was proposed by Luo *et al.* [26]. A temporal pyramid matching method was utilized to handle the temporal alignment problem.

Wang *et al.* [2, 45] introduced an actionlet ensemble representation to model the actions meanwhile capturing the intra-class variances, as shown in Fig. 2.1. A temporal pattern representation called Fourier Temporal Pyramid (FTP) was also introduced in [2, 45] to handle temporal sequence misalignment and noise.

Chen *et al.* [46] designed a part-based 5D feature vector to explore the relevant joints of body parts in skeleton sequences. Koniusz *et al.* [47] introduced tensor representations for capturing the high-order relationships among body joints. Wang *et al.* [48] proposed a graph-based motion representation in conjunction with a SPGK-

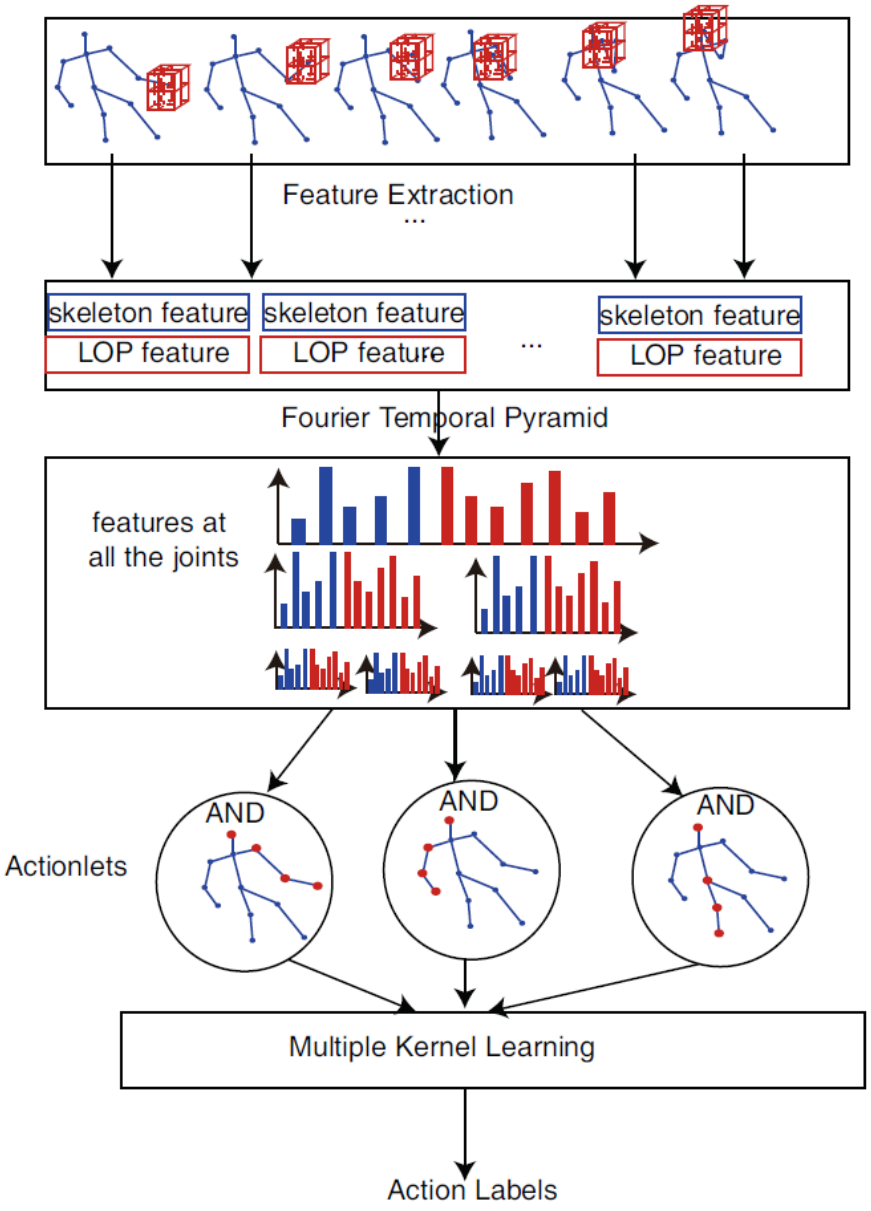


Fig. 2.1 Illustration of the actionlet ensemble framework [2] ©2012 IEEE.

kernel SVM for skeleton-based activity recognition. Zanfir *et al.* [49] developed a moving pose framework together with a modified k nearest neighbor (k-NN) classifier for low-latency action recognition.

2.1.2 Skeleton-Based Action Recognition with Deep Learning Models

Very recently, deep learning, especially recurrent neural network (RNN), based approaches have shown their strength in skeleton-based action recognition [3, 50, 5, 1, 51–54]. Our proposed spatio-temporal LSTM network and Global Context-aware Attention LSTM network are based on the LSTM model that is an extension of RNN. In this section, we review the RNN and LSTM based methods that are relevant to our proposed models.

Du *et al.* [3] proposed a Hierarchical RNN network by utilizing multiple bidirectional RNNs in a novel hierarchical fashion, as illustrated in Fig. 2.2. The human skeletal structure was divided to five major joint groups. Then each group was fed into the corresponding bidirectional RNN. The outputs of the RNNs were concatenated to represent the upper body and lower body, then each was further fed into another set of RNNs. By concatenating the outputs of two RNNs, the global body representation was obtained, which was fed to the next RNN layer. Finally, a softmax classifier was used in [3] to perform action classification.

Veeriah *et al.* [4] proposed to add a new gating mechanism for LSTM to model the derivatives of the memory states and explore the salient action patterns, as illustrated in Fig. 2.3. In this method, all of the input features were concatenated at each frame and were fed to the differential LSTM at each step.

Zhu *et al.* [5] introduced a regularization term to the objective function of the LSTM network to push the entire framework towards learning co-occurrence relations

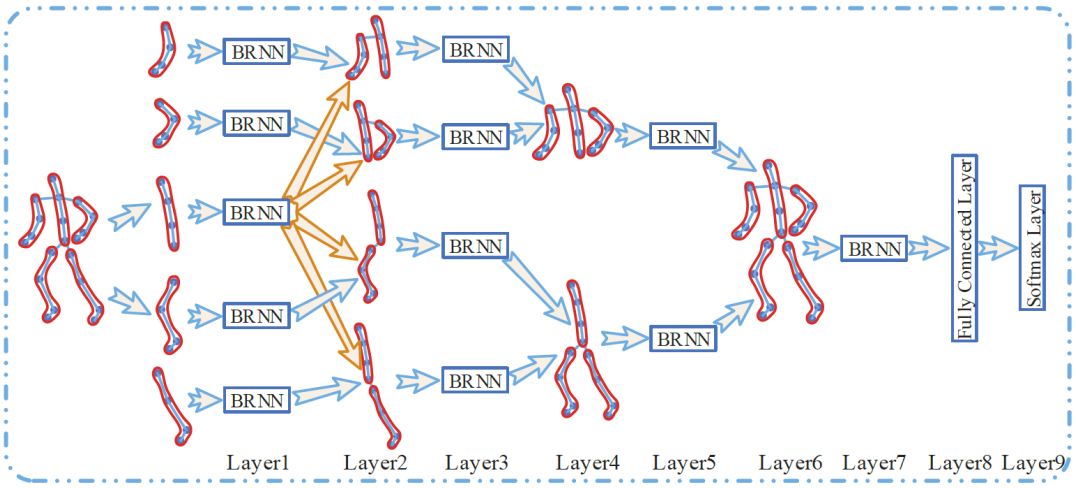


Fig. 2.2 Illustration of the Hierarchical RNN network [3] ©2015 IEEE.

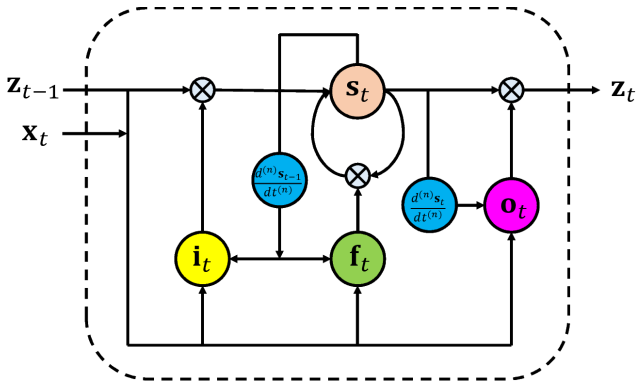


Fig. 2.3 Illustration of the LSTM network with a differential gating scheme [4] ©2015 IEEE.

among the joints for action recognition, as shown in Fig. 2.4. An internal dropout [55] technique within the LSTM unit was also introduced in [5].

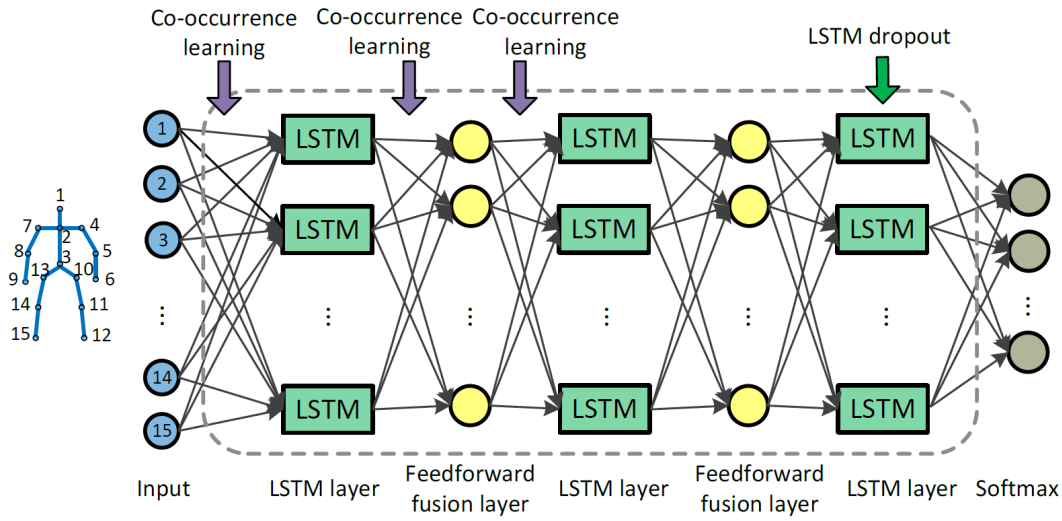


Fig. 2.4 Illustration of the deep LSTM network proposed by Zhu *et al.* [5] ©2015 AAAI.

Shahroudy *et al.* [1] proposed to split the LSTM's memory cell to sub-cells to push the network towards learning the context representations for each body part separately, as shown in Fig. 2.5. The output of the network was learned by concatenating the multiple memory sub-cells.

Harvey and Pal [56] adopted an encoder-decoder recurrent network to reconstruct the skeleton sequence and perform action classification at the same time. Their model showed promising results on motion capture sequences.

Mahasseni and Todorovic [57] proposed to use LSTM to encode a skeleton sequence as a feature vector. At each step, the input of the LSTM consists of the concatenation of the skeletal joints' 3D locations in a frame. They further constructed a feature manifold by using a set of encoded feature vectors. Finally, the manifold was used to assist and regularize the supervised learning of another LSTM for action recognition.

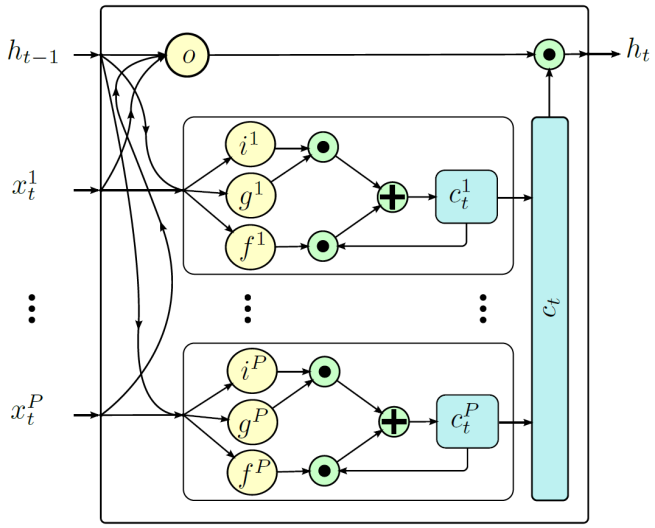


Fig. 2.5 Illustration of the Part-aware LSTM network [1] ©2016 IEEE.

Different from the aforementioned works, our proposed spatio-temporal LSTM (ST-LSTM) network (in Chapter 3) does not simply concatenate the joint-based input features to build the body-level feature representation. Instead, the context dependencies between the skeletal joints are explicitly modeled by applying recurrent analysis over temporal and spatial dimensions concurrently. Furthermore, a novel trust gate is introduced to make our ST-LSTM network more reliable against the noisy input data.

Besides, the aforementioned RNN/LSTM based approaches do not explicitly consider the informativeness of each skeletal joint with regarding to the global action sequence. Differently, our proposed GCA-LSTM network (in Chapter 4) utilizes the global context information to perform attention over all the evolution steps of LSTM to selectively emphasize the informative joints in each frame, and thereby generates an attention representation for the sequence, which can be used to improve the classification performance. Furthermore, a recurrent attention mechanism is proposed to iteratively optimize the attention performance.

2.1.3 Online Action Prediction in Untrimmed Sequences

In Section 2.1.1 and Section 2.1.2, we have reviewed the methods for skeleton-based **action recognition** in well-segmented sequences, i.e., each skeleton sequence contains a *whole observation* of *one action sample*, and we need to recognize the class of the action in this sequence. The proposed skeleton-based **online action prediction** method (introduced in Section 5) takes one step forward in dealing with the untrimmed continuous sequences, i.e., the input is an untrimmed streaming sequence that contains *multiple action instances*, and we need to early recognize the class of the current ongoing action from its *observed part* at each time step. In this section, we review the works on action prediction and action analysis with untrimmed sequences, which are relevant to our online action prediction method.

Action Prediction. Action prediction is to recognize the class label of an ongoing activity when only a part of it is perceived. Predicting (recognizing) an action before it gets fully performed has attracted a lot of research attention recently [58–64].

Cao *et al.* [60] formulated the prediction task as a posterior-maximization problem, and applied sparse coding for action prediction. Ryoo *et al.* [59] represented each action as an integral histogram of spatio-temporal features. They also developed a recognition methodology called dynamic bag-of-words (DBoW) for activity prediction. Li *et al.* [65] designed a predictive accumulative function. In their method, the human activities are represented as a temporal composition of constituent actionlets. Kong *et al.* [58] proposed a discriminative multi-scale model for early action recognition. Ke *et al.* [63] extracted deep features from the optical flow images for two-person interaction prediction.

Hu *et al.* [6] explored to incorporate 3D skeleton information for real-time action prediction in the *well-segmented* sequences, *i.e.*, each sequence includes only one action. They introduced a soft regression strategy for action prediction, as shown in

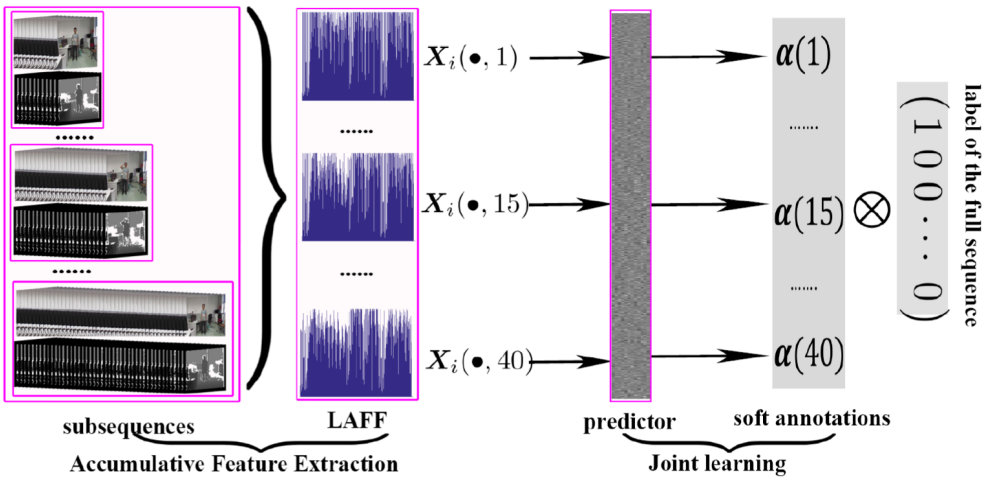


Fig. 2.6 Illustration of the soft regression model for action prediction [6] ©2016 Springer.

Fig. 2.6. An accumulative frame feature was also designed to make their method work efficiently. However, their framework is not suitable for online action prediction in the *untrimmed* continuous skeleton sequence that contains multiple action instances.

Action Analysis with Untrimmed Sequences. Beside the online action prediction task, the problem of temporal action detection [66–73, 66, 74–77] also copes with untrimmed videos. Several methods attempted online detection [78], while most of the action detection approaches are developed for handling offline mode that conducts detection after observing the whole long sequence [69, 79, 80].

The online action prediction task addressed in this thesis (in Chapter 5) is different from action detection, as action detection mainly addresses accurate spatio-temporal segmentation, while action prediction focuses more on predicting the class of the current ongoing action timely from its observed part, even when only a small ratio of it is performed.

Sliding window-based design [49, 72, 81, 82] and action proposals [71] have been adopted for action detection. Zanfir *et al.* [49] used a sliding window with one fixed

scale (obtained by cross validation) for action detection. Shou *et al.* [83] adopted multi-scale windows for action detection via multi-stage networks.

Differently, in the online action prediction task, determining the scale of the temporal window is challenging owing to the scale variations of the observed part of the ongoing action. Also, rather than using one fixed scale [49] or multi-scale multi-round scans [83, 84], a novel SSNet for online prediction is proposed in this thesis (in Chapter 5), which is supervised to choose the proper window scale for prediction at each time step. Moreover, the redundant computations within the sliding window-based design are efficiently shared over different steps in our approach.

2.2 Related Datasets

With the advent of the Microsoft Kinect, a decent number of datasets have been collected for the research on 3D skeleton-based human activity analysis. Below we introduce the benchmark datasets that are used for experimental evaluations in this thesis. Seven datasets for action recognition are introduced, namely, NTU RGB+D [1], UT-Kinect [7], SBU Interaction [39], SYSU-3D [85], ChaLearn Gesture (segmented) [86], MSR Action3D [87], and Berkeley MHAD [88]. Four datasets for online action prediction are introduced, namely, OAD [78], ChaLearn Gesture (untrimmed) [86], PKUMMD [89], and G3D [90].

NTU RGB+D dataset [1] was captured with Kinect (v2). It is currently the largest publicly available dataset for depth-based action recognition, which contains more than 56,000 video sequences and 4 million video frames. The samples in this dataset were collected from 80 distinct viewpoints. A total of 60 action classes (including daily actions, medical conditions, and pair actions) were performed by 40 different persons aged between 10 and 35. This dataset is very challenging due to the large intra-class and viewpoint variations. With a large number of samples, this dataset is

highly suitable for deep learning based activity analysis. The parameters learned on this dataset can also be used to initialize the models for smaller datasets to improve and speed up the training process of the network. RGB videos, depth videos, IR videos, and 3D skeleton sequences are provided in this dataset, as shown in Fig. 2.7.

Fig. 2.7 First row: sample frames from the NTU RGB+D dataset [1]. Second row: the RGB, RGB+skeleton, depth, depth+skeleton, and IR modalities of a sample frame. ©2016 IEEE.

UT-Kinect dataset [7] was captured with a stationary Kinect sensor. It contains 10 action classes. Each action was performed twice by every subject. The 3D locations of 20 skeletal joints are provided. The significant intra-class and viewpoint variations make this dataset very challenging.

SBU Interaction dataset [39] was collected with Kinect. It contains 8 classes of two-person interactions, and includes 282 skeleton sequences with 6822 frames. Each body skeleton consists of 15 joints. This dataset is challenging because of (1) the relatively low accuracies of the coordinates of skeletal joints recorded by Kinect, and (2) complicated interactions between two persons in many action sequences.

SYSU-3D dataset [85] contains 480 sequences and was collected with Kinect. In this dataset, 12 different activities were performed by 40 persons. The 3D coordinates of 20 joints are provided in this dataset.

ChaLearn Gesture (segmented) dataset [86] consists of 23 hours of videos captured with Kinect. A total of 20 Italian gestures were performed by 27 different subjects. Each skeleton in this dataset has 20 joints.

MSR Action3D dataset [87] is widely used for depth-based action recognition. It contains a total of 10 subjects and 20 actions. Each action was performed by the same subject two or three times. Each frame in this dataset contains 20 skeletal joints.

Berkeley MHAD dataset [88] was collected by using a motion capture network of sensors. It contains 659 sequences and about 82 minutes of recording time. Eleven action classes were performed by twelve subjects. The 3D coordinates of 35 skeletal joints are provided in each frame.

OAD dataset [78] was collected with a Kinect (v2) sensor that captures color images, depth images, and human skeletal data simultaneously. This dataset contains 10 daily actions that were performed by different actors. A total of 59 long video sequences are provided in this dataset, and each long video contains multiple action instances.

ChaLearn Gesture (untrimmed) dataset [86] is a large-scale dataset for human activity (body language) analysis. It contains 20 action classes. This dataset is very challenging, as the body motions of many action classes are very similar in the untrimmed long sequences.

PKUMMD dataset [89] was collected for continuous skeleton-based human activity understanding. This dataset contains 1076 untrimmed long video sequences and 51 action classes. Multi-modality data sources, including RGB, depth, IR, and 3D skeleton, are provided in this dataset.

G3D dataset [90] consists of untrimmed video sequences for the gaming activities. It contains 20 different gaming actions. These actions are further grouped into 7 major categories.

Chapter 3

Skeleton-Based Action Recognition Using Spatio-Temporal LSTM Network with Trust Gates

In our first work, we focus on human action recognition in 3D skeleton sequences, and a new network model, spatio-temporal LSTM (ST-LSTM), is introduced for this task. Besides, we design a novel gating framework for the ST-LSTM network to handle the noisy skeleton data.

3.1 Introduction

Recurrent neural networks (RNNs) which can handle the sequential data with variable lengths [91, 92], have shown their strength in language modeling [93–95], image captioning [96, 97], video analysis [98–105, 78, 106, 107], and RGB-based activity recognition [108–113]. Applications of these networks have also shown promising achievements in skeleton-based action recognition [3, 4, 1].

In the current skeleton-based action recognition literature, RNNs are mainly used to model the long-term context information across the temporal dimension by representing motion-based dynamics. However, there are often strong dependency relations among the skeletal joints in spatial domain also, and the spatial dependency structure is usually discriminative for action classification.

To model the dynamics and dependency relations in both temporal and spatial domains, we propose a spatio-temporal long short-term memory (ST-LSTM) network in this chapter. In our ST-LSTM network, each joint can receive context information from its stored data from previous frames and also from the neighboring joints at the same time frame to represent its incoming spatio-temporal context. Feeding a simple chain of joints to a sequence learner limits the performance of the network, as the human skeletal joints are not semantically arranged as a chain. Instead, the adjacency configuration of the joints in the skeletal data can be better represented by a tree structure. Consequently, we propose a traversal procedure by following the tree structure of the skeleton to exploit the kinematic relationship among the body joints for better modeling spatial dependencies.

Since the 3D positions of skeletal joints provided by depth sensors are not always very accurate, we further introduce a new gating framework, so called "trust gate", for our ST-LSTM network to analyze the reliability of the input data at each spatio-temporal step. The proposed trust gate gives better insight to the ST-LSTM network about when and how to update, forget, or remember the internal memory content as the representation of the long-term context information.

In addition, we introduce a feature fusion method within the ST-LSTM unit to better exploit the multi-modal features extracted for each joint.

We summarize the main contributions of this chapter as follows. (1) A novel spatio-temporal LSTM (ST-LSTM) network for skeleton-based action recognition is designed. (2) A tree traversal technique is proposed to feed the structured human

skeletal data into a sequential LSTM network. (3) The functionality of the ST-LSTM framework is further extended by adding the proposed "trust gate". (4) A multi-modal feature fusion strategy within the ST-LSTM unit is introduced. (5) The proposed method achieves state-of-the-art performance on seven benchmark datasets.

The remainder of this chapter is organized as follows. In section 3.2, we introduce our end-to-end trainable spatio-temporal recurrent neural network for action recognition. The experiments are presented in section 3.3. Finally, the chapter is concluded in section 3.4.

3.2 Spatio-Temporal Recurrent Networks

In a generic skeleton-based action recognition problem, the input observations are limited to the 3D locations of the major body joints at each frame. Recurrent neural networks have been successfully applied to this problem recently [3, 5, 1]. LSTM networks [114] are among the most successful extensions of recurrent neural networks. A gating mechanism controlling the contents of an internal memory cell is adopted by the LSTM model to learn a better and more complex representation of long-term dependencies in the input sequential data. Consequently, LSTM networks are very suitable for feature learning over time series data (such as human skeletal sequences over time).

We will briefly review the original LSTM model in this section, and then introduce our ST-LSTM network and the tree-structure based traversal approach. We will also introduce a new gating mechanism for ST-LSTM to handle the noisy measurements in the input data for better action recognition. Finally, an internal feature fusion strategy for ST-LSTM will be proposed.

3.2.1 Temporal Modeling with LSTM

In the standard LSTM model, each recurrent unit contains an input gate i_t, a forget gate f_t, an output gate o_t, and an internal memory cell state c_t, together with a hidden state h_t. The input gate i_t controls the contributions of the newly arrived input data at time step t for updating the memory cell, while the forget gate f_t determines how much the contents of the previous state (c_{t-1}) contribute to deriving the current state (c_t). The output gate o_t learns how the output of the LSTM unit at current time step should be derived from the current state of the internal memory cell. These gates and states can be obtained as follows:

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ u_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(M \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right) \tag{3.1}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{3.2}$$

$$h_t = o_t \odot \tanh(c_t) \tag{3.3}$$

where x_t is the input at time step t, u_t is the modulated input, $\odot$ denotes the element-wise product, and $M : \mathbb{R}^{D+d} \rightarrow \mathbb{R}^{4d}$ is an affine transformation. d is the size of the internal memory cell, and D is the dimension of x_t.

3.2.2 Spatio-Temporal LSTM

RNNs have already shown their strengths in modeling the complex dynamics of human activities as time series data, and achieved promising performance in skeleton-based human action recognition [3, 5, 4, 1]. In the existing literature, RNNs are mainly utilized in temporal domain to discover the discriminative dynamics and

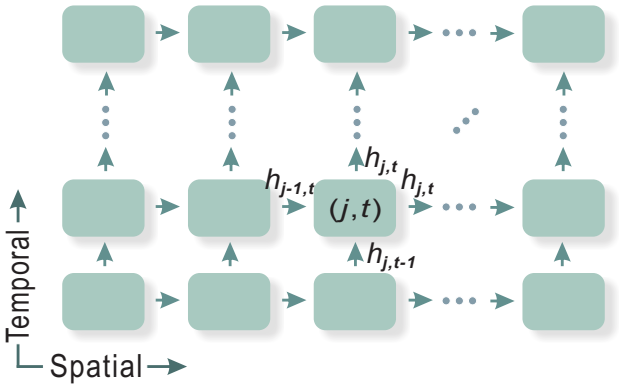


Fig. 3.1 Illustration of the spatio-temporal LSTM network. In temporal dimension, the corresponding body joints are fed over the frames. In spatial dimension, the skeletal joints in each frame are fed as a sequence. Each unit receives the hidden representation of the previous joints and the same joint from previous frames.

motion patterns for action recognition. However, there is also discriminative spatial information encoded in the joints' locations and posture configurations at each video frame, and the sequential nature of the body joints makes it possible to apply RNN-based modeling to spatial domain as well.

Different from the existing methods which concatenate the joints' information as the entire body's representation, we extend the recurrent analysis to spatial domain by discovering the spatial dependency patterns among different body joints. We propose a spatio-temporal LSTM (ST-LSTM) network to simultaneously model the temporal dependencies among different frames and also the spatial dependencies of different joints at the same frame. Each ST-LSTM unit, which corresponds to one of the body joints, receives the hidden representation of its own joint from the previous time step and also the hidden representation of its previous joint at the current frame. A schema of this model is illustrated in Fig. 3.1.

In this section, we assume the joints are arranged in a simple chain sequence, and the order is depicted in Fig. 3.3(a). In section 3.2.3, we will introduce a more advanced traversal scheme to take advantage of the adjacency structure among the skeletal joints.

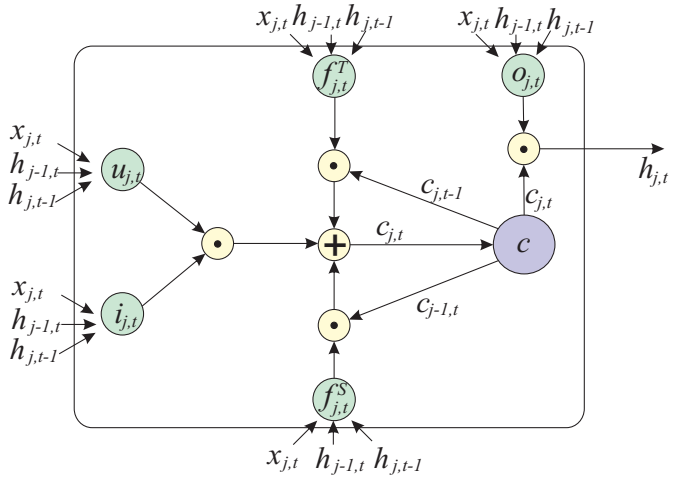


Fig. 3.2 Illustration of the proposed ST-LSTM with one unit.

We use j and t to respectively denote the indices of joints and frames, where $j \in \{1,...,J\}$ and $t \in \{1,...,T\}$. Each ST-LSTM unit is fed with the input ($x_{j,t}$, the information of the corresponding joint at current time step), the hidden representation of the previous joint at current time step $(h_{j-1,t})$, and the hidden representation of the same joint at the previous time step $(h_{j,t-1})$.

As depicted in Fig. 3.2, each unit also has two forget gates, $f^T_{j,t}$ and $f^S_{j,t}$, to handle the two sources of context information in temporal and spatial dimensions, respectively. The transition equations of ST-LSTM are formulated as follows:

$$\begin{pmatrix} i_{j,t} \\ f^S_{j,t} \\ f^T_{j,t} \\ o_{j,t} \\ u_{j,t} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(M \begin{pmatrix} x_{j,t} \\ h_{j-1,t} \\ h_{j,t-1} \end{pmatrix} \right) \tag{3.4}$$

$$c_{j,t} = i_{j,t} \odot u_{j,t} + f^S_{j,t} \odot c_{j-1,t} + f^T_{j,t} \odot c_{j,t-1} \tag{3.5}$$

$$h_{j,t} = o_{j,t} \odot \tanh(c_{j,t}) \tag{3.6}$$

3.2.3 Tree-Structure Based Traversal

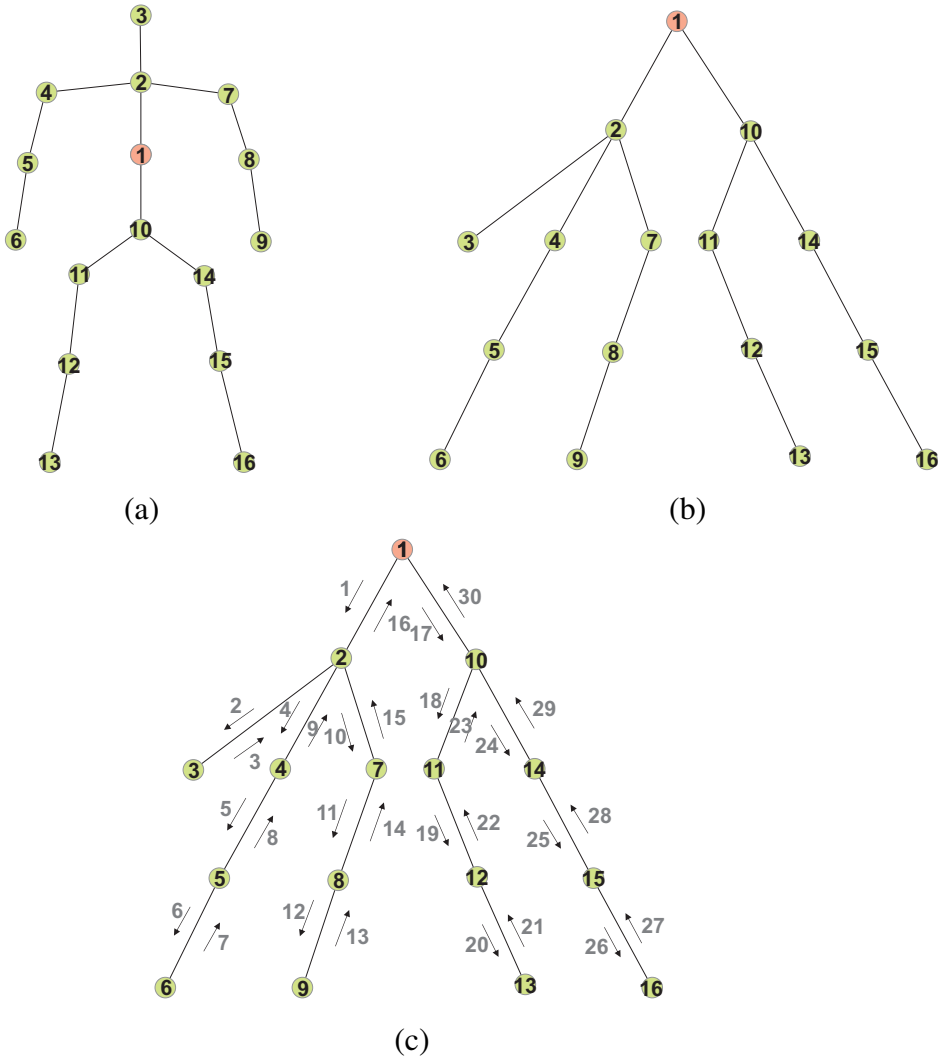


Fig. 3.3 (a) The skeleton of the human body. In the simple joint chain model, the joint visiting order is 1-2-3-...-16. (b) The skeleton is transformed to a tree structure. (c) The tree traversal scheme. The tree structure can be unfolded to a chain with the traversal scheme, and the joint visiting order is 1-2-3-2-4-5-6-5-4-2-7-8-9-8-7-2-1-10-11-12-13-12-11-10-14-15-16-15-14-10-1.

Arranging the skeletal joints in a simple chain order ignores the kinematic interdependencies among the body joints. Moreover, several semantically false connections between the joints, which are not strongly related, are added.

The body joints are popularly represented as a tree-based pictorial structure [115, 116] in human parsing, as shown in Fig. 3.3(b). It is beneficial to utilize the known

interdependency relations between various sets of body joints as an adjacency tree structure inside our ST-LSTM network as well. For instance, the hidden representation of the neck joint (joint 2 in Fig. 3.3(a)) is often more informative for the right hand joints (7, 8, and 9) compared to the joint 6, which lies before them in the numerically ordered chain-like model. Though using a tree structure for the skeletal data sounds more reasonable here, tree structures cannot be directly fed into our current form of the ST-LSTM network.

In order to mitigate the aforementioned limitation, a bidirectional tree traversal scheme is proposed. In this scheme, the joints are visited in a sequence, while the adjacency information in the skeletal tree structure will be maintained. At the first spatial step, the root node (central spine joint in Fig. 3.3(c)) is fed to our network. Then the network follows the depth-first traversal order in the spatial (skeleton tree) domain. Upon reaching a leaf node, the traversal backtracks in the tree. Finally, the traversal goes back to the root node.

In our traversal scheme, each connection in the tree is met twice, thus it guarantees the transmission of the context data in both top-down and bottom-up directions within the adjacency tree structure. In other words, each node (joint) can obtain the context information from both its ancestors and descendants in the hierarchy defined by the tree structure. Compared to the simple joint chain order described in section 3.2.2, this tree traversal strategy, which takes advantage of the joints' adjacency structure, can discover stronger long-term spatial dependency patterns in the skeleton sequence.

Our framework's representation capacity can be further improved by stacking multiple layers of the tree-structured ST-LSTMs and making the network deeper (see Fig. 3.4).

It is worth noting that at each step of our ST-LSTM framework, the input is limited to the information of a single joint at a time step, and its dimension is much smaller compared to the concatenated input features used by other existing methods. Therefore,

our network has much fewer learning parameters. This can be regarded as a weight sharing regularization for our learning model, which leads to better generalization in the scenarios with relatively small sets of training samples. This is an important advantage for skeleton-based action recognition, since the numbers of training samples in most existing datasets are limited.

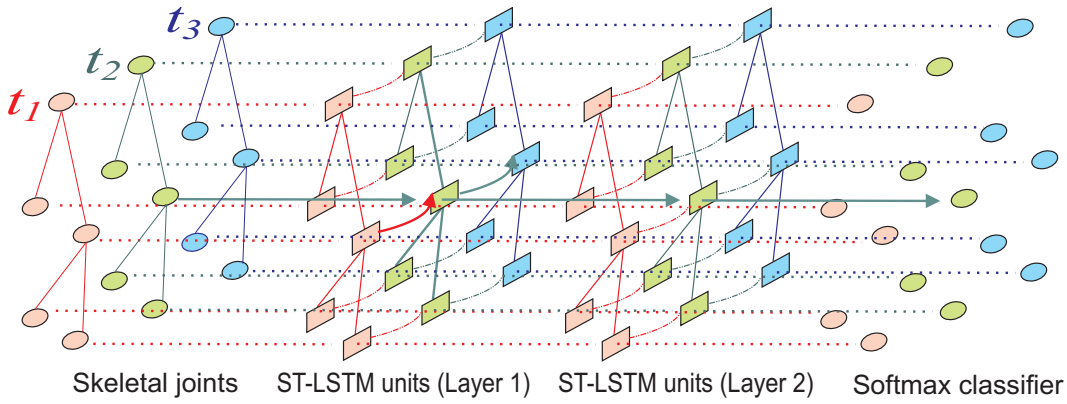


Fig. 3.4 Illustration of the deep tree-structured ST-LSTM network. For clarity, some arrows are omitted in this figure. The hidden representation of the first ST-LSTM layer is fed to the second ST-LSTM layer as its input. The second ST-LSTM layer's hidden representation is fed to the softmax layer for classification.

3.2.4 Spatio-Temporal LSTM with Trust Gates

In our proposed tree-structured ST-LSTM network, the inputs are the positions of body joints provided by depth sensors (such as Kinect), which are not always accurate because of noisy measurements and occlusion. The unreliable inputs can degrade the performance of the network.

To circumvent this difficulty, we propose to add a novel additional gate to our ST-LSTM network to analyze the reliability of the input measurements based on the derived estimations of the input from the available context information at each spatio-temporal step. Our gating scheme is inspired by the works in natural language processing [92], which use the LSTM representation of previous words at each step to predict the next coming word. As there are often high dependency relations among

the words in a sentence, this idea works decently. Similarly, in a skeletal sequence, the neighboring body joints often move together, and this articulated motion follows common yet complex patterns, thus the input data $x_{j,t}$ is expected to be predictable by using the contextual information ($h_{j-1,t}$ and $h_{j,t-1}$) at each spatio-temporal step.

Inspired by this predictability concept, we add a new mechanism to our ST-LSTM calculating a prediction of the input at each step and comparing it with the actual input. The amount of estimation error is then used to learn a new "trust gate". The activation of this new gate can be used to assist the ST-LSTM network to learn better decisions about when and how to remember or forget the contents in the memory cell. For instance, if the trust gate learns that the current joint has wrong measurements, then this gate can block the input gate and prevent the memory cell from being altered by the current unreliable input data.

Concretely, we introduce a function to produce a prediction of the input at step (j,t) based on the available context information as:

$$p_{j,t} = \tanh\left(M_p \begin{pmatrix} h_{j-1,t} \\ h_{j,t-1} \end{pmatrix}\right) \tag{3.7}$$

where M_p is an affine transformation mapping the data from $\mathbb{R}^{2d}$ to $\mathbb{R}^d$, thus the dimension of $p_{j,t}$ is d. Note that the context information at each step does not only contain the representation of the previous temporal step, but also the hidden state of the previous spatial step. This indicates that the long-term context information of both the same joint at previous frames and the other visited joints at the current frame are seamlessly incorporated. Thus this function is expected to be capable of generating reasonable predictions.

In our proposed network, the activation of trust gate is a vector in $\mathbb{R}^d$ (similar to the activation of input gate and forget gate). The trust gate $\tau_{j,t}$ is calculated as follows:

$$x'_{j,t} = \tanh\left(M_x\left(x_{j,t}\right)\right) \tag{3.8}$$

$$\tau_{j,t} = G(p_{j,t} - x'_{j,t}) \tag{3.9}$$

where $M_x : \mathbb{R}^D \to \mathbb{R}^d$ is an affine transformation. The activation function $G(\cdot)$ is an element-wise operation calculated as $G(z) = \exp(-\lambda z^2)$, for which λ is a parameter to control the bandwidth of Gaussian function ($\lambda > 0$). $G(z)$ produces a small response if z has a large absolute value and a large response when z is close to zero.

Adding the proposed trust gate, the cell state of ST-LSTM will be updated as:

$$\begin{aligned} c_{j,t} = \; & \tau_{j,t} \odot i_{j,t} \odot u_{j,t} \\ & + (\mathbf{1} - \tau_{j,t}) \odot f^S_{j,t} \odot c_{j-1,t} \\ & + (\mathbf{1} - \tau_{j,t}) \odot f^T_{j,t} \odot c_{j,t-1} \end{aligned} \tag{3.10}$$

This equation can be explained as follows: (1) if the input $x_{j,t}$ is not trusted (due to the noise or occlusion), then our network relies more on its history information, and tries to block the new input at this step; (2) on the contrary, if the input is reliable, then our learning algorithm updates the memory cell regarding the input data.

The proposed ST-LSTM unit equipped with trust gate is illustrated in Fig. 3.5. The concept of the proposed trust gate technique is theoretically generic and can be used in other domains to handle noisy input information for recurrent network models.

3.2.5 Feature Fusion within ST-LSTM Unit

As mentioned above, at each spatio-temporal step, the positional information of the corresponding joint at the current frame is fed to our ST-LSTM network. Here we

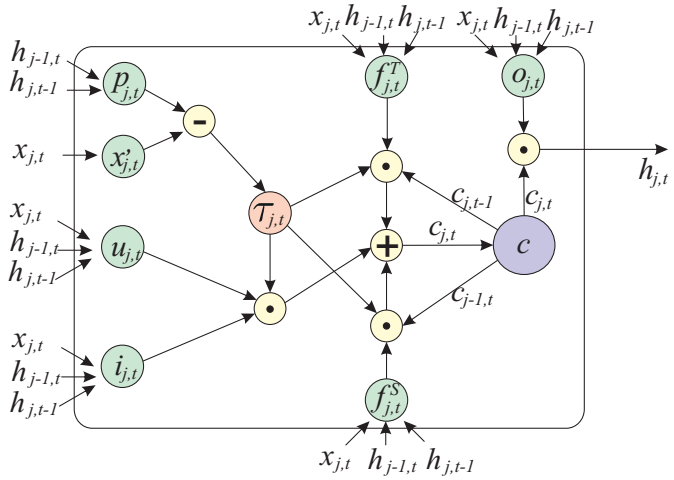


Fig. 3.5 Illustration of the proposed ST-LSTM with trust gate.

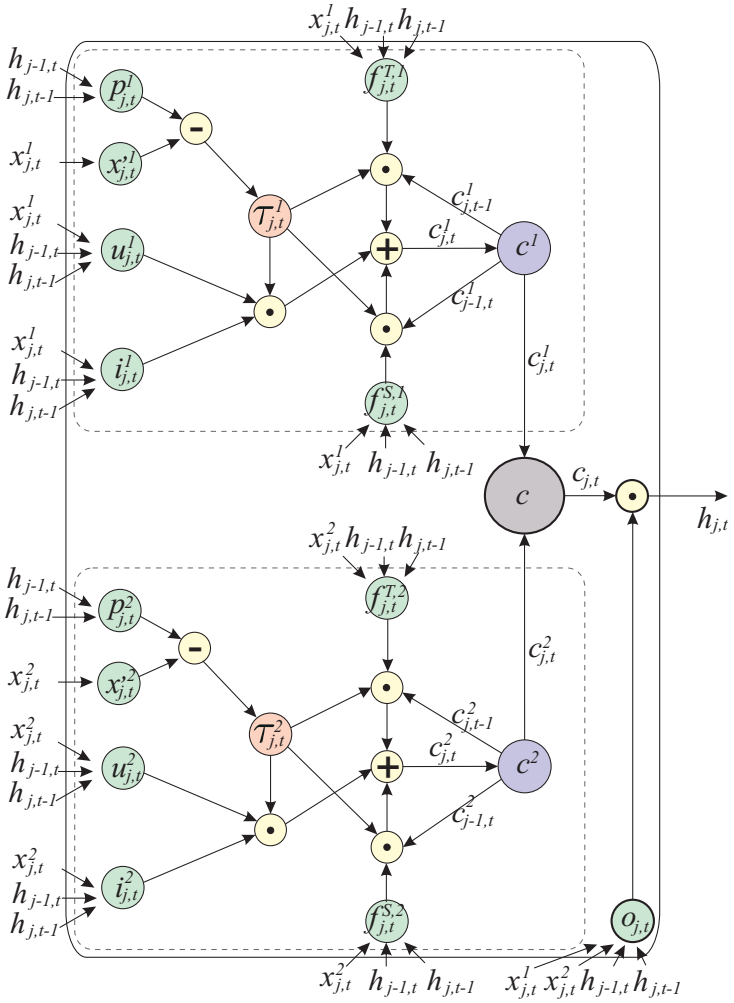


Fig. 3.6 Illustration of the proposed structure for feature fusion inside the ST-LSTM unit.

call joint position-based feature as a geometric feature. Beside utilizing the joint position (3D coordinates), we can also extract visual texture and motion features (*e.g.*, HOG, HOF [117, 118], or ConvNet-based features [119, 120]) from the RGB frames, around each body joint as the complementary information. This is intuitively effective for better human action representation, especially in the human-object interaction scenarios.

A naive way for combining geometric and visual features for each joint is to concatenate them in the feature level and feed them to the corresponding ST-LSTM unit as network's input data. However, the dimension of the geometric feature is very low intrinsically, while the visual features are often in relatively higher dimensions. Due to this inconsistency, simple concatenation of these two types of features in the input stage of the network causes degradation in the final performance of the entire model.

The work in [1] feeds different body parts into the Part-aware LSTM [1] separately, and then assembles them inside the LSTM unit. Inspired by this work, we propose to fuse the two types of features inside the ST-LSTM unit, rather than simply concatenating them at the input level.

We use $x_{j,t}^{\mathscr{F}}$ ($\mathscr{F} \in \{1,2\}$) to denote the geometric feature and visual feature for a joint at the t-th time step. As illustrated in Fig. 3.6, at step (j,t), the two features ($x_{j,t}^{1}$ and $x_{j,t}^{2}$) are fed to the ST-LSTM unit separately as the new input structure. Inside the recurrent unit, we deploy two sets of gates, input gates ($i_{j,t}^{\mathscr{F}}$), forget gates with respect to time ($f_{j,t}^{T,\mathscr{F}}$) and space ($f_{j,t}^{S,\mathscr{F}}$), and also trust gates ($\tau_{j,t}^{\mathscr{F}}$), to deal with the two heterogeneous sets of modality features. We put the two cell representations ($c_{j,t}^{\mathscr{F}}$) together to build up the multimodal context information of the two sets of modality features. Finally, the output of each ST-LSTM unit is calculated based on the multimodal context representations, and controlled by the output gate ($o_{j,t}$) which is shared for the two sets of features.

For the features of each modality, it is efficient and intuitive to model their context information independently. However, we argue that the representation ability of each modality-based sets of features can be strengthened by borrowing information from the other set of features. Thus, the proposed structure does not completely separate the modeling of multimodal features.

Let us take the geometric feature as an example. Its input gate, forget gates, and trust gate are all calculated from the new input ($x^1_{j,t}$) and hidden representations ($h_{j,t-1}$ and $h_{j-1,t}$), whereas each hidden representation is an associate representation of two features' context information from previous steps. Assisted by visual features' context information, the input gate, forget gates, and also trust gate for geometric feature can effectively learn how to update its current cell state ($c^1_{j,t}$). Specifically, for the new geometric feature input ($x^1_{j,t}$), we expect the network to produce a better prediction when it is not only based on the context of the geometric features, but also assisted by the context of visual features. Therefore, the trust gate ($\tau^1_{j,t}$) will have stronger ability to assess the reliability of the new input data ($x^1_{j,t}$).

The proposed ST-LSTM with integrated multimodal feature fusion is formulated as:

$$\begin{pmatrix} i^{\mathscr{F}}_{j,t} \\ f^{S,\mathscr{F}}_{j,t} \\ f^{T,\mathscr{F}}_{j,t} \\ u^{\mathscr{F}}_{j,t} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(M^{\mathscr{F}} \begin{pmatrix} x^{\mathscr{F}}_{j,t} \\ h_{j-1,t} \\ h_{j,t-1} \end{pmatrix} \right) \tag{3.11}$$

$$p^{\mathscr{F}}_{j,t} = \tanh\left(M^{\mathscr{F}}_{p} \begin{pmatrix} h_{j-1,t} \\ h_{j,t-1} \end{pmatrix} \right) \tag{3.12}$$

$$x'^{\mathscr{F}}_{j,t} = \tanh\left(M^{\mathscr{F}}_{x} \begin{pmatrix} x^{\mathscr{F}}_{j,t} \end{pmatrix} \right) \tag{3.13}$$

$$\tau^{\mathscr{F}}_{j,t} = G(x'^{\mathscr{F}}_{j,t} - p^{\mathscr{F}}_{j,t}) \tag{3.14}$$

$$\begin{aligned} c^{\mathscr{F}}_{j,t} = {} & \tau^{\mathscr{F}}_{j,t} \odot i^{\mathscr{F}}_{j,t} \odot u^{\mathscr{F}}_{j,t} \\ & + (\mathbf{1} - \tau^{\mathscr{F}}_{j,t}) \odot f^{S,\mathscr{F}}_{j,t} \odot c^{\mathscr{F}}_{j-1,t} \\ & + (\mathbf{1} - \tau^{\mathscr{F}}_{j,t}) \odot f^{T,\mathscr{F}}_{j,t} \odot c^{\mathscr{F}}_{j,t-1} \end{aligned} \tag{3.15}$$

$$o_{j,t} = \sigma\left(M_o \begin{pmatrix} x^1_{j,t} \\ x^2_{j,t} \\ h_{j-1,t} \\ h_{j,t-1} \end{pmatrix} \right) \tag{3.16}$$

$$h_{j,t} = o_{j,t} \odot \tanh \begin{pmatrix} c^1_{j,t} \\ c^2_{j,t} \end{pmatrix} \tag{3.17}$$

3.2.6 Learning the Classifier

As the labels are given at video level, we feed them as the training outputs of our network at each spatio-temporal step. A softmax layer is used by the network to predict the action class $\hat{y}$ among the given class set Y. The prediction of the whole video can be obtained by averaging the prediction scores of all steps. The objective function of our ST-LSTM network is as follows: $\mathscr{L} = \sum_{j=1}^{J} \sum_{t=1}^{T} l(\hat{y}_{j,t}, y)$, where $l(\hat{y}_{j,t}, y)$ is the

negative log-likelihood loss [121] that measures the difference between the prediction result $\hat{y}_{j,t}$ at step (j,t) and the true label y.

The back-propagation through time (BPTT) algorithm [121] is effective for minimizing the objective function for the RNN/LSTM models. As our ST-LSTM model involves both spatial and temporal steps, we adopt a modified version of BPTT for training. The back-propagation runs over spatial and temporal steps by starting at the last joint at the last frame. To clarify the error accumulation in this procedure, we use $e^T_{j,t}$ and $e^S_{j,t}$ to denote the error back-propagated from step $(j,t+1)$ to (j,t) and the error back-propagated from step $(j+1,t)$ to (j,t), respectively. Then the errors accumulated at step (j,t) can be calculated as $e^T_{j,t} + e^S_{j,t}$. Consequently, before back-propagating the error at each step, we should guarantee both its subsequent spatial step and subsequent temporal step have already been computed.

The left-most units in our ST-LSTM network do not have preceding units in the spatial dimension, as shown in Fig. 3.1. To update the cell states of these units in the feed-forward stage, a popular strategy is to feed zeros to them as the hidden representations from the preceding nodes. In our implementation, we link the last unit at the previous time step to the first unit at the current time step. We call the new connection as "last-to-first link". In the tree traversal, the first and last nodes refer to the same joint (root node of the tree). However, the last node contains holistic context information of the human skeleton structure in the corresponding frame. Linking the last node to the start node of the next time step provides the start node with the whole body structure configuration, rather than initializing it with less meaningful zero values. Thus, the network with last-to-first link has better ability to effectively learn the action patterns in the skeleton data.

3.3 Experiments

The proposed method is evaluated and empirically analyzed on seven benchmark datasets for which the coordinates of skeletal joints are provided. These datasets are NTU RGB+D, UT-Kinect, SBU Interaction, SYSU-3D, ChaLearn Gesture, MSR Action3D, and Berkeley MHAD. We conduct extensive experiments with different models to verify the effectiveness of individual technical contributions proposed, as follows:

(1) "ST-LSTM (Joint Chain)". In this model, the joints are visited in a simple chain order, as shown in Fig. 3.3(a);

(2) "ST-LSTM (Joint Chain) + Trust Gate". This model uses the trust gate to handle the noisy input.

(3) "ST-LSTM (Tree)". In this model, the tree traversal scheme illustrated in Fig. 3.3(c) is used to take advantage of the tree-based spatial structure of skeletal joints;

(4) "ST-LSTM (Tree) + Trust Gate". In this model, the tree traversal is used. The trust gate is also added.

The input to every unit of of our network at each spatio-temporal step is the location of the corresponding skeletal joint (i.e., geometric features) at the current time step. We also use two of the datasets (NTU RGB+D dataset and UT-Kinect dataset) as examples to evaluate the performance of our fusion model within the ST-LSTM unit by fusing the geometric and visual features. These two datasets include human-object interactions (such as making a phone call and picking up something) and the visual information around the major joints can be complementary to the geometric features for action recognition.

3.3.1 Implementation Details

In our experiments, each video sequence is divided to T sub-sequences with the same length, and one frame is randomly selected from each sub-sequence. This sampling strategy has the following advantages: (1) Randomly selecting a frame from each sub-sequence can add variation to the input data, and improve the generalization strengths of our trained network. (2) Assume each sub-sequence contains n frames, so we have n choices to sample a frame from each sub-sequence. Accordingly, for the whole video, we can obtain a total number of n^T sampling combinations. This indicates that the training data can be greatly augmented. We use different frame sampling combinations for each video over different training epochs. This strategy is useful for handling the over-fitting issues, as most datasets have limited numbers of training samples. We observe this strategy achieves better performance in contrast with uniformly sampling frames. We cross-validated the performance based on the leave-one-subject-out protocol on the large scale NTU RGB+D dataset, and found $T = 20$ as the optimum value.

We use Torch7 [122] as the deep learning platform to perform our experiments. We train the network with stochastic gradient descent, and set the learning rate, momentum, and decay rate to 2×10^{-3}, 0.9, and 0.95, respectively. We set the unit size d to 128, and the parameter λ used in $G(\cdot)$ to 0.5. Two ST-LSTM layers are used in our stacked network. Although there are variations in terms of joint number, sequence length, and data acquisition equipment for different datasets, we adopt the same parameter settings mentioned above for all datasets. Our method achieves promising results on all the benchmark datasets with these parameter settings untouched, which shows the robustness of our method. An NVIDIA TitanX GPU is used to perform our experiments. We evaluate the computational efficiency of our method on the NTU RGB+D dataset and set the batch size to 100. On average, within one second, 210,

100, and 70 videos can be processed by using "ST-LSTM (Joint Chain)", "ST-LSTM (Tree)", and "ST-LSTM (Tree) + Trust Gate", respectively.

3.3.2 Experiments on the NTU RGB+D Dataset

The NTU RGB+D dataset has two standard evaluation protocols [1]. The first protocol is the cross-subject (X-Subject) evaluation protocol, in which half of the subjects are used for training and the remaining subjects are kept for testing. The second is the cross-view (X-View) evaluation protocol, in which $2/3$ of the viewpoints are used for training, and $1/3$ unseen viewpoints are left out for testing. We evaluate the performance of our method on both of these protocols. The results are shown in Table 3.1.

Table 3.1 Experimental results on the NTU RGB+D Dataset. "X-S" and "X-V" denote X-Subject and X-View, respectively.

Method	Feature	X-S	X-V
Lie Group [42]	Geometric	50.1%	52.8%
Cippitelli *et al.* [123]	Geometric	48.9%	57.7%
Dynamic Skeletons [85]	Geometric	60.2%	65.2%
FTP [124]	Geometric	61.1%	72.6%
Hierarchical RNN [3]	Geometric	59.1%	64.0%
Deep RNN [1]	Geometric	56.3%	64.1%
Part-aware LSTM [1]	Geometric	62.9%	70.3%
ST-LSTM (Joint Chain)	Geometric	61.7%	75.5%
ST-LSTM (Joint Chain)+Trust Gate	Geometric	64.9%	76.8%
ST-LSTM (Tree)	Geometric	65.2%	76.1%
ST-LSTM (Tree) + Trust Gate	Geometric	**69.2%**	**77.7%**

In Table 3.1, the deep RNN model concatenates the joint features at each frame and then feeds them to the network to model the temporal kinetics, and ignores the spatial dynamics. As can be seen, both "ST-LSTM (Joint Chain)" and "ST-LSTM (Tree)" models outperform this method by a notable margin. It can also be observed that our approach utilizing the trust gate brings significant performance improvement,

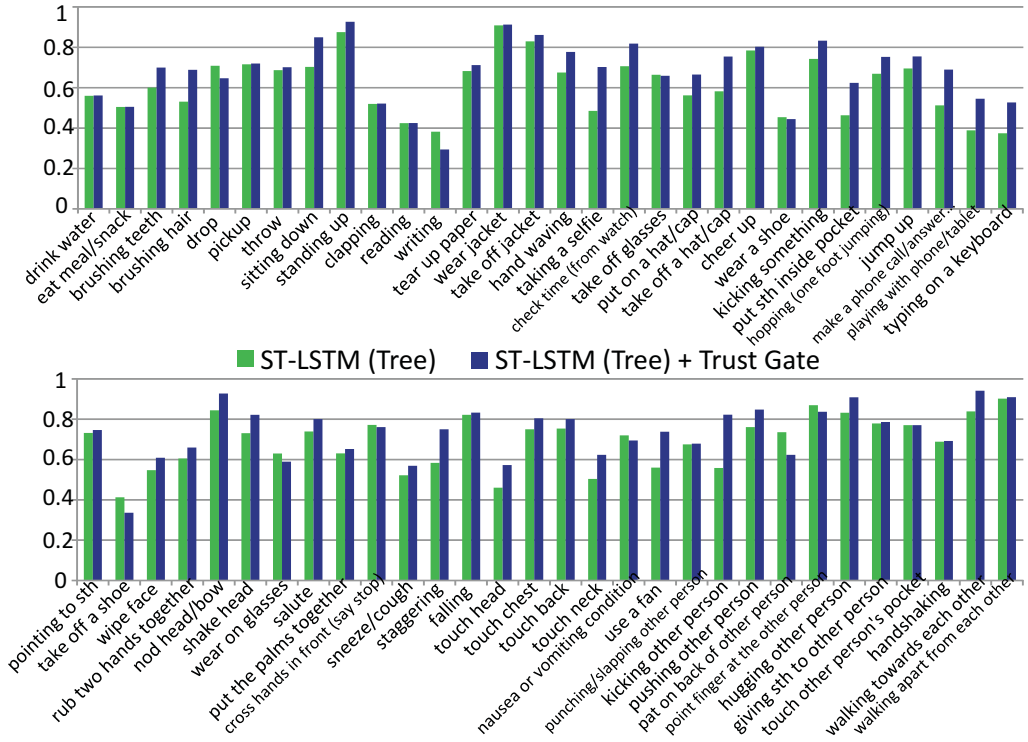


Fig. 3.7 Recognition accuracy per class on the NTU RGB+D dataset

because the data provided by Kinect is often noisy and multiple joints are frequently occluded in this dataset. Note that our proposed models (such as "ST-LSTM (Tree) + Trust Gate") reported in this table only use skeletal data as input.

We compare the class specific recognition accuracies of "ST-LSTM (Tree)" and "ST-LSTM (Tree) + Trust Gate", as shown in Fig. 3.7. We observe that "ST-LSTM (Tree) + Trust Gate" significantly outperforms "ST-LSTM (Tree)" for most of the action classes, which demonstrates our proposed trust gate can effectively improve the human action recognition accuracy by learning the degrees of reliability over the input data at each time step.

A notable portion of videos in the NTU RGB+D dataset were collected in side views. Due to the design of Kinect's body tracking mechanism, skeletal data is less accurate in side view compared to the front view. To further investigate the effectiveness of the proposed trust gate, we analyze the performance of the network by feeding the side views samples only. The accuracy of "ST-LSTM (Tree)" is 76.5%,

while "ST-LSTM (Tree) + Trust Gate" yields 81.6%. This shows how trust gate can effectively deal with the noise in the input data.

To verify the performance boost by stacking layers, we limit the depth of the network by using only one ST-LSTM layer, and the accuracies drop to 65.5% and 77.0% based on the cross-subject and cross-view protocol, respectively. This indicates our two-layer stacked network has better representation power than the single-layer network.

To evaluate the performance of our feature fusion scheme, we extract visual features from several regions based on the joint positions and use them in addition to the geometric features (3D coordinates of the joints). We extract HOG and HOF [117, 118] features from a 80×80 RGB patch centered at each joint location. For each joint, this produces a 300D visual descriptor, and we apply PCA to reduce the dimension to 20. The results are shown in Table 3.2. We observe that our method using the visual features together with the joint positions improves the performance. Besides, we compare our newly proposed feature fusion strategy within the ST-LSTM unit with two other feature fusion methods: (1) early fusion which simply concatenates two types of features as the input of the ST-LSTM unit; (2) late fusion which uses two ST-LSTMs to deal with two types of features respectively, then concatenates the outputs of the two ST-LSTMs at each step, and feeds the concatenated result to a softmax classifier. We observe that our proposed feature fusion strategy is superior to other baselines.

We also evaluate the sensitivity of the proposed network with respect to the variation of neuron unit size and λ values. The results are shown in Fig. 3.8. When trust gate is added, our network obtains better performance for all the λ values compared to the network without the trust gate.

Finally, we investigate the recognition performance with early stopping conditions by feeding the first p portion of the testing video to the trained network based on the

Table 3.2 Evaluation of different feature fusion strategies on the NTU RGB+D dataset. "Geometric + Visual (1)" indicates the early fusion scheme. "Geometric + Visual (2)" indicates the late fusion scheme. "Geometric $\bigoplus$ Visual" means our newly proposed feature fusion scheme within the ST-LSTM unit.

Feature Fusion Method	X-Subject	X-View
Geometric Only	69.2%	77.7%
Geometric + Visual (1)	70.8%	78.6%
Geometric + Visual (2)	71.0%	78.7%
Geometric $\bigoplus$ Visual	73.2%	80.6%

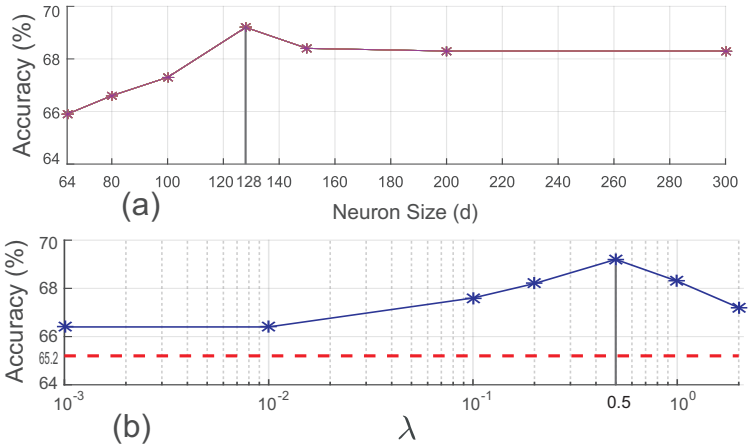


Fig. 3.8 (a) Performance comparison of our approach using different values of neuron size (d) on the NTU RGB+D dataset (X-subject). (b) Performance comparison of our method using different λ values on the NTU RGB+D dataset (X-subject). The blue line represents our results when different λ values are used for trust gate, while the red dashed line indicates the performance of our method when trust gate is not added.

cross-subject protocol ($p \in \{0.1, 0.2, ..., 1.0\}$). The results are shown in Fig. 3.9. We observe that the results are improved when a larger portion of the video is fed to our network.

3.3.3 Experiments on the UT-Kinect Dataset

There are two evaluation protocols for the UT-Kinect dataset in the literature. The first is the leave-one-out-cross-validation (LOOCV) protocol [7]. The second protocol is suggested by [8], for which half of the subjects are used for training, and the remaining are used for testing. We evaluate our approach using both protocols on this

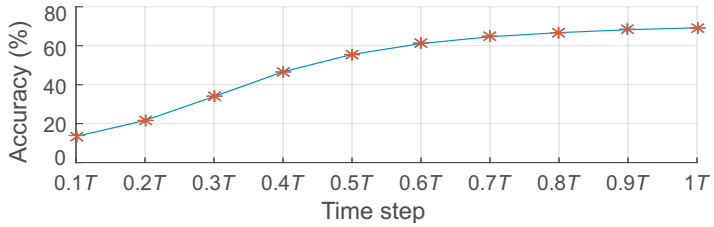


Fig. 3.9 Experimental results of our method by early stopping the network evolution at different time steps.

dataset. Using the LOOCV protocol, our method achieves better performance than other methods, as shown in Table 3.3. Using the second protocol (see Table 3.4), our method achieves competitive result (95.0%) to the Elastic functional coding method [38] (94.9%), which is an extension of the Lie Group model [42].

Table 3.3 Experimental results on the UT-Kinect dataset (LOOCV protocol [7])

Method	Feature	Acc.
Grassmann Manifold [125]	Geometric	88.5%
Jetley *et al.* [126]	Geometric	90.0%
Histogram of 3D Joints [7]	Geometric	90.9%
Space Time Pose [127]	Geometric	91.5%
Riemannian Manifold [128]	Geometric	91.5%
SCs (Informative Joints) [129]	Geometric	91.9%
Chrungoo *et al.* [130]	Geometric	92.0%
Key-Pose-Motifs Mining[131]	Geometric	93.5%
ST-LSTM (Joint Chain)	Geometric	91.0%
ST-LSTM (Joint Chain) + Trust Gate	Geometric	94.5%
ST-LSTM (Tree)	Geometric	92.4%
ST-LSTM (Tree) + Trust Gate	Geometric	**97.0%**

Table 3.4 Results on the UT-Kinect dataset (half-vs-half protocol [8])

Method	Feature	Acc.
Skeleton Joint Features [8]	Geometric	87.9%
Chrungoo *et al.* [130]	Geometric	89.5%
Lie Group [42] (reported by [38])	Geometric	93.6%
Elastic functional coding [38]	Geometric	94.9%
ST-LSTM (Tree) + Trust Gate	Geometric	**95.0%**

Some actions in the UT-Kinect dataset involve human-object interactions, thus appearance based features representing visual information of the objects can be complementary to the geometric features. Thus we can evaluate our proposed feature fusion approach within the ST-LSTM unit on this dataset. The results are shown in Table 3.5. Using geometric features only, the accuracy is 97%. By simply concatenating the geometric and visual features, the accuracy improves slightly. However, the accuracy of our approach can reach 98% when the proposed feature fusion method is adopted.

Table 3.5 Evaluation of our approach for feature fusion on the UT-Kinect dataset (LOOCV protocol [7]).

Feature Fusion Method	Acc.
Geometric Only	97.0%
Geometric + Visual (1)	97.5%
Geometric + Visual (2)	97.5%
Geometric $\bigoplus$ Visual	98.0%

3.3.4 Experiments on the SBU Interaction Dataset

We follow the standard evaluation protocol in [39] and perform 5-fold cross validation on the SBU Interaction dataset. As two human skeletons are provided in each frame of this dataset, our traversal scheme visits the joints throughout the two skeletons over the spatial steps. We report the results in terms of average classification accuracy in Table 3.6. The methods in [5] and [3] are both LSTM-based approaches, which are more relevant to our method.

The results show that the proposed "ST-LSTM (Tree) + Trust Gate" model outperforms all other skeleton-based methods. "ST-LSTM (Tree)" achieves higher accuracy than "ST-LSTM (Joint Chain)", as the latter adds some false links between less related joints.

Table 3.6 Experimental results on the SBU Interaction dataset

Method	Feature	Acc.
Yun *et al.* [39]	Geometric	80.3%
Ji *et al.* [132]	Geometric	86.9%
CHARM [133]	Geometric	83.9%
Hierarchical RNN [3]	Geometric	80.4%
Co-occurrence LSTM [5]	Geometric	90.4%
Deep LSTM [5]	Geometric	86.0%
ST-LSTM (Joint Chain)	Geometric	84.7%
ST-LSTM (Joint Chain) + Trust Gate	Geometric	89.7%
ST-LSTM (Tree)	Geometric	88.6%
ST-LSTM (Tree) + Trust Gate	Geometric	**93.3%**

Both Co-occurrence LSTM [5] and Hierarchical RNN [3] adopt the Svaitzky-Golay filter in temporal domain to smooth the skeletal joint positions and reduce the influence of noise in the data collected by Kinect. The proposed "ST-LSTM (Tree)" model without the trust gate mechanism outperforms Hierarchical RNN, and achieves comparable result (88.6%) to Co-occurrence LSTM. When the trust gate is used, the accuracy of our method jumps to 93.3%.

3.3.5 Experiments on the SYSU-3D Dataset

We follow the standard evaluation protocol in [85] on the SYSU-3D dataset. The samples from 20 subjects are used to train the model parameters, and the samples of the remaining 20 subjects are used for testing. We perform 30-fold cross validation and report the mean accuracy in Table 3.7.

The results in Table 3.7 show that our proposed "ST-LSTM (Tree) + Trust Gate" method outperforms all the baseline methods on this dataset. We can also find that the tree traversal strategy can help to improve the classification accuracy of our model. As the skeletal joints provided by Kinect are noisy in this dataset, the trust gate, which

Table 3.7 Experimental results on the SYSU-3D dataset

Method	Feature	Acc.
LAFF (SKL) [6]	Geometric	54.2%
Dynamic Skeletons [85]	Geometric	75.5%
ST-LSTM (Joint Chain)	Geometric	72.1%
ST-LSTM (Joint Chain) + Trust Gate	Geometric	74.8%
ST-LSTM (Tree)	Geometric	73.4%
ST-LSTM (Tree) + Trust Gate	Geometric	**76.5%**

aims at handling noisy data, brings significant performance improvement (about 3% improvement).

There are large viewpoint variations in this dataset. To make our model reliable against viewpoint variations, we adopt a similar skeleton normalization procedure as suggested by [1] on this dataset. In this preprocessing step, we perform a rotation transformation on each skeleton, such that all the normalized skeletons face to the same direction. Specifically, after rotation, the 3D vector from "right shoulder" to "left shoulder" will be parallel to the X axis, and the vector from "hip center" to "spine" will be aligned to the Y axis (please see [1] for more details about the normalization procedure).

We evaluate our "ST-LSTM (Tree) + Trust Gate" method by respectively using the original skeletons without rotation and the transformed skeletons. We observe the accuracy is 73.0% if we use the original skeletons, which is lower than the accuracy (76.5%) achieved by using the transformed skeletons. The results show that it is beneficial to use the transformed skeletons as the input for action recognition.

3.3.6 Experiments on the ChaLearn Gesture Dataset

We follow the evaluation protocol adopted in [134, 135] and report the F1-score measures on the validation set of the ChaLearn Gesture dataset.

Table 3.8 Experimental results on the ChaLearn Gesture dataset

Method	Feature	F1-Score
Portfolios [136]	Geometric	56.0%
Wu *et al.* [137]	Geometric	59.6%
Pfister *et al.* [138]	Geometric	61.7%
HiVideoDarwin [134]	Geometric	74.6%
VideoDarwin [135]	Geometric	75.2%
Deep LSTM [1]	Geometric	87.1%
ST-LSTM (Joint Chain)	Geometric	89.1%
ST-LSTM (Joint Chain) + Trust Gate	Geometric	90.6%
ST-LSTM (Tree)	Geometric	89.9%
ST-LSTM (Tree) + Trust Gate	Geometric	**92.0%**

As shown in Table 3.8, our method surpasses the state-of-the-art methods [136–138, 134, 135, 1], which demonstrates the effectiveness of our method in dealing with skeleton-based action recognition problem.

Compared to other methods, our method focuses on modeling both temporal and spatial dependency patterns in skeleton sequences. Besides, the proposed trust gate is also incorporated to our method to handle the noisy skeleton data captured by Kinect, which further improves the results.

3.3.7 Experiments on the MSR Action3D Dataset

We follow the experimental protocol in [3] on the MSR Action3D dataset, and show the results in Table 3.9.

On the MSR Action3D dataset, our proposed method, "ST-LSTM (Tree) + Trust Gate", achieves 94.8% of classification accuracy, which is superior to the Hierarchical RNN model [3] and other baseline methods.

Table 3.9 Experimental results on the MSR Action3D dataset

Method	Feature	Acc.
Histogram of 3D Joints [7]	Geometric	79.0%
Joint Angles Similarities [44]	Geometric	83.5%
SCs (Informative Joints) [129]	Geometric	88.3%
Oriented Displacements [139]	Geometric	91.3%
Lie Group [42]	Geometric	92.5%
Space Time Pose [127]	Geometric	92.8%
Lillo *et al.* [31]	Geometric	93.0%
Hierarchical RNN [3]	Geometric	94.5%
ST-LSTM (Tree) + Trust Gate	Geometric	**94.8%**

Table 3.10 Experimental results on the Berkeley MHAD dataset

Method	Feature	Acc.
Ofli *et al.* [37]	Geometric	95.4%
Vantigodi *et al.* [140]	Geometric	96.1%
Vantigodi *et al.* [141]	Geometric	97.6%
Kapsouras *et al.* [142]	Geometric	98.2%
Hierarchical RNN [3]	Geometric	100%
Co-occurrence LSTM [5]	Geometric	100%
ST-LSTM (Tree) + Trust Gate	Geometric	**100%**

3.3.8 Experiments on the Berkeley MHAD Dataset

We adopt the experimental protocol in [3] on the Berkeley MHAD dataset. 384 video sequences corresponding to the first seven persons are used for training, and the 275 sequences of the remaining five persons are held out for testing. The experimental results in Table 3.10 show that our method achieves very high accuracy (100%) on this dataset. Unlike [3] and [5], our method does not use any preliminary manual smoothing procedures.

3.3.9 Visualization of Trust Gates

In this section, to better investigate the effectiveness of the proposed trust gate scheme, we study the behavior of the proposed framework against the presence of noise in

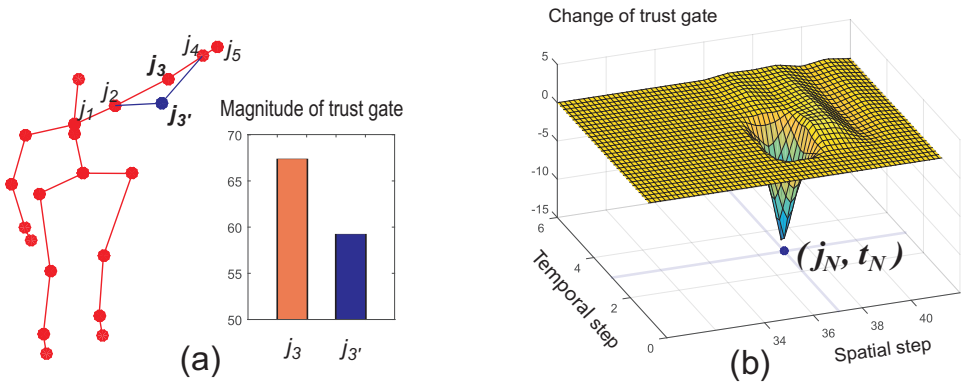


Fig. 3.10 Visualization of the trust gate's behavior when inputting noisy data. (a) $j_{3'}$ is a noisy joint position, and j_3 is the corresponding rectified joint location. In the histogram, the blue bar indicates the magnitude of trust gate when inputting the noisy joint $j_{3'}$. The red bar indicates the magnitude of the corresponding trust gate when $j_{3'}$ is rectified to j_3. (b) Visualization of the difference between the trust gate calculated when the noise is imposed at the step (j_N, t_N) and that calculated when inputting the original data.

skeletal data from the MSR Action3D dataset. We manually rectify some noisy joints of the samples by referring to the corresponding depth images. We then compare the activations of trust gates on the noisy and rectified inputs. As illustrated in Fig. 3.10(a), the magnitude of trust gate's output (l_2 norm of the activations of the trust gate) is smaller when a noisy joint is fed, compared to the corresponding rectified joint. This demonstrates how the network controls the impact of noisy input on its stored representation of the observed data.

In our next experiment, we manually add noise to one joint for all testing samples on the Berkeley MHAD dataset, in order to further analyze the behavior of our proposed trust gate. Note that the Berkeley MHAD dataset was collected with motion capture system, thus the skeletal joint coordinates in this dataset are much more accurate than those captured with Kinect sensors. We add noise to the right foot joint by moving the joint away from its original location. The direction of the translation vector is randomly chosen and the norm is a random value around 30cm, which is a significant noise in the scale of human body. We measure the difference in the magnitudes of trust

gates' activations between the noisy data and the original ones. For all testing samples, we carry out the same operations and then calculate the average difference. The results in Fig. 3.10(b) show that the magnitude of trust gate is reduced when the noisy data is fed to the network. This shows that our network tries to block the flow of noisy input and stop it from affecting the memory. We also observe that the overall accuracy of our network does not drop after adding the above-mentioned noise to the input data.

3.3.10 Evaluation of Different Spatial Joint Sequence Models

The previous experiments showed how "ST-LSTM (Tree)" outperforms "ST-LSTM (Joint Chain)", because "ST-LSTM (Tree)" models the kinematic dependency structures of human skeletal sequences. In this section, we further analyze the effectiveness of our "ST-LSTM (Tree)" model and compare it with other baseline models.

The "ST-LSTM (Joint Chain)" has fewer steps in the spatial dimension than the "ST-LSTM (Tree)". One question that may rise here is if the advantage of "ST-LSTM (Tree)" model could be only due to the higher length and redundant sequence of the joints fed to the network, and not because of the proposed semantic relations between the joints. To answer this question, we evaluate the effect of using a double chain scheme to increase the spatial steps of the "ST-LSTM (Joint Chain)" model. Specifically, we use the joint visiting order of 1-2-3-...-16-1-2-3-...-16, and we call this model as "ST-LSTM (Double Joint Chain)". The results in Table 3.11 show that the performance of "ST-LSTM (Double Joint Chain)" is better than "ST-LSTM (Joint Chain)", yet inferior to "ST-LSTM (Tree)".

This experiment indicates that it is beneficial to introduce more passes in the spatial dimension to the ST-LSTM for performance improvement. A possible explanation is that the units visited in the second round can obtain the global level context representation from the previous pass, thus they can generate better representations of the action patterns by using the context information. However, the performance of "ST-LSTM

Table 3.11 Performance comparison of different spatial sequence models

Dataset	NTU (X-Subject)	NTU (X-View)	UT-Kinect	SBU Interaction	ChaLearn Gesture
ST-LSTM (Random Chain)	59.9%	73.9%	88.4%	78.8%	86.2%
ST-LSTM (Joint Chain)	61.7%	75.5%	91.0%	84.7%	89.1%
ST-LSTM (Double Joint Chain)	63.5%	75.6%	91.5%	85.9%	89.2%
ST-LSTM (Tree)	65.2%	76.1%	92.4%	88.6%	89.9%

(Double Joint Chain)" is still worse than "ST-LSTM (Tree)", though the numbers of their spatial steps are almost equal.

Our proposed tree traversal scheme is superior because it connects the most semantically related joints and avoids false connections between the less-related joints (unlike the other baseline models).

To further investigate the importance of maintaining the adjacency structure among the skeletal joints, we also evaluate a model "ST-LSTM (Random Chain)". In this model, the adjacency structure of the body joints is totally ignored, and the joints are arranged with a random order in spatial dimension. Concretely, we generate five random chains, and report the mean accuracy of them in Table 3.11. The results show that the performance of "ST-LSTM (Random Chain)", which totally ignores the semantic spatial dependence, is obviously worse than other models.

In our "ST-LSTM (Tree)" model, the root node of the tree is the central spine joint (see Fig. 3.3(b)), thus the first visited joint (start joint) in spatial dimension is also this joint (see Fig. 3.3(c)). We also explore different methods to transform the skeleton to a tree structure. Concretely, we select different joints as the root nodes, and thereby generate different tree structures. We respectively set 'central spine joint', 'head joint' , 'left hand joint', 'right hand joint', 'left foot joint', and 'right foot joint' as the root nodes, and accordingly construct six tree structures, which correspond to six different tree traversal sequences. We compare their performance in Table 3.12. We find their results are comparable. Though the joint visiting orders are different in

Table 3.12 Comparison of different start joints (root nodes) of the tree traversal on the NTU RGB+D dataset. "S.D." denotes standard deviation.

Root node of the tree	NTU (X-Subject)	NTU (X-View)
Central spine joint	65.2%	76.1%
Head joint	65.4%	76.4%
Left hand joint	64.9%	75.9%
Right hand joint	65.2%	76.1%
Left foot joint	65.5%	75.9%
Right foot joint	65.7%	76.0%
Mean Acc. & S.D.	65.3±0.3%	76.1±0.2%

these tree traversal sequences, all of them can well maintain the semantic dependency information in the structured skeleton, thus they all obtain promising results.

3.3.11 Evaluation of Temporal Average, LSTM and ST-LSTM

To further investigate the effect of simultaneous modeling of dependencies in spatial and temporal domains, in this experiment, we replace our ST-LSTM with the original LSTM which only models the temporal dynamics among the frames without explicitly considering spatial dependencies. We report the results of this experiment in Table 3.13. As can be seen, our "ST-LSTM + Trust Gate" significantly outperforms "LSTM + Trust Gate". This demonstrates that the proposed modeling of the dependencies in both temporal and spatial dimensions provides much richer representations than the original LSTM.

The second observation of this experiment is that if we add our trust gate to the original LSTM, the performance of LSTM can also be improved, but its performance gain is less than the performance gain by adding trust gate to our ST-LSTM. A possible explanation is that we have both spatial and temporal context information at each step of ST-LSTM to generate a good prediction of the input at the current step (see Eq. (3.7)), thus our trust gate can achieve a good estimation of the reliability of the input at each step by using the prediction (see Eq. (3.9)). However, in the original

Table 3.13 Performance comparison of Temporal Average, LSTM, and our proposed ST-LSTM

Dataset	NTU (X-Subject)	NTU (X-View)	UT-Kinect	SBU Interaction	ChaLearn Gesture
Temporal Average	47.6%	52.6%	81.9%	71.5%	77.9%
LSTM	62.0%	70.7%	90.5%	86.0%	87.1%
LSTM + Trust Gate	62.9%	71.7%	92.0%	86.6%	87.6%
ST-LSTM	65.2%	76.1%	92.4%	88.6%	89.9%
ST-LSTM + Trust Gate	69.2%	77.7%	97.0%	93.3%	92.0%

LSTM, the available context at each step is from the previous temporal step, i.e., the prediction can only be based on the context in the temporal dimension, thus our trust gate is less effective when it is added to the original LSTM. This further demonstrates the effectiveness of our ST-LSTM framework for spatio-temporal modeling of the skeleton sequences.

In addition, we investigate the effectiveness of the LSTM structure for handling the sequential data. We evaluate a baseline method (called "Temporal Average") by averaging the features from all frames instead of using LSTM. Specifically, the geometric features are averaged over all the frames of the input sequence (i.e., the temporal ordering information in the sequence is ignored), and then the resultant averaged feature is fed to a two-layer network, followed by a softmax classifier. The performance of this scheme is much weaker than our proposed ST-LSTM with trust gate, and also weaker than the original LSTM, as shown in Table 3.13. The results demonstrate the representation strengths of the LSTM networks for modeling the dependencies and dynamics in sequential data, when compared to traditional temporal aggregation methods of input sequences.

3.3.12 Evaluation of the Last-to-first Link Scheme

In this section, we evaluate the effectiveness of the last-to-first link scheme in our model (see section 3.2.6). The results in Table 3.14 show that the performance is

Table 3.14 Evaluation of the last-to-first link in our proposed network

Dataset	NTU (X-Subject)	NTU (X-View)	UT-Kinect	SBU Interaction	ChaLearn Gesture
Without last-to-first link	68.5%	76.9%	96.5%	92.1%	90.9 %
With last-to-first link	69.2%	77.7%	97.0%	93.3%	92.0 %

improved after adding the last-to-first link to our network. We analyze the benefits of the last-to-first link as follows. In the spatial dimension of our ST-LSTM network (see Fig. 3.1), the hidden representation of each node is fed to its next node. However, at each time step, the first node does not have preceding node in the spatial dimension. Thus zero values are used to substitute the hidden representation of the preceding node when the last-to-first link is not used. However, when we add the last-to-first link, the first node can obtain the hidden representation from the last node of the previous time step, which contains the context information of the skeleton structure in a frame. Thus the performance after using the last-to-first link scheme is better compared to the alternative method by simply feeding zero values.

3.4 Chapter Summary

In this chapter, we have extended the RNN-based action recognition method to both spatial and temporal domains. Specifically, we have proposed a novel ST-LSTM network which analyzes the 3D locations of skeletal joints at each frame and at each processing step. A skeleton tree traversal method based on the adjacency graph of body joints is also proposed to better represent the structure of the input sequences and to improve the performance of our network by connecting the most related joints together in the input sequence. In addition, a new gating mechanism is introduced to improve the robustness of our network against the noise in input sequences. A multi-modal feature fusion method is also proposed for our ST-LSTM framework. The experimental results have validated the contributions and demonstrated the effectiveness of our approach

which achieves better performance over the existing state-of-the-art methods on seven challenging benchmark datasets.

Chapter 4

Skeleton-Based Action Recognition with Global Context-Aware Attention LSTM Networks

In the previous chapter, a spatio-temporal LSTM network is introduced for skeleton-based action recognition. However, it does not have explicit capability to selectively focus on the informative (important) body joints of each action sequence. In this chapter, we improve its design and propose a Global Context-Aware Attention LSTM network to emphasize the features of the informative joints that are more relevant to the action performed in a sequence.

4.1 Introduction

As shown in [143, 3], human actions can be represented by a combination of the motions of skeletal joints in 3D space. However, this does not indicate all joints in the skeleton sequence are informative for action recognition. For instance, the hand joints' motions are quite informative for the action *clapping*, while the movements of the foot

joints are not. Different action sequences often have different informative joints, and in the same sequence, the informativeness degree of a body joint may also change over the frames. Thus, it is beneficial to selectively focus on the informative joints in each frame of the sequence, and try to ignore the features of the irrelevant ones, as the latter contribute very little for action recognition, and even bring noise which corrupts the performance [129]. This selectively focusing scheme can also be called *attention*, which has been demonstrated to be quite useful for various tasks, such as speech recognition [144], image caption generation [96], machine translation [145], and so on.

Long Short-Term Memory (LSTM) networks have strong power in handling sequential data [114]. They have been successfully applied to language modeling [94], RGB based video analysis [103, 67, 108, 111, 109, 64, 146, 98, 104], and also skeleton-based action recognition [3, 5, 21]. However, the original LSTM does not have strong attention capability for action recognition. This limitation is mainly owing to LSTM's restriction in perceiving the global context information of the video sequence, which is, however, often very important for the global classification problem – skeleton-based action recognition.

In order to perform reliable attention over the skeletal joints, we need to assess the informativeness degree of each joint in each frame with regarding to the global action sequence. This indicates that we need to have global contextual knowledge first. However, the available context information at each evolution step of LSTM is relatively local. In LSTM, the sequential data is fed to the network as input step by step. Accordingly, the context information (hidden representation) of each step is fed to the next one. This implies the available context at each step is the hidden representation

from the previous step, which is quite local when compared to the global information [1].

In this chapter, we extend the original LSTM model and propose a Global Context-Aware Attention LSTM (GCA-LSTM) network which has strong attention capability for skeleton-based action recognition. In our method, the global context information is fed to all evolution steps of the GCA-LSTM. Therefore, the network can use it to measure the informativeness scores of the new inputs at all steps, and adjust the attention weights for them accordingly, i.e., if a new input is informative regarding to the global action, then the network takes advantage of more information of it at this step, on the contrary, if it is irrelevant, then the network blocks the input at this step.

Our proposed GCA-LSTM network for skeleton-based action recognition includes a global context memory cell and two LSTM layers, as illustrated in Fig. 4.1. The first LSTM layer is used to encode the skeleton sequence and initialize the global context memory cell. And the representation of the global context memory is then fed to the second LSTM layer to assist the network to selectively focus on the informative joints in each frame, and further generate an attention representation for the action sequence. Then the attention representation is fed back to the global context memory cell in order to refine it. Moreover, we propose a recurrent attention mechanism for our GCA-LSTM network. As a refined global context memory is produced after the attention procedure, the global context memory can be fed to the second LSTM layer again to perform attention more reliably. We carry out multiple attention iterations to optimize the global context memory progressively. Finally, the refined global context is fed to the softmax classifier to predict the action class.

In addition, we also extend the aforementioned design of our GCA-LSTM network in this chapter, and further propose a two-stream GCA-LSTM, which incorporates

[1] Although in LSTM, the hidden representations of the latter steps contain wider range of context information than that of the initial steps, their context is still relatively local, as LSTM has trouble in remembering information too far in the past [147].

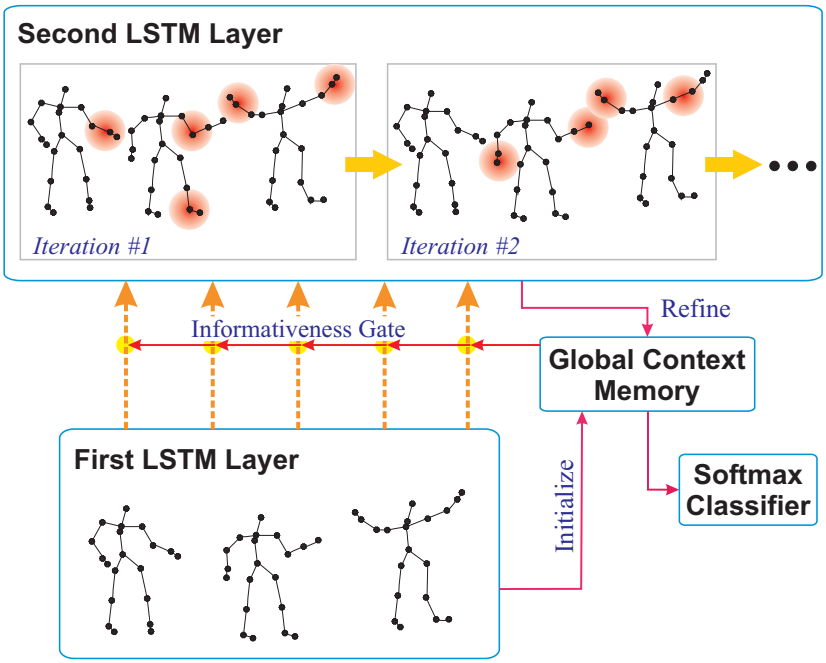


Fig. 4.1 Skeleton-based human action recognition with the Global Context-Aware Attention LSTM network. The first LSTM layer encodes the skeleton sequence and generates an initial global context representation for the action sequence. The second layer performs attention over the inputs by using the global context memory cell to achieve an attention representation for the sequence. Then the attention representation is used back to refine the global context. Multiple attention iterations are performed to refine the global context memory progressively. Finally, the refined global context information is utilized for classification.

fine-grained (joint-level) attention and coarse-grained (body part-level) attention, in order to achieve more accurate action recognition results.

The contributions of this chapter are summarized as follows: (1) A GCA-LSTM model is proposed, which retains the sequential modeling ability of the original LSTM, meanwhile promoting its selective attention capability by introducing a global context memory cell. (2) A recurrent attention mechanism is proposed, with which the attention performance of our network can be improved progressively. (3) A stepwise training scheme is proposed to more effectively train the network. (4) We further extend the design of our GCA-LSTM model, and propose a more powerful two-stream GCA-LSTM network. (5) The proposed end-to-end network yields state-of-the-art performance on the evaluated benchmark datasets.

The rest of this chapter is organized as follows. In Section 4.2, we introduce the proposed GCA-LSTM network. In Section 4.3, we introduce the two-stream attention framework. We provide the experimental results in Section 4.4. Finally, we conclude the chapter in Section 4.5.

4.2 GCA-LSTM Network

4.2.1 Spatio-Temporal LSTM

In order to put our proposed network into context, we first briefly review the spatio-temporal LSTM (ST-LSTM) model, since our GCA-LSTM network is based on this model.

In a generic skeleton-based human action recognition problem, the 3D coordinates of the major body joints in each frame are provided. The spatial dependence of different joints in the same frame and the temporal dependence of the same joint among different frames are both crucial cues for skeleton-based action analysis.

In Chapter 3, we introduce a ST-LSTM network for skeleton-based action recognition, which is capable of modeling the dependency structure and context information in both spatial and temporal domains simultaneously. In ST-LSTM model, the skeletal joints in a frame are arranged and fed as a chain (the spatial direction), and the corresponding joints over different frames are also fed in a sequence (the temporal direction).

Specifically, each ST-LSTM unit is fed with a new input ($x_{j,t}$, the 3D location of joint j in frame t), the hidden representation of the same joint at the previous time step ($h_{j,t-1}$), and also the hidden representation of the previous joint in the same frame ($h_{j-1,t}$), where $j \in \{1,...,J\}$ and $t \in \{1,...,T\}$ denote the indices of joints and frames, respectively. The ST-LSTM unit has an input gate ($i_{j,t}$), two forget gates corresponding to the two sources of context information ($f_{j,t}^{(T)}$ for the temporal dimension, and $f_{j,t}^{(S)}$ for the spatial domain), together with an output gate ($o_{j,t}$). The transition equations of ST-LSTM are formulated as:

$$\begin{pmatrix} i_{j,t} \\ f_{j,t}^{(S)} \\ f_{j,t}^{(T)} \\ o_{j,t} \\ u_{j,t} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left(W \begin{pmatrix} x_{j,t} \\ h_{j-1,t} \\ h_{j,t-1} \end{pmatrix} \right) \tag{4.1}$$

$$\begin{aligned} c_{j,t} &= i_{j,t} \odot u_{j,t} \\ &\quad + f_{j,t}^{(S)} \odot c_{j-1,t} \\ &\quad + f_{j,t}^{(T)} \odot c_{j,t-1} \end{aligned} \tag{4.2}$$

$$h_{j,t} = o_{j,t} \odot \tanh(c_{j,t}) \tag{4.3}$$

where $c_{j,t}$ and $h_{j,t}$ denote the cell state and hidden representation of the unit at the spatio-temporal step (j,t), respectively, $u_{j,t}$ is the modulated input, $\odot$ denotes the

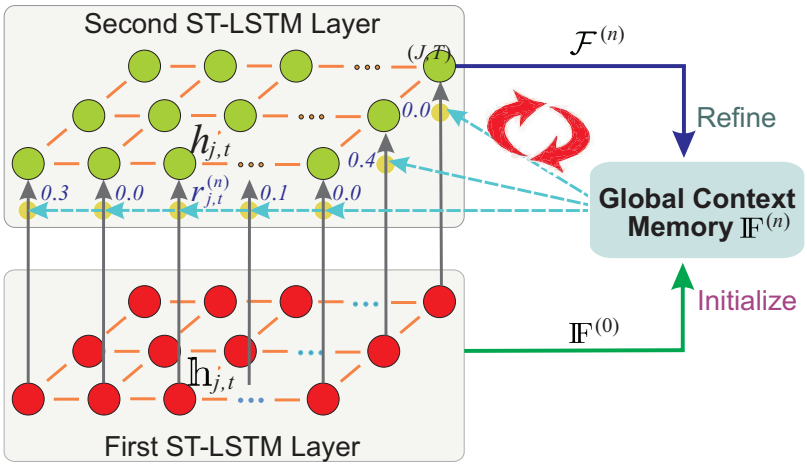


Fig. 4.2 Illustration of our GCA-LSTM network. Some arrows are omitted for clarity.

element-wise product, and W is an affine transformation consisting of model parameters.

4.2.2 Global Context-Aware Attention LSTM

Several previous works [129, 46] have shown that in each action sequence, there is often a subset of informative joints which are important as they contribute much more to action analysis, while the remaining ones may be irrelevant (or even noisy) for this action. As a result, to obtain a high accuracy of action recognition, we need to identify the informative skeletal joints and concentrate more on their features, meanwhile trying to ignore the features of the irrelevant ones, i.e., selectively focusing (*attention*) on the informative joints is useful for human action recognition.

Human action can be represented by a combination of skeletal joints' movements. In order to reliably identify the informative joints in an action instance, we can evaluate the informativeness score of each joint in each frame with regarding to the global action sequence. To achieve this purpose, we need to obtain the global context information first. However, the available context at each evolution step of LSTM is the hidden

representation from the previous step, which is relatively local when compared to the global action.

To mitigate the aforementioned limitation, we propose to introduce a global context memory cell for the LSTM model, which keeps the global context information of the action sequence, and can be fed to each step of LSTM to assist the attention procedure, as illustrated in Fig. 4.2. We call this new LSTM architecture as Global Context-Aware Attention LSTM (GCA-LSTM).

Overview of the GCA-LSTM network

We illustrate the proposed GCA-LSTM network for skeleton-based action recognition in Fig. 4.2. Our GCA-LSTM network contains three major modules. The *global context memory cell* maintains an overall representation of the whole action sequence. The *first ST-LSTM layer* encodes the skeleton sequence, and initializes the global context memory cell. The *second ST-LSTM layer* performs attention over the inputs at all spatio-temporal steps to generate an attention representation of the action sequence, which is then used to refine the global context memory.

The input at the spatio-temporal step (j,t) of the first ST-LSTM layer is the 3D coordinates of the joint j in frame t. The inputs of the second layer are the hidden representations from the first layer.

Multiple attention iterations (recurrent attention) are performed in our network to refine the global context memory iteratively. Finally, the refined global context memory can be used for classification.

To facilitate our explanation, we use $\mathbb{h}_{j,t}$ instead of $h_{j,t}$ to denote the hidden representation at the step (j,t) in the first ST-LSTM layer, while the symbols, including $h_{j,t}$, $c_{j,t}$, $i_{j,t}$, and $o_{j,t}$, which are defined in Section 4.2.1, are utilized to represent the components in the second layer only.

Initializing the Global Context Memory Cell

Our GCA-LSTM network performs attention by using the global context information, therefore, we need to obtain an initial global context memory first.

A feasible scheme is utilizing the outputs of the first layer to generate a global context representation. We can average the hidden representations at all spatio-temporal steps of the first layer to compute an initial global context memory cell ($\mathbb{F}^{(0)}$) as follows:

$$\mathbb{F}^{(0)} = \frac{1}{JT} \sum_{j=1}^{J} \sum_{t=1}^{T} \mathbb{h}_{j,t} \tag{4.4}$$

We may also concatenate the hidden representations of the first layer and feed them to a feed-forward neural network, then use the resultant activation as $\mathbb{F}^{(0)}$. We empirically observe these two initialization schemes perform similarly.

Performing Attention in the Second ST-LSTM Layer

By using the global context information, we evaluate the informativeness degree of the input at each spatio-temporal step in the second ST-LSTM layer.

In the n-th attention iteration, our network learns an informativeness score ($r_{j,t}^{(n)}$) for each input ($\mathbb{h}_{j,t}$) by feeding the input itself, together with the global context memory cell ($\mathbb{F}^{(n-1)}$) generated by the previous attention iteration to a network as follows:

$$e_{j,t}^{(n)} = W_{e_1} \left(\tanh \left(W_{e_2} \begin{pmatrix} \mathbb{h}_{j,t} \\ \mathbb{F}^{(n-1)} \end{pmatrix} \right) \right) \tag{4.5}$$

$$r_{j,t}^{(n)} = \frac{\exp(e_{j,t}^{(n)})}{\sum_{u=1}^{J} \sum_{v=1}^{T} \exp(e_{u,v}^{(n)})} \tag{4.6}$$

where $r_{j,t}^{(n)} \in (0,1)$ denotes the normalized informativeness score of the input at the step (j,t) in the n-th attention iteration, with regarding to the global context information.

The informativeness score $r_{j,t}^{(n)}$ is then used as a gate of the ST-LSTM unit, and we call it *informativeness gate*. With the assistance of the learned informativeness gate, the cell state of the unit in the second ST-LSTM layer can be updated as:

$$\begin{aligned} c_{j,t} &= r_{j,t}^{(n)} \odot i_{j,t} \odot u_{j,t} \\ &\quad + (1 - r_{j,t}^{(n)}) \odot f_{j,t}^{(S)} \odot c_{j-1,t} \\ &\quad + (1 - r_{j,t}^{(n)}) \odot f_{j,t}^{(T)} \odot c_{j,t-1} \end{aligned} \tag{4.7}$$

The cell state updating scheme in Eq. (4.7) can be explained as follows: (1) if the input $(\mathbb{h}_{j,t})$ is informative (important) with regarding to the global context representation, then we let the learning algorithm update the cell state of the second ST-LSTM layer by importing more information of it; (2) on the contrary, if the input is irrelevant, then we need to block the input gate at this step, meanwhile relying more on the history information of the cell state.

Refining the Global Context Memory Cell

We perform attention by adopting the cell state updating scheme in Eq. (4.7), and thereby obtain an attention representation of the action sequence. Concretely, the output of the last spatio-temporal step in the second layer is used as the attention representation ($\mathscr{F}^{(n)}$) for the action. Finally, the attention representation $\mathscr{F}^{(n)}$ is fed to the global context memory cell to refine it, as illustrated in Fig. 4.2. The refinement is formulated as follows:

$$\mathbb{F}^{(n)} = \mathrm{ReLu}\left(W_F^{(n)} \begin{pmatrix} \mathscr{F}^{(n)} \\ \mathbb{F}^{(n-1)} \end{pmatrix} \right) \tag{4.8}$$

where $\mathbb{F}^{(n)}$ is the refined version of $\mathbb{F}^{(n-1)}$. Note that $W_F^{(n)}$ is not shared over different iterations.

Multiple attention iterations (recurrent attention) are carried out in our GCA-LSTM network. Our motivation is that after we obtain a refined global context memory cell, we can use it to perform the attention again to more reliably identify the informative joints, and thus achieve a better attention representation, which can then be utilized to further refine the global context. After multiple iterations, the global context can be more discriminative for action classification.

Classifier

The last refined global context memory cell $\mathbb{F}^{(N)}$ is fed to a softmax classifier to predict the class label:

$$\hat{y} = \text{softmax}\left(W_c\left(\mathbb{F}^{(N)}\right)\right) \tag{4.9}$$

The negative log-likelihood loss function [121] is adopted to measure the difference between the true class label y and the prediction result $\hat{y}$. The back-propagation algorithm is used to minimize the loss function. The details of the back-propagation process are described in Section 4.2.3.

4.2.3 Training the Network

In this part, we first briefly describe the basic training method which directly optimizes the parameters of the whole network, we then propose a more advanced stepwise training scheme for our GCA-LSTM network.

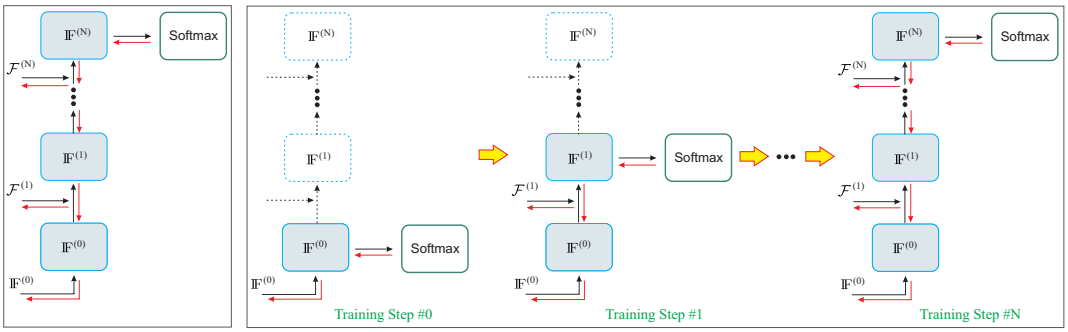


Fig. 4.3 Illustration of the two network training methods. (a) Directly train the whole network. (b) Stepwise optimize the network parameters. In this figure, the global context memory cell $\mathbb{F}^{(n)}$ is unfolded over the attention iterations. The training step #n corresponds to the n-th attention iteration. The black and red arrows denote the forward and backward passes, respectively. Some passes, such as those between the two ST-LSTM layers, are omitted for clarity. Better viewed in colour.

Directly Train the Whole Network

Since the classification is performed by using the last refined global context, to train such a network, it is natural and intuitive to feed the action label as the training output at the last attention iteration, and back-propagate the errors from the last step, i.e., directly optimize the whole network as shown in Fig. 4.3(a).

Stepwise Training

Owing to the recurrent attention mechanism, there are frequent mutual interactions among different modules (the two ST-LSTM layers and the global context memory cell, see Fig. 4.2) in our network. Moreover, during the progress of multiple attention iterations, new parameters are also introduced. Due to these facts, it is rather difficult to simply optimize all parameters and all attention iterations of the whole network directly as mentioned above.

Therefore, we propose a stepwise training scheme for our GCA-LSTM network, which optimizes the model parameters incrementally. The details of this scheme are depicted in Fig. 4.3(b) and Algorithm 1.

Algorithm 1 Stepwise train the GCA-LSTM network.

```
Randomly initialize the parameters of the whole network with zero-mean Gaussian.
for n = 0 to N do   // n is the training step
    Feed the action label as the training output at the
    attention iteration n.
    do
        Training an epoch: optimizing the parameters used
        in the iterations 0 to n via back-propagation.
    while Validation error is decreasing
end for
```

The proposed stepwise training scheme is effective and efficient in optimizing the parameters and ensuring the convergence of the GCA-LSTM network. Specifically, at each training step n, we only need to optimize a subset of parameters and modules which are used by the attention iterations 0 to n. [2] Training this shrunken network is more effective and efficient than directly training the whole network. At the step $n+1$, a larger scale network needs to be optimized. However, the training at step $n+1$ is also very efficient, as most of the parameters and passes have already been optimized (pre-trained well) by its previous training steps.

4.3 Two-stream GCA-LSTM Network

In the aforementioned design (Section 4.2), the GCA-LSTM network performs action recognition by selectively focusing on the informative joints in each frame, i.e., the attention is carried out at joint level (fine-grained attention). Beside fine-grained attention, coarse-grained attention can also contribute to action analysis. This is because some actions are often performed at body part level. For these actions, all the joints from the same informative body part tend to have similar importance degrees. For example, the postures and motions of all the joints (elbow, wrist, palm, and finger)

[2] Note that #0 is not an attention iteration, but the process of initializing the global context memory cell ($\mathbb{F}^{(0)}$). To facilitate the explantation of the stepwise training, we here temporally describe it as an attention iteration.

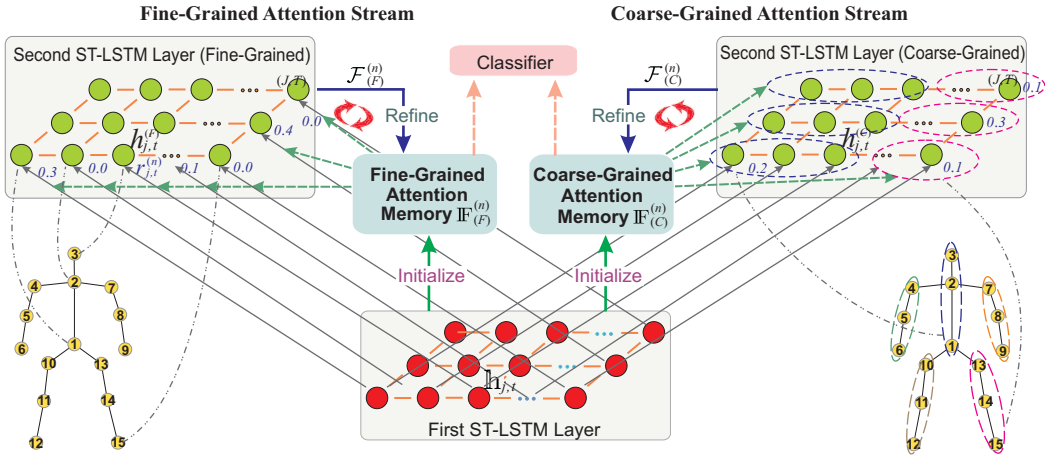


Fig. 4.4 Illustration of the two-stream GCA-LSTM network, which incorporates fine-grained (joint-level) attention and coarse-grained (body part-level) attention. To perform coarse-grained attention, the joints in a skeleton are divided into five body parts, and all the joints from the same body part share a same informative score. In the second ST-LSTM layer for coarse-grained attention, we only show two body parts at each frame, and other body parts are omitted for clarity.

from the right hand are all important for recognizing the action *salute* in the NTU RGB+D dataset [1], i.e., we need to identify the informative body part "right hand" here. This implies coarse-grained (body part-level) attention is also useful for action recognition.

As suggested by Du *et al.* [3], the human skeleton can be divided into five body parts (torso, left hand, right hand, left leg, and right leg) based on the human physical structure. These five parts are illustrated as the right part of Fig. 4.4. Therefore, we can measure the informativeness degree of each body part with regarding to the action sequence, and then perform coarse-grained attention.

Specifically, we extend the design of out GCA-LSTM model, and introduce a two-stream GCA-LSTM network here, which jointly takes advantage of a fine-grained (joint-level) attention stream and a coarse-grained (body part-level) attention stream.

The architecture of the two-stream GCA-LSTM is illustrated in Fig. 4.4. In each attention stream, there is a global context memory cell to maintain the global attention representation of the action sequence, and also a second ST-LSTM layer to perform

attention. This indicates we have two separated global context memory cells in the whole architecture, which are respectively the fine-grained attention memory cell ($\mathbb{IF}_{(F)}^{(n)}$) and the coarse-grained attention memory cell ($\mathbb{IF}_{(C)}^{(n)}$). The first ST-LSTM layer, which is used to encode the skeleton sequence and initialize the global context memory cells, is shared by the two attention streams.

The process flow (including initialization, attention, and refinement) in the fine-grained attention stream is the same as the GCA-LSTM model introduced in Section 4.2. The operation in the coarse-grained attention stream is also similar. The main difference is that, in the second layer, the coarse-grained attention stream performs attention by selectively focusing on the informative body parts in each frame.

Concretely, in the attention iteration n, the network learns an informativeness score ($r_{P,t}^{(n)}$) for each body part P ($P \in \{1,2,3,4,5\}$) as:

$$e_{P,t}^{(n)} = W_{e_3}\left(\tanh\left(W_{e_4}\begin{pmatrix}\bar{\mathbb{h}}_{P,t}\\ \mathbb{IF}_{(C)}^{(n-1)}\end{pmatrix}\right)\right) \tag{4.10}$$

$$r_{P,t}^{(n)} = \frac{\exp(e_{P,t}^{(n)})}{\sum_{u=1}^{5}\sum_{v=1}^{T}\exp(e_{u,v}^{(n)})} \tag{4.11}$$

where $\bar{\mathbb{h}}_{P,t}$ is the representation of the body part P at frame t, which is calculated based on the hidden representations of all the joints that belong to P, with average pooling as:

$$\bar{\mathbb{h}}_{P,t} = \frac{1}{J_P}\sum_{j\in P}\mathbb{h}_{j,t} \tag{4.12}$$

where J_P denotes the number of joints in body part P.

To perform coarse-grained attention, we allow each joint j in body part P to share the informativeness degree of P, i.e., at frame t, all the joints in P use the same informativeness score $r_{P,t}^{(n)}$, as illustrated in Fig. 4.4. Hence, in the coarse-grained attention stream, if $j \in P$, then the cell state of the second ST-LSTM layer is updated

at the spatio-temporal step (j,t) as:

$$\begin{aligned} c_{j,t} &= r_{P,t}^{(n)} \odot i_{j,t} \odot u_{j,t} \\ &\quad + (1 - r_{P,t}^{(n)}) \odot f_{j,t}^{(S)} \odot c_{j-1,t} \\ &\quad + (1 - r_{P,t}^{(n)}) \odot f_{j,t}^{(T)} \odot c_{j,t-1} \end{aligned} \tag{4.13}$$

Multiple attention iterations are also performed in the proposed two-stream GCA-LSTM network. Finally, the refined fine-grained attention memory $\mathbb{F}_{(F)}^{(N)}$ and coarse-grained attention memory $\mathbb{F}_{(C)}^{(N)}$ are both fed to the softmax classifier, and the prediction scores of these two streams are averaged for action recognition.

The proposed step-wise training scheme can also be applied to this two-stream GCA-LSTM network, and at the training step #n, we simultaneously optimize the two attention streams, both of which correspond to the n-th attention iteration.

4.4 Experiments

We evaluate our proposed method on the NTU RGB+D [1], SYSU-3D [85], UT-Kinect [7], SBU-Kinect Interaction [39], and Berkeley MHAD [88] datasets. To investigate the effectiveness of our approach, we conduct extensive experiments with the following different network structures:

- "ST-LSTM + Global (1)". This network architecture is similar to the original two-layer ST-LSTM network in [21], but the hidden representations at all spatio-temporal steps of the second layer are concatenated and fed to a one-layer feed-forward network to generate a global representation of the skeleton sequence, and the classification is performed on the global representation; while in [21], the classification is performed on single hidden representation at each spatio-temporal step (local representation).

- "ST-LSTM + Global (2)". This network structure is similar to the above "ST-LSTM + Global (1)", except that the global representation is obtained by averaging the hidden representations of all spatio-temporal steps.

- "GCA-LSTM". This is the proposed Global Context-Aware Attention LSTM network. Two attention iterations are performed by this network. The classification is performed on the last refined global context memory cell. The two training methods (*direct training* and *stepwise training*) described in Section 4.2.3 are also evaluated for this network structure.

In addition, we also adopt the large scale NTU RGB+D and the challenging SYSU-3D as two major benchmark datasets to evaluate the proposed "two-stream GCA-LSTM" network.

We use Torch7 framework [122] to perform our experiments. Stochastic gradient descent (SGD) algorithm is adopted to train our end-to-end network. We set the learning rate, decay rate, and momentum to 1.5×10^{-3}, 0.95, and 0.9, respectively. The applied dropout probability [55] in our network is set to 0.5. The dimensions of the global context memory representation and the cell state of ST-LSTM are both 128.

4.4.1 Experiments on the NTU RGB+D Dataset

There are two standard evaluation protocols for the NTU RGB+D dataset [1]: (1) Cross subject (CS): 20 subjects are used for training, and the remaining subjects are used for testing; (2) Cross view (CV): two camera views are used for training, and one camera view is used for testing. To extensively evaluate the proposed method, both protocols are tested in our experiment.

We compare the proposed GCA-LSTM network with state-of-the-art approaches, as shown in Table 4.1. We can observe that our proposed GCA-LSTM model outperforms the other skeleton-based methods. Specifically, our GCA-LSTM network outperforms

the original ST-LSTM network in [21] by 6.9% with the cross subject protocol, and 6.3% with the cross view protocol. This demonstrates that the attention mechanism in our network brings significant performance improvement.

Both "ST-LSTM + Global (1)" and "ST-LSTM + Global (2)" perform classification on the global representations, thus they achieve slightly better performance than the original ST-LSTM [21] which performs classification on local representations. We also observe "ST-LSTM + Global (1)" and "ST-LSTM + Global (2)" perform similarly.

The results in Table 4.1 also show that using the *stepwise training* method can improve the performance of our network in contrast to using the *direct training* method.

Table 4.1 Experimental results on the NTU RGB+D dataset.

Method	CS	CV
Skeletal Quads [43]	38.6%	41.4%
Lie Group [42]	50.1%	52.8%
Dynamic Skeletons [85]	60.2%	65.2%
HBRNN [3]	59.1%	64.0%
Deep RNN [1]	56.3%	64.1%
Deep LSTM [1]	60.7%	67.3%
Part-aware LSTM [1]	62.9%	70.3%
JTM CNN [148]	73.4%	75.2%
STA Model [149]	73.4%	81.2%
SkeletonNet [54]	75.9%	81.2%
Visualization CNN [150]	76.0%	82.6%
ST-LSTM [21]	69.2%	77.7%
ST-LSTM + Global (1)	70.5%	79.5%
ST-LSTM + Global (2)	70.7%	79.4%
GCA-LSTM (*direct training*)	74.3%	82.8%
GCA-LSTM (*stepwise training*)	**76.1%**	**84.0%**

We also evaluate the performance of the two-stream GCA-LSTM network, and report the results in Table 4.2. The results show that by incorporating fine-grained attention and coarse-grained attention, the proposed two-stream GCA-LSTM network achieves better performance than the GCA-LSTM with fine-grained attention only.

We also observe the performance of two-stream GCA-LSTM can be improved with the *stepwise training* method.

Table 4.2 Performance of the two-stream GCA-LSTM network on the NTU RGB+D dataset.

Method	CS	CV
GCA-LSTM (coarse-grained only)	74.1%	81.6%
GCA-LSTM (fine-grained only)	74.3%	82.8%
Two-stream GCA-LSTM	76.2%	84.7%
Two-stream GCA-LSTM with *stepwise training*	77.1%	85.1%

4.4.2 Experiments on the SYSU-3D Dataset

We follow the standard cross-validation protocol in [85] on the SYSU-3D dataset [85], in which 20 subjects are adopted for training the network, and the remaining subjects are kept for testing. We report the experimental results in Table 4.3. We can observe that our GCA-LSTM network surpasses the state-of-the-art skeleton-based methods in [6, 85, 21], which demonstrates the effectiveness of our approach in handling the task of action recognition in skeleton sequences. The results also show that our proposed *stepwise training* scheme is useful for our network.

Table 4.3 Experimental results on the SYSU-3D dataset.

Method	Accuracy
LAFF (SKL) [6]	54.2%
Dynamic Skeletons [85]	75.5%
ST-LSTM [151]	76.5%
ST-LSTM + Global (1)	76.8%
ST-LSTM + Global (2)	76.6%
GCA-LSTM (*direct training*)	77.8%
GCA-LSTM (*stepwise training*)	**78.6%**

Using this challenging dataset, we also evaluate the performance of the two-stream attention model. The results in Table 4.4 show that the two-stream GCA-LSTM network is effective for action recognition.

Table 4.4 Performance of the two-stream GCA-LSTM network on the SYSU-3D dataset.

Method	Accuracy
GCA-LSTM (coarse-grained only)	76.9%
GCA-LSTM (fine-grained only)	77.8%
Two-stream GCA-LSTM	78.8%
Two-stream GCA-LSTM with *stepwise training*	79.1%

4.4.3 Experiments on the UT-Kinect Dataset

We follow the standard leave-one-out-cross-validation protocol in [7] to evaluate our method on the UT-Kinect dataset [7]. Our approach yields state-of-the-art performance on this dataset, as shown in Table 4.5.

Table 4.5 Experimental results on the UT-Kinect dataset.

Method	Accuracy
Grassmann Manifold [125]	88.5%
Histogram of 3D Joints [7]	90.9%
Riemannian Manifold [128]	91.5%
Key-Pose-Motifs Mining [131]	93.5%
Action-Snippets and Activated Simplices [152]	96.5%
ST-LSTM [21]	97.0%
ST-LSTM + Global (1)	97.0%
ST-LSTM + Global (2)	97.5%
GCA-LSTM (*direct training*)	98.5%
GCA-LSTM (*stepwise training*)	**99.0%**

4.4.4 Experiments on the SBU-Kinect Interaction Dataset

We perform 5-fold cross-validation evaluation on the SBU-Kinect Interaction dataset by following the standard protocol in [39]. The experimental results are depicted in Table 4.6. In this table, HBRNN [3], Deep LSTM [5], Co-occurrence LSTM [5], and ST-LSTM [21] are all LSTM based models for action recognition in skeleton sequences, and are very relevant to our network. We can see that the proposed GCA-LSTM network achieves the best performance among all of these methods.

4.4.5 Experiments on the Berkeley MHAD Dataset

We adopt the standard experimental protocol on the Berkeley MHAD dataset, in which 7 subjects are used for training and the remaining 5 subjects are held out for testing. The results in Table 4.7 show that our method achieves very high accuracy (100%) on this dataset.

As the Berkeley MHAD dataset was collected with a motion capture system rather than a Kinect, thus the coordinates of the skeletal joints are relatively accurate. To evaluate the robustness with regarding to the input noise, we also investigate the performance of our GCA-LSTM network on this dataset by adding zero mean input

Table 4.6 Experimental results on the SBU-Kinect Interaction dataset.

Method	Accuracy
Yun *et al.* [39]	80.3%
CHARM [133]	83.9%
Ji *et al.* [132]	86.9%
HBRNN [3]	80.4%
Deep LSTM [5]	86.0%
Co-occurrence LSTM [5]	90.4%
SkeletonNet [54]	93.5%
ST-LSTM [21]	93.3%
GCA-LSTM (*direct training*)	94.1%
GCA-LSTM (*stepwise training*)	**94.9%**

Table 4.7 Experimental results on the Berkeley MHAD dataset

Method	Accuracy
Ofli *et al.* [37]	95.4%
Vantigodi *et al.* [140]	96.1%
Vantigodi *et al.* [141]	97.6%
Kapsouras *et al.* [142]	98.2%
ST-LSTM [21]	100%
GCA-LSTM (*direct training*)	100%
GCA-LSTM (*stepwise training*)	100%

noise to the skeleton sequences, and show the results in Table 4.8. We can see that even if we add noise with the standard deviation (σ) set to $12cm$ (which is significant noise in the scale of human body), the accuracy of our method is still very high (92.7%). This demonstrates that our method is quite robust against the input noise.

Table 4.8 Evaluation of robustness against the input noise. Gaussian noise $\mathcal{N}(0,\sigma^2)$ is added to the 3D coordinates of the skeletal joints.

Standard deviation (σ) of noise	$0.1cm$	$1cm$	$2cm$	$4cm$
Accuracy	100%	99.3%	98.5%	97.5%
Standard deviation (σ) of noise	$8cm$	$12cm$	$16cm$	$32cm$
Accuracy	95.6%	92.7%	80.4%	61.5%

4.4.6 Evaluation of Attention Iteration Numbers

We also test the effect of different attention iteration numbers on our GCA-LSTM network, and show the results in Table 4.9. We can observe that increasing the iteration number can help to strength the classification performance of our network (using 2 iterations obtains higher accuracies compared to using only 1 iteration). This demonstrates that the recurrent attention mechanism proposed by us is useful for the GCA-LSTM network.

Table 4.9 Performance comparison of different attention iteration numbers (N).

#Attention Iteration	NTU RGB+D (CS)	NTU RGB+D (CV)	UT-Kinect	SYSU-3D	Berkeley MHAD
1	72.9%	81.8%	98.0%	77.8%	**100%**
2	**76.1%**	**84.0%**	**99.0%**	**78.6%**	**100%**

Specifically, we also evaluate the performance of 3 attention iterations by using the large scale NTU RGB+D dataset, and the results are shown in Table 4.10. We find the performance of 3 attention iterations is slightly better than 2 iterations if we share the parameters over different attention iterations (see columns (a) and (b) in Table 4.10). This consistently shows using multiple attention iterations can improve the performance of our network progressively. We do not try more iterations due to the GPU's memory limitation.

We also find that if we do not share the parameters over different attention iterations (see columns (c) and (d) in Table 4.10), then too many iterations can bring performance degradation (the performance of using 3 iterations is worse than that of using 2 iterations). In our experiment, we observe the performance degradation is caused by over-fitting (increasing iteration number will introduce new parameters if we do not share parameters). But the performance of two iterations is still significantly better than one iteration in this case. We will also give the experimental analysis of the parameter sharing schemes detailed in Section 4.4.7.

4.4.7 Evaluation of Parameter Sharing Schemes

As formulated in Eq. (4.5), the model parameters W_{e_1} and W_{e_2} are introduced for calculating the informativeness score at each spatio-temporal step in the second layer. Also multiple attention iterations are carried out in this layer. To regularize the parameter number inside our network and improve the generalization capability, we investigate two parameter sharing strategies for our network: (1) Sharing within

Table 4.10 Performance comparison of different parameter sharing schemes.

#Attention Iteration	(a) w/o sharing within iteration w/ sharing cross iterations	(b) w/ sharing within iteration w/ sharing cross iterations
1	71.0%	72.9%
2	73.0%	74.3%
3	**73.1%**	**74.4%**

#Attention Iteration	(c) w/o sharing within iteration w/o sharing cross iterations	(d) w/ sharing within iteration w/o sharing cross iterations
1	71.0%	72.9%
2	**73.4%**	**76.1%**
3	69.3%	73.2%

iteration: W_{e_1} and W_{e_2} are shared by all spatio-temporal steps in the same attention iteration; (2) Sharing cross iterations: W_{e_1} and W_{e_2} are shared over different attention iterations. We investigate the effect of these two parameter sharing strategies on our GCA-LSTM network, and report the results in Table 4.10.

In Table 4.10, we can observe that: (1) Sharing parameters within iteration is useful for enhancing the generalization capability of our network, as the performance in columns (b) and (d) of Table 4.10 is better than (a) and (c), respectively. (2) Sharing parameters over different iterations is also helpful for handling the over-fitting issues, but it may limit the representation capacity, as the network with two attention iterations which shares parameters within iteration but does not share parameters over iterations achieves the best result (see column (d) of Table 4.10). As a result, in our GCA-LSTM network, we only share the parameters within iteration, and two attention iterations are used.

4.4.8 Evaluation of Training Methods

The previous experiments showed that using the *stepwise training* method can improve the performance of our network in contrast to using *direct training* (see Table 4.1, 4.5, 4.3, 4.6). To further investigate the performance of these two training methods, we plot the convergence curves of our GCA-LSTM network in Fig. 4.5.

We analyze the convergence curves (Fig. 4.5) of the *stepwise training* method as follows. By using the proposed *stepwise training* method, at the training step #0, we only need to train the subnetwork for initializing the global context ($\mathbb{F}^{(0)}$), i.e., only a subset of parameters and modules need to be optimized, thus the training is very efficient and the loss curve converges very fast. When the validation loss stops decreasing, we start the next training step #1. Step #1 contains new parameters and modules for the first attention iteration, which have not been optimized yet, therefore, loss increases immediately at this epoch. However, most of the parameters involved at this step have already been pre-trained well by the previous step #0, thus the network training is quite effective, and the loss drops to a very low value after only one training epoch.

By comparing the convergence curves of the two training methods, we can find (1) the network converges much faster if we use *stepwise training*, compared to *directly train* the whole network. We can also observe that (2) the network is easier to get over-fitted by using *direct training* method, as the gap between the train loss and validation loss starts to rise after the 20th epoch. These observations demonstrate that the proposed *stepwise training* scheme is quite useful for effectively and efficiently training our GCA-LSTM network.

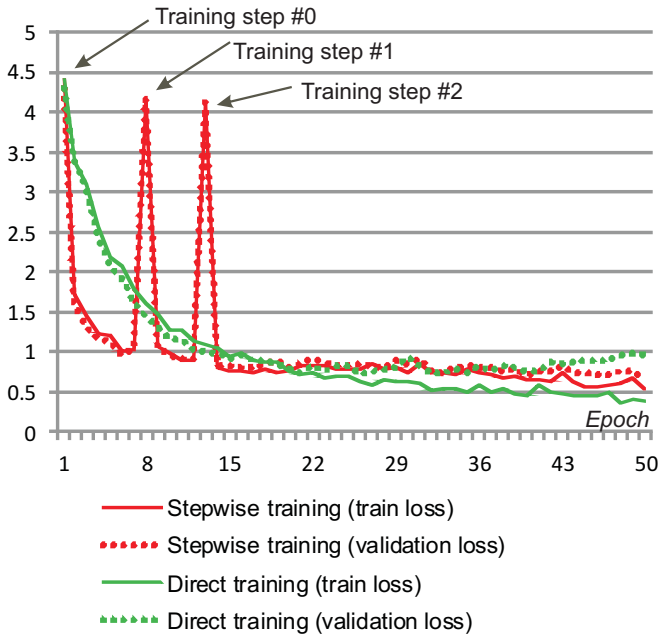


Fig. 4.5 Convergence curves of the GCA-LSTM network with two attention iterations by respectively using *stepwise training* (in red) and *direct training* (in green) on the NTU RGB+D dataset. Better viewed in colour.

4.4.9 Evaluation of Initialization Methods and Attention Designs

In Section 4.2.2, we introduce two methods to initialize the global context memory cell ($\mathbb{F}^{(0)}$). The first is averaging the hidden representations of the first layer (see Eq. (4.4)), and the second is using a one-layer feed-forward network to obtain $\mathbb{F}^{(0)}$. We compare these two initialization methods in Table 4.11. The results show that these two methods perform similarly. In our experiment, we also find that by using feed-forward network, the model converges faster, thus the scheme of feed-forward network is used to initialize the global context memory cell in our GCA-LSTM network.

Table 4.11 Performance comparison of different methods of initializing the global context memory cell.

Method	NTU RGB+D (CS)	NTU RGB+D (CV)
Averaging	73.8%	83.1%
Feed-forward network	74.3%	82.8%

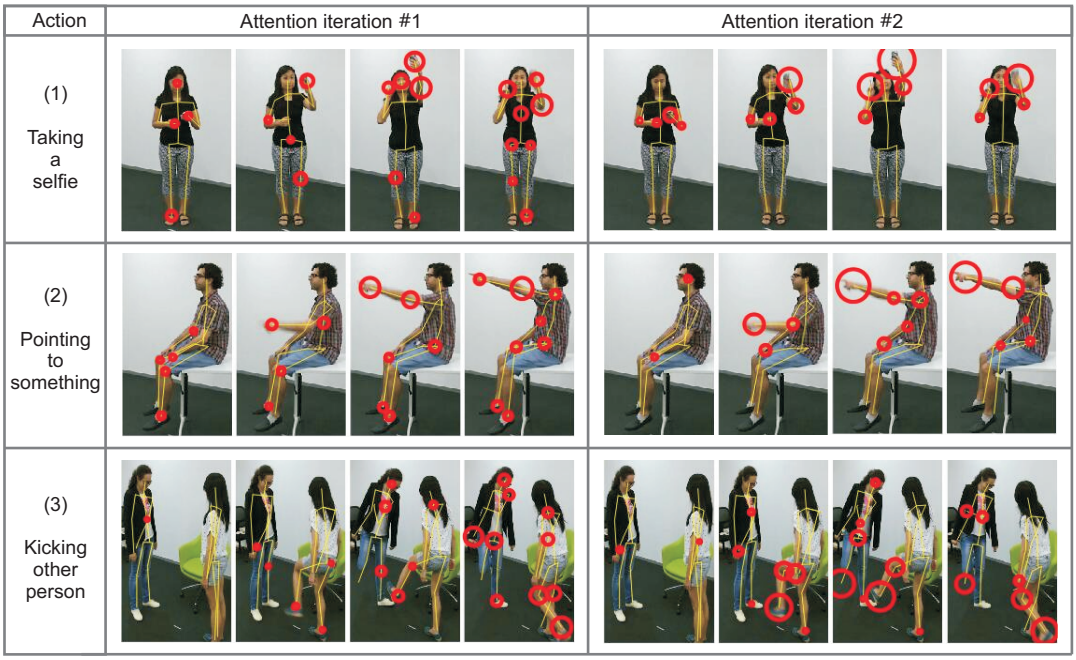


Fig. 4.6 Examples of qualitative results on the NTU RGB+D dataset. Three actions (*taking a selfie*, *pointing to something*, and *kicking other person*) are illustrated. The informativeness scores of two attention iterations are visualized. Four frames are shown for each iteration. The circle size indicates the magnitude of the informativeness score for the corresponding joint in a frame. For clarity, the joints with tiny informativeness scores are not shown.

In the GCA-LSTM network, the informativeness score $r_{j,t}^{(n)}$ is used as a gate within LSTM neuron, as formulated in Eq. (4.7). We also explore to replace this scheme with soft attention method [96, 153], i.e., the attention representation $\mathscr{F}^{(n)}$ is calculated as $\sum_{j=1}^{J}\sum_{t=1}^{T} r_{j,t}^{(n)} \mathbb{h}_{j,t}$. Using soft attention, the accuracy drops about one percentage point on the NTU RGB+D dataset. This can be explained as equipping LSTM neuron with gate $r_{j,t}^{(n)}$ provides LSTM better insight about when to update, forget or remember. In addition, it can keep the sequential ordering information of the inputs $\mathbb{h}_{j,t}$, while soft attention loses ordering and positional information.

4.4.10 Visualizations

To better understand our network, we analyze and visualize the informativeness score evaluated by using the global context information on the large scale NTU RGB+D dataset in this section.

We analyze the variations of the informativeness scores over the two attention iterations to verify the effectiveness of the recurrent attention mechanism in our method, and show the qualitative results of three actions (*taking a selfie*, *pointing to something*, and *kicking other person*) in Fig. 4.6. The informativeness scores are computed with soft attention for visualization. In this figure, we can see that the attention performance increases between the two attention iterations. In the first iteration, the network tries to identify the potential informative joints over the frames. After this attention, the network achieves a good understanding of the global action. Then in the second iteration, the network can more accurately focus on the informative joints in each frame of the skeleton sequence. We can also find that the informativeness score of the same joint can vary in different frames. This indicates that *our network performs attention not only in spatial domain, but also in temporal domain*.

In order to further quantitatively evaluate the effectiveness of the attention mechanism, we analyze the classification accuracies of the three action classes in Fig. 4.6 among all the actions. We observe if the attention mechanism is not used, the accuracies of these three classes are 67.7%, 71.7%, and 81.5%, respectively. However, if we use one attention iteration, the accuracies rise to 67.8%, 72.4%, and 83.4%, respectively. If two attention iterations are performed, the accuracies become 67.9%, 73.6%, and 86.6%, respectively.

To roughly explore which joints are more informative for the activities in the NTU RGB+D dataset, we also average the informativeness scores of the same joint in all the testing sequences, and visualize it in Fig. 4.7. We can observe that averagely, more

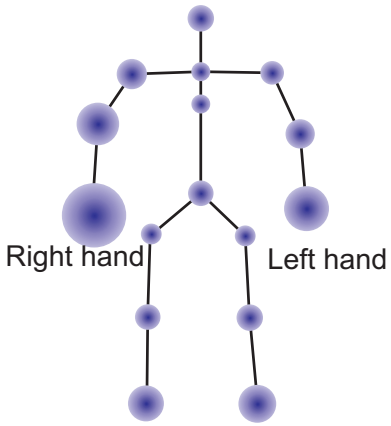


Fig. 4.7 Visualization of the average informativeness gates for all testing samples. The size of the circle around each joint indicates the magnitude of the corresponding informativeness score.

attention is assigned to the hand and foot joints. This is because in the NTU RGB+D dataset, most of the actions are related to the hand and foot postures and motions. We can also find that the average informativeness score of the right hand joint is higher than that of left hand joint. This indicates most of the subjects are right-handed.

4.5 Chapter Summary

In this chapter, we have extended the original LSTM network to construct a Global Context-Aware Attention LSTM (GCA-LSTM) network for skeleton-based action recognition, which has strong ability in selectively focusing on the informative joints in each frame of the skeleton sequence with the assistance of global context information. Furthermore, we have proposed a recurrent attention mechanism for our GCA-LSTM network, in which the selectively focusing capability is improved iteratively. In addition, a two-stream attention framework is also introduced. The experimental results validate the contributions of our approach by achieving state-of-the-art performance on five challenging datasets.

Chapter 5

Skeleton-Based Online Action Prediction Using Scale Selection Network

The ST-LSTM network introduced in Chapter 3 and the GCA-LSTM network introduced in Chapter 4 both focus on action recognition in well-segmented skeleton sequences, in which each sequence contains one action sample. In this chapter, we focus on a more challenging task: online action prediction in untrimmed skeleton sequences, in which each sequence contains multiple action instances.

5.1 Introduction

In action prediction (early action recognition), the goal is to predict the class label of an ongoing action using its observed part so far. Predicting actions before they get completely performed is a subset of a broader research domain on human activity analysis. It has attracted lots of attention due to its wide range of application in security surveillance, human-machine interaction, patient monitoring, etc [6, 60]. Most of the

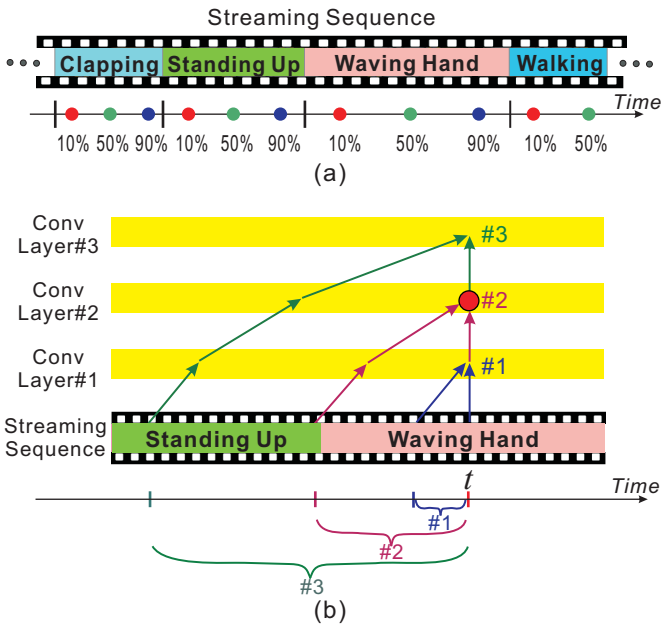


Fig. 5.1 Figure (a) illustrates an untrimmed streaming sequence that contains multiple action instances. We need to recognize the current ongoing action at each time step when only a part (*e.g.*, 10%) of it is performed. Figure (b) depicts our SSNet for online action prediction. At time t, only a part of the action *waving hand* is observed. Our SSNet selects the convolutional Layer #2 rather than #3 for prediction, as the perception window of #2 mainly covers the performed part of current action, while #3 involves too many frames from the previous action which can interfere the prediction at time step t.

existing works [6, 63, 154] focus on action prediction in well-segmented videos, for which each video contains only one action instance. However, in practical scenarios, such as online human-machine interaction systems, plenty of action instances are contained in a streaming sequence, which are not segmented. In this chapter, we address the more challenging task: "online action prediction in untrimmed video", *i.e.*, we want to recognize the current ongoing action from the observed part of it at each time step of the data stream, which can include multiple actions, as illustrated in Fig. 5.1(a).

We investigate real-time action prediction with the continuous 3D skeleton data in this chapter. To predict the class label of the current ongoing action at each time step,

we adopt a sliding window method over the frames of the streaming skeleton sequence, and the data frames inside the window are utilized to perform action prediction.

The sliding window design has been widely employed for a series of machine vision tasks, such as object recognition [155], pedestrian detection [156], activity detection [69, 80, 49, 82], etc. Most of these works utilize one fixed scale or combine multi-scale multi-pass scans at each sliding position. However, in our online action prediction task, we need to predict the ongoing action at each observation ratio, while there are significant temporal scale variations in the observed part of the ongoing action. This makes it quite difficult to determine the scale of the sliding window.

The untrimmed streaming sequence may contain multiple action instances, as shown in Fig. 5.1(a). The order of the actions can be arbitrary, and the durations of different instances are often not the same. Moreover, the observed (per whole) ratio of the ongoing action changes over time, which makes it even more challenging to obtain a proper temporal window scale for online prediction. For instance, at an early temporal stage, it is beneficial to use a relatively small scale, because the larger window sizes may include frames from the previous action instance which can mislead the recognition of the current instance. Conversely, if a large part of the current action has already been observed, it is beneficial to use a larger window size to cover more of its performed parts in order to achieve a reliable prediction.

In order to tackle the aforementioned challenges, in this chapter, a novel Scale Selection Network (SSNet) is proposed for online action prediction. Instead of using a fixed scale or multi-scale multi-pass scans at each time step, we supervise our network to dynamically choose the proper temporal window scale at each step to cover the performed part of the current action instance. In our approach, the network predicts the ongoing action at each frame. Beside predicting the class label, it also regresses the temporal distance to the beginning of current action instance, which indicates the

performed part of the ongoing action. Thus, at the next frame, we can utilize it as the temporal window scale for action class prediction.

In our network, we apply convolutional analysis in temporal dimension to model the motion dynamics over the frames for skeleton-based action prediction. A hierarchical architecture with dilated convolution layers is leveraged to learn a comprehensive representation over the frames within each perception window, such that different layers in our SSNet correspond to different temporal scales, as shown in Fig. 5.1(b). Therefore, at each time step, our network selects the *proper* convolutional layer which covers the most similar window scale regressed by its previous step. Then the activations of this layer can be used for action prediction. The proposed SSNet is designed to select the proper window in order to cover the performed part of current action and try to suppress the noisy data from the previous ones, hence it produces reliable predictions at each step. To the best of our knowledge, this is the first convolutional model with explicit temporal scale selection as its fundamental capability for handling scale variations in online activity analysis.

In many existing approaches that utilize sliding window designs, the computational efficiencies are often relatively low owing to the overlapping design and exhaustive multi-scale multi-round scans. In our method, the action prediction is performed with a regressed scale at each step, which avoids multi-pass scans. So the action prediction and scale selection are performed by a single convolutional network very efficiently. Moreover, we introduce an activation sharing scheme to deal with the overlapping computations over different time steps, which makes our SSNet run very fast for real-time online prediction.

In addition, to improve the performance of our network in handling the 3D skeleton data as input, we also propose a hierarchy of dilated tree convolutions to learn the multi-level structured semantic representations over the skeleton joints at each frame for our action prediction network.

The main contributions of this chapter are summarized as follows: (1) We study the new problem of real-time online action prediction in continuous 3D skeleton streams by leveraging convolutional analysis in temporal dimension. (2) Our proposed SSNet is capable of dealing with the scale variations of the observed portion of the ongoing action at different time steps. We propose a scale selection scheme to let our network choose the proper temporal scale at each step, such that the network can mainly focus on the performed part of the current action, and try to avoid the noise from the previous action samples. (3) A hierarchy of dilated tree convolutions are also proposed to learn multi-level structured representations for the input skeleton data and improve the performance of our SSNet for skeleton-based action prediction. (4) The proposed framework is very efficient for online action analysis thanks to the computation sharing over different time steps. (5) We perform action prediction with our SSNet which is end-to-end trainable, rather than using expensive multi-stage multi-network design at each step. (6) The proposed method achieves superior performance on four challenging datasets for 3D skeleton-based activity analysis.

The remainder of this chapter is organized as follows. In section 5.2, we introduce our proposed SSNet for skeleton-based online action prediction in detail. We present the experimental results and comparisons in section 5.3. Finally, we conclude the chapter in section 5.4.

5.2 The Proposed Method

We introduce the proposed network architecture, Scale Selection Network (SSNet), for skeleton-based online action prediction in this section. The overall schema of this method is illustrated in Fig. 5.2. In the proposed network, the one dimensional (1-D) convolutions are performed in temporal domain to model the motion dynamics over the frames. The inputs of SSNet are the frames within a temporal window at each time step. In order to tackle the scale variations in the partially observed action at different time

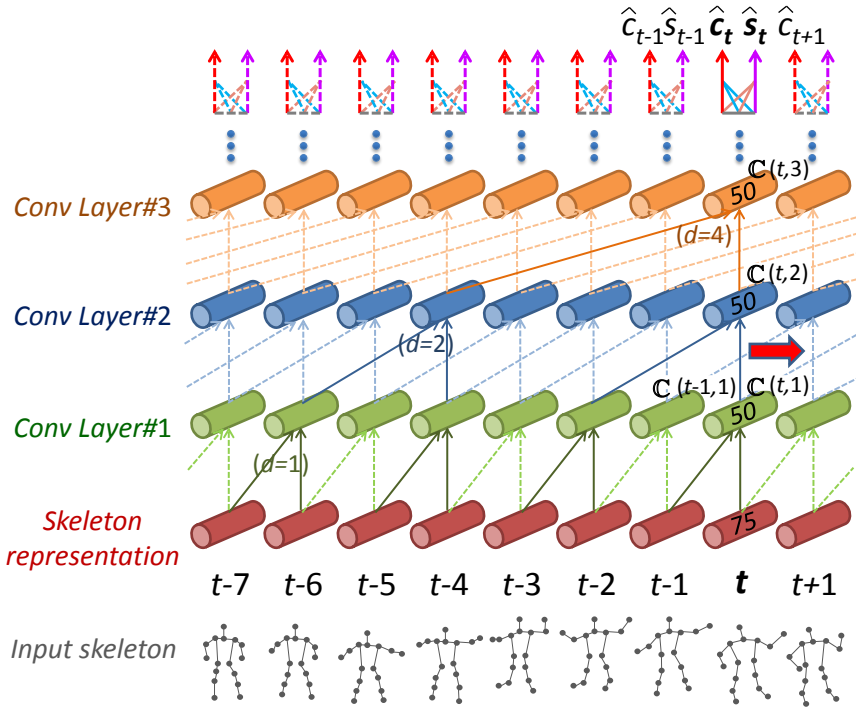


Fig. 5.2 Illustration of the proposed SSNet for action prediction over the temporal axis. The solid lines denote the SSNet links activated at current step t, and the dashed lines indicate the links activated at other time steps. Our SSNet has 14 1-D convolutional layers. Here we only show 3 layers for clarity. At each time step, SSNet predicts the class ($\hat{c}_t$) of the ongoing action, and also estimates the temporal distance ($\hat{s}_t$) to current action's start point. Calculation details of $\hat{c}_t$ and $\hat{s}_t$ are shown in Fig. 5.3.

steps, a scale selection method is proposed, which enables our SSNet to focus on the observed part of the ongoing action by picking the most suitable convolutional layers. To better deal with the input data modality, a hierarchy of dilated tree convolutions are also introduced to process the input skeleton data for our network.

5.2.1 Temporal Modeling with Convolutional Layers

Convolutional networks [157] have proven their superior strength in modeling the time series data [158, 159, 10]. For example, van den Oord *et al.* [158] proposed a convolutional model, called WaveNet, for audio signal generation, and Dauphin *et al.* [159] introduced a convolutional network for time series in language sequential modeling. Inspired by the success of convolutional approaches in the analysis of temporal sequential data, we leverage a stack of 1-D convolutional layers to model the motion dynamics and context dependencies over the video sequence frames, and inspired by the WaveNet model, we propose a network for the skeleton-based action prediction task. Specifically, a hierarchical architecture with dilated convolutional layers is leveraged in our model to learn a comprehensive representation over the video frames within a temporal window.

Dilated convolution. The main building blocks of our network model are dilated causal convolutions. Causal design [158] indicates action prediction at time t is based on the available information before t (including t) without using the future information. Dilated convolution [160] means the convolutional filter works over a larger field than the filter's length, and some input values inside the field are skipped with a certain step.

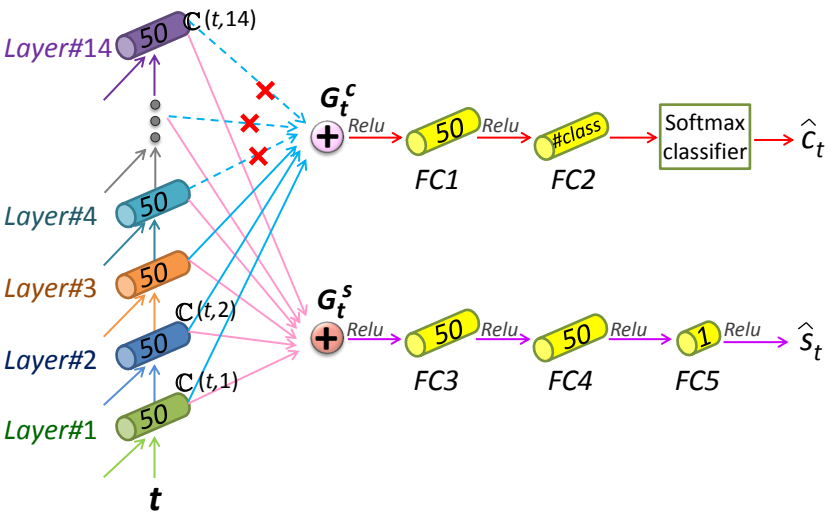


Fig. 5.3 Details of our SSNet that jointly predicts the class label $\hat{c}_t$ and regresses the start point's distance $\hat{s}_t$ for the current ongoing action **at time** t. If the regressed result $\hat{s}_{t-1}$ at the previous time step $(t-1)$ indicates that Layer #3 corresponds to the most *proper* window scale (*i.e.*, $l_t^p = 3$), then our network will use Layers #1-3 for class prediction, while the activations from the layers above #3 are dropped (marked with *cross* in the figure). In this figure, we only show a subset of convolutional nodes of our SSNet, and other ones in the hierarchical structure (depicted as the solid lines in Fig. 5.2) are omitted for clarity.

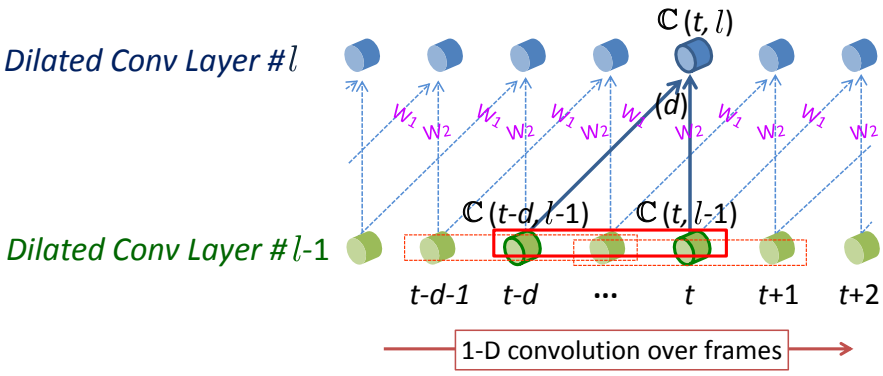


Fig. 5.4 Illustration of the dilated convolution layer used in our network. At each position, the 1-D convolutional filter covers a time range (labeled as a red box), and only the two boundary nodes (corresponding to two time steps) in the covered range are used, while the other nodes between these two nodes are not used by the dilated convolutional operation for this position.

Concretely, dilated convolution [160] is also called "convolution with holes", which can be formulated as presented in [160]:

$$(X *_d w)(\mathbf{p}) = \sum_{\mathbf{t}+d\mathbf{s}=\mathbf{p}} X(\mathbf{t})\, w(\mathbf{s}) \tag{5.1}$$

where $*_d$ indicates the dilated convolutional operation, X is the input, w is the filter, and d denotes the dilation rate of the convolution ($d = 1$ represents the standard convolution).

In order to show how the dilated convolution is used in our model, we illustrate the mechanism of a dilated convolutional layer in Fig. 5.4.

As shown in Fig. 5.4, at each position (*e.g.*, the position t), the dilated convolutional filter (with dilation rate d) works over two input time steps (t and $t-d$), and the other time steps between these two steps are not considered for the convolutional operation at this position. Let $\mathbb{C}(t,l)$ denote the activation of the convolutional node at the position t in the dilated convolutional Layer #l ($l \in [1, \mathscr{L}]$, and $\mathscr{L}$ denotes the number of 1-D convolutional layers in our network). Then $\mathbb{C}(t,l)$ can be calculated as:

$$\mathbb{C}(t,l) = f\Big(W_1\, \mathbb{C}(t-d, l-1) + W_2\, \mathbb{C}(t, l-1) + b\Big) \tag{5.2}$$

where $f(\cdot)$ is a non-linear activation function. W_1 and W_2 (together with the bias b) are the parameters of the dilated convolutional filter, which are shared at the same layer, as illustrated in Fig. 5.4.

It is intuitive to use the aforementioned dilated convolution for human action analysis, because the running time for longer actions can be very long and the convolutional network needs to be able to cover a large receptive field. Applying standard convolution, the network needs more layers or larger filter sizes to achieve a broader receptive field. However, both of these significantly increase the number of model parameters. In

contrast, by configuring the dilation rate (d), dilated convolution can support expansion of the receptive field very efficiently, without bringing more parameters [160]. In addition, it does not need any extra pooling operations, thus it can well maintain the ordering information of the inputs [160]. Therefore, dilated convolution is suitable for the task of action prediction.

Multiple dilated convolutional layers. In our method, we stack multiple dilated convolutional layers, as illustrated in Fig. 5.2. Specifically, the dilation rate increases exponentially over the layers in our network, *i.e.*, we set d to 1, 2, 4, 8, ... for Layers #1, #2, #3, #4, ..., respectively.

This design results in an **exponential** expansion of the perception scale across the network layers. For example, the perception temporal window of the convolutional operation node $\mathbb{C}(t,2)$ in Layer #2 (see Fig. 5.2) is $[t-3,t]$ (4 frames), while the node $\mathbb{C}(t,3)$ in Layer #3 corresponds to a larger scale of temporal window (8 frames: $[t-7,t]$).

It is worth mentioning that all the video frames in the window $[t-7,t]$ can be perceived by the node $\mathbb{C}(t,3)$ with the hierarchical structure. This shows how the field of view expands over the layers in our network, while the coverage of the input is kept.

5.2.2 Scale Selection

For the streaming sequences, we can utilize the frames in a temporal window $[t-s,t]$ (with scale s) to perform action prediction at the time step t. However, finding a proper temporal scale s for different steps and inputs is not easy. At the early stages of an action, a relatively small scale is preferred, because larger windows can involve too many frames from the previous action, which interfere the recognition. On the contrary, if a large ratio of the action is observed (especially when the duration of this action is long), to obtain a reliable prediction, we need a larger s to cover more of its

observed parts. This implies the importance of finding a proper scale value at each time step, rather than using a fixed scale at all steps.

We propose a scale selection scheme for online action prediction in this section. The core idea is to regress a *proper* window scale at each time step, and then at the next time step, the network can use this scale value to choose the *proper* layers for action prediction.

At each step, as shown in Fig. 5.2, the class label ($\hat{c}_t$) of the current action is predicted, and the temporal distance ($\hat{s}_t$) between the current action's start point and the current frame is also regressed. This distance indicates that the performed part of the current action is assumed to be $[t-\hat{s}_t, t]$ at step t.

Assuming that we have obtained the regression result $\hat{s}_{t-1}$ at step $(t-1)$, thus at frame t, our network selects the time range $[(t-1)-\hat{s}_{t-1}, t]$ for action prediction. Specifically, in our network design, the nodes in different layers correspond to different perception temporal window scales, thus we can select the node from the *proper* layer to cover the performed part of the current action. For this *proper* layer l, we make sure its perception window's scale equals to (or slightly larger than) $\hat{s}_{t-1}+1$, while the perception window of its previous layer $(l-1)$ is smaller than $\hat{s}_{t-1}+1$. For example, Layer #2 in Fig. 5.1 is the *proper* layer in this case.

Let l_t^p denote the selected *proper* layer at step t. Then we aggregate the activations of the nodes $\mathbb{C}(t,l)$ ($l \in [1, l_t^p]$) in our network to generate a comprehensive representation for the selected time range as:

$$G_t^c = \frac{1}{l_t^p} \sum_{l=1}^{l_t^p} \mathbb{C}(t,l) \tag{5.3}$$

Note that we connect multiple layers ($[1, l_t^p]$) together to compute G_t^c, rather than using l_t^p only. This skip connection design can speed up convergence and enables the training of much deeper models, as shown by [161, 162]. Besides, it can also help to improve

the representation capability of our network, as the information from multiple layers corresponding to multiple scales is fused for current action. Finally, G_t^c is fed to the fully connected layers followed by a softmax classifier to predict the class label ($\hat{c}_t$) for the current time step.

As shown in Fig. 5.3, beside predicting the action class ($\hat{c}_t$), our network also generates a representation (G_t^s) to regress the start point's distance ($\hat{s}_t$):

$$G_t^s = \frac{1}{\mathscr{L}} \sum_{l=1}^{\mathscr{L}} \mathbb{C}(t,l) \tag{5.4}$$

For the distance regression, we directly adopt the top convolutional layer $\mathscr{L}$ (*together with all the layers below it*), which has a large perception window (generally larger than the complete execution time of one action), rather than dynamically selecting a layer as in Eq (5.3). This is due to the essential difference between the regression task and the action label prediction task. Start point's distance regression can be regarded as regressing the position of the bonding [163] between the *current action* and its previous activities, thus involving information from the previous activity will not reduce (or even benefit) the regression performance for current action. Using Eq (5.4) also implies the distance regression is performed independently at each time step, and is not affected by the regression results of the previous steps.

In the domain of object detection [164], such as Fast-RCNN [165], the bounding box of the current object was shown to be accurately regressed by a learning scheme. Similarly, our proposed network learns to regress the bounding (start point) of the current ongoing action reliably.

The regression result produced by the previous step $(t-1)$ is used to guide the scale selection (with scale $\hat{s}_{t-1}+1$) for action prediction at the current step t. An alternative method can be: first regressing the scale $\hat{s}_t$ at step t, then using the scale $\hat{s}_t$ to directly perform action prediction for the same step t. We observe these two choices perform

Table 5.1 Details of the main structure of SSNet. Refer to Fig. 5.13 for the detailed architecture configurations of SSNet.

1-D convolutional layer index	#1	#2	#3	#4	#5	#6	#7
Dilation rate (d)	1	2	4	8	16	32	64
Perception temporal window scale (frames)	2	4	8	16	32	64	128
Output channels	50	50	50	50	50	50	50
1-D convolutional layer index	#8	#9	#10	#11	#12	#13	#14
Dilation rate (d)	1	2	4	8	16	32	64
Perception temporal window scale (frames)	129	131	135	143	159	191	255
Output channels	50	50	50	50	50	50	50

similarly in practice. This is intuitive as $\hat{s}_{t-1}+1$ is close to $\hat{s}_t$. The main difference of these two choices is the scale used at the beginning of a new action, because if we use the scale regressed by its previous step, the scale used at this step may be derived from the previous action, which is not proper. However, at the beginning frame of an action, too little information of the current action is observed, which makes prediction at this step very difficult even using the proper scale (only one frame), thus these two choices still perform similarly at this step. In the following frames, since more information is observed and proper scales can be used, both choices perform reliably. The framework will be less efficient if regressing for the same step, as the two tasks (regression and prediction) need to be conducted as two sequential stages at each time step (cannot be performed simultaneously).

5.2.3 Details of the Main Structure

The proposed SSNet has 14 dilated convolutional layers for temporal modeling. Specifically, we stack two similar sub-networks with dilation rates $(d) : 1,2,4,8,...,64$ over the layers of each sub-network, *i.e.*, the dilation rate (d) is reset to 1 at the beginning of each sub-network, as shown in Table 5.1 and Fig. 5.13. The motivation of this design is to achieve more variations for the temporal window scales (we obtain 14 different scales from 2 to 255 here). Besides, each sub-network can be intuitively regarded and

implemented as a large convolutional module. Moreover, such a design still guarantees the node at each layer to perceive all the video frames in its perception window (*i.e.*, without losing input coverage), due to the hierarchial structure of SSNet.

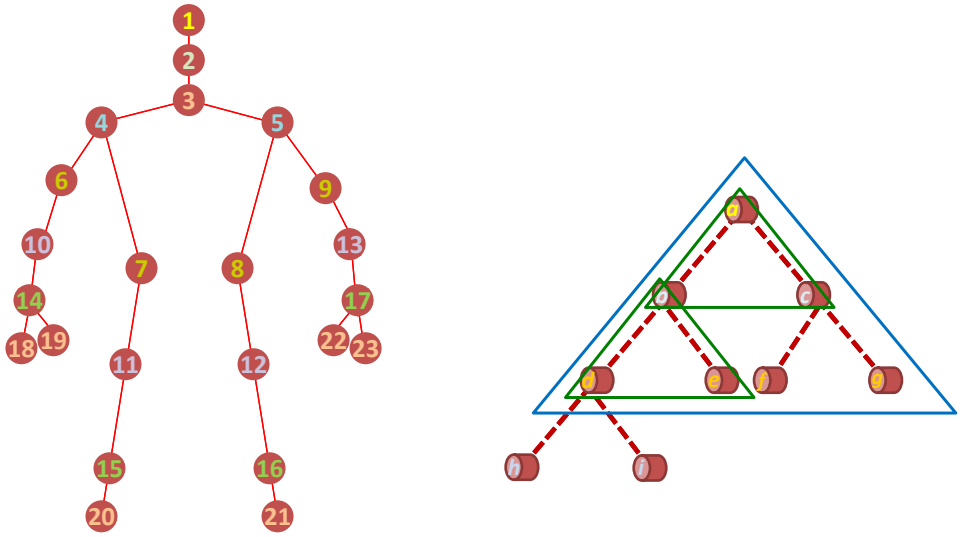
With such a design, the perception temporal window scale of the top layer in our network is 255 frames, which covers more than 8-second sequence at the recording frame rate of common video cameras like Kinect. Generally, the duration of a full single action in most existing datasets is less than 8 seconds. Thus, the temporal scale 255 is large enough for action analysis. Even if the whole duration time of an action is longer than 8 seconds, we believe the classification can be performed reliably when such a long segment (8 seconds) of the action has been perceived.

5.2.4 Activation Sharing Scheme

Our framework can be implemented in a very computation-efficient way. Although both action label prediction and distance regression are conducted on various window scales at each step, all of the computational steps are encapsulated in a single network with a hierarchical structure (see Fig. 5.2), *i.e.*, we do not need separated networks or multiple scanning passes for action prediction at each step.

In addition, although convolutional operations are performed over a sliding window at each step, the redundant computations of the overlapping regions among different sliding positions are avoided. With the causal convolution design, many features (activations of convolutional operations) computed in previous steps can be reused by the latter steps, which avoids redundant computation.

As depicted in Eqs (5.3) and (5.4), at time step t, the prediction and regression are based on the nodes $\mathbb{C}(t,l)$, $l \in [1, l_t^p]$ or $l \in [1, \mathscr{L}]$. Each node $\mathbb{C}(t,l)$ is calculated based on only two input nodes, $\mathbb{C}(t-d_l, l-1)$ and $\mathbb{C}(t, l-1)$, as shown in Fig. 5.2. $\mathbb{C}(t-d_l, l-1)$ has already been computed at time step $t-d_l$. Therefore, to obtain



(a) (b)

Fig. 5.5 (a) The skeleton joints of the human body form a tree structure. We set the head joint (joint 1) as the root node, and the height of the tree in this figure is 8. (b) Illustration of the convolution with triangular filters sliding over the tree structure. The green and the blue triangles indicate the convolutions with two different filter sizes.

$\mathbb{C}(t,l)$, we only need to calculate the activation of $\mathbb{C}(t,l-1)$. Similarly, $\mathbb{C}(t,l-1)$ can be computed after we get $\mathbb{C}(t,l-2)$.

As a result, although we feed a window of frames to SSNet at each time step (t), we only need to calculate the activations of the nodes in column t of Fig. 5.2, and all other convolutional operations in the hierarchical structure can be copied from the previous time steps. This activation sharing makes our network efficient enough to be used in real-time applications.

5.2.5 Multi-level Structured Skeleton Representations

As mentioned above, in our framework, the streaming 3D skeleton data is fed to the SSNet. A naive way to perform action prediction with such an input data structure is to concatenate the 3D coordinates of all joints at each frame to form a vector (that we call it as coordinate concatenation representation). We can then feed this coordinate concatenation representation of each frame to the SSNet as input (see

Fig. 5.2). However, the semantic structure amongst the skeleton joints in a frame is ignored in this representation. As illustrated in Fig. 5.5(a), the skeleton joints in every human body configuration are physically connected in a semantical tree structure in the spatial domain, and utilizing such structure information has shown to be quite helpful for human activity analysis [3, 21, 27].

Instead of directly using the method of coordinate concatenation, in this chapter, we model the spatial tree structure of the skeleton joints, in order to capture the posture information of the human body more effectively at each frame and thus strengthen the capability of our framework in skeleton-based action prediction.

Specifically, we propose a hierarchy of dilated tree convolutions in spatial domain to learn the multi-level (local, mid-level, and holistic) structured representations for the tree structure of the skeleton in each frame. The proposed hierarchical dilated tree convolution for spatial domain modeling is essentially an extension of the multi-layer 1-D dilated convolution that is introduced in section 5.2.1 for temporal modeling. Below we introduce this design in detail.

Convolution over tree structure. Convolutional networks are powerful tools in modeling the spatial visual structures [157]. Here to model the discussed semantic structure of the human skeleton, we propose to apply convolutions by using triangular filters sliding over the nodes of the tree, as shown in Fig. 5.5(b). At each step of the convolution, the triangular filter covers a sub-tree region, and the nodes in this region are used to produce an activation as a semantic representation of this position. This process is similar to the common convolutional operations that slide over the pixels of an input image or previous layer's feature maps. Different sizes of the triangular filters can also be used for this process, as shown in Fig. 5.5(b).

In our method, zero padding is adopted for the convolution over the skeleton tree, *i.e.*, if a certain node (*e.g.*, joint 2 in Fig. 5.5(a)) has only one child (joint 3), to perform convolution at this node position, we set this child (joint 3) as the left node, and its

right node is filled with zero. Similarly, for the leaf nodes (*e.g.*, joint 20), both of the child nodes are filled with zero.

A hierarchy of dilated tree convolutions. In order to learn representations that are effective and discriminative for representing the skeletal data in a frame, we stack multiple convolutional layers over the tree-structured skeleton joints, and perform convolution with triangular filters at each layer, as illustrated in Fig. 5.6.

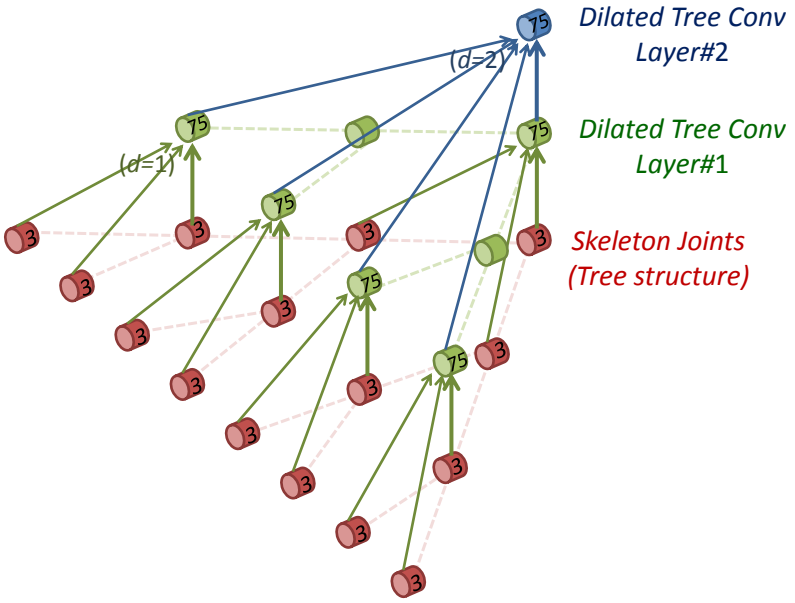


Fig. 5.6 Illustration of the hierarchy of dilated tree convolutions that learns the multi-level structured representations over the input skeleton joints (labeled in red) at each frame. The solid arrows denote the dilated tree convolutions with triangular filters. In our method, 3 dilated tree convolutional layers are used to cover the input skeleton tree with height 8, while in this figure, we only show 2 layers that cover the tree with height 4 for clarity. Note that the bottom of this figure shows a full binary tree, while the human skeleton only has a subset of the nodes of a full binary tree. Therefore, in implementation, the convolutional operations only need to be performed on a subset of the nodes. The channel number of the input skeleton is 3, namely, the 3D coordinates (x, y, z) of each joint. (Best viewed in color)

Dilated convolutions which are effective and efficient in computation are also used here (similar to section 5.2.1). Only the top and the bottom nodes in each triangular region of each position are used for activation calculation, as shown by the Layer #1

(with dilation rate set to 1) and Layer #2 (with dilation rate set to 2) in Fig. 5.6. Here we call this convolution design as dilated tree convolution.

Three dilated tree convolutional layers are stacked in our model, and their dilation rates are 1, 2, and 4, respectively. Therefore, a hierarchy of dilated tree convolutions are constructed over the skeletal data. The details of this hierarchy design are shown in Table 5.2.

With this design, the nodes in different layers of the hierarchy perceive different spatial ranges of the input skeleton joints. For example, *each node* in Layer #1 of Fig. 5.6 learns a representation from a very local region of neighbouring joints of the input skeleton (perception sub-tree height is 2), while *each node* in Layer #2 learns a representation over a larger region of the skeleton (perception sub-tree height is 4). Specifically, the top layer, #3, can learn a representation based on all the joints of the whole skeleton tree (perception tree height is 8). This implies that the multi-level (local, mid-level, and holistic) structured semantic representations of the skeleton data are learned at different layers in this hierarchy.

Finally, we aggregate the multi-level representations by averaging the activations of all the convolutional nodes in the hierarchy, and the aggregated result is fed to our SSNet as the representation of the skeleton data at each frame (see Fig. 5.2).

Since the multi-level structured semantic representations are learned, which are effective for representing the spatial structure and posture of the human skeleton at each frame, the performance of our SSNet for action prediction is improved. Moreover, this structured skeleton representation learning procedure can be attached to our SSNet as an input processing module of it (see Fig. 5.2), such that the whole model of our SSNet is still end-to-end trainable.

Table 5.2 Details of the hierarchy of dilated tree convolutions (corresponding to Fig. 5.6).

Dilated tree convolutional layer index	#1	#2	#3
Dilation rate (d)	1	2	4
Perception sub-tree height	2	4	8
Output channels	75	75	75

5.2.6 Objective Function

The objective function of our SSNet is formulated as:

$$\ell = \ell_c(\hat{c}_t, c_t) + \gamma \ell_s(\hat{s}_t, s_t) \tag{5.5}$$

where c_t is the ground truth class label, and s_t is the ground truth distance between the start point of the action and the current frame t. γ is the weight for the regression task. ℓ_c is the negative log-likelihood loss measuring the difference between the true class label c_t and the predicted result $\hat{c}_t$ at time step t. ℓ_s is the regression loss defined as $\ell_s(\hat{s}_t, s_t) = (\hat{s}_t - s_t)^2$. Our objective function is minimized by stochastic gradient descent.

To train our SSNet, we generate fixed-length clips from the annotated long sequences with sliding temporal windows. The length of each clip is equal to the perception temporal scale of the top convolutional layer (255 frames). Each clip can then be fed to the SSNet. In the training phase, class prediction is performed using the proper layer that is chosen based on the ground truth distance to the start point. We also observe adding small random noise to the layer choosing process during training is helpful for improving the generalization capability of our network for class prediction.

In the testing phase, the action prediction is performed frame-by-frame through a sliding window, and the proper layer for prediction at each time step is determined by the distance regression result of its previous step. The ground truth information of the start point is not used during testing.

5.3 Experiments

The proposed method is evaluated on four challenging datasets: the OAD dataset [78], the ChaLearn Gesture dataset [86], the PKUMMD dataset [89], and the G3D dataset [90]. In all the datasets, multiple action instances are contained in each long video. Beside the predefined action classes, these datasets also contain frames which belong to the background activity, thus we add a blank class to represent the frames in this situation.

We conduct extensive experiments with the following different architectures:

1. **SSNet.** This is our proposed network for skeleton-based action prediction, which can select a proper layer to cover the performed part of the current ongoing action at each time step *by using the start point regression result*. The multi-level structured skeleton representations are used in this network.

2. **FSNet ($\mathscr{S}$).** Fixed Scale Network (FSNet) is similar to SSNet, but the action prediction is directly performed using the top layer. This indicates scale selection scheme is not used, and the prediction is based on a fixed window scale ($\mathscr{S}$) at all steps. We configure the structure and propose a set of FSNets, such that they have different perception window scales at the top layer. Concretely, five FSNets with different fixed scales ($\mathscr{S} = 15,\ 31,\ 63,\ 127,\ 255$) are evaluated. To make a fair comparison, skip connections (see Eq (5.3)) are also used in each FSNet, *i.e.*, all layers (corresponding to different scales) in a FSNet are connected as Eq (5.3) for action prediction at each step.

3. **FSNet-MultiNet.** This baseline is a combination of multiple FSNets. A set of FSNets with different scales ($\mathscr{S} = 15,\ 31,\ 63,\ 127,\ 255$) are used for each time step. We then fuse the results of them, *i.e.*, exhaustive multi-scale multi-round scans are used to perform action prediction at each time step.

4. ***SSNet-GT*.** Beside the aforementioned models, we also evaluate an "ideal" baseline, *SSNet-GT*. Action prediction in *SSNet-GT* is also performed at the selected layer. However, we do not use the regression result to select the scale, instead, we directly use the *ground truth (GT)* distance of the start point to select the layer for action prediction at each step.

Note that the multi-level structured skeleton representations are used in all of the above architectures (SSNet, FSNet ($\mathscr{S}$), FSNet-MultiNet, and SSNet-GT) for fair comparisons.

Our proposed approach is also compared to other state-of-the-art methods for skeleton-based activity analysis:

1. **ST-LSTM** [151]. This network achieves superior performance on 3D skeleton-based action recognition task. We adapt it to our online action prediction task and generate a prediction of the action class at each frame of the streaming sequence.

2. **JCR-RNN** [78]. This network is a variant of LSTM, which models the context dependencies in temporal dimension of the untrimmed sequences. It obtains state-of-the-art performance of action detection in skeleton sequences on some benchmark datasets. A prediction of the current action class is provided at each frame of the streaming sequence.

3. **Attention Net** [166]. This network adopts an attention mechanism to dynamically assign weights to different frames and different skeletal joints for 3D skeleton-based action classification. A prediction of the action class is produced at each time step.

5.3.1 Implementation Details

The experiments are conducted with the Torch7 toolbox [122]. Our network is trained from scratch, *i.e.*, the network parameters are initialized with small random values (uniform distribution in [-0.08, 0.08]). The learning rate, momentum, and decay rate are set to 10^{-3}, 0.9, and 0.95, respectively. The output dimensions of FC1, FC3, FC4, and FC5 in Fig. 5.3 are set to 50, 50, 50, and 1, respectively. FC2's output dimension is determined by the class number of each specific dataset. GLU [159] is the activation function used for the convolutional operations in our network (see Eq (5.2)). Residual connections [161] are used over different convolutional layers. The output channels of the convolutional nodes for temporal modeling (see Fig. 5.2) are all 50. The output channels of the convolutional nodes for structured skeleton representation learning are equal to the dimension of the coordinate concatenation representation of a frame. In our experiment, γ in Eq (5.5) is set to 0.01. The above-mentioned parameters are obtained by cross-validation on the training sets.

In our SSNet, the proposed hierarchy of dilated tree convolution is used to learn the multi-level structured representation for each skeleton in a frame. If two skeletons are contained in a frame, then their structured representations are averaged. The averaged result is used as the representation for this frame.

We show the number of parameters of our SSNet with the two different skeleton representations in Table 5.3. By attaching the multi-level structured representation, the parameter number in the whole model of SSNet is only slightly larger than the configuration in which we use coordinate concatenation representation. This implies that the number of parameters in the hierarchy of dilated tree convolutions is quite small (only 13% of the whole model).

We also summarize the numbers of network parameters for different methods. The numbers of network parameters of SSNet, FSNet(15), FSNet(31), FSNet(63),

FSNet(127), FSNet(255), FSNet-MultiNet, SSNet-GT, ST-LSTM, JCR-RNN, and AttentionNet are 310K, 170K, 200K, 240K, 270K, 310K, 1M, 310K, 420K, 290K, and 3M, respectively.

We perform our experiments with a single NVIDIA TitanX GPU. We evaluate the efficiency of our method for online action prediction in the streaming sequence, and show the running speed of it in Table 5.3. Our network responds fast for online action prediction. The low computational cost of our method is partially due to (1) the concise skeleton data as input, (2) the efficient dilated convolution, and (3) our activation sharing scheme. Besides, even if we learn the multi-level structured representations, the overall speed of our SSNet is still very fast.

Table 5.3 Number of parameters and computational efficiency of our SSNet when using different skeleton representations within it.

Skeleton representations in SSNet	#Parameters	Speed
With coordinate concatenation	$270K$	$50fps$
With multi-level structured representation	$310K$	$40fps$

5.3.2 Experiments on the OAD Dataset

For the OAD dataset, 30 long sequences are used for training, and 20 long sequences are used for testing. The action prediction results on the OAD dataset are shown in Fig. 5.7 and Table 5.4. In the figures and tables, the prediction accuracy of an observation ratio $p\%$ denotes the average accuracy of the predictions during the observed segment ($p\%$) of the action instance.

Note that the special baseline *SSNet-GT* performs action prediction with the *ground truth* scale at each step, thus it provides the best results. Our SSNet with regressed scale even achieves comparable results to this "ideal" baseline (*SSNet-GT*), which indicates the effectiveness of our scale selection scheme for online action prediction at each progress level.

Table 5.4 Action prediction accuracies on the OAD dataset. Note that in the last row, *SSNet-GT* is an "ideal" baseline, in which the *ground truth (GT)* scales are used for action prediction. Our SSNet, which performs prediction with the regressed scales, is even comparable to *SSNet-GT*. Refer to Fig. 5.7 for more results.

Observation Ratio	10%	50%	90%
JCR-RNN	62.0%	77.3%	78.8%
ST-LSTM	60.0%	75.3%	77.5%
Attention Net	59.0%	75.8%	78.3%
FSNet (15)	58.5%	75.4%	75.9%
FSNet (31)	62.3%	75.2%	76.2%
FSNet (63)	62.2%	77.1%	78.9%
FSNet (127)	63.6%	76.3%	78.9%
FSNet (255)	57.2%	70.3%	71.2%
FSNet-MultiNet	62.6%	79.1%	81.6%
SSNet	**65.8**%	**81.3**%	**82.8**%
SSNet-GT	*66.7%*	*81.7%*	*83.0%*

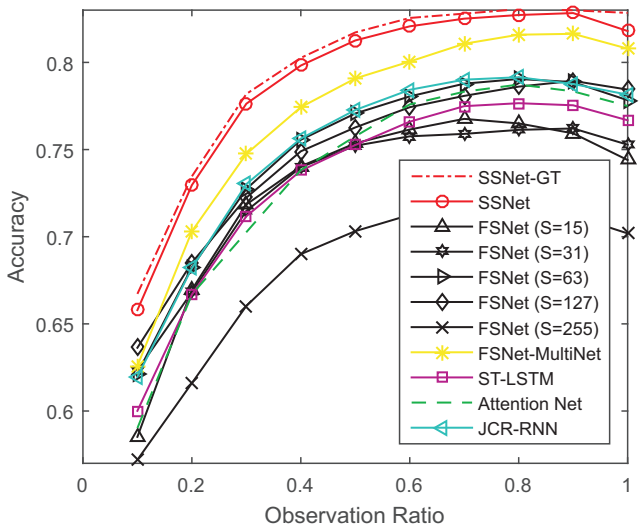


Fig. 5.7 Action prediction results on the OAD dataset.

Apart from the "ideal" *SSNet-GT* model, our proposed SSNet yields the best prediction results among all methods at all observation ratios. Specifically, our SSNet can even produce a quite reliable prediction (about 66% accuracy) at the early stage when only a small ratio (10%) of the action instance is observed.

The performance of our SSNet is much better than FSNets which perform prediction with fixed-scale windows at each time step. Even fusing a set of FSNets with different scales, FSNet-MultiNet is still weaker than our single SSNet at all progress levels. This demonstrates that our proposed scale selection scheme, which guides the SSNet to dynamically cover the performed part of the current action at each step, is very effective for online action prediction.

The proposed SSNet significantly outperforms the state-of-the-art RNN/LSTM based methods, JCR-RNN [78] and ST-LSTM [151], which can handle continuous streaming skeleton sequences. The performance disparity could be explained as: (1) At the early stages (eg. 10%), our SSNet can focus on the performed part of current action by using the selected scale, while RNN models [78, 151] may bring information from the previous actions which can interfere the prediction for current action. (2) At the latter stages (eg. 90%), the context information from the early part of current action may vanish in RNN model with its hidden state evolving frame by frame, while our SSNet, which uses convolutional layers to model the temporal dependencies over the frames, can still handle the long-term context dependency information in the temporal window. Our SSNet also outperforms the Attention Net [166] which assigns weights to differen frames and joints. This indicates the superiority of our SSNet with explicit scale selection.

We also observe the average action prediction accuracy decreases at the ending stages. A possible explantation is that the frames at the ending stages of some action instances contain postures and motions that are not very relevant to the current action's class label.

5.3.3 Experiments on the ChaLearn Gesture Dataset

On the ChaLearn Gesture dataset, $3/4$ of the annotated videos are used for training, and the remaining annotated videos are held for testing. We sample 1 frame from every 4 frames considering the large amount of data. We report the action prediction results in Fig. 5.8 and Table 5.5. Our SSNet outperforms other methods at all observation ratios on this large-scale dataset.

Table 5.5 Action prediction accuracies on the ChaLearn Gesture dataset. Refer to Fig. 5.8 for more results.

Observation Ratio	10%	50%	90%
JCR-RNN	15.6%	51.6%	64.7%
ST-LSTM	15.8%	51.3%	65.1%
Attention Net	16.8%	52.1%	65.3%
FSNet (15)	16.6%	50.8%	62.0%
FSNet (31)	16.9%	53.2%	64.4%
FSNet (63)	15.8%	49.8%	60.8%
FSNet (127)	14.8%	46.4%	56.4%
FSNet (255)	14.5%	45.7%	55.4%
FSNet-MultiNet	17.5%	54.1%	65.9%
SSNet	**19.5%**	**56.2%**	**69.1%**
SSNet-GT	*20.1%*	*56.8%*	*70.0%*

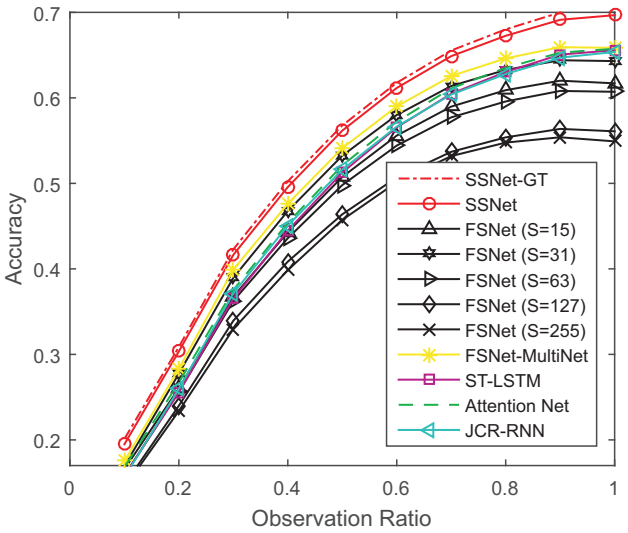


Fig. 5.8 Action prediction results on the ChaLearn Gesutre dataset.

5.3.4 Experiments on the PKUMMD Dataset

Cross-subject evaluation protocol is used for the PKUMMD dataset [89], in which 57 subjects are used for training, and the remaining 9 subjects are for testing. Considering the large amount of data, we use the videos that contain the challenging interaction actions for our experiment, and sample 1 frame from every 4 frames for these videos. The comparison results of the prediction performance on this dataset are presented in Fig. 5.9 and Table 5.6.

Our method achieves the best results at all the progress levels on this dataset. Specifically, our SSNet outperforms other methods significantly, even when only a very small ratio (10%) of the action is observed. This indicates that our method can produce a much better prediction at the early stage by focusing on the current action, compared to other methods which do not explicitly consider the scale selection.

Another observation is that the FSNet with fixed scale at each time step is quite sensitive to the scale used, as different scales provide very different results. This further demonstrates that our SSNet, which dynamically chooses the proper scale at each step to perform prediction, is effective for online action prediction.

Table 5.6 Action prediction accuracies on the PKUMMD dataset. Refer to Fig. 5.9 for more results.

Observation Ratio	10%	50%	90%
JCR-RNN	25.3%	64.0%	73.4%
ST-LSTM	22.9%	63.0%	74.5%
Attention Net	19.8%	62.9%	74.9%
FSNet (15)	27.1%	67.4%	76.2%
FSNet (31)	30.6%	69.9%	79.8%
FSNet (63)	25.3%	63.5%	72.1%
FSNet (127)	25.9%	60.6%	71.0%
FSNet (255)	20.2%	50.9%	62.4%
FSNet-MultiNet	27.4%	71.8%	80.3%
SSNet	**33.9**%	**74.1**%	**82.9**%
SSNet-GT	*34.8%*	*74.2%*	*83.1%*

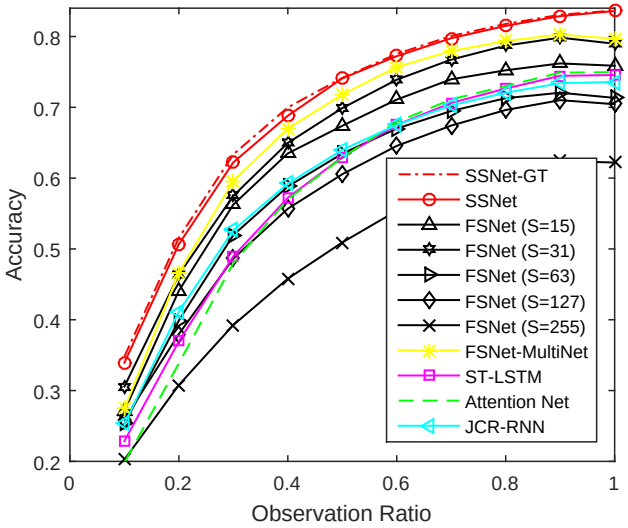


Fig. 5.9 Action prediction results on the PKUMMD dataset.

5.3.5 Experiments on the G3D Dataset

For the G3D dataset [90], we use 104 videos for training, and the remaining ones are used for testing. Our SSNet achieves superior performance on this challenging dataset, as shown in Fig. 5.10 and Table 5.7.

Table 5.7 Action prediction accuracies on the G3D dataset. Refer to Fig. 5.10 for more results.

Observation Ratio	10%	50%	90%
JCR-RNN	70.0%	79.1%	81.9%
ST-LSTM	67.3%	75.6%	76.8%
Attention Net	67.4%	76.9%	79.3%
SSNet	**72.0**%	**81.2**%	**83.7**%
SSNet-GT	*73.5%*	*81.5%*	*84.0%*

5.3.6 Evaluation of Skeleton Representations

We compare the performance of our SSNet when using the multi-level structured skeleton representations to that when using the coordinate concatenation representation, and report the results in Table 5.8.

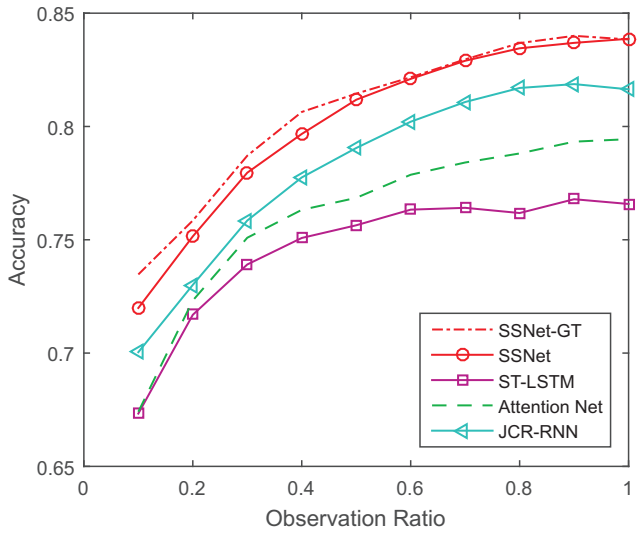


Fig. 5.10 Action prediction results on the G3D dataset.

The comparison results show that by using the hierarchy of dilated tree convolutions to learn the multi-level structured representation for the skeleton data in each frame, the action prediction performance of our SSNet is significantly improved. This clearly demonstrates the effectiveness of our newly proposed method in learning a discriminative representation of the human skeleton data in the spatial domain.

It is worth noting that, even if we do not use the powerful multi-level structured representation, but directly use the coordinate concatenation representation, our SSNet still outperforms the state-of-the-art skeleton-based activity analysis methods, JCR-RNN [78], ST-LSTM [151], and Attention Net [166], on all the four datasets.

5.3.7 Evaluation of Distance Regression

We adopt the metric *SL-Score* proposed in [78] to evaluate the distance regression performance of our network, which is calculated as $e^{-|\hat{s}-s|/d}$, where s and $\hat{s}$ are respectively the ground truth distance and regressed distance to the action's start point, and d is the length of the action instance. For false classification samples, the score is set to 0.

Table 5.8 Action prediction accuracies (%) of SSNet with different skeleton representations.

Skeleton representations	OAD			ChaLearn Gesture		
	Observation Ratio			Observation Ratio		
	10%	50%	90%	10%	50%	90%
Coordinate concatenation	65.6	79.2	81.6	17.5	53.5	65.9
Multi-level structured representation	**65.8**	**81.3**	**82.8**	**19.5**	**56.2**	**69.1**
Skeleton representations	PKUMMD			G3D		
	Observation Ratio			Observation Ratio		
	10%	50%	90%	10%	50%	90%
Coordinate concatenation	30.0	68.5	78.6	70.1	79.1	82.0
Multi-level structured representation	**33.9**	**74.1**	**82.9**	**72.0**	**81.2**	**83.7**

Table 5.9 Start point regression performance (*SL-Score*). SSNet^C indicates that we use the coordinate concatenation representation for the network.

Dataset	JCR-RNN	SSNet^C	SSNet
OAD	0.42	0.69	**0.71**
ChaLearn Gesture	0.49	0.58	**0.60**
PKUMMD	0.61	0.72	**0.75**
G3D	0.62	0.72	**0.74**

We report the regression performance of our SSNet in Table 5.9. As the action detection method, JCR-RNN [78], also estimates the start point, we also compare our method with it. Besides, we investigate the regression performance of the SSNet when we do not use multi-level structured representation but directly use coordinate concatenation for it (here we denote this case as SSNet^C).

The results show that our SSNet provides the best regression performance. Specifically, we observe that the regression result of SSNet (with multi-level structured representation) is better than SSNet^C (with coordinate concatenation). This indicates that by effectively learning the spatial tree structure of the input skeleton data, the accuracy of temporal distance regression can also be improved.

Table 5.10 Start point regression errors.

Observed Segment	5%	10%	30%	50%	70%	90%
Error (frames)	6	4	3	3	3	3

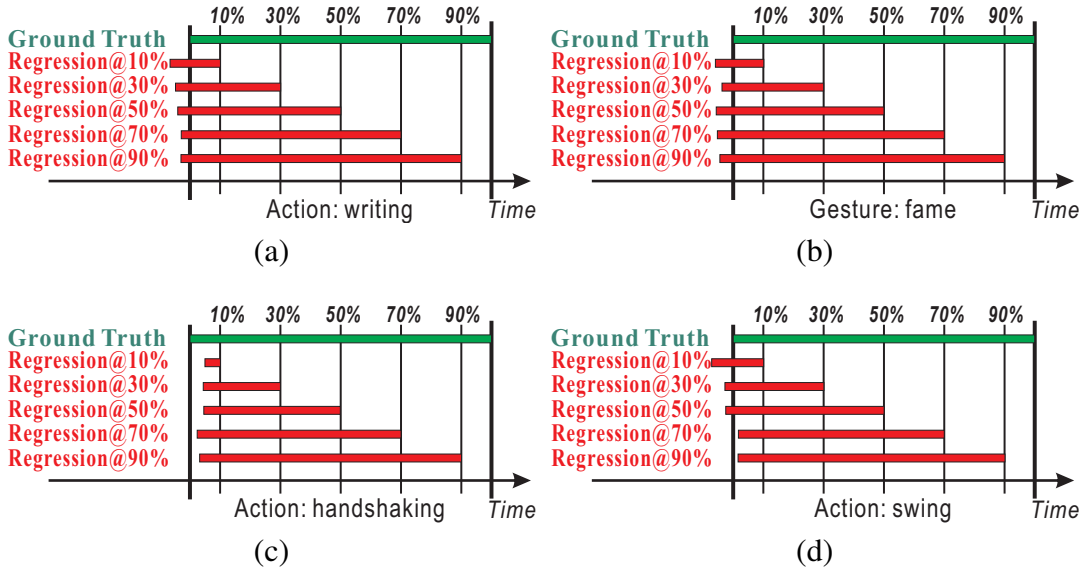


Fig. 5.11 Examples of the start point regression results on the four datasets. The leftmost point of the green bar is the ground truth start point position of the current ongoing action. The leftmost point of each red bar (ending at $p\%$) is the regressed start point position when $p\%$ of the action instance is observed.

We also evaluate the average regression errors during the observed segment ($p\%$) on the large-scale ChaLearn Gesture dataset in Table 5.10. The regression error is calculated as $|\hat{s}-s|$. We find our method regresses the distance reliably. When only a small ratio (5%) of the action instance has been observed, the average regression error is 6 frames. The regression becomes more reliable when more frames are observed. We also visualize some examples in Fig. 5.11. It shows that our SSNet achieves promising regression performance.

5.3.8 Evaluation of Network Configurations

We configure the maximum dilation rate and the layer number to generate a set of SSNets, which have different maximum perception window scales at the top layers.

Table 5.11 Evaluation of different configurations of the proposed network on the OAD dataset.

Number of 1-D convolutional layers	8	10	12	14	16
Max. dilation rate	8	16	32	64	128
Max. perception window scale (frames)	*31*	*63*	*127*	*255*	*511*
Start point regression (*SL-Score*)	0.65	0.68	0.70	0.71	0.71
Prediction accuracy (%)	75.2	77.8	79.0	80.6	80.6

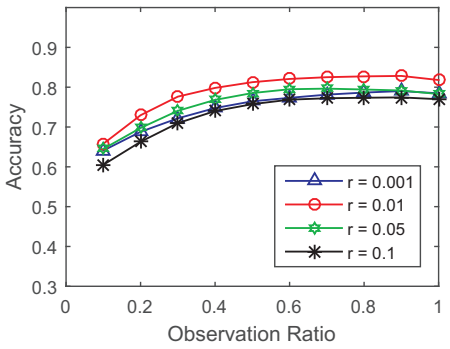


Fig. 5.12 Action prediction results with different γ values on the OAD dataset.

The results in Table 5.11 show that using more layers are beneficial for performance as the perception temporal window scale of the top layer increases. However, the performance of 16 layers is almost the same as 14 layers. A possible explanation is that the duration time of most actions is less than 255 frames. Besides, 255 frames are long enough for activity analysis. Thus using the SSNet with 14 layers (with maximum window scale 255) is suitable.

We also evaluate the performance of our SSNet with different γ values (see Eq (5.5)) in Fig. 5.12. We observe our SSNet yields the best performance when γ is set to 0.01.

As shown in Eq (5.3) and Eq (5.4), in the modules of generating representations for class prediction and distance regression, instead of using the activation from one convolutional layer only, we add skip connections (links from the bottom convolutional layers). In our experiment, we observe that using this skip connection design, the

Table 5.12 Frame-level classification accuracies. FSNet(best) denotes the FSNet that gives the best results among all FSNets.

Dataset	ST-LSTM	AttentionNet	JCR-RNN	FSNet(best)	SSNet
OAD	0.77	0.75	0.79	0.80	**0.82**
ChaLearn	0.62	0.63	0.62	0.64	**0.66**
PKUMMD	0.78	0.80	0.79	0.82	**0.85**
G3D	0.70	0.71	0.74	0.75	**0.76**

action prediction accuracy can be improved by about 1.5%. We also investigate to further add batch normalization (BN) layers [167] to our network, and we do not see obvious performance improvement, thus BN layers are not used in our model.

5.3.9 Frame-level Classification Accuracies

As the action classification is performed at each frame of the videos, the average classification accuracies over all frames are also evaluated, and the results are reported in Table 5.12. The results show the superiority of our SSNet over the compared approaches.

5.4 Chapter Summary

In this chapter, we have proposed a network model, SSNet, for online action prediction in untrimmed skeleton sequences. A stack of convolutional layers are introduced to model the dynamics and dependencies in temporal dimension. A scale selection scheme is also proposed for SSNet, with which our network can choose the proper layer corresponding to the most proper window scale for action prediction at each time step. Besides, a hierarchy of dilated tree convolutions are designed to learn the multi-level structured representations for the skeleton data in order to improve the performance of our network. Our proposed method yields superior performance on all the evaluated benchmark datasets.

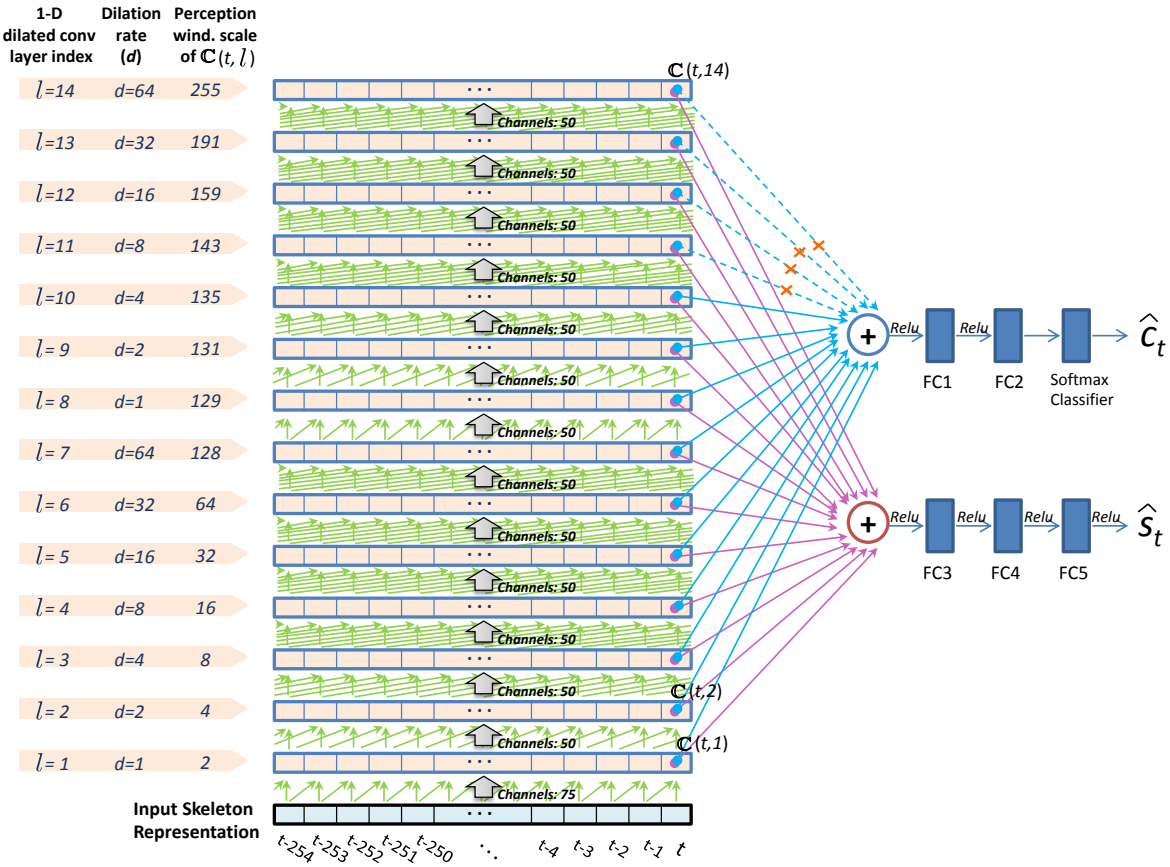


Fig. 5.13 Detailed network architecture configurations of SSNet (for action prediction at the time step t). The distance regression is performed based on the top convolutional layer (together with the layers below it with the skip connections), which has a large perception window. The class prediction is performed based on the selected *proper* layer (together with the layers below it), which is selected based on the estimated window scale.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we have proposed three network models for human activity analysis in 3D skeleton sequences.

First, we introduce a new ST-LSTM network to analyse the 3D location of each individual joint in each video frame at each processing step, which models the context dependency information in both the spatial and temporal dimensions. For better representation of the structured input to the network, a skeleton tree traversal algorithm is designed, which improves the performance of the ST-LSTM network. Due to the unreliability of the 3D input data, a new gating mechanism is proposed to improve the robustness of the network against noise and occlusion. In addition, a novel multi-modal feature fusion strategy is described. The provided experimental results validate the effectiveness of this network that achieves superior performance over the existing state-of-the-art methods on the evaluated datasets for 3D human action recognition.

Second, we introduce a GCA-LSTM network to selectively focus on the informative joints in the action sequence with the assistance of global contextual information. To achieve a reliable attention representation for the action sequence, a recurrent atten-

tion mechanism is designed for the GCA-LSTM network, with which the attention performance is improved iteratively. Besides, a two-stream framework that leverages coarse-grained attention and fine-grained attention is proposed. A stepwise training scheme is also introduced to more effectively train this network. Experiments show that the GCA-LSTM network is able to reliably focus on the most informative joints in each frame of the skeleton sequence. Moreover, this network yields state-of-the-art performance on five challenging datasets for action recognition.

Finally, we introduce a SSNet model for online action prediction in streaming 3D skeleton sequences. A dilated convolutional architecture is introduced to model the motion dynamics and context dependencies in temporal dimension via a sliding window over the time axis, and a hierarchy of tree convolutions are used to learn the multi-level structured representations for the 3D skeleton data to improve the performance of our SSNet. Besides, a novel window scale selection scheme is proposed to make our SSNet focus on the performed part of the ongoing action at each time step. In addition, an activation sharing scheme is proposed to deal with the overlapping computations among the adjacent steps, which allows our model to run more efficiently. The extensive experiments show the effectiveness of the proposed action prediction framework.

6.2 Future Work

In this section, we introduce some of our possible future research directions.

6.2.1 Multi-Modal Feature Fusion for Activity Analysis

The 3D skeleton data provides the geometrical posture and motion information of the human actions. However, in some scenarios, especially the human-object interaction cases, the appearance information can also be useful for human activity understanding. This indicates it could be beneficial to fuse the RGB data, depth data, and skeleton

data for activity analysis, since they provide appearance and geometrical information respectively, which are often complementary for understanding the human behaviours.

Specifically, in most of the existing datasets for skeleton-based human activity analysis, beside the 3D skeleton sequences, the corresponding RGB sequences and depth sequences are also provided. As a result, we can design feature fusion models to aggregate the features extracted from different data modalities provided by these datasets for more accurate human activity analysis.

6.2.2 One-Shot Action Recognition in Skeleton Data

Existing works [168] show that once some categories have been learned, the knowledge gained in this process can be abstracted and used to learn novel classes efficiently, even if *only one learning example per new class is given* (i.e., via one-shot learning).

Since the samples of certain action categories (especially their 3D skeleton data) may be difficult to collect [169], one-shot action recognition in skeleton data could also be an important research branch of human activity analysis.

Specifically, we can define the one-shot skeleton-based action recognition scenario as follows. We can setup an auxiliary set and an evaluation set, and there are no overlaps of classes between these two sets. The auxiliary set contains multiple action classes, and the samples of these classes can be used for learning (e.g., learning a feature generation network). The evaluation set consists of the novel action classes for one-shot recognition evaluation, and one sample from each novel class is picked as the exemplar, while the remaining samples are used to test the recognition performance.

6.2.3 Online Action Detection in Untrimmed Skeleton Sequences

In our proposed Scale Selection Network (SSNet) for online action prediction (in Chapter 5), at each time step, the proper temporal window scale is estimated (this scale

indicates the distance to the starting point's location of current action). Besides, the current action's label is also predicted at each step.

This indicates the proposed SSNet could be further extended to address the task of action detection in streaming 3D skeleton data, which requires to identify the temporal location of each action and meanwhile predict the class label of the action.

Author's Publications

- **Jun Liu**, Amir Shahroudy, Dong Xu, Gang Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," European Conference on Computer Vision (**ECCV**), 2016.
- **Jun Liu**, Gang Wang, Ping Hu, Ling-Yu Duan, Alex C. Kot, "Global context-aware attention LSTM networks for 3D action recognition," IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**), 2017.
- **Jun Liu**, Amir Shahroudy, Gang Wang, Ling-Yu Duan, Alex C. Kot, "SSNet: Scale selection network for online 3D action prediction," IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**), 2018.
- **Jun Liu**, Amir Shahroudy, Dong Xu, Alex C. Kot, Gang Wang, "Skeleton-Based Action Recognition Using Spatio-Temporal LSTM Network with Trust Gates," IEEE Transactions on Pattern Analysis and Machine Intelligence (**TPAMI**), 2018.
- **Jun Liu**, Gang Wang, Ling-Yu Duan, Kamila Abdiyeva, Alex C. Kot, "Skeleton-Based Human Action Recognition with Global Context-Aware Attention LSTM Networks," IEEE Transactions on Image Processing (**TIP**), 2018.

- **Jun Liu**, Henghui Ding, Amir Shahroudy, Ling-Yu Duan, Xudong Jiang, Gang Wang, Alex C. Kot, "Feature Boosting Network for 3D Pose Estimation," IEEE Transactions on Pattern Analysis and Machine Intelligence (**TPAMI**), 2019.

- **Jun Liu**, Amir Shahroudy, Gang Wang, Ling-Yu Duan, Alex C. Kot, "Skeleton-Based Online Action Prediction Using Scale Selection Network," IEEE Transactions on Pattern Analysis and Machine Intelligence (**TPAMI**), 2019.

- **Jun Liu**, Amir Shahroudy, Mauricio Lisboa Perez, Gang Wang, Ling-Yu Duan, Alex C. Kot, "NTU RGB+D 120: A Large-Scale Benchmark for 3D Human Activity Understanding," IEEE Transactions on Pattern Analysis and Machine Intelligence (**TPAMI**), 2019.

- Amir Shahroudy, **Jun Liu**, Tian-Tsong Ng, Gang Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**), 2016.

- Ping Hu, Bing Shuai, **Jun Liu**, Gang Wang, "Deep level sets for salient object detection," IEEE Conference on Computer Vision and Pattern Recognition (**CVPR**), 2017.

References

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

[2] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Mining actionlet ensemble for action recognition with depth cameras," in *CVPR*, pp. 1290–1297, 2012.

[3] Y. Du, W. Wang, and L. Wang, "Hierarchical recurrent neural network for skeleton based action recognition," in *CVPR*, pp. 1110–1118, 2015.

[4] V. Veeriah, N. Zhuang, and G.-J. Qi, "Differential recurrent neural networks for action recognition," in *ICCV*, pp. 4041–4049, 2015.

[5] W. Zhu, C. Lan, J. Xing, W. Zeng, Y. Li, L. Shen, and X. Xie, "Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks," in *AAAI*, vol. 2, p. 6, 2016.

[6] J.-F. Hu, W.-S. Zheng, L. Ma, G. Wang, and J. Lai, "Real-time rgb-d activity prediction by soft regression," in *ECCV*, pp. 461–470, 2016.

[7] L. Xia, C. Chen, and J. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *CVPRW*, pp. 20–27, 2012.

[8] Y. Zhu, W. Chen, and G. Guo, "Fusing spatiotemporal features and joints for 3d action recognition," in *CVPRW*, pp. 486–491, 2013.

[9] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, pp. 568–576, 2014.

[10] G. Varol, I. Laptev, and C. Schmid, "Long-term temporal convolutions for action recognition," *TPAMI*, vol. 40, no. 6, pp. 1510–1517, 2018.

[11] F. Han, B. Reily, W. Hoff, and H. Zhang, "Space-time representation of people based on 3d skeletal data: a review," *CVIU*, vol. 158, pp. 85–105, 2017.

[12] L. L. Presti and M. La Cascia, "3d skeleton-based human action classification: A survey," *PR*, vol. 53, pp. 130–147, 2016.

[13] M. Liu, Q. He, and H. Liu, "Fusing shape and motion matrices for view invariant action recognition using 3d skeletons," in *ICIP*, pp. 3670–3674, 2017.

[14] S. Zhang, X. Liu, and J. Xiao, "On geometric features for skeleton-based action recognition using multilayer lstm networks," in *WACV*, pp. 148–157, 2017.

[15] H. Rahmani and M. Bennamoun, "Learning action recognition model from depth and skeleton videos," in *ICCV*, pp. 5832–5841, 2017.

[16] H. Rahmani, A. Mahmood, D. Q. Huynh, and A. Mian, "Real time action recognition using histograms of depth gradients and random decision forests," in *WACV*, pp. 626–633, 2014.

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

[18] P. Zhang *et al.*, "View adaptive recurrent neural networks for high performance human action recognition from skeleton data," *arXiv*, 2017.

[19] Q. Ma, L. Shen, E. Chen, S. Tian, J. Wang, and G. W. Cottrell, "Walking walking walking: Action recognition from action echoes," in *IJCAI*, pp. 2457–2463, 2017.

[20] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

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

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

[23] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, "Real-time human pose recognition in parts from single depth images," in *CVPR*, pp. 1297–1304, 2011.

[24] J. K. Aggarwal and L. Xia, "Human activity recognition from 3d data: A review," *Pattern Recognition Letters*, vol. 48, pp. 70–80, 2014.

[25] J. Zhang, W. Li, P. O. Ogunbona, P. Wang, and C. Tang, "Rgb-d-based action recognition datasets: A survey," *Pattern Recognition*, vol. 60, pp. 86–105, 2016.

[26] J. Luo, W. Wang, and H. Qi, "Group sparsity and geometry constrained dictionary learning for action recognition from depth maps," in *ICCV*, pp. 1809–1816, 2013.

[27] A. Shahroudy, T. Ng, Q. Yang, and G. Wang, "Multimodal multipart learning for action recognition in depth videos," *TPAMI*, vol. 38, no. 10, pp. 2123–2129, 2016.

[28] M. Meng, H. Drira, M. Daoudi, and J. Boonaert, "Human-object interaction recognition by learning the distances between the object and the skeleton joints," in *FG*, pp. 1–6, 2015.

[29] X. Yang and Y. Tian, "Effective 3d action recognition using eigenjoints," *JVCIR*, vol. 25, no. 1, pp. 2–11, 2014.

[30] J. Wang and Y. Wu, "Learning maximum margin temporal warping for action recognition," in *ICCV*, pp. 2688–2695, 2013.

[31] I. Lillo, J. Carlos Niebles, and A. Soto, "A hierarchical pose-based approach to complex action understanding using dictionaries of actionlets and motion poselets," in *CVPR*, pp. 1981–1990, 2016.

[32] C. Chen, R. Jafari, and N. Kehtarnavaz, "Fusion of depth skeleton and inertial data for human action recognition," in *ICASSP*, pp. 2712–2716, 2016.

[33] L. Tao and R. Vidal, "Moving poselets: A discriminative and interpretable skeletal motion representation for action recognition," in *ICCVW*, pp. 61–69, 2015.

[34] J. Weng, C. Weng, and J. Yuan, "Spatio-temporal naive-bayes nearest-neighbor (st-nbnn) for skeleton-based action recognition," in *CVPR*, 2017.

[35] A. Shahroudy, G. Wang, and T. Ng, "Multi-modal feature fusion for action recognition in rgb-d sequences," in *ISCCSP*, pp. 1–4, 2014.

[36] P. Wang, W. Li, P. Ogunbona, Z. Gao, and H. Zhang, "Mining mid-level features for action recognition based on effective skeleton representation," in *DICTA*, 2014.

[37] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Sequence of the most informative joints (smij): A new representation for human skeletal action recognition," *JVCIR*, vol. 25, no. 1, 2014.

[38] R. Anirudh, P. Turaga, J. Su, and A. Srivastava, "Elastic functional coding of human actions: from vector-fields to latent variables," in *CVPR*, pp. 3147–3155, 2015.

[39] K. Yun, J. Honorio, D. Chattopadhyay, T. L. Berg, and D. Samaras, "Two-person interaction detection using body-pose features and multiple instance learning," in *CVPRW*, 2012.

[40] G. Yu, Z. Liu, and J. Yuan, "Discriminative orderlet mining for real-time recognition of human-object interaction," in *ACCV*, pp. 50–65, 2014.

[41] R. Chaudhry, F. Ofli, G. Kurillo, R. Bajcsy, and R. Vidal, "Bio-inspired dynamic 3d discriminative skeletal features for human action recognition," in *CVPRW*, 2013.

[42] R. Vemulapalli, F. Arrate, and R. Chellappa, "Human action recognition by representing 3d skeletons as points in a lie group," in *CVPR*, pp. 588–595, 2014.

[43] G. Evangelidis, G. Singh, and R. Horaud, "Skeletal quads: Human action recognition using joint quadruples," in *ICPR*, 2014.

[44] E. Ohn-Bar and M. Trivedi, "Joint angles similarities and hog^2 for action recognition," in *CVPRW*, pp. 465–470, 2013.

[45] J. Wang, Z. Liu, Y. Wu, and J. Yuan, "Learning actionlet ensemble for 3d human action recognition," *TPAMI*, vol. 36, no. 5, pp. 914–927, 2014.

[46] H. Chen, G. Wang, J.-H. Xue, and L. He, "A novel hierarchical framework for human action recognition," *Pattern Recognition*, vol. 55, pp. 148–159, 2016.

[47] P. Koniusz, A. Cherian, and F. Porikli, "Tensor representations via kernel linearization for action recognition from 3d skeletons," in *ECCV*, 2016.

[48] P. Wang, C. Yuan, W. Hu, B. Li, and Y. Zhang, "Graph based skeleton motion representation and similarity measurement for action recognition," in *ECCV*, pp. 370–385, 2016.

[49] M. Zanfir, M. Leordeanu, and C. Sminchisescu, "The moving pose: An efficient 3d kinematics descriptor for low-latency action recognition and detection," in *ICCV*, pp. 2752–2759, 2013.

[50] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "Computer vision for human-machine interaction," in *Computer Vision for Assistive Healthcare*, pp. 127–145, 2018.

[51] C. Li, Z. Cui, W. Zheng, C. Xu, and J. Yang, "Spatio-temporal graph convolution for skeleton based action recognition," *arXiv*, 2018.

[52] T. S. Kim and A. Reiter, "Interpretable 3d human action analysis with temporal convolutional networks," *arXiv*, 2017.

[53] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *arXiv*, 2018.

[54] Q. Ke, S. An, M. Bennamoun, F. Sohel, and F. Boussaid, "Skeletonnet: Mining deep part features for 3-d action recognition," *IEEE Signal Processing Letters*, vol. 24, no. 6, pp. 731–735, 2017.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR*, vol. 15, pp. 1929–1958, 2014.

[56] F. G. Harvey and C. Pal, "Semi-supervised learning with encoder-decoder recurrent neural networks: Experiments with motion capture sequences," *arXiv*, 2016.

[57] B. Mahasseni and S. Todorovic, "Regularizing long short term memory with 3d human-skeleton sequences for action recognition," in *CVPR*, 2016.

[58] Y. Kong, D. Kit, and Y. Fu, "A discriminative model with multiple temporal scales for action prediction," in *ECCV*, pp. 596–611, 2014.

[59] M. S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *ICCV*, pp. 1036–1043, 2011.

[60] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. Mark Siskind, and S. Wang, "Recognize human activities from partially observed videos," in *CVPR*, 2013.

[61] Z. Xu, L. Qing, and J. Miao, "Activity auto-completion: Predicting human activities from partial videos," in *ICCV*, pp. 3191–3199, 2015.

[62] K. Li and Y. Fu, "Prediction of human activity by discovering temporal sequence patterns," *T-PAMI*, vol. 36, no. 8, pp. 1644–1657, 2014.

[63] Q. Ke, M. Bennamoun, S. An, F. Boussaid, and F. Sohel, "Human interaction prediction using deep temporal features," in *ECCV*, pp. 403–414, 2016.

[64] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid, "Leveraging structural context models and ranking score fusion for human interaction prediction," *arXiv*, 2017.

[65] K. Li, J. Hu, and Y. Fu, "Modeling complex temporal composition of actionlets for activity prediction," in *ECCV*, pp. 286–299, 2012.

[66] P. Wei, N. Zheng, Y. Zhao, and S.-C. Zhu, "Concurrent action detection with structural prediction," in *ICCV*, pp. 3136–3143, 2013.

[67] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei, "End-to-end learning of action detection from frame glimpses in videos," in *CVPR*, 2016.

[68] J. Gao, Z. Yang, C. Sun, K. Chen, and R. Nevatia, "Turn tap: Temporal unit regression network for temporal action proposals," in *ICCV*, 2017.

[69] D. Oneata, J. Verbeek, and C. Schmid, "The lear submission at thumos 2014," 2014.

[70] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," *arXiv*, 2016.

[71] X. Dai, B. Singh, *et al.*, "Temporal context network for activity localization in videos," *arXiv*, 2017.

[72] A. Sharaf, M. Torki, M. E. Hussein, and M. El-Saban, "Real-time multi-scale action detection from 3d skeleton data," in *WACV*, pp. 998–1005, 2015.

[73] J. Gao, Z. Yang, and R. Nevatia, "Red: Reinforced encoder-decoder networks for action anticipation," in *BMVC*, 2017.

[74] Z. Shou, J. Chan, A. Zareian, K. Miyazawa, and S.-F. Chang, "Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos," in *CVPR*, 2017.

[75] J. Gao, Z. H. Yang, and R. Nevatia, "Cascaded boundary regression for temporal action detection," in *BMVC*, 2017.

[76] L. Wang, Y. Xiong, D. Lin, and L. Van Gool, "Untrimmednets for weakly supervised action recognition and detection," in *CVPR*, 2017.

[77] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, D. Lin, and X. Tang, "Temporal action detection with structured segment networks," in *ICCV*, 2017.

[78] Y. Li, C. Lan, J. Xing, W. Zeng, C. Yuan, and J. Liu, "Online human action detection using joint classification-regression recurrent neural networks," in *ECCV*, 2016.

[79] B. Li, H. Chen, Y. Chen, Y. Dai, and M. He, "Skeleton boxes: Solving skeleton based action detection with a single deep convolutional neural network," *arXiv*, 2017.

[80] P. Siva and T. Xiang, "Weakly supervised action detection.," in *BMVC*, vol. 2, p. 6, 2011.

[81] S. Baek, K. I. Kim, and T.-K. Kim, "Real-time online action detection forests using spatio-temporal contexts," in *WACV*, pp. 158–167, 2017.

[82] M. Hoai and F. De la Torre, "Max-margin early event detectors," *IJCV*, vol. 107, no. 2, pp. 191–202, 2014.

[83] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *CVPR*, pp. 1049–1058, 2016.

[84] Y. Zhu and S. Newsam, "Efficient action detection in untrimmed videos via multi-task learning," in *WACV*, pp. 197–206, 2017.

[85] J.-F. Hu, W. Zheng, J. Lai, and J. Zhang, "Jointly learning heterogeneous features for rgb-d activity recognition," in *CVPR*, pp. 5344–5352, 2015.

[86] S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, O. Lopes, I. Guyon, V. Athitsos, and H. Escalante, "Multi-modal gesture recognition challenge 2013: Dataset and results," in *ICMI*, pp. 445–452, 2013.

[87] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3d points," in *CVPRW*, 2010.

[88] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, and R. Bajcsy, "Berkeley mhad: A comprehensive multimodal human action database," in *WACV*, pp. 53–60, 2013.

[89] C. Liu, Y. Hu, *et al.*, "Pku-mmd: A large scale benchmark for continuous multi-modal human action understanding," *arXiv*, 2017.

[90] V. Bloom, D. Makris, and V. Argyriou, "G3d: A gaming action dataset and real time action recognition evaluation framework," in *CVPRW*, pp. 7–12, 2012.

[91] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*, pp. 6645–6649, 2013.

[92] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[93] T. Mikolov, S. Kombrink, L. Burget, J. H. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *ICASSP*, pp. 5528–5531, 2011.

[94] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *INTERSPEECH*, pp. 194–197, 2012.

[95] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding.," in *INTERSPEECH*, pp. 3771–3775, 2013.

[96] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, pp. 2048–2057, 2015.

[97] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 mscoco image captioning challenge," *TPAMI*, vol. 39, no. 4, pp. 652–663, 2017.

[98] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *ICML*, pp. 843–852, 2015.

[99] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao, "A multi-stream bi-directional recurrent neural network for fine-grained action detection," in *CVPR*, 2016.

[100] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *CVPR*, 2016.

[101] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *CVPR*, 2016.

[102] Z. Deng, A. Vahdat, H. Hu, and G. Mori, "Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition," in *CVPR*, 2016.

[103] M. S. Ibrahim, S. Muralidharan, Z. Deng, A. Vahdat, and G. Mori, "A hierarchical deep temporal model for group activity recognition," in *CVPR*, 2016.

[104] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *CVPR*, 2016.

[105] B. Ni, X. Yang, and S. Gao, "Progressively parsing interactional objects for fine grained action detection," in *CVPR*, 2016.

[106] A. Richard, H. Kuehne, and J. Gall, "Weakly supervised action learning with rnn based fine-to-coarse modeling," in *CVPR*, 2017.

[107] Z. Luo, B. Peng, D.-A. Huang, A. Alahi, and L. Fei-Fei, "Unsupervised learning of long-term motion dynamics for videos," in *CVPR*, 2017.

[108] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *CVPR*, pp. 4694–4702, 2015.

[109] 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*, pp. 2625–2634, 2015.

[110] Q. Li, Z. Qiu, T. Yao, T. Mei, Y. Rui, and J. Luo, "Action recognition by learning deep multi-granular spatio-temporal video representation," in *ICMR*, pp. 159–166, 2016.

[111] Z. Wu, X. Wang, Y.-G. Jiang, H. Ye, and X. Xue, "Modeling spatial-temporal clues in a hybrid deep learning framework for video classification," in *ACM MM*, pp. 461–470, 2015.

[112] T. Shu, S. Todorovic, and S.-C. Zhu, "Cern: Confidence-energy recurrent network for group activity recognition," in *CVPR*, 2017.

[113] M. Wang, B. Ni, and X. Yang, "Recurrent modeling of interaction context for collective activity recognition," in *CVPR*, 2017.

[114] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[115] B. Zou, S. Chen, C. Shi, and U. M. Providence, "Automatic reconstruction of 3d human motion pose from uncalibrated monocular video sequences based on markerless human motion tracking," *PR*, vol. 42, no. 7, pp. 1559–1571, 2009.

[116] Y. Yang and D. Ramanan, "Articulated pose estimation with flexible mixtures-of-parts," in *CVPR*, 2011.

[117] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *ECCV*, pp. 428–441, 2006.

[118] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *CVPR*, pp. 3169–3176, 2011.

[119] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[120] G. Chéron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *ICCV*, pp. 3218–3226, 2015.

[121] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. 2012.

[122] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *NIPS Workshop*, 2011.

[123] E. Cippitelli, E. Gambi, S. Spinsante, and F. Flórez-Revuelta, "Evaluation of a skeleton-based method for human activity recognition on a large-scale rgb-d dataset," in *TechAAL*, 2016.

[124] H. Rahmani and A. Mian, "3d action recognition from novel viewpoints," in *CVPR*, 2016.

[125] R. Slama, H. Wannous, M. Daoudi, and A. Srivastava, "Accurate 3d action recognition using learning on the grassmann manifold," *PR*, vol. 48, no. 2, 2015.

[126] S. Jetley and F. Cuzzolin, "3d activity recognition using motion history and binary shape templates," in *ACCVW*, pp. 129–144, 2014.

[127] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. Del Bimbo, "Space-time pose representation for 3d human action recognition," in *ICIAP*, pp. 456–464, 2013.

[128] M. Devanne, H. Wannous, S. Berretti, P. Pala, M. Daoudi, and A. Bimbo, "3-d human action recognition by shape analysis of motion trajectories on riemannian manifold," *TCYB*, vol. 45, no. 7, 2015.

[129] M. Jiang, J. Kong, G. Bebis, and H. Huo, "Informative joints based human action recognition using skeleton contexts," *SPIC*, vol. 33, pp. 29–40, 2015.

[130] A. Chrungoo, S. Manimaran, and B. Ravindran, "Activity recognition for natural human robot interaction," in *ICSR*, pp. 84–94, 2014.

[131] C. Wang, Y. Wang, and A. L. Yuille, "Mining 3d key-pose-motifs for action recognition," in *CVPR*, 2016.

[132] Y. Ji, G. Ye, and H. Cheng, "Interactive body part contrast mining for human interaction recognition," in *ICMEW*, pp. 1–6, 2014.

[133] W. Li, L. Wen, M. Choo Chuah, and S. Lyu, "Category-blind human action recognition: a practical recognition system," in *ICCV*, pp. 4444–4452, 2015.

[134] H. Wang, W. Wang, and L. Wang, "Hierarchical motion evolution for action recognition," in *ACPR*, pp. 574–578, 2015.

[135] B. Fernando, E. Gavves, J. Oramas, A. Ghodrati, and T. Tuytelaars, "Modeling video evolution for action recognition," in *CVPR*, pp. 5378–5387, 2015.

[136] A. Yao, L. Van Gool, and P. Kohli, "Gesture recognition portfolios for personalization," in *CVPR*, pp. 1915–1922, 2014.

[137] J. Wu, J. Cheng, C. Zhao, and H. Lu, "Fusing multi-modal features for gesture recognition," in *ICMI*, pp. 453–460, 2013.

[138] T. Pfister, J. Charles, and A. Zisserman, "Domain-adaptive discriminative one-shot learning of gestures," in *ECCV*, pp. 814–829, 2014.

[139] M. A. Gowayyed, M. Torki, M. E. Hussein, and M. El-Saban, "Histogram of oriented displacements (hod): Describing trajectories of human joints for action recognition," in *IJCAI*, 2013.

[140] S. Vantigodi and R. V. Babu, "Real-time human action recognition from motion capture data," in *NCVPRIPG*, pp. 1–4, 2013.

[141] S. Vantigodi and V. B. Radhakrishnan, "Action recognition from motion capture data using meta-cognitive rbf network classifier," in *ISSNIP*, pp. 1–6, 2014.

[142] I. Kapsouras and N. Nikolaidis, "Action recognition on motion capture data using a dynemes and forward differences representation," *JVCIR*, vol. 25, no. 6, pp. 1432–1445, 2014.

[143] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall, "A survey on human motion analysis from depth data," in *Time-of-flight and depth imaging. sensors, algorithms, and applications*, pp. 149–187, 2013.

[144] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIPS*, pp. 577–585, 2015.

[145] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.

[146] Q. Ke, M. Bennamoun, S. An, F. Bossaid, and F. Sohel, "Spatial, structural and temporal feature learning for human interaction prediction," *arXiv*, 2016.

[147] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *ICLR*, 2015.

[148] P. Wang, Z. Li, Y. Hou, and W. Li, "Action recognition based on joint trajectory maps using convolutional neural networks," in *ACM MM*, pp. 102–106, 2016.

[149] 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*, pp. 4263–4270, 2017.

[150] M. Liu, H. Liu, and C. Chen, "Enhanced skeleton visualization for view invariant human action recognition," *Pattern Recognition*, vol. 68, pp. 346–362, 2017.

[151] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang, "Skeleton-based action recognition using spatio-temporal lstm network with trust gates," *TPAMI*, 2018.

[152] C. Wang, J. Flynn, Y. Wang, and A. L. Yuille, "Recognizing actions in 3d using action-snippets and activated simplices," in *AAAI*, 2016.

[153] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *EMNLP*, 2015.

[154] Y. Kong, Z. Tao, and Y. Fu, "Deep sequential context networks for action prediction," in *CVPR*, 2017.

[155] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," in *CVPR*, vol. 1, pp. 11–18, 2006.

[156] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *T-PAMI*, vol. 31, no. 12, pp. 2179–2195, 2009.

[157] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10, p. 1995, 1995.

[158] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, 2016.

[159] Y. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *ICML*, 2017.

[160] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016.

[161] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, pp. 770–778, 2016.

[162] K. He, X. Y. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *ECCV*, pp. 630–645, 2016.

[163] S. Liu, L. Feng, Y. Liu, H. Qiao, J. Wu, and W. Wang, "Manifold warp segmentation of human action," *TNNLS*, 2017.

[164] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *CVPR*, 2017.

[165] R. Girshick, "Fast r-cnn," in *ICCV*, pp. 1440–1448, 2015.

[166] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, "Global context-aware attention lstm networks for 3d action recognition," in *CVPR*, 2017.

[167] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, pp. 448–456, 2015.

[168] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *TPAMI*, vol. 28, no. 4, pp. 594–611, 2006.

[169] P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera, "Rgb-d-based human motion recognition with deep learning: A survey," *CVIU*, 2018.