Recognizing and predicting human actions with depth camera

Weng, Junwu

2020

Weng, J. (2020). Recognizing and predicting human actions with depth camera. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/138384>

<https://doi.org/10.32657/10356/138384>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 23:23:45 SGT

Recognizing and Predicting Human Actions with Depth Camera

Junwu Weng

School of Electrical & Electronic Engineering

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

2020

Recognizing and Predicting Human Actions with Depth Camera

Junwu Weng

School of Electrical & Electronic Engineering

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

2020

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.

Aug. 16th, 2019

.....

Date

Junwu Weng

.....

Junwu WENG

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.

Aug. 16th, 2019

.....

Date

.....

Prof. Xudong JIANG

Authorship Attribution Statement

This thesis contains material from four paper(s) published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as [Junwu Weng, Chaoqun Weng, and Junsong Yuan. Spatio-Temporal Naive-Bayes Nearest-Neighbor for Skeleton-Based Action Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition \(CVPR\), 4171-4180 \(2017\).](#)

The contributions of the co-authors are as follows:

- Junwu Weng proposed the idea, designed the experiments, and prepared the manuscript.
- Chaoqun Weng and Junwu Weng have discussions of the details of the idea.
- The manuscript was revised by Chaoqun Weng and Junsong Yuan.

Chapter 4 is published as [Junwu Weng, Chaoqun Weng, Junsong Yuan, and Zicheng Liu. Discriminative Spatio-Temporal Pattern Discovery for 3D Action Recognition. IEEE Transactions on Circuits and Systems for Video Technology \(T-CSVT\), 1077-1089 \(2018\).](#)

The contributions of the co-authors are as follows:

- Junwu Weng proposed the idea, designed the experiments, and prepared the manuscript.
- Chaoqun Weng conducted the experiment for Two-persons Interaction on SBU Interaction dataset.
- The manuscript was revised by Chaoqun Weng, Junsong Yuan and Zicheng liu.

Chapter 5 is published as [Junwu Weng, Mengyuan Liu, Xudong Jiang, and Junsong Yuan. Deformable Pose Traversal Convolution for 3D Action and Gesture Recognition. Proceedings of the European Conference on Computer Vision \(ECCV\), 136-152 \(2018\).](#)

The contributions of the co-authors are as follows:

- Junwu Weng proposed the idea, designed the experiments, and prepared the manuscript.
- Mengyuan Liu and Junwu Weng have discussions of the details of the idea.
- The manuscript was revised by Mengyuan Liu, Xudong Jiang and Junsong Yuan.

Chapter 6 is published as Junwu Weng, Xudong Jiang, Wei-Long Zheng, and Junsong Yuan. Early Action Recognition with Category Exclusion using Policy-based Reinforcement Learning. IEEE Transactions on Circuits and Systems for Video Technology (T-CSVT), (2020).

The contributions of the co-authors are as follows:

- Junwu Weng proposed the idea, designed the experiments, and prepared the manuscript.
- Xudong Jiang, Wei-Long Zheng and Junwu Weng have discussions of some details of the idea.
- The manuscript was revised by Wei-Long Zheng, Xudong Jiang and Junsong Yuan.

Aug. 16th, 2019

.....

Date

.....

Junwu WENG

Acknowledgements

It seems that I have experienced a really long summer in Singapore. In these four years, there are frustration and success, as well as tears and laughter. However at the end of this journey, the only thing left is gratitude.

I wish to express my sincere gratitude to my advisors, Prof. Junsong Yuan and Prof. Xudong Jiang for the continuous support in my pursuit for the PhD degree, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my Ph.D study. I would like to also thank my TAC members Prof. Jianfei Cai and Prof. Justin Dauwels for not only their insightful comments in each yearly meeting, but for the hard question which incited me to widen my research from various perspectives as well. Besides, I would also express my thanks to my mentors Dr. Cuiling Lan and Prof. Wenjun Zeng for their encouragement and guidance during my internship in Microsoft Research Asia (MSRA).

My sincere thanks go to Dr. Gang Yu, Dr. Yuning Jiang, Dr. Jianyu Yang, Dr. Hongxing Wang, Dr. Chaoqun Weng, Dr. Hui Liang, Dr. Chunluan Zhou, Dr. Kang Dang, and Dr. Zhiwen Fang for their insightful suggestions, sharing and precious support in these four years. I also thank my fellow lab mates Ankang Liu, Bo Hu, Liuhao Ge, Tan Yu, Zhenzhen Wang, Jiong Yang, Suchen Wang, Yujun Cai, Yanjun Zhu, Jiaqi Zhang, Mengjia Yan, as well as Chuanxia Zheng, Guoxian Song, Yuedong Zheng, and Zhijie Zhang who made me feel like being home though I am in Singapore.

My dear friends Dr. Wei-Long Zheng, Renqin Cai, Qiuqiang Kong, Qinzhong Yao, Jiayi Chen, Guan Pan. Thank you so much for the warm support, discussion and sharing. The Ph.D life would not have been full of laughter without your companions.

Lastly, I would like to thank my parents. Without their unconditional love, support, encouragement, guidance and understanding, this four-year long journey would not have been easy. Love you, my parents.

Junwu Weng
Nanyang Technological University
August 2019

“Success is how high you bounce when you hit the bottom.”

—George S. Patton

To my dear family

Contents

Acknowledgements	ix
Summary	xiii
List of Figures	xxi
List of Tables	xxiii
Symbols and Acronyms	xxv
1 Introduction	1
1.1 Background	1
1.2 Motivations and Contributions	4
1.2.1 What is Action?	4
1.2.2 How to Differentiate Actions?	6
1.3 Outline of the Thesis	7
2 Literature Review	11
2.1 Action Recognition	11
2.1.1 Image-based Method	12
2.1.1.1 Traditional Method	12
2.1.1.2 Deep-learning-based Method	13
2.1.2 Pose-based Method	14
2.1.2.1 Traditional Method	15
2.1.2.2 Deep-learning-based Method	16
2.2 Action Prediction	18
2.3 Conclusion	20
3 Spatio-Temporal Naive-Bayes Nearest Neighbor	21
3.1 Introduction	21
3.2 Related Work	23
3.2.1 Spatio-Temporal Modeling for 3D Action Recognition	23
3.2.2 Naive-Bayes Nearest-Neighbor	24
3.3 Methodology	25

3.3.1	3D Action Representation	25
3.3.2	Naive Bayes Nearest Neighbor	26
3.3.3	Spatio-Temporal NBNN	27
3.4	Experiment	30
3.4.1	Implementations	31
3.4.2	Results and Analysis	32
3.5	Conclusion	37
4	Spatio-Temporal Naive Bayes Mutual Information Maximization	39
4.1	Introduction	39
4.2	Related Work	40
4.3	Methodology	42
4.3.1	3D Interactive Actions Representation	43
4.3.2	Naive-Bayes Mutual Information Maximization	44
4.3.3	Spatio-Temporal NBMIM	45
4.4	Experiment	49
4.4.1	Implementations	49
4.4.2	Datasets	50
4.4.3	Results and Analysis	53
4.5	Conclusion	62
5	Deformable Pose Traversal Convolution	65
5.1	Introduction	65
5.2	Related Work	68
5.3	Methodology	69
5.3.1	Pose Traversal Convolution	69
5.3.2	Deformable Pose Traversal Convolution	71
5.3.3	Learning and Classification	73
5.4	Experiments	74
5.4.1	Implementation	74
5.4.2	Datasets	75
5.4.3	Results and Analysis	76
5.5	Conclusions	81
6	Early Action Recognition with Category Exclusion	83
6.1	Introduction	83
6.2	Related Work	85
6.3	Proposed Method	86
6.3.1	Problem Formulation	87
6.3.2	Category Exclusion Modeling	87
6.3.3	Agent Training	90
6.4	Experiment	91
6.4.1	Implementation	92
6.4.2	Dataset	93

6.4.3 Results and Analysis	94
6.5 Conclusion	103
7 Conclusion & Future Work	105
7.1 Conclusion	105
7.2 Future Work	107
 List of Author's Awards, Patents, and Publications	 109
 Bibliography	 111

Summary

Understanding human behavior from videos is a very important task in computer vision community. It is a significant sub-branch of video analysis. Human behavior analysis is widely applied in many application scenarios like human-computer interaction, video surveillance, video retrieval and autonomous driving. Thanks to the development of commodity depth cameras, skeleton-based human behavior analysis has drawn considerable attention in the computer vision community recently. Skeleton action sequences are extracted from depth camera based on pose estimation algorithms or directly detected from motion capture devices. Compared with RGB-based action sequences, skeleton-based action instances are more simplified and semantic. However, the limitation is that there is less appearance and few scene information provided in skeletal data. How to design suitable and stable models to understand human behavior, both body action and hand gesture, using skeletal data is an interesting and challenging topic. To well understand human behavior through action sequence, two tasks are very important, namely the action recognition and the action prediction. In this thesis, four different models are proposed to handle these distinctive tasks.

Due to the success of deep learning models in image recognition, most of the state-of-the-arts choose to utilize deep learning as the tool for skeleton-based action recognition. However, compared with images and videos which are composed of millions or billions of pixels, the skeleton is composed by only tens of joints which is thus of much less complexity than images and videos. For such a light-weight data, non-parametric models like Naive-Bayes Nearest Neighbor (NBNN) may be more suitable than the deep learning models with high complexity. In the first two works of this thesis, two robust NBNN-based models, ST-NBNN and ST-NBMIM, are proposed to characterize skeleton sequences. Besides, to better understand skeleton-based actions, the bilinear classifiers are adopted to identify both key temporal stages as well as spatial joints for action classification. Although only using a linear classifier, experiments on five benchmark datasets show that by combining the strength of both non-parametric and parametric models, ST-NBNN and

ST-NBMIM can achieve competitive performance compared with state-of-the-art results using sophisticated models such as deep learning. Moreover, by identifying key skeleton joints and temporal stages for each action class, the two NBNN-based models can capture the essential spatio-temporal patterns that play key roles of recognizing actions, which is not always achievable by using end-to-end models.

When facing the large-scale skeleton data, the non-parametric model reaches its limitation, and the deep-learning-based models demonstrate their superior performance on dataset with large size. Meanwhile, human body movements exhibit spatial patterns among pose joints. It is thus of great importance to identify those motion patterns and avoid the non-informative joints, via identifying the key combinations of joints that matter for the recognition. Although key spatio-temporal patterns discovery has been explored previously for skeleton-based action recognition, the temporal dynamics modeling of key joint combinations is not well researched in the community. In the third work of this thesis, a CNN model is proposed to adaptively search key pose joints for each action sequence. The work utilizes the deep-learning technique to train a deformable CNN model to discover sample-related key spatio-temporal patterns for action recognition. This deformable convolution better utilizes the contextual joints for action and gesture recognition and is more robust to noisy joints. The proposed model is evaluated on three benchmark datasets and the experimental results show the effectiveness of introducing temporal dynamics modeling of key joint combinations into the skeleton-based action recognition.

The goal of early action recognition is to predict action label when the sequence is partially observed. The existing methods treat the early action recognition task as sequential classification problems on different observation ratios of an action sequence. Since these models are trained by differentiating positive categories from all negative classes, the diverse information of different negative categories is ignored, which we believe can be collected to help improve the recognition performance. In the last work of this thesis, a new direction, introducing category exclusion to early action recognition, is explored. The category exclusion is modeled as a mask operation on the classification probability output of a pre-trained early action recognition classifier. Specifically, policy-based reinforcement learning is utilized to train an agent. The agent generates a series of binary masks to exclude interfering

negative categories during action execution and hence help improve the recognition accuracy. The proposed method is evaluated on three benchmark recognition datasets, and it enhances the recognition accuracy consistently over all different observation ratios on the three datasets, where the accuracy improvements on the early stages are especially significant.

In summary, this thesis demonstrates the superior performance of the proposed four methods, including the ST-NBNN, ST-NBMIM, Deformable Pose Traversal Convolution, and the Category Exclusion Agent, for the tasks of action recognition and action prediction of skeleton-based sequences. These four models are extensively evaluated on well-known benchmark datasets and the experimental results show the effectiveness of these models on their corresponding tasks.

List of Figures

1.1	Taxonomy for Human Behavior Analysis. The red rectangle marks the hierarchy this thesis focuses on.	2
1.2	Property of Different Video Data Sources	3
1.3	Basic Workflow of Human Action Classification	4
3.1	Illustration of Key Stage, Joints (waving right hand action)	22
3.2	Overview of the Proposed Method	25
3.3	Parameter Sensitivity Analysis on MSR-Action Dataset	31
3.4	An Example of Spatio-Temporal Weight Matrix and Squared Distance Matrix (One pose feature of <i>Pull</i> Action)	34
3.5	Influence of Noisy Joints on Accuracy of Berkeley MHAD Dataset	35
3.6	Key Stages and Key Joints with Related Key Motions from MSR-Action3D	36
4.1	Overview of ST-NBMIM	40
4.2	Parameter Sensitivity Analysis on SBU Interaction Dataset	52
4.3	Comparison of Confusion Matrix between ST-NBNN and ST-NBMIM. 1. Approaching, 2. Departing, 3. Kicking, 4. Pushing, 5. Shaking Hands, 6. Hugging, 7. Exchanging, 8. Punching	55
4.4	An Example of Spatio-Temporal Weight Matrix and Comparison between Squared Distance Matrix from NBNN (first row) and Mutual Information Matrix from NBMIM (second row)	56
4.5	Influence of Noisy Joints on Accuracy of Berkeley MHAD Dataset	57
4.6	Training Convergence Curve of the ST-NBMIM	59
4.7	Key Stages and Key Joints with Related Key Motions from MSR-Action3D	60
5.1	Illustration of the Pose Traversal Representation, the Pose Traversal Convolution and the Deformable Pose Traversal Convolution	66
5.2	Structures of the Pose Traversal Convolution Network and the Deformable Pose Traversal Convolution Network	70
5.3	Illustration of the Deformable Pose Traversal Convolution	71
5.4	(A) Comparison of Confusion Matrices between Pose Traversal Convolution and Deformable Pose Traversal Convolution on DHG-14. (B) Comparison of Sub Confusion Matrices between Pose Traversal Convolution and Deformable Pose Traversal Convolution.	77

5.5	Impact of Rotation on Accuracy	78
5.6	Impact of Noise on Accuracy	78
5.7	Visualization of the Learned Offsets	80
6.1	Illustration of Category Exclusion in Early Action Recognition . . .	85
6.2	Illustration of Linear Feature Embedding Layers (FPHA)	92
6.3	Comparison of Early Recognition Curves	94
6.4	The Confusion Matrices before and after Exclusion and Their Dif- ference at the Observation Ratio 5% (FPHA)	96
6.5	Mask Statistics on FPHA	97
6.6	The Learned Weights Distribution of Agent's Input Layer on Input Embeddings	99
6.7	Accuracy Difference and Reward Curve of FPHA	100
6.8	Visualization of the Generated Mask and the Recognition before and after Exclusion	102

List of Tables

3.1	Results on MSR-Action3D (%)	32
3.2	Results on UTKinect (%)	33
3.3	Results on Berkeley MHAD (%)	33
4.1	Comparison with State-of-the-arts on MSR-Action3D (%)	51
4.2	Comparison with State-of-the-arts on UT-Kinect (%)	51
4.3	Comparison with State-of-the-arts on MHAD (%)	51
4.4	Comparison with State-of-the-arts on SBU (%)	52
4.5	Comparison with State-of-the-arts on NTU RGB+D (%)	53
4.6	Comparison with Baselines on Five Datasets (%)	54
4.7	Combination of CNN feature and ST-NBMIM (%)	56
4.8	Time Cost in Training Phase	59
4.9	Time Cost in Testing Phase	59
5.1	Comparison of Results on Hand Gesture Dataset - DHG (%)	75
5.2	Comparison on NTU (%)	76
5.3	Comparison on MHAD (%)	76
5.4	Comparison between 2D and 3D Data on DHG (%)	79
6.1	Early recognition accuracy comparison with baseline on two settings (CS & CV) of NTU-RGBD, FPHA and UCF-101 datasets (%)	95
6.2	Early recognition ablation study on FPHA (%)	98
6.3	Parameter Sensitivity Analysis of the Reward η on FPHA (%)	101

Symbols and Acronyms

Acronyms

NN	Nearest Neighbor
NBNN	Naive-Bayes Nearest-Neighbor
ST-NBNN	Spatio-Temporal Naive-Bayes Nearest-Neighbor
NBMIM	Naive-Bayes Mutual Information Maximization
ST-NBMIM	Spatio-Temporal Naive-Bayes Mutual Information Maximization
PTC	Pose Traversal Convolution
DPTC	Deformable Pose Traversal Convolution
MAP	Maximum a Posteriori
SVM	Support Vector Machine
STM	Support Tensor Machine
CNN	Convolution Neural Network
G-CNN	Graph Convolution Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ResNet	Residual Network
ConvLSTM	Convolutional LSTM
TCN	Temporal Convolution Network
ReLU	Rectified Linear Unit

Chapter 1

Introduction

1.1 Background

Human activity analysis is of great importance in video analysis. In the past decades, this research topic has attracted much attention from researchers in the computer vision community, and it has grown dramatically. It is widely applied in many real world scenarios. For example, in video surveillance, it is important to capture outstanding human behaviors such as theft stealing in the crowd. For healthcare, detecting the indoor activity of the elderly can enable timely rescue. For autonomous driving, detecting the activity of people on the street in real time can help guarantee the driver's safety. For human-computer interaction, a good human behavior analysis algorithm is able to help machine easily understand user's command and response instantly. For all these applications, detecting and understanding human behaviors are the core problems that need to be solved. Though human activity analysis has achieved great success, to well understand human behavior visually and artificially is still a challenging problem.

The analysis of human behavior can be performed at various levels of abstraction. Based on the taxonomies discussed in [1, 2], we further define the action hierarchy as *Pose*, *Primitive*, *Action*, *Activity*, *Interaction*, and *Group Activity*. The relationship among these six categories based on time scale and complexity is illustrated in Fig. 1.1. *Pose* is defined as the static pose or appearance of action subject in a single frame. In this hierarchy, there is no temporal information involved. *Primitive* is defined as the movement of a certain part of a human body in a short temporal

FIGURE 1.1: Taxonomy for Human Behavior Analysis. The red rectangle marks the hierarchy this thesis focuses on.

duration. For example, Raising the Left Arm in a short temporal duration can be regarded as an action primitive of left arm. *Action* is a term to describe the simple movement of a whole human body in a certain duration. Movements like Running, Jumping, *etc.*, can be regarded as actions. *Activity* consists of multiple *Actions* organized in a certain temporal order. For example the activity Jumping Hurdles. It is an activity that contains starting, jumping, and running actions. The term *Interaction* involves the interactive *Actions* between two persons. Two-persons movements like Hugging, Shaking Hands can be treated as interaction. *Group Activity* is the most complex hierarchy and bears the longest duration in the taxonomy. It is defined as the movement of a crowd. The research conducted in this thesis mainly focuses on the *Action*, *Activity*, and *Interaction*, as marked in Fig. 1.1.

To well understand human actions, three basic tasks [3] should be considered and solved, namely *Action Recognition*, *Action Prediction*, and *Action Detection*. In human action analysis, the input sequence can be categorized into two types, Segmented Video and Un-segmented Video. In segmented video, there is only one action instance included, and the starting and ending time of the action is known. In un-segmented video, multiple action instances are involved in a sequence, and the starting and ending time of each action instance is not provided and need to be estimated. Given an input visual sequence V , *Action Recognition* is to determined the action label c of the input by learning a mapping $f : V \rightarrow c \in \{1, 2, \dots, C\}$,

where C is the number of action categories. The input of the action recognition task is always a segmented video. *Action Prediction* is also called *Early Action Recognition*. The goal of *Action Prediction* is to predict the action label of an ongoing activity, which is always a partially observed sequence $V_o \subset V$. Therefore, action prediction is to learn a mapping $f : V_o \rightarrow c \in \{1, 2, \dots, C\}$. The input of action prediction task can be both segmented and un-segmented video sequences. Different from *Action Recognition* and *Action Prediction*, *Action Detection* aims to not only determine the action label of the input un-segmented sequence, but predict the spatio-temporal locations $\mathbf{l} = [x, y, w, h, s, d]$ of the action instance as well. Hence *Action Detection* learns the mapping $f : V \rightarrow \mathbf{l} = [x, y, w, h, s, d]$. The estimated location includes the two-dimensional coordinates (x, y) on the image frame, the width w and the height h of the detection rectangle bounding box, the starting timing t , and the temporal duration d of the detected action instance. For pose sequences in which the spatial location of the actor is known, the *Action Detection* is then degraded as temporal localization problem. In this thesis, the research only focuses on the *Action Recognition* and *Action Prediction* tasks.

FIGURE 1.2: Property of Different Video Data Sources

Thanks to the development of commodity depth cameras, to date, the action sequences are not only RGB-based videos, but they can also be depth-based and pose-based videos. Different from the original videos which consist of multiple RGB image frames, each frame of depth sequence records the distance between the scene to the capture device. The depth images can help well segment the foreground, which, in most of the case, is the action performer, from the background. Compared with RGB-based videos, which bear rich texture information and redundant action information, the pose data that consists of the coordinates of body

joints is semantic and concise. A brief comparison of these three data types is shown in Fig. 1.2. In this thesis, the research focuses mainly on the pose data and the RGB data.

1.2 Motivations and Contributions

Human action recognition or action prediction is to extract features that describe the essence of actions from the video and to issue a corresponding action class label. Therefore, for the human action analysis, two parts owe to be the focuses of the research, *i.e.* *Visual Representation* and *Action Classification* as illustrated in Fig. 1.3. How to design discriminative visual representations (what is action?) and depict suitable classification boundaries (how to differentiate them?) to distinguish one action from others become the critical questions to ask in the day-to-day research. In this thesis, a few possible considerations to these two questions are trying to be provided.

FIGURE 1.3: Basic Workflow of Human Action Classification

1.2.1 What is Action?

The Intra-class and Inter-class variations is one of the main challenges in all classification problems. Due to the high complexity of human actions, human behavior analysis problem is not easy to be solved. For increasing numbers of action classes, the intra- and inter-class variations problem will be more challenging as the overlap between classes will be higher. A well-designed discriminative representation can greatly reduce the variation within each class, and enlarge the margins between classes.

Many factors may cause the intra-class variations of human actions. For example, difference in the performing style of the movements of each action, anthropometric

differences between individuals, the visual noise caused by unstable recording environment, capture devices and estimation errors, the various duration of different action sequence, and the label noise in training data, *etc.*, can all be the reasons causing large intra-class variations and inter-class similarities. Hence, it is critical to discover the key common patterns among these factors to represent actions.

Back to the question “what is action?”, our answer is an action is the combination of key common spatio-temporal patterns. To be more specific, an action can be represented by the movements of informative body parts from key frames. For example, different persons have different drinking styles. One can choose to drink water by sitting, standing or walking. No matter which style a person choose to drink, the raising of one hand with a glass of water to the mouth is the key common patterns of the Drinking Water action. If a method has the ability to search the common patterns to recognize actions, the similarities of samples from the same class can be further boosted and the margins between different classes can be further enlarged. Therefore the discovery of the key common patterns can mollify the negative impact from intra- and inter-class variations. The Attention Mechanism [4, 5] is one of the strategies to solve the problem. In this thesis, we propose three methods towards this direction. These methods model the attention mechanism into the classifier and achieve good recognition performance. The contributions are summarized as below.

- The proposed methods Spatio-Temporal Naive Bayes Nearest Neighbor (ST-NBNN), and Spatio-Temporal Naive Bayes Mutual Information Maximization (ST-NBMIM) in Chapter 3 and Chapter 4 introduce the idea of spatial temporal attention in a non-parametric classifier for better action recognition. The proposed methods utilize Support Tensor Machine (STM) to learn the attention weights on the spatial domain and the temporal domain respectively. With the strengths of both the STM, who introduces the attention mechanism, and the non-parametric model, the proposed ST-NBNN and ST-NBMIM outperform the baselines Naive Bayes Nearest Neighbor (NBNN) and Naive Bayes Mutual Information Maximization (NBMIM) by large margin, both 12% on the MHAD dataset for example, and they achieve competitive performance compared with state-of-the-art results using sophisticated models such as deep learning. The accuracy improvement shows the efficiency of

learned spatial-temporal weights for solving intra- and inter-class variations problem.

- Considering that the spatial and temporal weights learned in ST-NBNN and ST-NBMIM are class-related, they are not adaptive to each action sequence. Besides, as human actions are temporal dynamics of body movements, the key body parts and the coordination of key body parts vary during action performing. We therefore propose the Deformable Pose Traversal Convolution in Chapter 5 to introduce the attention mechanism into the convolution operation. The deformation of the kernel shapes are learned by a Convolutional LSTM, which is able to remember the historical information of action dynamics and predict suitable kernel deformation parameters for 3D pose feature extraction. This proposed deformable pose traversal convolution network outperforms the baseline by 4.5% under the Both-28 setting of DHG dataset, and it achieves comparable accuracy with or beats the state-of-the-arts on three different benchmark datasets, which shows the effectiveness of the proposed joint combination attention learning module.

1.2.2 How to Differentiate Actions?

With discriminative visual representations, how to well depict classification boundary to separate one class from others is another problem to think about. Due to the complexity of human action, it is challenging to determine the boundaries between different classes and generalize it to recognize testing samples. To date, most of the existing methods utilize supervised learning to recognize actions, as the method proposed in Chapter 5. In the setting of supervised learning, one-hot vector ground truth is always involved to activate the probability “response” of positive category and suppress the “responses” of negative ones. In this learning mode, the goal is to distinguish the positive category from all negative categories. However in the real world, the relationships between the positive category and different negative categories are more sophisticated than we thought. From this perspective, we attempt two directions to make contributions.

- Compared with the RGB-based action sequences, which consist of frames of images, the pose-based actions are sequences of human poses. The image data includes hundreds of thousands of pixels, while there are just tens of joints

bear in each pose. It is not easy to face the *curse of dimension* for pose-based feature, and we claim that the pose-based action recognition task is a lightweight problem. We therefore introduce the non-parametric models as the classification methods in Chapter 3 and Chapter 4. Without learning classification boundaries, nearest-neighbor-based methods (NN-based methods) directly utilize the label distribution of local neighbors to make label prediction, which construct complicated “classification boundaries”. The proposed NN-based methods, ST-NBNN and ST-NBMIM, explore the co-operation between non-parametric model and parametric model, and the extensive experiments show the potential of using non-parametric model for pose-based human action analysis. The ST-NBNN and ST-NBMIM are extensively evaluated on up to five benchmark datasets, and the results show that though there are not many parameters to learn, our proposed nearest-neighbor-based models can achieve comparable performance with the sophisticated models like deep neural network.

- In supervised learning, the goal is to enforce the positive category to be far away from all negative categories. However, in real-world applications, the relationships between the positive category and different negative categories are not identical. Due to visual similarity and noise from the input data, the positive samples may be close to some negative clusters, while be far away from the other ones. We therefore introduce the idea of Category Exclusion and apply it on the Action Prediction task in Chapter 6. The exclusion strategy is able to distinguish those negative categories that are similar to positive samples from those not. By excluding negative categories that are similar to the positive one during action performing, our proposed method can effectively improve the overall accuracy of action prediction task. This new auxiliary classification strategy is extensively evaluated on three action recognition datasets, and the results verify the effectiveness of this new thinking of learning.

1.3 Outline of the Thesis

Chapter 1 introduces the background of Human Behavior Analysis to show the great importance of this topic. Then the action hierarchy, basic related tasks, and

the data type used are briefly introduced to locate the research area this thesis focuses on.

Chapter 2 provides a brief literature survey on the recent developments in human action recognition on both RGB data and pose data, as well as the human action prediction task. Both the traditional methods and the deep-learning-based methods are summarized to provide clear view of where the methods proposed in this thesis locate in the community.

Chapter 3 extends the NBNN to ST-NBNN for skeleton-based action recognition. Compared with NBNN, ST-NBNN takes the spatio-temporal structure of 3D actions into consideration and combines the strength of both non-parametric model and parametric model to achieve better performance. We theoretically show that the original NBNN is just a special case of ST-NBNN when the spatial weights and temporal weights are all set to one. Despite using only a linear classifier, the proposed ST-NBNN performs good on three benchmark datasets and achieves competitive results compared with state-of-the-arts using sophisticated end-to-end models. Besides, our proposed method can discover critical spatial joints and temporal stages, which are essential to capture the spatio-temporal patterns of 3D actions, and which is not always achievable using end-to-end models such as deep learning.

Chapter 4 further extends the ST-NBNN classifier proposed in Chapter 3 to ST-NBMIM for skeleton-based action recognition. The proposed ST-NBMIM further explores the relationships of query sequence to the positive and negative clusters, and involves the concept of mutual information to boost the discriminative power of action-snippet descriptor. Besides, we further combine this traditional method with the powerful deep-learning representation and find that ST-NBMIM can cooperate well with the deep-learning feature. Extensive experiments on five benchmark datasets show the efficiency of our proposed ST-NBMIM, which shade light to the direction of NN-based method applied on skeleton-based action recognition task.

Chapter 5 proposes a Deformable Pose Traversal Convolution Network that applies one-dimensional convolution to traverse the 3D pose for its representation. Instead of fixing the receptive field when performing traversal convolution, it optimizes the convolution kernel for each joint, by considering contextual joints with various weights. This deformable convolution better utilizes the contextual joints for action

and gesture recognition and is more robust to noisy joints. Moreover, by feeding the learned pose feature to a LSTM, we perform end-to-end training that jointly optimizes 3D pose representation and temporal sequence recognition. Experiments on three benchmark datasets validate the competitive performance of our proposed method, as well as its efficiency and robustness to handle noisy joints of pose.

Chapter 6 steps towards a new direction for human early action recognition task. The goal of early action recognition is to predict action label when the sequence is partially observed. The existing methods treat the early action recognition task as sequential classification problems on different observation ratios of an action sequence. Since these models are trained by differentiating positive category from all negative classes, the diverse information of different negative categories is ignored, which we believe can be collected to help improve the recognition performance. In this chapter, we step towards to a new direction by introducing category exclusion to early action recognition. We model the exclusion as a mask operation on the classification probability output of a pre-trained early action recognition classifier. Specifically, we use policy-based reinforcement learning to train an agent. The agent generates a series of binary masks to exclude interfering negative categories during action execution and hence help improve the recognition accuracy. The proposed method is evaluated on three benchmark recognition datasets, NTU-RGBD, First-Person Hand Action, as well as UCF-101. The proposed method enhances the recognition accuracy consistently over all different observation ratios on the three datasets, where the accuracy improvements on the early stages are especially significant.

Chapter 7 summarizes this thesis and envisions some future work.

Chapter 2

Literature Review

The methods proposed in this thesis focus on the Action Recognition and Action Prediction. The literature review in this chapter therefore narrows down the scope to these two tasks. The approaches for action analysis can be roughly categorized into the model-based methods and the appearance-based methods. In appearance-based methods, different appearance features extracted directly from the RGB video data are design for better action representation. While in model-based methods, human body tracking or pose estimation on images is utilized to model the motion dynamics of human body parts for action understanding. Due to the success of pose estimation methods [6–10], pose-based action analysis becomes a popular research topic recently. Considering the existing the pose-based methods learn many ideas from the RGB-based methods, in this chapter, both the RGB-based methods and the pose-based methods will be briefly reviewed.

2.1 Action Recognition

In action recognition, the data to be analyzed is fully observed segmented video. Each video includes only one action instance. The goal of action recognition is to take a segmented video as input and output a corresponding category label. This task has been widely explored in the community.

2.1.1 Image-based Method

2.1.1.1 Traditional Method

Considering that action recognition is a basic classification problem, the feature representation of videos is a very important part in research, and it attracts heated attention from researchers. Traditional methods mainly focus on how to design discriminative hand-crafted features for action description. The action representation can be further categorized as low-level feature and mid-level feature.

Low-level video feature is usually extracted from local path or densely sampled from the videos without the physical significance. To extract low-level feature, there are always two steps involved, local interest point detection and interest point description. The well-known approaches for local interest point detection include the Harris 3D detector proposed in [11], the Hessian 3D detector presented in [12], the Cuboids [13, 14] and the Dense Trajectory [15–17]. The Harris 3D detector [11] is designed for the Spatial-Temporal Interest Point (STIP). In STIP, the normalized Laplace operator is maximized iteratively to determine the scale of the interest points. Compared with Harris 3D detector, Hessian 3D detector [12] is able to extract dense interest points. However, the box filter and integral video is involved to reduce the computational cost. The Cuboids detectors [13, 14] utilize the responses of Gabor filter for local interest points detection. The STIPs detected by the Harris 3D detector and the Hessian 3D detector are relatively local compared with Sparse Trajectory [15–17]. Sparse Trajectory models the longer temporal dynamics of interest points, and achieves promising results but with high computational cost. With spatial-temporal positions (interest points or trajectories), the next step is to extract feature from 3D local features around them. The histogram-based descriptors include Histogram of Gradient (HoG), Histogram of Flow (HoF) [11], 3D Histogram of Gradient (HoG3D) [18], and Motion Boundary Histogram (MBH) [19] which utilize the measurement of histogram to summarize the gradient distribution of local patches. 3D Scale Invariant Feature Transform (3D SIFT) [20] is one the descriptors similar to the histogram-based ones. It is extended from the well-known 2D SIFT [21] designed for image description.

Mid-level feature encodes the video content by middle-level components like templates or attributes. It is usually constructed based on the low-level features. The well-known traditional mid-level features include Bag-of-Words (BoW) [22], Vector

of Locally Aggregated Descriptors (VLAD) [23], Fisher Vector [24, 25]. All these three methods introduce the K-means or Gaussian Mixture Model (GMM) to quantize the local descriptors from vast distributions, and represent the set of local descriptors of a video as a histogram. These local descriptor aggregations are inspired from the mid-level representation of images, and they ignore the spatio-temporal relationship among the descriptors. Context and Appearance Distribution Feature is proposed in [26] to take the spatio-temporal relationship among local interest points into consideration. The spatio-temporal context distributions of pairwise interest points over multiple space-time scale are characterized by a few GMMs in this method. The Video Covariance Matrix Logarithm (VCML) [27] is based on a covariance matrix representation to construct relationships between different low-level features. In [28] a simple and efficient idea, Rank Pooling, is introduced to model the temporal evolution of low-level representation and it achieves promising results.

In traditional models, low-level or mid-level features are further fed into classifiers like Support Vector Machine (SVM) [28–30], Hidden Markov Model (HMM) [31], Conditional Random Field (CRF) [32] for classification.

2.1.1.2 Deep-learning-based Method

With the development of Deep Learning, various deep-learning-based models are proposed for action recognition. Different from the traditional methods based on hand-crafted feature, the deep-learning-based methods involve both the feature learning and the classification into one framework by end-to-end learning, which is more efficient than the traditional methods. Until now the deep-learning-based methods have achieved promising performance.

The Convolutional Neural Network (CNN) achieves great success in image analysis and it reaches new level of recognition. Encouraged by the success in image analysis, researchers in video analysis therefore consider how to apply CNN model for action recognition. One intuitive idea is to utilize CNN directly for frame-level feature extraction, and then fuse the features or classification score from each frame for action classification. In [33], different fusion techniques are explored to be applied in video recognition, which initially verifies the feasibility of the usage of CNN on video analysis. In AdaScan [34], an adaptive scan pooling is proposed

to pick the informative frames for better action recognition. CNN model with Rank Pooling fusion is proposed in [35] to model the temporal dynamics of CNN feature from each frame. Considering human action is data with spatio-temporal structure, both the appearance information and motion dynamics should be considered in the recognition. In Two-stream models [36, 37], two CNNs are involved to extract the appearance and motion features of action sequences respectively. However, the temporal dynamics of actions are weakly involved in these models. Another direction of CNN feature fusion is to involve the Recurrent Neural Network (RNN) [38, 39] in which the extracted CNN feature from each frame is further fed into a RNN for temporal modeling. The three dimensional convolution neural network (3D CNN) [40–43] is another representative model. In 3D CNN, the two-dimensional convolution operator in 2D CNN is extended to fully utilize the spatio-temporal information of action sequence, which achieves satisfactory recognition performance. However, one drawback of the 3D CNN model is that it bears many parameters to be learned. During training, a large number of training samples are required to feed the model.

2.1.2 Pose-based Method

In pose-based action recognition, the input data is a sequence of human 3D poses. Each pose consists of the three-dimensional coordinates of each body joint. The pose data mainly comes from two sources: 1) Directly from the motion capture system; 2) Estimated from the RGB-based video [9, 10] or depth-based video [6–8, 44]. The state-of-the-arts can be separated into two groups, the traditional models, and the deep-learning-based models. The traditional methods focus on artificially designing discriminative features for good classification performance. However, these hand-crafted features need a lot of effort on the design, and given the limited complexity of the classifier, these methods are only suitable for small-size dataset. The renaissance of deep learning technique boosts the development of the machine learning community. The increased structure complexity and the sophisticated non-linearity of deep learning model bring effectiveness of machine learning on large-scale datasets.

2.1.2.1 Traditional Method

Wang *et al.* propose Actionlet Ensemble Mining [45] to search for the discriminative action representation of 3D action. Compared with Action Ensemble Mining [45], which groups joint locations into actionlet ensembles, the approach proposed in [46] is built on body parts. Besides, both the spatial pose structure and temporal pose evolutions are modeled in [46]. In SMIJ [47], the most informative joints are selected simply based on measures such as mean or variance of joint angle trajectories. The sequence of these informative joints is then used as the representation of actions. SMIJ shows that it is of great importance to find the informative joints for better recognition. EigenJoints [48, 49] follows the similar direction as SMIJ, however with a unsupervised method Principal Component Analysis (PCA). As each dimension of the joint feature bears physical significance, the dimensions of the feature with high variance picked by PCA are directly related to the informative joints. In Orderlet [50], Yu *et al.* show that for discriminative action recognition, to discover the informative coordinations among the joints is as critical as to search for the individual informative joints. Therefore, Orderlet is defined as the interaction between joints by involving a few comparisons of joints' primitive feature. The informative components discovery is not only involved on spatial domain of pose action sequence, but also the temporal domain. Wang *et al.* [51] propose a sequential pattern mining method to model the temporal structures of a set key poses, which is defined as the key-motifs of 3D actions. Inspired by these methods focusing on key joints discovery, the two methods ST-NBNN [52] and ST-NBMIM [53] proposed in this thesis introduce the attention mechanism into the non-parametric classifier and achieve good performance. The proposed ST-NBNN and ST-NBMIM can jointly learn the key spatio-temporal patterns for different action categories, and the learned patterns can well interpret the actions to be recognized, which is not well explored in the previous traditional methods.

Apart from key patterns mining as introduced above, another direction for 3D action modeling is to explore how to fully utilize the graph structure (geometry) of 3D pose. In [54], the rotation and translations between various body parts in 3D space are used to describe human pose. With this representation, human actions can be regarded as curves in Lie Group. Considering that classification of curves in Lie Group is not easy, Vemulapalli *et al.* [54] therefore map the action curves into a vector space, Lie Algebra. In [55], the idea is similar to [54]. The main difference

is that the action curves in Lie Group are unwrapped while rolling, which achieves better recognition performance than [54]. Before the Graph Convolution Neural Network (GCN) is warmly discussed in the community, Wang *et al.* [56] propose a traditional graph-based pose motion representation for 3D action recognition, which fully utilizes the topological structure of 3D human pose as a graph. In [57], the tensor representations are explored to capture the high-order relationships between 3D skeleton joints.

Other traditional methods are introduced as below. Xia *et al.* propose the histograms of 3D joints to recognize 3D actions. The histograms of 3D joints can well handle the situation of multi-view. Wang *et al.* [58] represent an action sequence by multiple action-snippets, each of which consists of fixed number of poses. The action-snippets are normalized and projected to the surface of a hyper unit ball. The authors propose to use the simplices to represent clusters of action-snippets on the hyper unit ball. Garcia-Hernando *et al.* extend the traditional Random Forest to model the temporal transitions of 3D pose and propose the Transition Forest [59], which has good performance on both the action recognition and action detection tasks.

2.1.2.2 Deep-learning-based Method

The deep-learning-based methods can be further categorized as RNN-based model, and CNN-based model, which are the two representative models in deep learning.

The RNN-based models mainly use the Recurrent Neural Network (RNN) to model the temporal dynamics of action sequence. In Hierarchical Bidirectional Recurrent Neural Network (HBRNN) [60], the 3D human pose is decomposed into five body parts, two arms, one torso, and two legs. Five bidirectional RNNs are then involved to extract features for five body parts, and the hidden features from these five RNNs are further fed into the higher-level RNNs hierarchically to model the relationship among these body parts. Similar to HBRNN, Shahroudy *et al.* propose a part-aware LSTM [61] to construct the relationship between body parts. Considering that the co-occurrences of body joints intrinsically characterize human actions, Zhu *et al.* [62] introduce a novel regularization scheme into LSTM to learn the co-occurrence features of 3D body joints. Li *et al.* propose a Classification and Regression RNN [63] to recognize action and regress the start and end time of

action instances simultaneously. The traditional LSTM model is usually used for temporal modeling. However, in ST-LSTM [64], the LSTM is re-designed to construct the spatio-temporal dependency of pose joints in action sequence. This work is further extended in GCA-LSTM [65] by introducing the global attention mechanism. Song *et al.* propose a Spatio-Temporal Attention LSTM (STA-LSTM) [66] for 3D action recognition, in which two sub-LSTMs are involved for spatial and temporal attention dynamics learning respectively. Inspired by the Two-stream models [36, 37] proposed for image-based action recognition, Wang *et al.* [67] propose to model the spatial configuration and the temporal dynamics of pose actions by two RNNs. An adaptive RNN tree is designed in [68] to classify actions in a hierarchical way, which is able to distinguish fine-grained action categories and is flexible when new class is involved. The RNN-based methods introduced above only model the long-term dependency, and the short-term and mid-term dependencies of actions are ignored. In Temporal Sliding LSTM (TS-LSTM) proposed in [69], the short-term, mid-term, and long-term LSTMs are combined as one to model the multiple temporal scale of human actions.

The convolution neural network is applied in pose-based action recognition in three different ways. In Deformable Pose Traversal Convolution Network [70], one-dimensional convolutional operator is used for pose feature extraction. Each convolutional kernel can deform to capture the key joint combinations and therefore better represents poses. This kernel deformation design can effectively discover key joint combinations and fully exploit the rich joint combination in each convolution operation. This is a spatial modeling. In Scale Selection Network (SSNet) [71], one-dimensional convolution network [72] is involved to extract temporal features of partial action sequence for temporal modeling. Considering that different layers of the CNN model bear the information of temporal receptive fields with different size, the proposed SSNet can well handle the variation of action scale during online action recognition. Another direction of applying CNN into the pose-based action recognition task is to transform a pose sequence into an image, and apply 2D CNN to extract spatial temporal features. In [73], each 3D pose sequence is transformed to trajectory maps, and 2D CNN is used to extract sequence-level representation for action recognition, which can be regarded as spatio-temporal modeling. The similar framework is also proposed in [74, 75]. In [76], Liu *et al.* propose to represent actions as the evolution of pose estimation maps, and utilize the 2D CNN to extract sequence-level features.

Compared with an image, a pose is actually an undirected acyclic graph, which is not suitable to be processed directly by the CNN model. The recently discussed Graph Convolutional Neural Network (GCN) is able to well handle this problem. To date, the GCN models are designed in two perspectives, the spectral perspective, and the spatial perspective. In spectral perspective, the input graph is transformed into a spectral domain for convolution, while in spatial perspective, the convolution operation is performed directly on the input graph. The GCN model proposed in [77] follows the spectral perspective. Li *et al.* combine the strength of local convolution filtering and the recursive learning power of autoregressive moving average, and propose a spatio-temporal graph convolution (STGC) approach for skeleton-based action recognition. The ST-GCN [78] follows the spatial perspective. In this method, different graph convolution partitioning strategies are explored to achieve better action recognition performance. In [79], a reinforcement framework is proposed to select the key frames of action sequences for action recognition, in which a GCN is involved as the classifier for both the classification and the critic in agent training. Due to the success of GCN, in [80], the graph convolution operation is involved in LSTM for better temporal dynamics modeling of pose graph.

2.2 Action Prediction

Dissimilar to the action recognition task, where actions are well-segmented and assumed to be fully executed and observed, early action recognition, or action prediction a relatively new research topic in action analysis, is another important task aiming to predict the action label contained in a partially observed video. Due to the specific setting of the action prediction, the category recognition needs to be performed at each observation ratio. To better recognize the actions at early stages, the visual representation from each observation ratio should be discriminative enough to distinguish one action from others. Therefore, how to improve the discriminative ability of the visual representation from each stage, or how to fully utilize the information from the partially observed action sequences becomes the core problem in the research community.

Before deep learning technique is widely-applied in computer vision, there are already many traditional methods proposed to solve the early action recognition

task. To improve the performance of action prediction, there are two ways to consider. 1) Design good visual feature to better represent partially observed actions from different observation ratios. 2) Calibrate classifier or learning strategy to well draw decision boundaries between action clusters. On the direction of feature design, Ryoo *et al.* [81] propose to involve the well-known integral bag-of-words and dynamic bag-of-words into the action prediction task. The integral and dynamic bag-of-words progressively accumulate the bag-of-words representation of previous stages and construct the temporal dynamics of visual representations from different observation ratios. Cao *et al.* [82] introduce the Sparse Coding to solve the action classification of partially observed sequences. The partial observed action in [82] is defined as the video with unobserved temporal gap at any time, which is more general than the early action recognition task. Meanwhile, the observation ratio or the happening time of each part of an action sequence is well encoded. In [83], a new representation, hierarchical movemes is proposed to describe human movements at multiple levels of granularities so that the large temporal extent of human actions can be covered. Li *et al.* [84] propose a novel framework for long-duration complex activity prediction, in which the action and object co-occurrence is modeled as a context cue for recognition, and a predictive accumulative function (PAF) is proposed to show the predictability of activities.

To better calibrate the classifier and design a good learning strategy, Hoai *et al.* [85, 86] re-design the Structured SVM to process the sequential data. The classification response of the partially observed action sequence is required to be lower than the the one of the full sequence so that the temporal order of the visual representation is modeled in the classifier. Similar to [85], Kong *et al.* [87] introduce the SVM into the early action recognition to enforce the large-margin between action categories. In [88], the early action recognition task is formulated as a query auto-completion (QAC) problem in information retrieval. A soft regression framework is proposed in [89] for activity prediction. Considering that actions at the early stage are not easy to differentiate, the author propose to learn the feature with soft label supervision at the early stages.

Thanks to the development of deep learning techniques, the representation learning and classifier training can be combined in an end-to-end manner, which boosts the performance of action prediction models. In [90], deep learning model CNN is utilized to predict the visual representations of future actions, and the predicted

representations are further used as the feature for classification. Hu *et al.* extend their soft regression framework [89] into the RNN model [91], which achieves great recognition performance. Kong *et al.* attempt to bridge the relationship between partial video features and full video features, and propose the Deep Sequential Context Network (DeepSCN). The Long-Short-Term-Memory (LSTM) model is re-designed in [92] to memorize hard-to-predict samples for early action recognition.

The previous works consider the early action recognition as a series of classification problems. These methods mainly focus on how to distinguish the positive category from all the negative ones. All the negative categories are treated equally without any bias. However, considering the diversity of actions, the representations of negative categories are clustered in the feature space in various way with different impacts on the recognition of the positive category, and this situation is even severe when taking the different observation ratios into consideration. In this thesis, the diverse information of negative categories is taken into consideration. A reinforcement-learning-based method is proposed for negative category exclusion. With the proposed method, the informations from the action sequences are not only used for positive category recognition, but also for negative categories exclusion, and therefore the information from action sequences can be fully utilized.

2.3 Conclusion

This chapter gives a summarization of the so far proposed methods for RGB-based and Pose-based action recognition. The existing methods for action prediction (early action recognition) is also summarized in the chapter. The specific properties of these methods are introduced, and their differences with the proposed methods in this thesis are pointed out. More detailed and critical analysis of the related works and their relationship to the works proposed in this thesis will be involved in each corresponding chapter.

Chapter 3

Spatio-Temporal Naive-Bayes Nearest Neighbor

3.1 Introduction

Despite the great progress of using learning-based methods for 3D action recognition, on the other hand, non-parametric classifiers, which make classification decision directly on the data and require no learning/training of parameters, are not well explored for 3D action recognition. Interestingly, for image recognition, non-parametric methods, e.g., *Naive-Bayes Nearest-Neighbor* (NBNN) [93], have shown an impressive performance of classifying images of local visual primitives by using image-to-class distance. Motivated by the previous success of NBNN, in this chapter, we extend it to recognize actions.

Two observations motivate our exploration of using NBNN for skeleton-based action recognition: (1) similar to images that are composed by local visual primitives, actions are composed by spatio-temporal primitives too, e.g., each action instance is a collection of skeleton poses and each pose is further a collection of spatial joints. We can easily apply *primitive-to-class* distance to perform action recognition, which can generalize according to NBNN. (2) Compared with images and videos which are composed of millions or billions of pixels, the skeleton is composed by only tens of joints which is thus of much less complexity than images and videos. We argue that instead of relying on a sophisticated end-to-end model, a simple non-parametric model may still obtain promising results for such a light-weight problem.

FIGURE 3.1: Illustration of Key Stage, Joints (waving right hand action)

In this chapter, we propose *Spatio-Temporal Naive-Bayes Nearest-Neighbor* (ST-NBNN), a new variation of NBNN, to classify 3D actions. Each 3D action instance is represented by a collection of temporal stages composed by 3D poses, and each pose in stages is presented by a collection of spatial joints. Following NBNN, our ST-NBNN applies stage-to-class distance to classify actions. It can well handle the length variations of 3D actions and also the large intra-class variations. However, not every temporal stage and spatial joints are of equal importance to the recognition of the action. Thus it is of great importance to identify the key stages and skeleton joints that matter for the recognition. To extend NBNN for 3D action classification, our ST-NBNN considers the spatio-temporal structure of actions. Instead of simply summing up all stage-to-class distances with a Naive-Bayes assumption, we present these distances as a spatio-temporal matrix of NN distances which represents the action instance. ST-NBNN further adopts a bilinear classifier [94] to identify key joints and stages and classify the spatio-temporal matrix of NN distances. Our proposed formulation can be iteratively optimized to learn the linear classification weight for both spatial joints and temporal stages.

We use Fig.3.1 to illustrate the idea of using key spatial joints and temporal stages for action recognition. When performing right hand waving action, only the right hand and arm (key joints) are activated. And when observing the timing (key stage) at which the right hand and arm raise up and move horizontally towards left, we can claim that waving right hand action is performing. Such a spatio-temporal pattern described by key temporal stages and spatial joints is critical to identify action classes. The discovery of such patterns not only can improve recognition accuracy but also answer what composes such an action instance and why we make a recognition of it.

By using both stage-to-class distance and bilinear classifier [94], our proposed ST-NBNN combines the strengths of non-parametric model and parametric model. Although only using a linear classifier, experiments on three benchmark datasets show that ST-NBNN using raw skeleton features can already obtain very competitive performance compared with state-of-the-art end-to-end models which optimize for feature representations. Moreover, by identifying key temporal stages and spatial joints which play key roles of recognizing the action, our ST-NBNN can capture the essential spatio-temporal patterns for each action class, and provide a physical interpretation of the action behavior. Such a spatio-temporal pattern discovery and explicit interpretation, however, is not always available via end-to-end models, which mainly focus on achieving higher recognition accuracy instead of better interpreting patterns.

3.2 Related Work

3.2.1 Spatio-Temporal Modeling for 3D Action Recognition

In recent years, skeleton-based action recognition problem attracts a lot of attention, and many learning-based methods [60–64, 95] have been proposed. Due to the tremendous amount of these works, we only limit our review to the spatio-temporal modeling of skeleton-based action recognition.

The modeling in spatial domain is mainly driven by the fact that an action is usually only characterized by the interactions or combinations of a subset of skeleton joints [62]. Two categories of approaches are often used to model the spatial pattern of actions: part-based model and sub-pose model. In part-based model, a skeleton is partitioned into several groups, and the joints in each group are skeletal neighbors of each other. In HBRNN [60], skeletons are decomposed into five parts, two arms, two legs and one torso, and a hierarchical recurrent neural network is build to model the relationship among these parts. Similarly, in [61] a part-aware LSTM is proposed to construct the relationship between body parts. In sub-pose model, the focus is mainly on the interaction of informative joints. In SMIJ [47], the most informative joints are selected simply based on measures such as mean or variance

of joint angle trajectories. The sequence of these informative joints is then used as the representation of actions. In Orderlet [96], interactions between joints are modeled by a few comparisons of joints’ primitive feature, and in action recognition only a subset of joints are involved. On the temporal domain, graphical models [97, 98], temporal pyramid matching [45, 99], and dynamic time warping [100] are the common methods for temporal modeling. While in [51], sequential pattern mining method is used to model temporal structures of a set of key poses. Besides spatial modeling or temporal modeling, we also see efforts on spatio-temporal modeling. In [64], LSTM model is extended to spatio-temporal domain to analyze skeletons. Compared with these methods, our proposed ST-NBNN is able to discover key factors of an action on spatial and temporal domain simultaneously.

3.2.2 Naive-Bayes Nearest-Neighbor

NBNN [93] is first proposed for image classification. Two key factors help NBNN achieve remarkably good results. First, it avoids using vector quantization to code primitive features so that rare but discriminative descriptors counts in classification. Second, it chooses ‘image-to-class’ distance rather than ‘image-to-image’ distance to make decision that it bears a good generalization ability. Due to its success, there are a few variations of the original NBNN algorithm. In [101] Behmo *et al.* parameterize NBNN to relax its restrictive assumption that all classes have similar densities in feature space. In kernelized version of NBNN [102], independent assumption is criticized by introducing kernel between image representations. Compared with NBNN kernel [102], we also break the independent assumption by incorporating the spatio-temporal structure of the action data. Following NBNN, we also see its application in video analysis. In [103], NBNN is re-designed as *Naive-Bayes based Mutual Information Maximization* (NBMIM) to solve action detection problem. Negative samples are involved in nearest neighbor matching to improve discriminative ability of descriptors. Recently, the combination of NBNN and CNN [104] as well as the effort to speeding up NN search [105] revive the possibility of NBNN’s return in computer vision.

FIGURE 3.2: Overview of the Proposed Method

3.3 Methodology

In this section, we introduce how the proposed method predicts actions and discovers key joints and stages as well. We first introduce a set of stage-descriptors to represent a 3D sequence (Sec. 3.3.1). Then NBNN [93] is used as a basic framework to classify actions (Sec. 3.3.2). Finally, the learning of spatial and temporal weights is introduced to discover key poses and spatial joints for action recognition (Sec. 3.3.3). The overview of our method is illustrated in Fig. 3.2. 1) An action video is uniformly divided into a fixed number of stages, and is represented by a set of stage-descriptors (orange query points); 2) Distances of stage-descriptors to action class sets (blue, green and red) are calculated by NN search; 3) Distances of stage-descriptors are gathered in temporal order to generate class-related squared distance matrices (marked by class-related dashed rectangular boxes); 4) Weights on the spatial (left side of the matrix) and the temporal (top of the matrix) domain are learnt to discover key factors of actions and predict action labels.

3.3.1 3D Action Representation

In skeleton-based action recognition, each 3D action is a sequence of 3D poses, but different actions may have different temporal lengths. To provide a unified presentation, we partition each action into N temporal windows of equal length as

shown in Fig. 3.2. Each one of the temporal windows is called a *temporal stage*, which is characterized by the 3D poses in its corresponding window. Assuming each 3D pose has J joints for its skeleton, for a temporal stage descriptor $\mathbf{x}$, the 3D pose in its j th frame is denoted as $\mathbf{p}_j \in \mathbb{R}^{3J}$, and the related velocity of that pose is denoted as $\mathbf{v}_j \in \mathbb{R}^{3J}$. Then the pose part $\mathbf{x}_p$ and the velocity part $\mathbf{x}_v$ of $\mathbf{x}$ is defined as below:

$$\begin{aligned}\mathbf{x}_p &= [(\mathbf{p}_1)^\top, \dots, (\mathbf{p}_l)^\top]^\top \\ \mathbf{x}_v &= [(\mathbf{v}_1)^\top, \dots, (\mathbf{v}_l)^\top]^\top\end{aligned}\tag{3.1}$$

Similar to [58], we also normalize $\mathbf{x}_p$ and $\mathbf{x}_v$ to have l_2 norm equals to one. In our experiment, it shows such a normalization can perform slightly better than using the original features. As we use both original 3D pose and its velocity to represent 3D actions, a temporal stage descriptor $\mathbf{x}$ of l frames is presented as:

$$\mathbf{x} = [(\mathbf{x}_p)^\top, (\mathbf{x}_v)^\top]^\top\tag{3.2}$$

Finally, a 3D action video is described by its N stage-descriptors $V = \{\mathbf{x}^i\}_{i=1}^N$.

3.3.2 Naive Bayes Nearest Neighbor

Given a query action video $V_q = \{\mathbf{x}^i\}_{i=1}^N$, the goal is to find which class $c \in \{1, 2, \dots, C\}$ the video V_q belongs to. NBNN follows maximum a posteriori (MAP) rule for classification. When assuming equal prior $p(c) = \frac{1}{C}$, the predicted class is:

$$c^* = \arg \max_c p(c|V_q) = \arg \max_c p(V_q|c)\tag{3.3}$$

With the Naive-Bayes assumption (data samples are independent with each other), Eq. 3.3 can be written as

$$\begin{aligned}c^* &= \arg \max_c p(V_q|c) \\ &= \arg \max_c p(\mathbf{x}^1, \dots, \mathbf{x}^N|c) \\ &= \arg \max_c \prod_{i=1}^N p(\mathbf{x}^i|c)\end{aligned}\tag{3.4}$$

Based on the analysis of [93], the probability density of each primitive $\mathbf{x}$ in a class c , namely $p(\mathbf{x}|c)$, can be estimated according to the distance between $\mathbf{x}$ and the nearest neighbor of $\mathbf{x}$ in class c , and Eq. 3.4 is then re-written as

$$c^* = \arg \min_c \sum_{i=1}^N \|\mathbf{x}^i - NN_c(\mathbf{x}^i)\|^2 \quad (3.5)$$

where $NN_c(\mathbf{x}^i)$ is the nearest neighbor of $\mathbf{x}^i$ in class c .

As a non-parametric model, NBNN has no training phase. For a query 3D action $V_q = \{\mathbf{x}^i\}_{i=1}^N$, each of its temporal stages will match against C classes separately by finding the best matched temporal stage, i.e., nearest neighbor, in that class. The better the match, i.e., the smaller the NN distance, the stronger this temporal stage will vote for that class c . Otherwise, if a temporal stage cannot find a good match in that class, it will not provide a strong vote to that class. As V_q has in total N temporal stages, the final decision is the summation over all of the N votes towards to C classes, as explained in Eq. 3.5.

3.3.3 Spatio-Temporal NBNN

For a specific action, usually only a subset of joints are activated for action performing, and for different actions, their activated joints are different. Hence in action classification, we can only focus on the activated spatial joints, and at the same time suppress those that are not discriminative or easily affected due to occlusions or capturing noises. Meanwhile, for a set of temporal stages, not every stage is of equal importance neither. Depending on action class, a certain temporal stage can be more discriminative than others for classification. As illustrated in Fig. 3.2, the descriptor (shadowed orange query square) of stage i is more discriminative than the beginning one and the ending one.

To identify important spatial joints and temporal stages simultaneously, we propose to leverage a bilinear classifier [94] to learn spatio-temporal weights for both of them in the framework of NBNN. Although previous works have studied how to pick important spatial joints or temporal poses, not many works can address both of them simultaneously. In the following section, we will introduce how the proposed ST-NBNN can effectively learn the spatial attentive weights and temporal attentive weights jointly.

Spatio-temporal Matrix

Though we assume that stage descriptors are independent in NBNN, they actually depend on each other in a certain spatio-temporal structure. Therefore, to capture the spatio-temporal structure of 3D actions representation, we represent a 3D action from a set $V = \{\mathbf{x}^i\}_{i=1}^N$ to a matrix. For a given video sample with N stages, its spatio-temporal matrix is defined as

$$\mathbf{X} = [\mathbf{x}^1, \dots, \mathbf{x}^N] \quad (3.6)$$

Stage-descriptors of an action are re-organized column by column following the temporal order. We further define the nearest neighbor matrix of $\mathbf{X}$ in c as $\mathbf{X}_c^{NN} = [NN_c(\mathbf{x}^1), \dots, NN_c(\mathbf{x}^N)]$, and the squared distance matrix to class c is defined as

$$\mathbf{X}_c = (\mathbf{X} - \mathbf{X}_c^{NN}) \odot (\mathbf{X} - \mathbf{X}_c^{NN}) \quad (3.7)$$

where $\odot$ is an element-wise product. $\mathbf{X}_c$ is regarded as a representation of $\mathbf{X}$ for class c , as illustrated in Weights Learning block of Fig. 3.2, and it is a combination of element-wise stage-to-class distances of the testing sample. Summation of all the elements in $\mathbf{X}_c$ is equivalent to $\sum_{i=1}^N \|\mathbf{x}^i - NN_c(\mathbf{x}^i)\|^2$ in Eq. 3.5.

Considering elements of $\mathbf{X}_c$ bear different contributions to classification, NBNN should be parameterized to emphasize those discriminative ones. We can simply vectorize $\mathbf{X}_c$ as $\boldsymbol{\chi}_c$, and the NBNN decision function Eq. 3.5 is then re-defined as $c^* = \arg \min_c \mathbf{w}^\top \boldsymbol{\chi}_c$, where the weight $\mathbf{w}$ can be learnt by linear SVM. However, as $\mathbf{X}_c$ is a large matrix, the number of weights to be determined is too many. This strategy is not only time consuming but also has the risk of over-fitting. Hence, a bilinear classifier is adopted in our formulation.

Based on the squared distance matrix in Eq. 3.7, the classification score of a query matrix $\mathbf{X}$ to class c is then determined by a bilinear function $f_c(\cdot)$, which is defined as

$$f_c(\mathbf{X}_c) = (\mathbf{u}_c^s)^\top \mathbf{X}_c \mathbf{u}_c^t \quad (3.8)$$

where $\mathbf{u}_c^s \in \mathbb{R}^M$ and $\mathbf{u}_c^t \in \mathbb{R}^N$ are the spatial and temporal weights of action class c . As a result, the classification becomes

$$c^* = \arg \min_c f_c(\mathbf{X}_c) \quad (3.9)$$

As can be seen from Eq. 3.8, the proposed method provides weights for both temporal stages and spatial joints. After a rearrangement, Eq. 3.9 can be represented as,

$$c^* = \arg \min_c \sum_{i=1}^N \mathbf{u}_c^t(i) \left\| (\mathbf{x}^i - NN_c(\mathbf{x}^i))^T \sqrt{\mathbf{u}_c^s} \right\|^2 \quad (3.10)$$

where $\sqrt{\cdot}$ is an element-wise square-root of a vector. It is worth noting that, NBNN is a special case of ST-NBNN. When $\mathbf{u}_c^s$ and $\mathbf{u}_c^t$ are assigned to $\mathbf{1}$, Eq. 3.10 becomes NBNN in Eq. 3.5. ST-NBNN generalizes NBNN by breaking the Naive-Bayes rule. Instead of assuming each stage is independent, we introduce the spatio-temporal structure of 3D action into our framework.

Spatial and Temporal Weights Learning

Our objective function is similar to tensor SVM. Following the learning strategy of [106], we adopt the one-vs.-all strategy to classify actions. With empirical loss, the objective function of spatio-temporal weight learning is defined as

$$\begin{aligned} \min_{\mathbf{u}_c^s, \mathbf{u}_c^t} \quad & \frac{1}{2} \|\mathbf{u}_c^s(\mathbf{u}_c^t)^T\|^2 + \lambda \sum_{i=1}^K \xi_i \\ \text{s.t.} \quad & \sum_{i=1}^N \mathbf{u}_c^t(i) = N, \quad \mathbf{u}_c^t \succeq \mathbf{0} \\ & \xi_i \geq \max(0, 1 - c_i f_c(\mathbf{X}_c^i))^2 \\ & \xi_i \geq 0, \quad i = 1, \dots, K \end{aligned} \quad (3.11)$$

in which K is the number of training video samples, and $c_i \in \{-1, 1\}$ is the action label of the corresponding sample. $\mathbf{X}_c^i$ is the i th training sample in class c . λ is a parameter for classification error penalty.

The reason why we set linear constraints to temporal weights but not to spatial weights is as following. For the spatial domain we do not know how many key joints will be involved (some joints do not have any contribution in recognition), and how important those activated joints compared with others, while on the temporal domain, each stage of an action counts in classification. Experiment results also

show that linear constraints on the spatial domain do not bear any contribution to performance, but the temporal constraints do.

The optimization of Eq. 3.11 is regarded as an iterative process. There are two steps in each iterative round, 1) fix $\mathbf{u}_c^t$ and update $\mathbf{u}_c^s$, 2) fix $\mathbf{u}_c^s$ then update $\mathbf{u}_c^t$. $\mathbf{u}_c^t$ is initialized to $\mathbf{1}$.

Fix $\mathbf{u}_c^t$ and Update $\mathbf{u}_c^s$: With $\mathbf{u}_c^t$ fixed, Eq. 3.11 is treated as a l_2 -regularized l_2 loss SVM problem shown below

$$\min_{\mathbf{u}_c^s} \frac{1}{2}\beta_1\|\mathbf{u}_c^s\|^2 + \lambda \sum_{i=1}^K \max(0, 1 - c_i f_c(\mathbf{X}_c^i))^2 \quad (3.12)$$

where $\beta_1 = \|\mathbf{u}_c^t\|^2$.

Fix $\mathbf{u}_c^s$ and Update $\mathbf{u}_c^t$: With updated $\mathbf{u}_c^s$, Eq. 3.11 is regarded as a convex optimization problem with linear constraints shown below

$$\begin{aligned} \min_{\mathbf{u}_c^t} \quad & \frac{1}{2}\beta_2\|\mathbf{u}_c^t\|^2 + \lambda \sum_{i=1}^K \max(0, 1 - c_i f_c(\mathbf{X}_c^i))^2 \\ \text{s.t.} \quad & \sum_{i=1}^N \mathbf{u}_c^t(i) = N, \quad \mathbf{u}_c^t \succeq \mathbf{0} \end{aligned} \quad (3.13)$$

where $\beta_2 = \|\mathbf{u}_c^s\|^2$.

This optimization process is operated iteratively until the objective function Eq. 3.11 converges.

3.4 Experiment

In this section, we experiment the proposed method on three 3D action datasets and compare its performance to existing methods. Implementation details are provided in Sec. 3.4.1. Comparison results on the MSR-Action3D dataset [107], the UTKinect dataset [98], and the Berkeley MHAD dataset [108] are provided and discussed in Sec. 3.4.2. Experiment results show that though ST-NBNN is simple, it is able to achieve state-of-the-arts performance of 3D action recognition and effectively discover key factors of actions.

FIGURE 3.3: Parameter Sensitivity Analysis on MSR-Action Dataset

3.4.1 Implementations

3D Action Representation. The one-vs.-all strategy is used in this method. To ensure the responses of linear functions $f_c(\cdot)$ are comparable with each other, each sample $\mathbf{X}_c^i$ is mean-centralized by $\mu^i = \sum_{c=1}^C \text{sum}(\mathbf{X}_c^i) / (C \times M \times N)$, where $\text{sum}(\cdot)$ sums up entries of the input matrix.

The setting of stage number is indicated in the Sec. 3.4.2. Considering the variation of action sequences' duration, stages defined in Sec. 3.3.1 may have overlaps with each other stages when given sequences are not too long.

To ensure that the representation introduced in Sec. 3.3.1 is location-invariant, each joint of the skeleton is centralized by subtracting coordinates of the hip joint.

Nearest Neighbor Search. The nearest neighbor search is implemented by the KD-tree implementation [109]. Representations of each class cluster are re-organized by a KD-tree so that the nearest neighbor search process can be boosted.

Spatio-Temporal Weights Learning. The training matrices $\mathbf{X}_c$ are generated by a leave-one-video-out strategy, which means all the stage-descriptors of query

Method	AS1	AS2	AS3	Ave.
NBNN- ∞	85.8	92.0	96.4	91.4
NBNN-15	86.8	92.0	96.4	91.7
NBNN+SVM	90.6	90.3	96.4	92.4
Lie Group [54]	95.4	83.9	98.2	92.5
SCK+DCK [57]	—	—	—	94.0
HBRNN [60]	93.3	94.6	95.5	94.5
ST-LSTM [64]	—	—	—	94.8
Graph-Based [56]	93.6	95.5	95.1	94.8
Ours	91.5	95.6	97.3	94.8

TABLE 3.1: Results on MSR-Action3D (%)

training video are excluded in searching region when query stage-descriptors search for nearest neighbors.

In our optimization, $\mathbf{u}_c^s$ and $\mathbf{u}_c^t$ are learnt iteratively. To solve the SVM problem of Eq. 4.16, we use a SVM toolbox [110] implemented by Chang *et al.*, and to update $\mathbf{u}_c^t$, a convex optimization toolbox [111] is used.

3.4.2 Results and Analysis

MSR-Action3D

Here we use the evaluation protocol described in [107]. 20 actions are grouped into three subsets AS1, AS2 and AS3. Each of the action set contains eight actions. In this experiment, the number of local pose l is 10 and the number of stage N is 15. We compare ST-NBNN with three baseline methods and five state-of-the-art skeleton-based methods. The results are shown in Table. 3.1.

Three baseline methods included in the comparison are (1) NBNN without stage setting (NBNN- ∞); (2) NBNN with 15 stages (NBNN-15); and (3) NBNN with weights learning by linear SVM (NBNN+SVM). In NBNN- ∞ , the number of stage N is determined by $N = P - l + 1$, where P is the length of the action video. Table. 3.1 shows that there is a slight improvement from NBNN- ∞ to NBNN-15, which indicates that the informative stages do exist to help differentiate actions. In NBNN+SVM, we use linear SVM to learn weights for every element of $\mathbf{X}_c$. As we can see, learnt weights do help improve the performance, especially in AS1.

Method	Accuracy
NBNN- ∞	95.5
NBNN-15	95.5
NBNN+SVM	94.0
Key-Motif [51]	93.5
Simplices [58]	96.5
ST-LSTM [64]	97.0
Lie Group [54]	97.1
Graph-Based [56]	97.4
SCK+DCK [57]	98.2
Ours	98.0

TABLE 3.2: Results on UTKinect (%)

Method	Accuracy
NBNN- ∞	88.0
NBNN-20	88.0
NBNN+SVM	100.0
SMIJ [47]	95.4
Meta-cognitive [112]	97.6
Kapsouras <i>et al.</i> [113]	98.2
HBRNN [60]	100.0
ST-LSTM [64]	100.0
Ours	100.0

TABLE 3.3: Results on Berkeley MHAD (%)

However, it also causes over-fitting. In AS2, although the training accuracy has already achieved 100%, the testing accuracy drops. Besides, NBNN+SVM takes much more time to learn as it has much more parameters compared with the proposed method. The results in Table. 3.1 show that ST-NBNN bears the ability to achieve even better performance than NBNN+SVM. The comparisons with five state-of-the-art methods show that the proposed method achieves the current best performance. In addition, our method is better than the non-linear models [54, 56, 57], and the deep learning based methods [60, 64].

We also evaluate the two main parameters, N and l of ST-NBNN on this dataset. We range l from 1 to 13 and set N from 1 to 21. In Fig. 3.3 the x-axis indicates the chosen number of local poses. The sub-title indicates the chosen number of temporal stages. As Fig. 3.3 shows, ST-NBNN needs a sufficient number of stages to learn the spatio-temporal weights and obtain good performance. When the number of stages is larger than 11, ST-NBNN can still help improve the performance with only 3 poses in each stage. However, further increasing N and l will not improve the performance.

The key stages, joints discovered from MSR-Action 3D dataset are shown in Fig. 3.6. Colored joints are with weights larger than average weights. The most informative joints are marked by bright color, and the second most informative joints are marked by light color. The global key motions are indicated by different colors. For example, the key motion directing in the x direction is colored by bright red

FIGURE 3.4: An Example of Spatio-Temporal Weight Matrix and Squared Distance Matrix (One pose feature of *Pull* Action)

for the 1st most informative joint, and by light red for the 2nd most informative joint. Only the motion of 1st and 2nd key joints are marked. The temporal weights for each action are shown as gray images. Each square in that image represents a temporal stage. The whiter the square, the higher the temporal weight. The key stage is highlighted by a red box. We illustrate each key stage using its 4 representative 3D poses. The bottom two actions have two key stages each. The results are quite interesting. In this dataset, *Hand Catch* and *Side Boxing* are not easy to be differentiated. However, ST-NBNN focuses on different motions of these two actions to differentiate them. For the *Hand Catch*, ST-NBNN mainly focuses on the x -direction motion of the key joint (right hand). While for the *Side Boxing*, ST-NBNN cares much about the y and z directions though x is the main direction of this action. A similar situation happens in *Forward Kick*. ST-NBNN does not choose the foot joints to focus on, but put more weight to the right hand since in *Forward Kick* performing, the right hand always move upward (y direction) concomitantly. Besides, as shown in Fig. 3.6 i) and j), the proposed method can also indicate different phases of actions. The two peaks of the temporal weight of *Pick Up* and *Throw* are related to the *Pick Up* and *Throw* two phases respectively.

UTKinect

We use the leave-one-out validation protocol described in [98] to evaluate our proposed method. Based on the description, there are 20 rounds of testing in our experiment. The parameters chosen for spatio-temporal weights learning are the same in each round. The number of local pose l is set to 3, and the number of stage N is 15.

FIGURE 3.5: Influence of Noisy Joints on Accuracy of Berkeley MHAD Dataset

Table. 3.2 shows that our method achieves 2.5% improvement from baseline NBNN, and NBNN+SVM is again inferior to ST-NBNN. With a large number of parameters, NBNN+SVM leads to over-fitting. The comparison with existing methods [51, 54, 56–58, 64] shows that ST-NBNN can achieve competitive performance.

In Fig. 3.4, we provide an example of learnt spatio-temporal weight matrix and squared distance matrices $\mathbf{X}_c$ from *Pull* action in UTKinect dataset. ST-Weight Matrix is on the top-left corner, and squared distance matrices are on the right side. Each matrix is 60 by 15. The related feature of discovered joints are marked by red box. 0 Walk, 1 Sit Down, 2 Stand Up, 3 Pick Up, 4 Carry, 5 Throw, 6 Push, 7 Pull, 8 Wave Hand, 9 Clap Hands. Due to the limitation of space, we only provide the first position feature of each stage and their related weights. Elements a_{ij} of the spatio-temporal weight matrix are determined by $a_{ij} = \mathbf{u}_c^s(i) \times \mathbf{u}_c^t(j), i = 1, \dots, M, j = 1, \dots, N$. The brighter the elements of the matrix, the larger the value of the elements. The red-square-marked region is related to the x, y, z coordinates of right hand (joint 11 and 12). In this dataset, subjects are required to perform *Pull* action by right hand, and the figure shows that ST-NBNN can discover it. From the right side, matrix 7 is the darkest one, which means that the testing sample has the smallest distance from action 7 (*Pull*), and therefore the testing sample belongs to *Pull*. As can be seen, matrix 7 bears many different parts compared with matrices from other categories. The most different parts are closely related to the right hand. Interestingly, the weight matrix selects the commonly most discriminative part of matrices, and it proves that our proposed method is able to discover key factors of actions.

FIGURE 3.6: Key Stages and Key Joints with Related Key Motions from MSR-Action3D

Berkeley MHAD

We follow the experimental protocol described in [108] on this dataset. The sequences performed by the first seven subjects are for training while the ones performed by the rest subjects are for testing. Due to the high sampling rate, most of the data is redundant. We down-sample each sequence by picking one frame of each ten frames. Under this setting, the number of local pose l is 20, and the number of stage N is 20. We compare ST-NBNN with three baseline methods and five state-of-the-art skeleton-based methods. The results are shown in Table. 3.3.

Table. 3.3 shows that although the accuracy of NBNN is only 88%, we can achieve 100% accuracy with the help of weights learning. Besides, comparisons with previous works [47, 60, 64, 112, 113] show that our spatio-temporal weights learning method is able to effectively discover key factors of actions and achieve the best performance.

Considering that the skeletal data captured by motion capture system is more accurate than Kinect depth sensor, we evaluate the tolerance and robustness of ST-NBNN to random noise of skeleton data. We further add noise to joints of skeletal data to see whether our method is robust to joint noises and can meanwhile still

pick out the informative joints. We randomly choose 10%, 20%, 30%, 40% and 50% joints of 35 joints of a pose, and for each selected joint we add noise ranging from -5 to 5 to each dimension of coordinates of joints. The involvement of noise will result in mismatches of related dimension of feature when searching for nearest neighbors. The influence of noisy joint on accuracy is shown in Fig. 3.5. The curve shows that with the increasing of noisy joints percentage, the accuracy of NBNN drops dramatically, while ST-NBNN can maintain high performance on this dataset. Meanwhile, NBNN+SVM is again inferior to ST-NBNN under the setting of noisy joints. ST-NBNN can still pick up the informative joints and keep the performance on high level.

3.5 Conclusion

In this chapter, we extend NBNN to ST-NBNN for skeleton-based action recognition. Compared with NBNN, ST-NBNN considers the spatio-temporal structure of 3D actions and combines the strength of both non-parametric model and also parametric model to achieve better performance. Despite using only a linear classifier, the proposed method works surprisingly well on three benchmark datasets and achieves competitive results compared with state-of-the-arts using sophisticated end-to-end models. Moreover, our proposed method can discover critical spatial joints and temporal stages, which are essential to capture the spatio-temporal patterns of 3D actions, and which is not always achievable via using end-to-end models such as deep learning. Our results demonstrate the potential of using non-parametric method for skeleton-based action recognition.

Chapter 4

Spatio-Temporal Naive Bayes Mutual Information Maximization

4.1 Introduction

The previous success of applying Nearest-Neighbor-based models, e.g., *Naive-Bayes Mutual Information Maximization* (NBMIM) [103], to action detection problem motivates us to develop it to recognize skeleton-based action sequences. Each action instance can be represented by a set of temporal stage descriptors composed of features from spatial joints in 3D poses, and each temporal stage bears positive or negative vote for the action class, based on its mutual information w.r.t the action class. Our proposed *Spatio-temporal NBMIM* takes the spatio-temporal structure of 3D action into consideration and helps further improve action recognition performance. Specifically, the proposed method ST-NBMIM adopts bilinear classifiers [94] to identify both critical temporal stages as well as spatial joints for action classification. Although only using a linear classifier, experiments on four benchmark datasets show that ST-NBMIM can achieve competitive performance compared with state-of-the-art results using sophisticated models such as deep learning. Besides, our ST-NBMIM bears the ability to discover essential spatio-temporal patterns that play key roles of recognizing actions, which is not always achievable by using end-to-end models.

The motivation of applying NBMIM [103] in 3D action recognition is on the basis of the following three observations. (1) Compared with RGB-based image or video

FIGURE 4.1: Overview of ST-NBMIM

analysis problem which always faces millions or billions of pixels, skeletal data only consists of tens of joints. We believe that compared with sophisticated end-to-end model, a simple NN-based model can still well handle such a lightweight problem; (2) Similar to images that are the composition of local visual primitives, actions can be represented as a set of temporal primitives, as the temporal stage-descriptor we defined in Sec. 4.3.1. Therefore, we can generalize NBNN [93] in 3D action problem by applying the *primitive-to-class* distance to recognized actions; However, (3) actions from different action classes may share a great number of similar temporal primitives, which are not helpful to action classification. In NBMIM [103], the involvement of negative primitives in nearest neighbor matching, namely mutual information calculation, can help boost the discriminative ability of temporal primitives. We can involve this idea into the task of 3D action recognition.

4.2 Related Work

The modeling of spatial domain can be divided into two categories, part-based model and sub-pose model. To date, the spatial domain modeling is mainly driven by the fact that an action is usually only characterized by the interactions or combinations of a subset of skeleton joints [62]. In the part-based model, the joints of a skeleton are partitioned into several groups, and in each group the joints are skeletal neighbors of each other. In [61] a part-aware LSTM is proposed to construct the relationship between body parts. Similarly, in HBRNN [60], skeletons

are decomposed into five parts, two arms, two legs, and one torso, and a hierarchical recurrent neural network is built to model the relationship among these parts. In sub-pose model, the focus is mainly on the informative joints or their interactions. In SMIJ [47], the most informative joints are selected simply based on measures such as mean or variance of joint angle trajectories. The sequence of these informative joints is then used as the representation of actions. In Orderlet [50], interactions between joints are modeled by a few comparisons of joints' primitive feature, and in action recognition only a subset of joints is involved. On the temporal domain, temporal pyramid matching [45, 99], dynamic time warping [100], and graphical models [97, 98] are the commonly used methods for temporal modeling. While in [51], sequential pattern mining method is used to model temporal structures of a set of key poses.

Besides spatial modeling or temporal modeling, we also see efforts on spatio-temporal modeling. Wang *et al.* [46] apply data mining techniques to discover co-occurring distinctive spatial body-part structures and temporal pose evolutions. In classification, bag-of-words model based on the mined spatial-part-sets and temporal-part-sets is utilized for action representation. Compared to [46], our proposed method focuses on discriminative and class-related individual spatial joints and temporal stages discovery. For a certain action category, the classification decision is mainly determined by the data from the discovered discriminative joints and stages. In [114], a hierarchical model is proposed to recognize pose-based, composable and concurrent actions and activities. Based on the learned motions poselets and actionlets dictionaries, the hierarchical model can provide spatio-temporal annotations of complex actions. The annotation can tell when the related body parts are activated for which atomic action, but these annotations may not be discriminative in classification. While in ST-NBMIM, key joints and key stages discovered in ST-NBMIM are discriminative for classification. In [64], a LSTM model is extended to spatio-temporal domain to analyze skeletons. In *Spatio-Temporal Naive Bayes Nearest Neighbor* ST-NBNN [52], bilinear classifier is utilized to discover the spatio-temporal structure of 3D action. Another track on spatio-temporal modeling is the CNN-based 3D action recognition model [73, 75, 115]. In these models, a 3D action sequence is first visualized as an image. The pixels of these image samples are directly or indirectly related to the joint coordinates, which means the spatio-temporal information of a 3D sequence is re-organized as the combination of pixels. A further CNN-model is applied to extract features and predict the label

of actions based on the input image samples. The CNN model in these works plays the role of implicitly extracting spatio-temporal information from 3D sequences. Compared with these CNN-based methods, the spatio-temporal modeling of ST-NBMIM is more explicit, and spatio-temporal patterns discovered by ST-NBMIM are physically interpretable.

Our ST-NBMIM is an extension of ST-NBNN [52]. In this work, we introduce the idea of mutual information into ST-NBNN. The involvement of negative samples can help boost the discriminative ability of action representation. The idea of applying mutual information calculation in NBNN framework was first proposed in [103], in which NBNN was re-designed as *Naive-Bayes based Mutual Information Maximization* (NBMIM) to solve action detection problem. One interesting property of NBMIM is that negative samples are involved in nearest neighbor matching to improve the discriminative ability of descriptors. NBMIM is a nearest-neighbor-based (NN-based) method since the calculation of mutual information in NBMIM relies on the nearest neighbor search. Even though NN-based methods are simple and non-parametric, their successes in image classification and action detection prove the effectiveness of these approaches.

4.3 Methodology

In this section, we introduce how ST-NBMIM predicts actions and improve the discriminative ability of descriptors. We first introduce how to represent a 3D sequence of two persons' interactive actions (Sec. 4.3.1). Then NBMIM [103] is used as a basic framework to classify actions (Sec. 4.3.2). Finally, the learning of spatial and temporal weights is introduced to discover key poses and spatial joints for action recognition (Sec. 4.3.3). The overview of our method is illustrated in Fig. 4.1. First, an action video is uniformly divided into a fixed number of stages, and is represented by a set of stage-descriptors (orange query points). After that, distances of stage-descriptors to action class sets (blue, green and red), namely distance descriptor, are calculated by NN search. Further, mutual information are estimated by calculating the difference between positive distance descriptor and the nearest negative distance descriptor to generate mutual information descriptors. Moreover, mutual information descriptors are gathered in temporal order to generate class-related mutual information matrices (marked by class-related dashed

rectangular boxes); At last, Weights on the spatial (left side of the matrix) and the temporal (bottom of the matrix) domain are learnt to discover key factors of actions and predict action labels.

4.3.1 3D Interactive Actions Representation

The description of two persons' interactive action is similar to the one of single person action. We partition each action into N temporal windows of equal length. Each one of the temporal stage is characterized by the 3D poses in its corresponding window. Assuming each 3D pose has J joints for its skeleton, for a temporal stage descriptor $\mathbf{x}$, the 3D pose in its j th frame is denoted as $\mathbf{p}_j \in \mathbb{R}^{3J}$, and the related velocity of that pose is denoted as $\mathbf{v}_j \in \mathbb{R}^{3J}$. Then the pose part $\mathbf{x}_p$ and the velocity part $\mathbf{x}_v$ of $\mathbf{x}$ are defined as below,

$$\begin{aligned}\mathbf{x}_p &= [(\mathbf{p}_1)^\top, \dots, (\mathbf{p}_l)^\top]^\top \\ \mathbf{x}_v &= [(\mathbf{v}_1)^\top, \dots, (\mathbf{v}_l)^\top]^\top\end{aligned}\tag{4.1}$$

As we use both original 3D pose and its velocity to represent 3D actions, a temporal stage descriptor $\mathbf{x}$ of l frames is presented as:

$$\mathbf{x} = [(\mathbf{x}_p)^\top, (\mathbf{x}_v)^\top]^\top\tag{4.2}$$

Inspired by the work in [116], we notice that the relative position between joints from two different persons is much more informative in interaction recognition than the individual position. On the basis of this idea, we use the difference between stages-descriptors of person A and person B involved in an interactive action, namely $\mathbf{x}_a$ and $\mathbf{x}_b$, as the interaction representation. The delta descriptor is defined as,

$$\mathbf{x}_\delta = abs(\mathbf{x}_a - \mathbf{x}_b)\tag{4.3}$$

where $abs(\cdot)$ denotes the element-wise absolute operation for the input descriptor. Considering we do not bear any information about which person is “active” actor and which one is “inactive” actor, the absolute operation can well handle this problem. Finally, a 3D interaction video is described by its N stage-descriptors $V = \{\mathbf{x}_\delta^i\}_{i=1}^N$.

4.3.2 Naive-Bayes Mutual Information Maximization

Given a query action video $V_q = \{\mathbf{x}^i\}_{i=1}^N$, the goal is to find which class $c \in \{1, 2, \dots, C\}$ the video V_q belongs to. *Naive-Bayes Mutual Information Maximization* (NBMIM) evaluates the mutual information I between the query video and a specific class c to identify the action label:

$$c^* = \arg \max_c I(V_q; c) = \arg \max_c \log \frac{p(V_q|c)}{p(V_q)} \quad (4.4)$$

With the Naive-Bayes assumption, Eq. 4.4 can be written as Eq. 4.5

$$\begin{aligned} c^* &= \arg \max_c \log \frac{p(V_q|c)}{p(V_q)} \\ &= \arg \max_c \log \prod_{i=1}^N \frac{p(\mathbf{x}^i|c)}{p(\mathbf{x}^i)} \\ &= \arg \max_c \sum_{i=1}^N \log \frac{p(\mathbf{x}^i|c)}{p(\mathbf{x}^i)} \\ &= \arg \max_c \sum_{i=1}^N I(\mathbf{x}^i; c) \end{aligned} \quad (4.5)$$

where $I(\mathbf{x}^i; c)$ is the mutual information between the i th stage-descriptor and the class c , and with the prior $p(c) = 1/C$ it can be defined as

$$I(\mathbf{x}^i; c) = \log \frac{C}{1 + \frac{p(\mathbf{x}^i|\bar{c})}{p(\mathbf{x}^i|c)}(C-1)} \quad (4.6)$$

where $\bar{c} \neq c, \bar{c} \in \{1, 2, \dots, C\}$.

Based on the analysis of [103], the ratio between $p(\mathbf{x}^i|\bar{c})$ and $p(\mathbf{x}^i|c)$ can be estimated according to the distance between $\mathbf{x}$ and the nearest neighbor of $\mathbf{x}$ in class c and $\bar{c}$,

$$\frac{p(\mathbf{x}^i|\bar{c})}{p(\mathbf{x}^i|c)} \approx \gamma^c \exp^{-\frac{1}{2\delta^2}(\|\mathbf{x}^i - NN_{\bar{c}}(\mathbf{x}^i)\|^2 - \|\mathbf{x}^i - NN_c(\mathbf{x}^i)\|^2)} \quad (4.7)$$

where $\gamma^c = N_c/N_{\bar{c}}$, in which N_c is the number of stage-descriptor from class c and $N_{\bar{c}}$ is the number of stage-descriptor from negative class $\bar{c}$. $NN_c(\mathbf{x})$ and $NN_{\bar{c}}(\mathbf{x})$ indicate the nearest neighbor of the query descriptor $\mathbf{x}$ in positive class c and negative class $\bar{c}$ respectively.

Based on Eq. 4.6 and Eq. 4.7, we can conclude that $I(\mathbf{x}^i; c) \propto \|\mathbf{x}^i - NN_{\bar{c}}(\mathbf{x}^i)\|^2 - \|\mathbf{x}^i - NN_c(\mathbf{x}^i)\|^2$, and we then simplify Eq. 4.5 as

$$\begin{aligned} c^* &= \arg \max_c \sum_{i=1}^N (\|\mathbf{x}^i - NN_{\bar{c}}(\mathbf{x}^i)\|^2 - \|\mathbf{x}^i - NN_c(\mathbf{x}^i)\|^2) \\ &= \arg \max_c \sum_{i=1}^N \tilde{I}(\mathbf{x}^i; c) \end{aligned} \quad (4.8)$$

where $\tilde{I}(\mathbf{x}^i; c)$ is the estimated mutual information between stage-descriptor $\mathbf{x}^i$ and class c .

For a query 3D action $V_q = \{\mathbf{x}^i\}_{i=1}^N$, each of its stage-descriptor will match against C classes separately by finding the best matched temporal stage, i.e., nearest neighbor, in that class. Considering different action classes may share similar temporal stages, these similar stage-descriptors $\mathbf{x}^i$ are not discriminative. For those discriminative ones, the difference between their distances to the positive class and negative classes are large. Eq. 4.8 helps suppress those similar temporal stages among C classes by applying difference operation between negative and positive nearest neighbor distances. The larger the difference between negative and positive distances, the stronger this temporal stage will vote for that class c . As V_q has in total N temporal stages, the final decision is the summation (average) over all of the N suppressed votes towards to C classes, as described in Eq. 4.8.

4.3.3 Spatio-Temporal NBMIM

When performing a specific action, in most of the case only a subset of joints are activated, and for actions from different classes the activated joints are different. Meanwhile, only those joints with high mutual information, as estimated in Eq. 4.8, bears strong classification ability. Based on this phenomenon, in action classification, we can only pick up those spatial joints with high mutual information and ignore the ones that is not informative. On the temporal domain the situation is similar, for a set of temporal stages, not every stage is of equal importance neither. Depending on action class, a certain temporal stage can be more discriminative than others for classification. As shown in Fig. 4.1, the stage-descriptor (shaded orange query square) of stage i bears higher mutual information and is more discriminative than the beginning stage and the ending stage.

To simultaneously identify informative spatial joints and temporal stages, bilinear classifier [94] is used to mine spatio-temporal patterns in the framework of NBMIM.

Mutual Information Matrix

Though in Eq. 4.5 we assume that the stages are independent with each other (Naive Bayes assumption), they actually depend on each other in a certain spatio-temporal structure. Hence, to discover the spatio-temporal structure of 3D actions representation, we first represent a 3D action instance from a set $V = \{\mathbf{x}^i\}_{i=1}^N$ to a matrix. For a given video sample with N stages, its spatio-temporal matrix is defined as

$$\mathbf{X} = [\mathbf{x}^1, \dots, \mathbf{x}^N] \quad (4.9)$$

Stage-descriptors of an action instance are re-organized column by column following the temporal order. Then we define the nearest neighbor matrix of $\mathbf{X}$ in c as $\mathbf{X}_c^{NN} = [NN_c(\mathbf{x}^1), \dots, NN_c(\mathbf{x}^N)]$, and the squared distance matrix to class c is defined as

$$\mathbf{X}_c = (\mathbf{X} - \mathbf{X}_c^{NN}) \odot (\mathbf{X} - \mathbf{X}_c^{NN}) \quad (4.10)$$

where $\odot$ is an element-wise product. $\mathbf{X}_c$ is regarded as a representation of $\mathbf{X}$ for class c , and it is a combination of element-wise stage-to-class distances of the testing sample. Similarly, if we regard class c as the positive class, then the negative squared distance matrix is $\mathbf{X}_{\bar{c}} = (\mathbf{X} - \mathbf{X}_{\bar{c}}^{NN}) \odot (\mathbf{X} - \mathbf{X}_{\bar{c}}^{NN})$. Based on the definition above, the mutual information matrix is defined as

$$\mathbf{X}_c^I = \mathbf{X}_{\bar{c}} - \mathbf{X}_c \quad (4.11)$$

Summation of all the elements in $\mathbf{X}_c^I$ is equivalent to the $\sum_{i=1}^N \tilde{I}(\mathbf{x}^i; c)$ in Eq. 4.8. The mutual information matrix $\mathbf{X}_c^I$, as illustrated in Weights Learning block of Fig. 4.1, is the representation of the action instance V in class c .

Since the summation of elements in $\mathbf{X}_c^I$ determines the final classification decision, as shown in Eq. 4.8, each element bears contribution to the recognition task. However, only the discriminative elements have great impacts to the final decision and therefore this NBMIM framework, Eq. 4.8, should be parameterized to emphasize

those discriminative ones. Let's simply vectorize $\mathbf{X}_c^I$ as $\boldsymbol{\chi}_c^I$, and the NBMIM decision function Eq. 4.8 is then redefined as $c^* = \arg \min_c \mathbf{w}^\top \boldsymbol{\chi}_c^I$, where the weight $\mathbf{w}$ can be learnt by linear SVM. However, considering the mutual information matrix $\mathbf{X}_c^I$ is a large matrix, there will be considerable weights to be determined. Learning weights by linear SVM is not only time-consuming but also has the risk of over-fitting. Therefore in our work, we leverage bilinear classifier [94] to solve the weight learning problem.

With the mutual information matrix $\mathbf{X}_c^I$ of a query matrix $\mathbf{X}$, the classification score is then determined by a bilinear function $f_c(\cdot)$, which is defined as

$$f_c(\mathbf{X}_c^I) = (\mathbf{u}_c^s)^\top \mathbf{X}_c^I \mathbf{u}_c^t \quad (4.12)$$

where $\mathbf{u}_c^s \in \mathbb{R}^M$ and $\mathbf{u}_c^t \in \mathbb{R}^N$ are the spatial and temporal weights of action class c . As a result, the classification becomes

$$c^* = \arg \min_c f_c(\mathbf{X}_c^I) \quad (4.13)$$

As can be seen from Eq. 4.12, the proposed method provides weights for both temporal stages and spatial joints. After a rearrangement, Eq. 4.13 can be represented as,

$$c^* = \arg \min_c \sum_{i=1}^N \mathbf{u}_c^t(i) \|\tilde{I}(\mathbf{x}^i; c)^\top \sqrt{\mathbf{u}_c^s}\|^2 \quad (4.14)$$

where $\sqrt{\cdot}$ is an element-wise square-root of a vector. As we can see, NBMIM is a special case of ST-NBMIM. When $\mathbf{u}_c^s$ and $\mathbf{u}_c^t$ are assigned to $\mathbf{1}$, Eq. 4.14 becomes NBMIM in Eq. 4.8. With the break of Naive-Bayes rule, ST-NBMIM becomes a generalization of NBMIM. We introduce the spatio-temporal structure of 3D action into our framework to break the assumption of stage independence.

Spatial and Temporal Weights Learning

For the learning of $\mathbf{u}_c^s$ and $\mathbf{u}_c^t$, we introduce the objective function that is similar to tensor SVM. Following the learning strategy of [106], we adopt the one-vs.-all

strategy to classify actions. With empirical loss, the objective function of spatio-temporal weight learning is defined as

$$\begin{aligned}
\min_{\mathbf{u}_c^s, \mathbf{u}_c^t} \quad & \frac{1}{2} \|\mathbf{u}_c^s(\mathbf{u}_c^t)^\top\|^2 + \lambda \sum_{i=1}^K \xi_i \\
s.t. \quad & \sum_{i=1}^N \mathbf{u}_c^t(i) = N, \quad \mathbf{u}_c^t \succeq \mathbf{0} \\
& \xi_i \geq \max(0, 1 - c_i f_c(\mathbf{X}_c^{I(i)}))^2 \\
& \xi_i \geq 0, \quad i = 1, \dots, K
\end{aligned} \tag{4.15}$$

in which K is the number of training video samples, and $c_i \in \{-1, 1\}$ is the action label of the corresponding sample. $\mathbf{X}_c^{I(i)}$ is the i th training sample (mutual information matrix) in class c . λ is a parameter for classification error penalty.

Here we set linear constraints to the temporal domain but not to the spatial domain. The reason is that for the spatial domain the number of involved key joints is uncertain. Some spatial joints like the *hip* joint do not have any contribution in recognition in most of the case. Compared with spatial key joints, on temporal domain each stage of an action counts in classification. Experiment results also show that linear constraints on the spatial domain do not bear any contribution to performance, but the temporal constraints do.

The optimization of Eq. 4.15 is regarded as an iterative process. There are two steps in each iterative round, 1) fix $\mathbf{u}_c^t$ and update $\mathbf{u}_c^s$, 2) fix $\mathbf{u}_c^s$ then update $\mathbf{u}_c^t$. $\mathbf{u}_c^t$ is initialized to $\mathbf{1}$.

Fix $\mathbf{u}_c^t$ and Update $\mathbf{u}_c^s$: With $\mathbf{u}_c^t$ fixed, Eq. 4.15 is treated as a l_2 -regularized l_2 loss SVM problem shown below

$$\min_{\mathbf{u}_c^s} \quad \frac{1}{2} \beta_1 \|\mathbf{u}_c^s\|^2 + \lambda \sum_{i=1}^K \max(0, 1 - c_i f_c(\mathbf{X}_c^{I(i)}))^2 \tag{4.16}$$

where $\beta_1 = \|\mathbf{u}_c^t\|^2$.

Fix $\mathbf{u}_c^s$ and Update $\mathbf{u}_c^t$: With updated $\mathbf{u}_c^s$, Eq. 4.15 is regarded as a convex

optimization problem with linear constraints shown below

$$\begin{aligned} \min_{\mathbf{u}_c^t} \quad & \frac{1}{2}\beta_2\|\mathbf{u}_c^t\|^2 + \lambda \sum_{i=1}^K \max(0, 1 - c_i f_c(\mathbf{X}_c^{I(i)}))^2 \\ \text{s.t.} \quad & \sum_{i=1}^N \mathbf{u}_c^t(i) = N, \quad \mathbf{u}_c^t \succeq \mathbf{0} \end{aligned} \quad (4.17)$$

where $\beta_2 = \|\mathbf{u}_c^s\|^2$.

This optimization process is operated iteratively until the objective function Eq. 4.15 converges.

4.4 Experiment

In this section, we evaluate the proposed method on five 3D action datasets and compare its performance to existing methods. Implementation details are provided in Sec. 4.4.1. The description of the five benchmark datasets, the MSR-Action3D dataset [107], the UT-Kinect dataset [98], the Berkeley MHAD dataset [108], the SBU-Interaction dataset [116], and the NTU RGB+D dataset [61], is provided in Sec. 4.4.2. Among the five datasets, SBU-Interaction and NTU RGB+D contain interactive actions of two people. Comparison results on these datasets are provided and discussed in Sec. 4.4.3. The experiment results show that the introduction of mutual information helps to improve the action recognition accuracy over ST-NBNN. The discriminative matching helps boosting the discriminative ability of action representation. Although ST-NBMIM is simple, it is able to achieve comparable performance with state-of-the-arts and also effectively discover the key factors of an action class.

4.4.1 Implementations

3D Action Representation. The one-vs.-all strategy is utilized in this method. To ensure the responses of linear functions $f_c(\cdot)$ are comparable with each other, each sample $\mathbf{X}_c^{I(i)}$ is mean-centralized by $\mu^i = \sum_{c=1}^C \text{sum}(\mathbf{X}_c^{I(i)}) / (C \times M \times N)$, where $\text{sum}(\cdot)$ sums up entries of the input matrix.

In Sec. 4.4.2, the setting of N is indicated. Due to the variation of action sequences' duration, stages defined in Sec. 4.3.1 may have overlaps when a sequence is too short.

To ensure that the representation introduced in Sec. 4.3.1 is location-invariant, for actions of single person each joint of the skeleton is centralized by subtracting coordinates of the hip joint. For interactive actions of two persons, two skeletal poses in each frame are centralized by subtracting the average coordinates of the two hip joints.

Nearest Neighbor Search. In our experiment, KD-tree implementation [109] and FLANN library [117] are used to boost the nearest neighbor searching process.

Spatio-Temporal Weight Learning. The training matrices $\mathbf{X}_c^I$ are generated by a leave-one-video-out strategy, which means all the stage-descriptors of a query training video are excluded from the nearest neighbor search. In our optimization, $\mathbf{u}_c^s$ and $\mathbf{u}_c^t$ are learned iteratively. To solve the SVM problem of Eq. 4.16, we use a SVM toolbox [110] implemented by Chang *et al.*, and to update $\mathbf{u}_c^t$, a convex optimization toolbox [111] is used.

4.4.2 Datasets

MSR-Action3D

There are 557 skeletal action sequences included in this dataset, and 20 human actions are involved. The actions recorded are common indoor daily actions. Each action is performed by 10 subjects twice or three times. The evaluation protocol we use is described in [107]. In this protocol, the 20 actions are grouped into three subsets AS1, AS2, and AS3, where each subset consists of eight actions. In this dataset, the number of poses in each local window is 10, and the number of stages N is set to 15.

UT-Kinect

This dataset contains 10 action classes performed by 10 subjects. Each action is performed by each subject twice. We use the leave-one-out validation protocol described in [98] to evaluate our proposed method. Based on the description, there

Method	AS1	AS2	AS3	Ave.
Lie Group [54]	95.4	83.9	98.2	92.5
SCK+DCK [57]	–	–	–	94.0
HBRNN [60]	93.3	94.6	95.5	94.5
ST-LSTM [64]	–	–	–	94.8
Graph-Based [56]	93.6	95.5	95.1	94.8
ST-NBNN	91.5	95.6	97.3	94.8
ST-NBMIM	92.5	95.6	98.2	95.3

TABLE 4.1: Comparison with State-of-the-arts on MSR-Action3D (%)

are 20 rounds of testing in our experiment. The parameters chosen for spatio-temporal weight learning are the same in each round. The number of local poses l is set to 3, and the number of stages N is 15.

Method	Accuracy
Key-Motif [51]	93.5
Simplices [58]	96.5
ST-LSTM [64]	97.0
Lie Group [54]	97.1
Graph-Based [56]	97.4
SCK+DCK [57]	98.2
ST-NBNN	98.0
ST-NBMIM	98.0

TABLE 4.2: Comparison with State-of-the-arts on UT-Kinect (%)

Method	Accuracy
SMIJ [47]	95.4
Meta-cognitive [112]	97.6
Kapsouras <i>et al.</i> [113]	98.2
HBRNN [60]	100.0
ST-LSTM [64]	100.0
ST-NBNN	100.0
ST-NBMIM	100.0

TABLE 4.3: Comparison with State-of-the-arts on MHAD (%)

Berkeley MHAD

The actions in this dataset are captured by a motion capture system. 11 action classes are included, and each action is performed by 12 subjects. We follow the experimental protocol described in [108] on this dataset. The sequences performed by the first seven subjects are used for training while the ones performed by the rest subjects are used for testing. Due to the high sampling rate, around 30 frames per second, most of the data is redundant. We down-sample each sequence by selecting one frame for every ten frames. Under this setting, the number of local poses l is 20, and the number of stages N is 20.

FIGURE 4.2: Parameter Sensitivity Analysis on SBU Interaction Dataset

SBU-Interaction

This dataset contains eight classes of two-person interactive actions. 282 skeleton sequences are captured by Kinect depth sensor. For each skeleton, there are 15 joints in total. We follow the protocol proposed in [116] to evaluate our method. There is a five-fold cross validation. The evaluation is based on the average accuracy on these five folds. The number of stages N is 17, and there are three poses in each stage. Considering that there are pair actions in this dataset, when performing the nearest neighbor search, only the related stages are involved.

Method	Accuracy
HBRNN [60]	80.4
CHARM [118]	83.9
Deep LSTM reported in [62]	86.0
Co-occurrence [62]	90.4
ST-LSTM [64]	93.3
ST-NBNN	89.3
ST-NBMIM	93.3

TABLE 4.4: Comparison with State-of-the-arts on SBU (%)

NTU RGB+D

NTU-RGBD dataset is currently the most challenging dataset in 3D action recognition. It is collected with Kinect V2 depth camera. There are around 56 thousand sequences in total. 60 different action classes are performed by 40 subjects aged from 10 to 35. 25 joints are included in each skeletal pose. We follow the protocol introduced in [61] to conduct the experiment. In the nearest neighbor search, we only search for the related stage since there are 10 pair actions in this dataset. This dataset has two standard evaluation settings, the cross-subject (CS) evaluation and the cross-view (CV) evaluation. In cross-subject setting, half of the subjects are used for training and the remaining are for testing. In cross-view setting, two of the three views are used for training and the left one is for testing. The number of local poses l is set to 5, and the number of stages N is 20.

Method	CS	CV
Skeleton Quads [119]	38.6	41.4
Lie Group [54]	50.1	52.8
HBRNN [60]	59.1	64.0
Part-Aware LSTM [61]	62.9	70.3
ST-LSTM [64]	69.2	77.7
Two Streams [67]	71.3	79.5
GCA-LSTM [65]	74.4	82.8
New Representation [74]	79.6	84.8
Enhanced Vis. [75]	80.0	87.2
ST-NBNN	56.6	56.0
ST-NBMIM	64.5	64.1
ST-NBMIM + CNN Feature	80.0	84.2

TABLE 4.5: Comparison with State-of-the-arts on NTU RGB+D (%)

4.4.3 Results and Analysis

Comparison with Baselines

We compare the proposed method with spatio-temporal naive-bayes nearest neighbor (ST-NBNN) and four related baselines on five benchmark datasets, MSR-Action3D (M.), UT-Kinect (U.), SBU-Interaction (S.), Berkeley MHAD (B.), and NTU RGBD (N.CS for cross-subject setting, N.CV for cross-view setting). The

baselines are (1) NBNN with N stages (NBNN-N); (2) NBNN with weight learning by linear SVM (NBNN+SVM); (3) Spatio-temporal NBNN (ST-NBNN); (4) NBMIM with N stages (NBMIM-N); and (5) NBMIM with weight learning by linear SVM (NBMIM+SVM). The results are shown in Table. 4.6

Method	M.	U.	S.	B.	N.CS	N.CV
NBNN-N	91.7	95.5	88.1	88.0	59.9	59.6
NBNN+SVM	92.4	94.0	83.0	100.0	44.4	43.0
ST-NBNN	94.8	98.0	89.3	100.0	56.6	56.0
NBMIM-N	93.3	96.0	89.2	88.0	60.8	60.3
NBMIM+SVM	92.7	95.0	89.3	99.3	55.1	54.1
ST-NBMIM	95.3	98.0	93.3	100.0	64.5	64.1

TABLE 4.6: Comparison with Baselines on Five Datasets (%)

As we can see from Table. 4.6, ST-NBMIM maintains or improves the performance of ST-NBNN. On SBU interaction dataset, ST-NBMIM improves by 4% over ST-NBNN. We can also see the accuracy improvement or maintenance from NBNN to NBMIM on all five benchmark datasets. As we discussed in Sec. 4.3.3, if we only use linear SVM as the weight learning method, there will be a large number of parameters to be determined, and this strategy may cause over-fitting. From Table. 4.6, in most of the cases, there are drops from NBMIM to NBMIM+SVM, which indicates the over-fitting caused by SVM, and we can see that the performance of ST-NBMIM is better than NBMIM+SVM.

The results from NTU-RGBD dataset show that the weight learning does not work for NBNN in this dataset. There are 3.3% and 3.6% drops from NBNN to ST-NBNN under cross-subject setting and cross-view setting respectively. However for NBMIM, the improvement is significant (3.7% improvement under cross-subject setting, 3.8% improvement under cross-view setting). The reason is that there are many actions that are very similar to each other in terms of the skeleton motion in NTU-RGBD dataset, e.g., *drinking water* and *eating snack*. The involvement of mutual information can help suppress elements that are similar among stage-descriptors and emphasize the elements that are discriminative. A further weight learning by our proposed method can help pick out those discriminative elements and improve the performance.

FIGURE 4.3: Comparison of Confusion Matrix between ST-NBNN and ST-NBMIM. 1. Approaching, 2. Departing, 3. Kicking, 4. Pushing, 5. Shaking Hands, 6. Hugging, 7. Exchanging, 8. Punching

The motivation of extending ST-NBNN to ST-NBMIM via involving mutual information is to boost the discriminative ability of action representation. Fig. 4.3 shows the comparison of confusion matrix between ST-NBNN and ST-NBMIM on SBU Interaction dataset. As can be seen from the figure, there are great improvements of classification accuracy on action “Pushing” and “Shaking Hand”. The actions “Shaking Hand” and “Exchanging” are very similar with each other. After involving mutual information into ST-NBNN, the confusion between these two action becomes less than before, and the overall performance of ST-NBMIM is better than ST-NBNN.

Combination with Convolutional Neural Network

In this section, we combine the proposed method ST-NBMIM with CNN model, ResNet18 [120]. Since our method mainly focuses on skeleton-based action recognition, we can not apply CNN model on the data we use directly. Therefore we first transform the pose data of each video sample to an image by using the visualization method proposed in [121]. Then a ResNet[120] with 18 layers is used to extract CNN features for representing each video sample. In order to learn the spatio-temporal weights, we randomly pick 20% of the training samples as the validation set, and 80% of the training samples for fine-tuning ResNet18. Then the fine-tuned ResNet is used to extract CNN feature for both the validation set and

FIGURE 4.4: An Example of Spatio-Temporal Weight Matrix and Comparison between Squared Distance Matrix from NBNN (first row) and Mutual Information Matrix from NBMIM (second row)

testing set. We use the proposed ST-NBMIM as the classifier to predict the label of each sample based on the extracted CNN feature. We evaluate the combination version on NTU-RGBD dataset [61], and the results are shown in Table. 4.7. As can be seen from the table, CNN feature with ST-NBMIM can perform better than ResNet18, which means that ST-NBMIM can benefit CNN model. Besides, as can be seen from the Table. 4.5, the performance of ST-NBMIM with CNN feature is comparable with the state-of-the-arts.

Method	CS	CV
NBMIM-N	60.8	60.3
ST-NBMIM	64.5	64.1
Res CNN Feature + FC (ResNet18)	78.9	83.2
Res CNN Feature + ST-NBMIM	80.0	84.2

TABLE 4.7: Combination of CNN feature and ST-NBMIM (%)

Comparison with the State-of-the-arts

In this section we compare the proposed method ST-NBMIM with the existing

methods on five benchmark datasets. The results are shown in Table. 4.1, Table. 4.2, Table. 4.3, Table. 4.4, and Table. 4.5. We can see that ST-NBMIM achieves the best accuracy on the MSR-Action3D dataset, Berkeley MHAD dataset and SBU-Interaction dataset. On the UT-Kinect dataset, the result is comparable with the state-of-the-arts.

On the NTU RGBD dataset, the proposed method ST-NBMIM can still perform better than the deep-learning-based method HBRNN [60] and Part-Aware LSTM [61]. However, we can also see that ST-NBMIM is not as good as other deep-learning-based model like [64, 67, 74]. The reason why ST-NBMIM is not better than these models is that, compared with them, our proposed method only uses raw features directly from skeletal data, and only a linear method is utilized as the classifier, which does not have such large model capacity as the deep learning models. We also try to combine CNN feature from ResNet18 [120] with our ST-NBMIM classifier, and we witness great improvement from ST-NBMIM with raw data feature. From Table. 4.5 we can see that our combination version is comparable with the state-of-the-arts.

FIGURE 4.5: Influence of Noisy Joints on Accuracy of Berkeley MHAD Dataset

Parameter Sensitivity Analysis

There are two main parameters in ST-NBMIM, the number of temporal stages N and the number of poses in each local stage l . The evaluation of parameter sensitivity is conducted on the SBU interaction dataset in this section. We change l from 1 to 10 and change N from 1 to 21. In Fig. 4.2, the x-axis indicates the chosen number of local poses. The sub-title indicates the chosen number of temporal stages. As Fig. 4.2 shows, ST-NBMIM needs a sufficient number of stages to

learn the spatio-temporal weights and obtain good performance on action recognition. When the number of stages is larger than three, ST-NBMIM can still help improve the performance with only one pose in each stage. However, when the number of stages is larger than nine, further increasing N and l will not improve the performance explicitly. Fig. 4.2 also shows that when the stage number N is sufficient, ST-NBMIM achieves better accuracy than ST-NBNN, which indicates the effectiveness and robustness of the proposed method.

Robustness to Noise

In this section, we evaluate the tolerance and robustness of ST-NBMIM to random noise of skeleton data on Berkeley MHAD dataset. We randomly choose 10%, 20%, 30%, 40% and 50% of the 35 joints. For the randomly selected joints, we add noises ranging from -5 to 5 to each dimension of joints coordinates. This setting will result in mismatches of nearest neighbor search. The influence of noisy joint on accuracy is shown in Fig. 4.5. As we can see, as the percentage of noisy joints increases, the performance of NBMIM drops dramatically. Compared with NBMIM, ST-NBMIM can still pick out the informative elements and maintain the accuracy at high level. The average accuracy of ST-NBMIM in these six situations is 98.24%. We can also see that in most of the cases, ST-NBNN can not perform better than ST-NBMIM. The average accuracy of ST-NBNN in these six situations is 97.58%, which is not as good as ST-NBMIM.

Time Cost Analysis

In this section, we also experimented on SBU Interaction dataset to test the time cost of each step of the proposed method. In the training phase, the nearest neighbor search is conducted first to obtain the mutual information matrices for each training sample. Then the spatio-temporal weights learning part is conducted. In the testing phase, each stage-descriptor of a testing sample will be a query to search its nearest neighbors. After that class-related mutual information matrices will be generated. The classification part is then conducted to apply learned spatio-temporal weights on mutual information matrices. The test results are shown in Table. 4.8 and Table. 4.9 respectively. There are five data splits in SBU Interaction dataset. We record the time cost of these five data splits on training phase and testing phase. The recorded time in Training part is from just one round (including one spatial weight update and one temporal weight update). In all the datasets

Split	# Sample	NN	Training
1	227	19.8 s	9.1 ms
2	230	19.7 s	13.7 ms
3	226	18.8 s	12.2 ms
4	228	20.3 s	10.8 ms
5	217	18.5 s	16.9 ms
Ave.	225.6	19.4 s	12.5 ms

TABLE 4.8: Time Cost in Training Phase

Split	# Sample	NN	Testing
1	55	1.3 s	0.9 ms
2	52	1.2 s	0.8 ms
3	56	1.4 s	0.9 ms
4	54	1.3 s	1.6 ms
5	65	1.4 s	1.0 ms
Ave.	56.4	1.3 s	1.0 ms

TABLE 4.9: Time Cost in Testing Phase

we test, two rounds are already enough for spatio-temporal weight learning. This experiment is conducted on a Intel Xeon E5-2609 CPU with 2.50GHz clock frequency. As we can see, the proposed method does not take much time on training and testing. However, the most time-consuming part is the nearest neighbor search. As the number of stage-descriptors increases in the search area, the searching time will increase relatively. Therefore, our method is not suitable for real-time application. However, as we witness works like [105] focusing on boosting nearest neighbor search speed, we believe that the situation caused by low searching speed will be alleviated.

FIGURE 4.6: Training Convergence Curve of the ST-NBMIM

FIGURE 4.7: Key Stages and Key Joints with Related Key Motions from MSR-Action3D

Convergence Analysis

In this section, we record the objective function value of each iteration in training on SBU Interaction dataset. The result is shown in Fig. 4.6. The convergence curve shown in Fig. 4.6 is the average curve of eight binary classifiers of ST-NBMIM. In most of the case, only two or three iterations are enough for the training process to converge.

Visualization

In this section, we visualize the mutual information matrix and learned spatio-temporal weight matrix in Fig. 4.4 to help better understand the proposed method. Besides, the discovered key joints and key temporal stages are shown in Fig. 4.7. In Fig. 4.4, we provide an example of the learned spatio-temporal weight matrix and the estimated mutual information matrices $\mathbf{X}_c^I$ from *Pushing* action in SBU Interaction dataset. ST-Weight Matrix is on the top-left corner, and squared distance matrices as well as mutual information matrices are on the right side. Each matrix is 45 by 13. The related feature of discovered joints are marked by red box. Due to space limitation, we only provide the first position feature of each

stage and their related weights. Elements a_{ij} of the spatio-temporal weight matrix are determined by $a_{ij} = \mathbf{u}_c^s(i) \times \mathbf{u}_c^t(j)$, $i = 1, \dots, M, j = 1, \dots, N$. The brighter the elements of the matrix, the larger the value of the elements. We can see from Fig. 4.4 that the *Pushing matrix* is the darkest one in NBNN and the brightest one in NBMIM. For NBNN described in ST-NBNN [52] and NBMIM described in Eq. 4.8, the classification is based on the summation of elements of each class-related squared-distance matrices of NBNN and mutual information matrices of NBMIM respectively. As we discussed in Sec. 4.3.3, each element of the matrices in NBNN and NBMIM bears the same contribution to the classification, and the motivation of the proposed weight learning method is to pick out those discriminative elements shared in each class-related matrix. Let's take the *Pushing* action for example. The ideal situation is that the elements that picked out by our weights learning method bear very high value in *matrix 5*, shown in Fig. 4.4, and very low value in other class-related matrices. In this case, the learned weight can help the summation of positive matrix, *matrix 5 (Pushing)* in this case, be the maximum one, and therefore help the classifier predict the true label. If the elements selected by our proposed weight learning method bear high values in both positive matrix and negative matrices, the selected ones are not discriminative enough for the classification task. From Fig. 4.4 we can see that the discriminative elements as we described above in the mutual information matrices of NBMIM have more sparse pattern than those in NBNN, which is easier for the spatio-temporal weight learning method to discover, and therefore the mutual information representation is more suitable than squared distance representation in NBNN for the weights learning method. Meanwhile, compared with the ones in NBNN, the “negative” matrices in NBMIM have similar “sparse pattern” making it easy for the proposed method to discover discriminative elements and help distinguishing positive matrices from the negative ones. We can also see that the learned weights shown on the upper-left corner of Fig. 4.4 is able to correctly discover those discriminative elements. The red-square-marked region of the spatio-temporal weight matrix is related to the x, y, z coordinates of right and left hands (joint 6 and 9), which are the most active joints in “Pushing” action.

Fig. 4.7 shows the key spatial joints and temporal stages discovered by the proposed method. Colored joints are with weights larger than average weights. The most informative joints are marked by bright color, and the second most informative joints are marked by light color. Other key joints are marked by dark blue. The

global key motions are indicated by different colors. For example, the key motion directing in the x direction is colored by bright red for the 1st most informative joint, and by light red for the 2nd most informative joint. Only the motion of 1st and 2nd key joints are marked. The temporal weight for each action is shown as a gray image. Each square in this image represents a temporal stage. The whiter the square, the higher the temporal weight. The key stage is highlighted by a red box. We illustrate each key stage using its 4 representative 3D poses. The bottom two actions have two key stages each. For the “Forward Kick” action, ST-NBMIM selects the right hand as the most discriminative joint and right toe as the second most discriminative joint. When performing “Forward Kick”, the dominant direction of joints’ motion is z , and we can see that the proposed method discovers the direction. For the “Tennis Swing” action, the motion direction that is discovered by the proposed method is y - z , since when performing this action, the right hand mainly moves “down” (y) and “forward” (z). In the MSR-Action 3D dataset, “Side Boxing” and “Hand Catch” are similar to each other. ST-NBMIM selects the x direction of the second most discriminative joints for both of these two actions. In order to differentiate these two actions, the difference is on the 1st most informative joint (right hand). ST-NBMIM focuses on the y - z motion direction for the “Side Boxing” action, but on the x motion direction for the “Hand Catch” action. Interestingly, as shown in Fig. 4.7 i) and j), the proposed method can also indicate different phases of actions. The two peaks of the temporal weight of “Hand Clap” are related to the stages when two hands are close to each other and when the two hands are far apart from each other. For “Pick Up and Throw”, the two peaks of the temporal weight are related to the “Pick Up” and “Throw” two phases respectively.

4.5 Conclusion

In this work, we combine the idea of spatio-temporal pattern discovery with the non-parametric model NBMIM to recognize 3D action. The spatio-temporal pattern mining in the proposed method ST-NBMIM is capable of discovering critical spatial joints and temporal stages of action instances simultaneously, which help not only increase the action recognition performance, but also physically explain each action recognized. We introduce the idea of mutual information into our

framework. The involvement of negative stage-descriptors in mutual information calculation helps to improve the discriminative ability of action representation. Experiments show that ST-NBMIM can achieve better performance than baseline like ST-NBNN. Despite using only a linear classifier, the proposed method works surprisingly well on four benchmark datasets and beats some sophisticated end-to-end models on large scale dataset NTU RGB+D. Our results demonstrate the efficiency of the proposed spatio-temporal pattern discovery method for skeleton-based action recognition.

Chapter 5

Deformable Pose Traversal Convolution

5.1 Introduction

With the success of pose estimation methods [6–8] using depth sensor, 3D action and hand gesture recognition have drawn considerable attention. To recognize 3D action and gestures, each 3D pose is often characterized by its joints with 3D locations.

However, previous work [52, 65, 66] show that not every spatial joint is of equal importance to the recognition of actions, and human body movements exhibit spatial patterns among pose joints [50]. It is thus of great importance to identify those motion patterns and avoid the non-informative joints, via identifying the key combinations of joints that matter for the recognition. For instance, to recognize hand gesture “Okay”, the “approaching of index fingertip and thumb tip” as well as the “stretching of other three fingers apart from the palm” should be observed. The coordination of these five key fingertips is important to the recognition of gesture “Okay”.

To identify the joint patterns for 3D action and gesture recognition, deep-learning-based methods have been popularly utilized recently [62, 65, 66, 68, 69, 95]. For example, [65, 66] apply the attention mechanism to 3D action recognition by assigning weights to joints and use the key joints to represent 3D poses. However, each

FIGURE 5.1: Illustration of the Pose Traversal Representation, the Pose Traversal Convolution and the Deformable Pose Traversal Convolution

joint is considered individually in these works, and the discriminative cues in the spatial configuration of pose are not fully utilized. Part-based models [60, 61] apply recurrent neural network (RNN) to explore the relationship among body parts. Liu *et al.* [64] and Wang *et al.* [67] feed one 3D pose joint into a RNN at each step to model the spatial dependency of joints, and show such spatial dependency modeling can improve the recognition performance. However, considering in each step the current available spatial context is only the hidden state from the previous steps, the spatial dependency of joints is not fully utilized by the sequential way of these two models.

In this work, we propose to identify joint patterns via traditional convolution. In image recognition, Convolution Neural Network (CNN) operates weighted summation in each local window of input image, which involves the local spatial relation of pixels. Meanwhile, each convolution is independent, and CNN is suitable to model the spatial dependency of pixels in parallel rather than sequentially. Similar to the appearance similarity measure among neighboring pixels via convolution, given the coordinates of joints, we can use convolution to obtain the spatial configuration among them. We follow the tree traversal rule in [64] to preserve the spatial neighboring relation of joints, as shown in Fig. 5.1a. The traversal starts from the red torso joint, and follows the guidance of orange arrows. Then we apply one-dimensional convolution to traverse the pose to extract spatial representation.

We use Fig. 5.1b to illustrate this Pose Traversal Convolution. The red dots connected with the orange line indicate a 3×1 pose convolution kernel. The green dot indicates the convolution anchor. The kernel anchored at right elbow operates convolution on the right arm, and it continues to slide along the pose joints to obtain pose information following the traversal guides in Fig. 5.1a. Meanwhile, the

Pose Traversal Convolution can be optimized to search appropriate spatial context for each individual joint, which we call it Deformable Pose Traversal Convolution. With Deformable Pose Traversal Convolution, we no longer fix the regular structure of the kernel for convolution, and the convolution can easily involve key joints that are not neighbors of each other, thus can capture essential dependency among joints. Also, each joint will identify its own spatial context by identifying a suitable kernel.

Our Deformable Pose Traversal Convolution is inspired by Deformable Convolution Neural Network [122] that introduces the attention mechanism into convolution. As illustrated in Fig. 5.1c, the kernel in Pose Traversal Convolution is deformed to involve two hands and the right shoulder in a convolution. The convolution kernel anchored on the right elbow is deformed so that the left and right hands as well as the right shoulder are also included in one convolution operation. We consider, due to the complexity of actions and gestures, the key combination of joints varies during the action performing. Therefore, the deformation offsets for the convolution kernel are predicted by a one-layer ConvLSTM [123] based on the previously observed pose data. The final extracted pose feature will be further fed into a LSTM network [124]. The whole network utilizes Deformable Pose Traversal Convolution to learn the spatial dependency among joints and use LSTM to model the long-term evolution of the pose sequence. This network is trained end-to-end. The performance on benchmark action and gesture datasets, as well as the comparison with the state of the arts, demonstrates the effectiveness of the proposed Deformable Pose Traversal Convolution for 3D action and gesture recognition.

Our contributions can be summarized as follows:

- We introduce a one-dimensional convolution neural network, Deformable Pose Traversal Convolution, to represent 3D pose. It can extract pose feature by identifying key combinations of joints, which is interpretable for action and gesture understanding.
- We apply the ConvLSTM [123] to learn the deformation offsets of convolution kernel. It models the temporal dynamics of key combinations of joints.

5.2 Related Work

3D Action and Gesture Recognition

3D action and gesture recognition task attracts a lot of attention in these years. The recently proposed methods can be categorized into traditional model [45, 47, 50, 52, 53, 55, 56, 59, 125–129] and deep-learning-based model [60, 62–69, 73, 74, 76]. Due to the tremendous amount of these works, we limit the review to spatial pose modeling in deep learning.

Part-based models consider 3D actions as the interaction between body parts. In HBRNN [60], a 3D pose is decomposed into five parts, and multiples bi-RNNs are stacked hierarchically to model the relationship between these parts. In [64, 65, 67], the pose graph is flattened by the tree traversal rule, and the joints are fed into LSTM sequentially to model the spatial dependency among them. In 2D CNN-based methods [73–76], a pose sequence is first visualized as images. Each pixel of the image is related to a joint. Then a two-dimensional CNN is applied on the generated images to extract sequence-level CNN feature, by which the spatio-temporal dependency of joints is implicitly learned. Compared with the 2D CNN methods, we apply one-dimensional convolution on pose traversal data to extract frame-level pose feature, which is more flexible especially for tasks requiring frame-level feature.

Attention Mechanism in 3D Action and Gesture

In 3D action and gesture recognition, not every spatial joint is of equal importance to the recognition task. Hence it is essentially important to identify the key joints that matter for recognition. In STA-LSTM [64], two sub-LSTMs are trained to predict spatial and temporal weights for the discovery of key joints and frames. In ST-NBNN [52], support tensor machine is introduced to assign spatial and temporal weights on nearest-neighbor distance matrices for action classification. In Orderlet [50], actions are described by a few key comparisons of joints' primitive features, and only a subset of joints is involved. The key combinations of joints are fixed during testing. In contrast, the proposed method allows the optimal spatial context of each joint changes during the action performing since we use a RNN structure to model the temporal dynamics of key joint combinations.

5.3 Methodology

In this section, we introduce how the proposed Deformable Pose Traversal Concolution processes the pose data in each frame and recognizes the 3D actions and gestures. The feature maps and the convolution kernels are two-dimensional in our model. One dimension for the spatial domain and the other for the channel domain. The deformable pose convolution operates on the spatial domain, and the deformation of kernel remains the same across the channel dimension. For notation clarity, we describe the module on spatial dimension, omitting the index of the channel dimension.

Each 3D action/gesture is represented as a sequence of 3D poses. Assuming that each pose consists of J joints, a single pose can be represented as $\mathbf{X} \in \mathbb{R}^{J \times C}$, where C is the number of channel. A 3D pose has six channels if the coordinates and velocities of the joints are both involved, as shown in Fig. 5.3. We define $\mathbf{x} \in \mathbb{R}^J$ as the one-channel version of $\mathbf{X}$. Then a 3D action/gesture sequence can be described by a set of poses, $V = \{\mathbf{x}^t\}_{t=1}^T$, where T is the length of a sequence sample. Given a 3D pose sequence, our goal is to predict the corresponding label $k \in \{1, 2, \dots, K\}$, where K is the number of categories. The problem can be formulated following the maximum a posteriori (MAP) rule,

$$k^* = \arg \max_k p(k|V) \quad (5.1)$$

5.3.1 Pose Traversal Convolution

To fully utilize the spatial kinematic dependency of joints in a single pose, we follow the tree traversal rule in [64] to represent the pose data in our method, as illustrated in Fig. 5.1a. By this manner, the length of the pose data is extended from J to J_e , and the arrangement of joints becomes a loop. We name it the pose traversal data.

In our method, we apply the modified Temporal Convolution Network [130] (TCN) on the pose traversal data to perform spatial pose convolution, which is named as Pose Traversal Convolution. Each layer of the TCN is composed of one-dimensional convolution, one-dimensional pooling, and channel-wise normalization operations.

FIGURE 5.2: Structures of the Pose Traversal Convolution Network and the Deformable Pose Traversal Convolution Network

After each convolution operation, there is a Rectified Linear Unit (ReLU) included to involve nonlinearity. The TCN is an encoder-decoder structure designed to solve temporal segmentation task. In this work, we only use the encoder part of TCN and modify it to a two-layer one-dimensional convolution neural network. The structure of the neural network with Pose Traversal Convolution is shown in Fig. 5.2a. Each block in the structure represents an operation of the network. The “in” and “out” of each block indicate the number of input channel and output channel, respectively. The “kernel” indicates the kernel size, and the “stride” indicates the stride of convolution operation. The size of the LSTM hidden neuron is indicated by “hidden”. The two-layer convolution structure extracts pose feature from each single frame, and the pose feature is then further fed into the main LSTM network to model the long-term temporal dynamics of the input pose sequence.

There are two steps for a regular one-dimensional convolution: 1) sampling using a regular set $\mathbf{G}$ on the input pose data; 2) summation of sampled joints’ value weighted by $\mathbf{w}$. The sampling set $\mathbf{G}$ defines the receptive field size and dilation. For a $N \times 1$ regular convolution kernel with dilation 1, where $N = 2 * M + 1$ and $M \in \mathbb{N}^+$, the regular sampling set can be defined as:

$$\mathbf{G} = \{-M, \dots, -1, 0, 1, \dots, M\} \quad (5.2)$$

For the location i_0 of the output feature map $\mathbf{y}$, we have

FIGURE 5.3: Illustration of the Deformable Pose Traversal Convolution

$$\mathbf{y}(i_0) = \sum_{i_n \in \mathbf{G}} \mathbf{w}(i_n) \cdot \mathbf{x}(i_0 + i_n) \quad (5.3)$$

where i_n enumerates the elements in $\mathbf{G}$.

5.3.2 Deformable Pose Traversal Convolution

Regular pose convolution focuses on the spatial neighboring joints. This traditional operation may involve non-informative joints and cannot well construct the relationship of key joints far away from each other. We consider that the optimal spatial context of a joint does not always come from its neighbors. It is of great importance to break the limitation of regular convolution kernel to learn the optimal spatial context. Inspired by the Deformable Convolution Network [122], which is designed for image recognition, we propose a deformable version of the Pose Traversal Convolution and apply it on pose data to discover better combinations of joints. More specifically, we replace the first layer of the Pose Traversal Convolution network with a one-dimensional deformable convolution, and a ConvLSTM [123] is involved to learn offsets δ for the kernel of each convolution anchor, as illustrated in Fig. 5.3. The gray rectangles represent the convolution kernels and

the dark gray rectangles inside of them indicate the convolution anchor. The pose traversal data is fed into the Offset Learning module, which consists of a ConvLSTM and an Offset Convolution. The “in” and “out” in the figure indicate the numbers of the input and output channels, respectively. The orange rectangles are the learned offsets to deform the convolution kernel at each anchor. The Deformable Pose Traversal Convolution is operated at the bottom of the figure based on the learned offsets. The offsets are the adjustments of convolution sampling locations on feature map.

By using a $N \times 1$ deformable convolution kernel with new irregular sampling set $\tilde{\mathbf{G}} = \{(i_n, \delta_n)\}_{n=1}^N$, the output feature $\mathbf{y}$ at the location i_0 is defined as,

$$\mathbf{y}(i_0) = \sum_{(i_n, \delta_n) \in \tilde{\mathbf{G}}} \mathbf{w}(i_n) \cdot \mathbf{x}(i_0 + i_n + \delta_n) \quad (5.4)$$

Now, the sampling is on the irregular locations $i_0 + i_n + \delta_n$. The Pose Traversal Convolution is a special case of the deformable version. When $\{\delta_n\}_{n=1}^N$ are all set to zeros, the set $\tilde{\mathbf{G}}$ becomes $\mathbf{G}$. Considering that the learned δ_n could be none-integer, bi-linear interpolation is used to sample the input feature map,

$$\mathbf{x}(i) = \alpha \cdot \mathbf{x}(\lfloor i \rfloor) + (1 - \alpha) \cdot \mathbf{x}(\lceil i \rceil) \quad (5.5)$$

where $i = i_0 + i_n + \delta_n$ denotes the fractional location (sub-joint), and $\alpha = \lceil i \rceil - i$.

Offset Learning

We involve a sub-path network to learn the offsets δ . Considering that to decide which joints to pay attention to should be inferred from the previously observed action/gesture, the sub-path network is constructed based on a RNN structure, ConvLSTM [123]. With the involved RNN model, the sub-path is able to learn the offsets δ on a temporal progress. ConvLSTM is first proposed for image sequences. Each convolution in ConvLSTM is two dimensional. Here in our proposed method, we modify it to one-dimensional version so that it can be well applied on the pose traversal data. With the pose traversal representation, ConvLSTM takes the spatial neighboring relationship of joints into consideration in each convolution operation.

In each time step, the input of offset learning module is the pose traversal data with C channels, and the corresponding output is the offset with N channels for the kernel on each anchor location. The hidden and memory cell tensor inside ConvLSTM store significant information of an action/gesture. An Offset Convolution is located at the output end of ConvLSTM which transfers the hidden tensor to the offsets at each time step. The illustration of offset learning is shown in Fig. 5.3, and the key equations of ConvLSTM are detailed in Eq. 5.6.

$$\begin{aligned}
\mathbf{g}_i^t &= \sigma(\mathbf{w}_{xi} * \mathbf{x}^t + \mathbf{w}_{hi} * \mathbf{h}^{t-1} + \mathbf{b}_i) \\
\mathbf{g}_f^t &= \sigma(\mathbf{w}_{xf} * \mathbf{x}^t + \mathbf{w}_{hf} * \mathbf{h}^{t-1} + \mathbf{b}_f) \\
\mathbf{m}^t &= \mathbf{g}_f^t \circ \mathbf{h}^{t-1} + \mathbf{g}_i^t \circ \tanh(\mathbf{w}_{xm} * \mathbf{x}^t + \mathbf{w}_{hm} * \mathbf{h}^{t-1} + \mathbf{b}_m) \\
\mathbf{g}_o^t &= \sigma(\mathbf{w}_{xo} * \mathbf{x}^t + \mathbf{w}_{ho} * \mathbf{h}^{t-1} + \mathbf{b}_o) \\
\mathbf{h}^t &= \mathbf{g}_o^t \circ \tanh(\mathbf{m}^t)
\end{aligned} \tag{5.6}$$

where the input, forget and output gates, $\mathbf{g}_i^t$, $\mathbf{g}_f^t$ and $\mathbf{g}_o^t$, of ConvLSTM are vectors. $\mathbf{m}^t$ and $\mathbf{h}^t$ are memory cell tensor and hidden state tensor respectively. $\sigma(\cdot)$ is the sigma function. $\mathbf{w}$ and $\mathbf{b}$ are the corresponding weights and bias in each operation. $*$ is a symbol for one-dimensional convolution, and $\circ$ denotes the element-wise product.

If the size of the convolution kernel is 1, the learned offset is defined as,

$$\mathbf{o}^t = \mathbf{w}_{of} * \mathbf{h}^t + \mathbf{b}_{of} \tag{5.7}$$

where $\mathbf{o}^t \in \mathbb{R}^{J_e}$ is the learned offset tensor. If the size of convolution kernel is N , the output offset is a matrix $\mathbf{O}^t \in \mathbb{R}^{J_e \times N}$. Each column of $\mathbf{O}^t$ corresponds to an offset set $\{\delta_n\}_{n=1}^N$ on a convolution anchor.

5.3.3 Learning and Classification

After passing through the Deformable Pose Traversal Convolution network, the pose traversal data is transferred to an abstract pose feature. The feature is then

fed into the main LSTM to model the long-term temporal evolution of the input action or gesture. We use the hidden state $\mathbf{h}_m^T$ from the last time step to predict the label. The hidden state $\mathbf{h}_m^T$ is passed into a fully-connected layer and a softmax layer to generate $\hat{\mathbf{z}} \in \mathbb{R}^K$ for action and gesture classification. The k -th element of $\hat{\mathbf{z}}$ is the estimated probability of the input sequence V belonging to class k , namely $\hat{z}_k = p(k|V)$. The objective function is to minimize the cross-entropy loss, which can be optimized via back-propagation through time (BPTT) [131] in an end-to-end manner.

5.4 Experiments

In this section, we evaluate and analyze the proposed method on three 3D action and gesture datasets. The implementation details are introduced in Sec. 5.4.1. Comparison results on the Dynamic Hand Gesture 14/28 dataset (DHG) [129], the NTU-RGB+D dataset (NTU) [61], and the Berkeley Multi-modal Human Action dataset (MHAD) [108] are provided and discussed in Sec. 5.4.3. Experiment results show that the proposed Deformable Pose Traversal Convolution effectively searches the optimal joint combinations on-line and achieves the state-of-the-arts performance for both 3D action and gesture recognition.

5.4.1 Implementation

Representation. The three datasets include single actions, two-person interactions, and hand gestures. All the body actions and hand gestures are represented as 3D poses. To ensure location and view invariance of the representation, each joint of the pose is centralized by subtracting the temporal average pose-center, and each pose is pre-rotated. For single-person actions and hand gestures, the pose-centers are defined as the hip-joint and palm-joint respectively. For two-person interaction, the pose-center is the average hip-joint of the two involved persons in each frame. The two-person interaction is represented by the absolute difference of corresponding joints in two persons.

Network. The convolution networks for pose feature extraction in all the experiments share the same network parameters. We modify the Temporal Convolution

Method	Fine	Coarse	Both-14	Both-28
Skeletal Quads [119]	70.6	92.2	84.5	79.4
SoCJ+HoHD+HoWR [129]	73.6	88.3	83.1	80.0
Pose Chain Conv	76.2	90.4	80.4	75.7
Pose Traversal Conv	77.1	91.8	81.1	76.6
D-Pose Traversal Conv	81.9	95.2	85.8	80.2

TABLE 5.1: Comparison of Results on Hand Gesture Dataset - DHG (%)

Network [130] to extract pose feature of each frame. The main parameters of the network are shown in Fig. 5.2. In our experiments, we use one-layer ConvLSTM for offset learning. The numbers of the main LSTM layer are two, three, and two for DHG, NTU, and MHAD respectively.

Training. Our neural network is implemented by PyTorch. The stochastic optimization method Adam [132] is adapted to train the network. We use gradient clipping similar to [133] to avoid the exploding gradient problem. The initial learning rate of the training is set to 0.001. The batch size for the DHG, NTU, and MHAD dataset are 64, 64, and 32 respectively. For efficient learning, we train the Pose Traversal Convolution network first and use the learned parameters to initialize the deformable network.

5.4.2 Datasets

Dynamic Hand Gesture 14/28. The Dynamic Hand Gesture 14/28 dataset is collected with Intel Real Sense Depth Camera. It includes 14 hand gesture categories, which are performed in two ways, using one finger and the whole hand. Following the protocol introduced in [129], the evaluation experiment is conducted under four settings, Fine, Coarse, Both-14, and Both-28. In each experiment setting, we use leave-one-subject-out cross-validation strategy.

NTU-RGB+D. The NTU-RGB+D dataset is collected with Kinect V2 depth camera. There are 60 different action classes. We follow the protocol described in [61] to conduct the experiments. There are two standard evaluation settings, the cross-subject (CS) and the cross-view (CV) evaluation. In CS setting, half of the subjects are used for training and the remaining are for testing. In CV setting,

Method	CS	CV	Method	Accuracy
ST-LSTM [64]	69.2	77.7	SMIJ [47]	95.4
Two Streams [67]	71.3	79.5	Meta-Cognitive [112]	97.6
GCA-LSTM [65]	74.4	82.8	Dynemes and Forward [113]	98.2
RNN Tree [68]	74.6	83.2	HBRNN [60]	100.0
CNN+MTLN [74]	79.6	84.8	ST-LSTM [64]	100.0
Li <i>et al.</i> [134]	83.2	89.3	ST-NBNN [52]	100.0
Pose Chain Conv	75.2	83.4	Pose Chain Conv	96.4
Pose Traversal Conv	76.1	84.3	Pose Traversal Conv	98.6
D-Pose Traversal Conv	76.8	84.9	D-Pose Traversal Conv	100.0

TABLE 5.2: Comparison on NTU (%)

TABLE 5.3: Comparison on MHAD (%)

two of the three views are used for training and the remaining one is for testing.

Berkeley MHAD. The action sequences in Berkeley MHAD dataset are captured by a motion capture system. There are 11 action categories in this dataset. We follow the experimental protocol introduced in [108] to evaluate the proposed method. The sequences performed by the first seven subjects are for training while the ones performed by the rest subjects are for testing.

5.4.3 Results and Analysis

Comparison with Baselines

We compare the proposed method with two baselines on three datasets, DHG, NTU and MHAD. The two baseline methods are the Pose Convolution with single chain representation (Pose Chain Conv), which simply flatten the pose graph without considering the neighboring relation of joints, and the Pose Convolution with Pose Traversal representation (Pose Traversal Conv). The comparison results are shown in Table. 5.1, Table. 5.2 and Table. 5.3 respectively. From the tables we can see that Pose Traversal Convolution achieves better performance than the Pose Chain Convolution, which verifies the effectiveness of the involvement of traversal representation in pose convolution. In these three datasets, we can also witness that the Deformable Pose Traversal Convolution (D-Pose Traversal Conv) performs better than the Pose Traversal Convolution. The Deformable Pose Traversal Convolution

FIGURE 5.4: (A) Comparison of Confusion Matrices between Pose Traversal Convolution and Deformable Pose Traversal Convolution on DHG-14. (B) Comparison of Sub Confusion Matrices between Pose Traversal Convolution and Deformable Pose Traversal Convolution.

is able to find good combinations of key joints in each convolution operation and to avoid non-informative or noisy joints, which can effectively help improve the recognition accuracy. Moreover, the performance improvement is significantly great on dynamic hand gestures which involve coordination of more pose parts than body actions.

Fig. 5.4a shows the comparison of confusion matrices between Pose Traversal Convolution and Deformable Pose Traversal Convolution on DHG-14 dataset, under the Both-14 setting. The red rectangle marks the sub part of the matrix, which includes the two similar gestures “Grab” and “Pinch”. The gestures in DHG-14 are 1. Grab, 2. Tap, 3. Expand, 4. Pinch, 5. Rotation CW, 6. Rotation CCW, 7. Swipe Right, 8. Swipe Left, 9. Swipe Up, 10. Swipe Down, 11. Swipe X, 12. Swipe V, 13. Swipe +, 14. Shake. As can be seen from this figure, compared with the Pose Traversal Convolution, the confusion matrix of the Deformable Pose Traversal Convolution is clearer, which means that the confusion between gestures is reduced by using Deformable Pose Traversal Convolution. We can also see that there is great confusion between the gesture “Grab” and gesture “Pinch”. These two gestures both belong to the “Fine” category, and they are very similar to each other. The Deformable Pose Traversal Convolution can greatly reduce the number of “Pinch” testing samples wrongly classified as “Grab”, as shown in Fig. 5.4b. The recognition accuracy of the “Pinch” gesture is greatly improved from 49% to 71.5%.

FIGURE 5.5: Impact of Rotation on Accuracy

FIGURE 5.6: Impact of Noise on Accuracy

Comparison with the State-of-the-arts

In this section, the proposed method is compared with the existing methods on three benchmark datasets, DHG-14/28, NTU-RGBD+D, and Berkeley MHAD. As can be seen from Table. 5.1, Table. 5.2 and Table. 5.3, our method achieves comparable performance with the state-of-the-arts on NTU-RGB+D dataset and Berkeley MHAD dataset. On the Dynamic Hand Gesture dataset, the proposed method performs significantly better than the existing methods. It is worth noting that the proposed method, Deformable Pose Traversal Convolution network, performs better than the ST-LSTM [64] and Two Streams network [67] which use Long-Short-Term-Memory network (LSTM) to model the spatial context of joints.

Robustness Analysis

1) Rotation. As 2D pose estimation methods [9, 10] reach a new level recently, we consider whether the proposed method is able to handle 2D pose data well. We conduct an experiment to compare the performance of the Pose Traversal Convolution (PTC) and the Deformable Pose Traversal Convolution (DPTC) on both 2D and 3D data of DHG dataset. The results are shown in Table. 5.4. As can be seen, the accuracies of the proposed methods on 2D data are comparable with the ones on 3D data under different settings. The reason is that the DHG dataset is collected for human-computer interaction, and all the recorded gestures are performed with the palm facing to the camera. Although there is one dimension missing, the proposed method still achieves the performance that is similar to the one on 3D data. The results verify the effectiveness of DPTC over PTC.

Method	Fine	Coarse	Both-14	Both-28
Pose Traversal Conv-2D	74.6	90.1	81.2	77.1
Pose Traversal Conv-3D	77.1	91.8	81.1	76.6
D-Pose Traversal Conv-2D	81.9	94.9	85.1	80.4
D-Pose Traversal Conv-3D	81.9	95.2	85.8	80.2

TABLE 5.4: Comparison between 2D and 3D Data on DHG (%)

We further evaluate the performance of our method under different data rotations to see its robustness to rotation. We rotate the 3D hand pose along the y -axis with -90° , -60° , -30° , 30° , 60° , as well as 90° , and record the recognition accuracies under these settings. The results are shown in Fig 5.5. As can be seen, though with different rotations, the proposed methods can still maintain the performance on 3D data, while on 2D data, the performance changes as along the increase of the rotation degree. Under the 2D data setting, the Deformable Pose Traversal Convolution still finds the optimal joint combinations and performs better than the Pose Traversal Convolution. Considering the rotation of the users' hand is just around $\pm 30^\circ$ in human-computer interaction, our method is able to handle the daily situation well.

2) Noise. Although pose estimation methods achieve good performance recently, the pose noise caused by the estimation errors and occlusion still cannot be ignored in 3D action and gesture recognition. In this section, we conduct experiments to show the tolerance of the proposed method to the pose noise on Berkeley MHAD dataset. We randomly select 10%, 20%, 30%, 40% and 50% of the pose joints and add severe random noise with maximum amplitude up to 50 on the selected joints. We evaluate the performance of Pose Traversal Convolution and Deformable Pose Traversal Convolution with different percentages of noisy joint. The results are shown in Fig. 5.6. As can be seen from the figure, though the accuracy of PTC drops a lot due to the impact of noise, DPTC still can avoid the noisy joints and achieve good performance. Under the setting of 50% noisy joints, the accuracy improvement from PTC to DPTC is 7.64%. Here the noise we use is severe one with amplitude up to 50, and the proposed DPTC can still perform well. Under the setting of 50% noisy joints, we also evaluate PTC and DPTC by using small noise with amplitude up to only 5. The accuracies of PTC and DPTC are 96.36% and 98.18% respectively. We can see that Pose Traversal Convolution is able to

FIGURE 5.7: Visualization of the Learned Offsets

well handle small noise by using the neighboring spatial context of joints.

Visualization

In this section, we visualize the learned offsets of a single frame from the offset learning module in Fig. 5.7. The experiment is conducted on the cross-view setting of the NTU dataset. To simplify the visualization of the experiment, we set the size of the convolution kernel to one. Under this setting, the learned offsets re-arrange the sampling points of the convolution and shift these sampling points to key joint locations that matter to the action. In the figure, the blue curve shows the offset for convolution kernel on each anchor. The sketch on the corner of each sub-figure shows the part of the pose that has high offset response in offset learning. The colors of the joints in the sketch correspond to the colored marker on the x -axis of the figures. The red arrows in the sketch guide the direction of the traversal. For the action “Throw” in Fig. 5.7a, the high response offsets are located around the right hand, marked by colored points. The sketch part at the corner of the sub-figure is the right hand. As can be seen, the kernels on the both side of the index finger are moving toward it, which indicates that for the action “Throw” the key joints are near the right index finger, and the model shifts its kernels to obtain better information. For the action “Kick Something” in Fig 5.7d, the high response

offsets are located in the left foot part, which guide the kernels to move toward the location between the left knee and the left ankle, not the left tiptoe. The reason is that the pose estimation of feet is not always stable in the NTU-RGB+D dataset especially under the cross-view setting, and hence the offset module chooses the point between the knee and the ankle to represent the “Kick” action. The action “Cross Hand” is performed by two hands, and as can be seen from Fig. 5.7f, there are high responses on the both hand parts.

5.5 Conclusions

In this work, we introduce a deformable one-dimensional convolution neural network to traverse 3D pose for pose representation. The convolution kernel is guided by ConvLSTM to deform for discovering optimal convolution context. Due to the recurrent property of ConvLSTM, it can model the temporal dynamics of kernel deformation. The proposed Deformable Pose Traversal Convolution is able to discover the optimal key joint combinations, as well as avoid non-informative joints, and hence achieves better recognition accuracy. Experiments validate the proposed contribution and verify the effectiveness and robustness of the proposed Deformable Pose Traversal Convolution.

Chapter 6

Early Action Recognition with Category Exclusion

6.1 Introduction

Action recognition is a fundamental task of video analysis in the computer vision community. In the past decades, this task has grown dramatically and achieved great success. However, action recognition focuses only on the after-the-fact detection. In many real-world scenarios like surveillance, human-computer interaction, autonomous driving, *etc.*, systems are required to identify an intended activity of human before the activity is fully executed in real-time, which gives birth to a relatively new research topic, Early Action Recognition. Early action recognition, or action prediction, aims to predict the action label from partially observed video. As many actions are very similar to each other at the beginning few frames, it is not easy to extract discriminative features to distinguish one action from others. This visual similarity, especially at the early stages, makes early action recognition very challenging.

To date, the existing methods [81, 87, 89, 91, 92, 135–137] consider the early action recognition as classification problems at different time stages of an action sequence. Features extracted from each timing point are supervised by labels to activate the probability “response” of positive category and suppress the “responses” of negative ones. In this learning mode, the goal is to distinguish the positive category from all negative categories. Negative categories are treated equally. However in reality,

the relationships between the positive category and different negative categories are not identical. Due to visual similarity, capturing error, illumination variation, various performing style, *etc.*, the positive cluster may be quite close to some negative clusters, but be far away from other negative ones. This situation is more prominent at the early stages of actions. The diverse information of different negative categories is ignored in the existing action learning mode, and therefore the corresponding information bear in the input data is filtered out. We believe that this diverse information of negative categories can be leveraged to boost the recognition performance.

When we human recognize a continuous action, we often make a decision by unconsciously excluding candidate categories. Let's take recognizing triple jump from track-and-field sports for example. When we observe an athlete running on a track, which is the early stage of triple jump, we are sure that this is not a field-based sport, and therefore the field-based categories can be excluded. By this exclusion, we can eliminate interferences and improve the accuracy of guessing. In early action recognition, a model can be trained to learn the exclusion of the negative categories similar to the positive category, which further separates the negative categories into hard-to-differentiate ones and easy-to-differentiate ones. This reasoning can be regarded as *eliminative induction*. To the best of our knowledge, there is no previous methods introducing this reasoning into the task, and this chapter is one step, and the first attempt, towards this direction.

In this chapter, we pre-train a classifier of all categories, and model the category exclusion as a mask operation on the classification probability output of the classifier. Since there is no ground truth of category exclusion, and Reinforcement Learning is an efficient way for learning an optimal policy based on future reward, we modify the reinforcement learning framework to train an agent to learn the category exclusion strategies with multiple rewards for each operation. During training, the agent explores different combinations of excluded categories and searches for the optimal policy that helps improve the recognition performance. Fig. 6.1 illustrates the framework of the proposed method.

Frames of the action sequence are fed into the pre-trained LSTM classifier (orange squares) one by one to extract features and output action classification probability (gray, white, and green bars). The input *state* includes the action classification probability, the mask from previous stage, and the features from local segment.

The *state* is further fed into the agent to generate binary masks to mask out the classification probabilities of the interfering negative categories, i.e., category exclusion.

FIGURE 6.1: Illustration of Category Exclusion in Early Action Recognition

By introducing the idea of *eliminative induction* into the task of early action recognition, which is different from the existing methods, we propose a category exclusion agent model to cooperate with a sequential classifier for early recognition. Due to the mechanism behind the agent is different from the traditional sequential classifier, the collaboration of the agent and the classifier thus utilizes complementary information of time dynamic in action sequence, and achieves better performance than the classifier alone. We evaluate the proposed method on three benchmark datasets with extensive experiments, and the results show the efficiency of our proposed method.

6.2 Related Work

Dissimilar to the action recognition task, where actions are well-segmented and assumed to be fully executed and observed, early action recognition is another important task aiming to predict the action label contained in a partially observed video.

Before deep learning technique is widely-applied in computer vision, many traditional methods are proposed to solve the early action recognition task. Ryoo *et al.* [81] propose to involve the well-known integral bag-of-words and dynamic bag-of-words into the action prediction task. Kong *et al.* [87] and Lan *et al.* [83] introduce the Support Vector Machine (SVM) into the early recognition. A soft regression framework is proposed in [89] for activity prediction. Recently, many deep-learning-based methods come out. Hu *et al.* extend their soft regression framework into the Recurrent Neural Network (RNN) model [91], which achieves great recognition performance. Kong *et al.* attempt to bridge the relationship between partial video features and full video features, and propose the Deep Sequential Context Network (DeepSCN) [136]. The Long Short-Term Memory (LSTM) model is re-designed in [92] to memorize hard-to-predict samples for early action recognition. Considering the changing scale of partial actions, a Scale Selection Network is proposed in [71] for action prediction of video stream.

All these methods consider the early action recognition as a series of classification problems. Compared with these methods, we consider to take the diverse information of negative categories into consideration. The proposed coordination between category exclusion strategy and sequential classifiers utilizes more information from action sequence than that utilized by classifiers alone, which helps better recognize early actions.

6.3 Proposed Method

In this section, we introduce how the proposed method excludes categories during action performing and meanwhile predicts actions at different stages of the sequence. The framework of our method is illustrated in Fig. 6.1. The formulation of this proposed method for the early action recognition task is first introduced in Sec. 6.3.1. Then we describe how to model the category exclusion in action recognition in Sec. 6.3.2. The training of the category exclusion agent is detailed in Sec. 6.3.3. To differentiate the action performed by subjects in the action sequence and the *action* determined by the agent as a policy, we apply italics on the second term.

6.3.1 Problem Formulation

Given a pre-trimmed sequence $V = \{\mathbf{z}_t\}_{t=1}^T$ with its sequence-level label $c \in \mathbb{C} = \{1, 2, \dots, C\}$, where C is the number of category, the goal of early recognition task is to predict the frame-level label $c_t = c$ at each timing based on the observation $\mathbf{z}_{1:t}$, and provide the accurate recognition as early as possible. Following the existing setting in [81, 89, 135, 136], we uniformly partition a video sequence which includes complete action execution into N shorter segments. Each segment contains $\lceil T/N \rceil$ frames, where $\lceil \cdot \rceil$ is the rounding up operation. The observation ratio o_n is then defined as n/N , which indicates the first n segments are observed. The n -th stage corresponds to the observation ratio o_n . There are N stages for each sequence. The index of the last frame in the n -th segment is $t = n\lceil T/N \rceil$. We denote $\mathbf{x}_n$ as the feature extracted from the first n segments, and therefore $\mathbf{x}_N$ is the representation for the whole sequence. With these notations, our task can be formulated based on maximum a posteriori (MAP) rule,

$$c_n^* = \arg \max_{c \in \mathbb{C}(\mathbf{s}_n)} p(c|\mathbf{x}_n). \quad (6.1)$$

Considering that the number of candidate categories from the set $\mathbb{C}$ may decrease during action execution based on the situation of category exclusion, this set is defined as a function of the *state* $\mathbf{s}_n$ observed by the agent. As defined in Eq. 6.1, there are two parts in our proposed method, a classifier for frame-level action prediction and an agent for category exclusion. At each stage n , the agent takes a *state* $\mathbf{s}_n$ as input and decides which categories to be excluded, and the classifier will predict action category from the modified category set $\mathbb{C}(\mathbf{s}_n)$.

6.3.2 Category Exclusion Modeling

The category exclusion is modeled as a mask operation on the action prediction output from the classifier. Specifically, the classifier in our framework is an action prediction function $f : \mathbf{x}_n \rightarrow \mathbf{d}_n \in [0, 1]^C$ involved to estimate the probability of classification in Eq. 6.1. $\mathbf{d}_n$ consists of the estimated probability of each action category, namely $d_n^c = p(c|\mathbf{x}_n)$. To exclude categories, a corresponding binary vector $\mathbf{m}_n \in \{0, 1\}^C$ is generated at each observation ratio o_n to mask out some

negative categories (dimensions) of $\mathbf{d}_n$. Each element of $\mathbf{m}_n$ is denoted as m_n^c . $m_n^c = 1$ indicates that the category c is not excluded, and $m_n^c = 0$ indicates that the category c is excluded. Hence, the formulation in Eq. 6.1 can be rewritten as

$$c_n^* = \arg \max_{c \in \mathbb{C}} p(c|\mathbf{x}_n) m_n^c = \arg \max_{c \in \mathbb{C}} d_n^c m_n^c. \quad (6.2)$$

We model the progression of category exclusion as a Markov Decision Process. In considering that the set of excluded category is specific for each action sequence, it is impossible to annotate the exclusion mask ground truth for each of them, and train the agent by supervised learning. We therefore utilize reinforcement learning to learn the category exclusion strategy. The agent explores various exclusion strategies and discovers the optimal ones guided by the designed *reward*. The process of reinforcement learning is to train the agent to map the input *states* to *actions* for optimal strategy. The three key parts of the reinforcement learning, *state*, *action*, and *reward* are detailed as below.

State: To ensure there is enough information from the partially observed sequence for the agent, we introduce three different *state* parts in our method. The feature $\mathbf{h}_n$ extracted from the partially observed video, the current classification probability $\mathbf{d}_n$, and the previous generated exclusion mask $\mathbf{m}_{n-1}$, i.e., $\mathbf{s}_n = (\mathbf{h}_n, \mathbf{d}_n, \mathbf{m}_{n-1})$. At each observation ratio o_n , these three parts are concatenated together, and fed into the agent for mask generation.

Action: As described above, the *action* is defined as a mask operation. The main part of the agent is a function $g : \mathbf{s} \rightarrow \mathbf{u} \in \{0, 1\}^C$, which is approximated by a neural network $\hat{g} : \mathbf{s} \rightarrow \mathbf{v} \in [0, 1]^C$. This network is designed and trained to select the categories to be excluded or kept based on the *state* it observes. The c -th dimension of this network's output $\hat{g}^c(\mathbf{s})$ denotes the inclusion probability with which we keep the estimated classification probability of the c -th category. The lower the inclusion probability, the higher possibility that we exclude the corresponding category. The network induces a probability distribution over the mask space, i.e., $\{0, 1\}^C$, and the probability of a mask $\mathbf{u}$ is given by

$$\pi(\mathbf{s}) = \prod_{c=1}^C \hat{g}^c(\mathbf{s})^{u^c} (1 - \hat{g}^c(\mathbf{s}))^{1-u^c}. \quad (6.3)$$

At the beginning of each action execution, the binary mask $\mathbf{m}$ is initialized as a vector with all elements set to one, i.e., $\mathbf{m}_0 = [1, \dots, 1]$. During action execution, the agent generates a series of immediate binary masks $\{\mathbf{u}_n\}_{n=1}^N$, $\mathbf{u}_n \in \{0, 1\}^C$, which help accumulate excluded categories recorded in $\mathbf{m}$. The update of the accumulated mask $\mathbf{m}$ is defined as

$$\mathbf{m}_n = \mathbf{m}_{n-1} \odot \mathbf{u}_n, \quad (6.4)$$

where $\odot$ is an element-wise multiplication operation. Since $\mathbf{u}_n$ is a binary mask, the accumulated mask $\mathbf{m}_n$ is also binary.

Reward: During the training, the agent explores different *action* combinations to learn exclusion strategies. The reward is designed as the guidance for the agent to learn the optimal strategy. In early action recognition, information might not be sufficient to exclude all negative categories. To achieve better recognition performance, especially at low observation ratio, it is essential to exclude the interfering categories, which are negative categories but bearing estimated class probabilities on a par with the positive category. Therefore, the reward design should encourage the agent to exclude those negative categories with high estimated classification probability and ignore those with low estimated one. However, in the traditional reinforcement learning framework, there is just one reward $\mathcal{R} \in \mathbb{R}$ for one operation, which is not suitable for our task. We therefore introduce a *Multi-rewards for One Operation* learning strategy to learn category exclusion patterns. Meanwhile, to avoid the positive category is wrongly excluded, exclusion of any positive category should be strongly punished in our design. Based on the consideration above, we set reward for each individual category dimension, and the immediate reward $\mathbf{r} \in \mathbb{R}^C$ is defined as

$$r_n^c = \begin{cases} -\text{sign}(m_n^c) \eta, & \text{if } c = c^+ \\ \text{sign}(m_n^c) d_n^c, & \text{if } c \neq c^+, \end{cases} \quad (6.5)$$

where η is a constant encouraging the agent to maintain the positive category c^+ . The function $\text{sign}(x)$ is a sign function introduced to steer the direction between encouragement and punishment of chosen *action*. The function $\text{sign}(x)$ returns 1 when x equals to 0, and returns -1 when x equals to 1. With the immediate

reward defined in Eq. 6.5, the accumulated reward $\mathbf{R} \in \mathbb{R}^C$ is defined as $\mathbf{R}_n = (\sum_{i=n}^N \mathbf{r}_i)/(N - n + 1)$, which is the average value of the immediate rewards from current stage n to the end.

6.3.3 Agent Training

The goal of the agent training in our proposed method is to learn an optimal category exclusion strategy. This learned strategy can be directly applied on the testing sequences to exclude negative interfering categories, and thus helps improve early action recognition performance. To achieve this, we use the well-known policy-gradient learning algorithm REINFORCE [138] to train the agent and maximize the accumulated rewards. During agent training, the agent generates binary masks $\mathbf{u}_n$ by choosing categories to be excluded randomly following the estimated inclusion probability outputted from the network $\hat{g}(\cdot)$. Each *action* the agent chooses is punished or encouraged on the basis of the obtained reward. In traditional policy-gradient reinforcement learning, the agent receives a scalar reward $\mathcal{R}$ for optimization. To optimize the agent, the goal is to minimize the following loss function

$$\begin{aligned} \mathcal{L}_n^r &= -\log(\pi(\mathbf{s}_n))\mathcal{R}_n \\ &= -\sum_{c=1}^C \left[u_n^c \log(\hat{g}_n^c) + (1 - u_n^c) \log(1 - \hat{g}_n^c) \right] \mathcal{R}_n, \end{aligned} \quad (6.6)$$

where the $\hat{g}_n^c$ is short for $\hat{g}^c(\mathbf{s}_n)$. The loss defined above can be regarded as a reward-weighted binary cross entropy. The higher the reward $\mathcal{R}_n$, the more important the activation of the probability $\pi(\mathbf{s}_n)$ to the related *action* $\mathbf{u}_n$.

While in our method, as we described in Sec. 6.3.2, to improve the searching efficiency in exclusion strategy space, we introduce the *Multi-rewards for One Operation* strategy in our training, and therefore the optimization of the agent is to minimize the loss function Eq. 6.7.

$$\mathcal{L}_n^r = -\sum_{c=1}^C \left[u_n^c \log(\hat{g}_n^c) + (1 - u_n^c) \log(1 - \hat{g}_n^c) \right] \tilde{\mathcal{R}}_n^c, \quad (6.7)$$

We normalize $\mathbf{R}$ to be $\tilde{\mathbf{R}}$ here to strengthen the gradient descent. For the reward set $\{\mathbf{R}_n\}_{n=1}^N$ of a certain action sequence, we pick the highest reward value as the normalization factor to rescale these rewards to range $[0, 1]$. The training procedure of the proposed method is summarized in Alg. 1.

Algorithm 1: Category Exclusion Learning

Input: Training videos $\{V_k\}_{k=1}^K$, Labels $\{c_k\}_{k=1}^K$

Output: Exclusion agent $\hat{g}(\cdot)$

Train $f(\cdot)$ with V and c by supervised learning

Obtain 1) Class probability $\{\mathbf{d}_n\}_{n=1}^N$ of every sample

2) Feature $\{\mathbf{h}_n\}_{n=1}^N$ of every sample

Initialize the agent $\hat{g}(\cdot)$

for $epoch \leftarrow 1$ **to** E **do**

for $k \leftarrow 1$ **to** K **do**

 initialize mask $\mathbf{m}_0 \leftarrow \mathbf{1}$

for $n \leftarrow 1$ **to** N **do**

 get $\mathbf{d}_n$ and $\mathbf{h}_n$ from sample V_k

 explore $\mathbf{u}_n$ with $\hat{g}(\mathbf{d}_n, \mathbf{m}_{n-1}, \mathbf{h}_n)$

 update $\mathbf{m}_n$ using $\mathbf{u}_n$ by Eq. 6.4

 obtain reward $\mathbf{r}_n$ using $\mathbf{m}_n$ and $\mathbf{d}_n$ by Eq. 6.5

 compute the normalized reward $\tilde{\mathbf{R}}$ by $\{\mathbf{r}_n\}_{n=1}^N$

 compute the loss $\mathcal{L}^r$ by Eq. 6.7

 update agent $\hat{g}(\cdot)$

6.4 Experiment

In this section, we evaluate the proposed method on three action benchmark datasets, the NTU-RGBD dataset [61], the First-Person Hand Action (FPHA) dataset [139] as well as the UCF-101 dataset [140], and compare its performance to the existing methods. We briefly introduce the datasets in Sec. 6.4.2. The implementation details are described in Sec. 6.4.1. Comparison experiments and analysis on the three datasets are discussed in Sec. 6.4.3. The evaluation results show the efficiency of the proposed method.

6.4.1 Implementation

Features: There are two models in the proposed method, the baseline classifier and the agent for category exclusion. We use the 3D pose data in the NTU-RGBD and the First-Person Hand Action dataset. We follow the pre-processing method described in [53, 139] to process the pose data. For the training of $f(\cdot)$ on the 3D datasets, we directly use the raw pose as the input feature $\mathbf{z}_t$. The accumulated hidden feature bear in LSTM is regarded as feature $\mathbf{x}_n$ for classification. We denote w_n as the temporal local window of length l at the end of the n -th partial video segment. For the classifier training on the UCF-101 dataset, images $\mathbf{z}_t$ in w_n are concatenated together and fed into 3D ResNeXt-101 network [43] pre-trained on Kinetics dataset [141] for visual features $\bar{\mathbf{z}}_n$.

FIGURE 6.2: Illustration of Linear Feature Embedding Layers (FPHA)

The trained classifier $f(\cdot)$ is further used to generate classification probability $\mathbf{d}_n$ for agent training. During agent training, we suppress those classification probability lower than random guess, i.e., $d_n^c < 1/C$, by setting them to zeros to reduce the data noise and avoid overfitting in training. For the two 3D pose datasets, $\mathbf{h}_n$ is the concatenated raw pose data from local window w_n . For UCF-101, the feature $\mathbf{h}_n$ for agent is the same as $\bar{\mathbf{z}}_n$.

To inform the agent to be aware of the exclusion situation in the previous one stage, we add one more information $\bar{\mathbf{d}}_n = \mathbf{d}_n \odot \mathbf{m}_{n-1}$ to the classification probability part of the input *state*. Considering that the input to the agent consists of multiple parts, $(\mathbf{h}_n, \mathbf{d}_n, \mathbf{m}_{n-1}, \bar{\mathbf{d}}_n)$, it is necessary to project these different features into the same embedding space for better representation. We therefore add linear embedding layers before the flow of these features into the agent. The feature embedding layers are illustrated in Fig. 6.2, which takes the case of FPHA dataset as an example. The classification probability part is embedded as a 64-dim

vector, the exclusion mask part is embedded as a 32-dim vector, and the local data is embedded as a 128-dim vector in the FPHA dataset, as shown in Fig. 6.2. The embedded version of different parts are then concatenated together and fed into the agent.

Training: Without loss of generality, we use LSTM [124] as the sequential classifier the agent cooperates with. The classifier is trained with frame-level label supervision, and optimized by Adam [132]. The output of $f(\cdot)$ is further processed by Softmax function to have classification probability. For NTU dataset, $f(\cdot)$ is a three-layer LSTM with 100 neurons in each layer. The learning rate is set to 0.005. For FPHA dataset, the LSTM classifier has one layer with 100 neurons. The learning rate is set to 0.005. For the UCF-101 dataset, the classifier has one layer and the number of neuron is 512. The learning rate is set to 0.1.

The agent is a two-layer RNN with 100 neurons in each layer for the NTU and FPHA dataset. For the UCF-101 dataset, the agent is a one-layer RNN with 100 neurons. To ensure that the range of each dimension of agents' output value is $[0, 1]$ to have inclusion probabilities, we locate Sigmoid functions at the output end of the agent. Inclusion probabilities that larger than 0.5 are re-assigned as 1, and the ones lower than 0.5 are re-assigned as 0. The consequent binary vector is the $\mathbf{u}$ defined in Sec. 6.3.2. The learning rate is set to 0.0005 in the training. The batch sizes are 128, 16 and 512 for the NTU FPHA and UCF-101 dataset respectively.

6.4.2 Dataset

NTU-RGB+D. The NTU-RGB+D dataset is currently the most challenging RGB-D dataset in 3D action recognition. It is collected with Kinect V2 depth camera. There are around 56,000 sequences in total. 60 different action classes are performed by 40 subjects aged from 10 to 35. There are 25 joints included in each skeletal pose. We follow the protocol introduced in [61] to conduct the experiment. This dataset has two standard evaluation settings, the cross-subject (CS) evaluation and the cross-view (CV) evaluation. In cross-subject setting, half of the subjects are used for training and the remaining ones are for testing. In cross-view setting, two of the three views are used for training and the left one is for testing. In this dataset, we set N to 20, and the results of NTU dataset are

all based on this setting. The size of the local window l is set to 10. The positive reward constant η is set as 1.2.

First-Person Hand Action. This is a publicly available dataset for 3D hand-object interaction recognition that contains labels for 3D hand pose, 6D object pose and action categories. We use this dataset to evaluate the early recognition performance of our proposed method. In this dataset, each hand pose consists of 21 joints. There are 45 indoor hand-object interaction categories included. This dataset contains 1,175 action videos performed by six actors. We follow the protocol described in [139] to evaluate our method. There are 600 action sequences for training and 575 samples for testing. We set N to 20 and l to 10 in this dataset. The positive reward constant η is set as 1.

UCF-101. The UCF-101 dataset is an unconstrained RGB video-based dataset. It contains 13,320 videos distributed in 101 action categories. We follow the settings in [136] to evaluate our method. The first 15 groups of videos are used for training, the next three groups are used for validation, and the rest ones are for testing. There are 3,682 full videos in total for testing. We set N to 10 and l to 16 in this dataset. The positive reward constant η is set as 1.

6.4.3 Results and Analysis

Comparison with Baseline and the State-of-the-arts

We compare the early action recognition performance of the proposed method with

FIGURE 6.3: Comparison of Early Recognition Curves

the baseline on the three datasets, the NTU-RGBD dataset, the FPHA dataset and the UCF-101 dataset. The comparison results are shown in Table. 6.1 with different observation ratios. The AUC is abbreviated for Area under Curve, which is equivalent to the average accuracy over all observation ratios. The accuracies higher than the baseline are emboldened. As can be seen, the proposed method achieves better performance consistently on all observation ratios which indicates the effectiveness and reliability of our method. The AUC or the average accuracy improvements are 1.69%, 1.78%, 2.13% and 0.41% for the two settings of the NTU, FPHA, and UCF-101 datasets respectively. The performance gains at the early stages are significant, higher than the ones at the late stages. On the observation ratio 10%, the accuracy improvements are much higher with 2.98%, 3.16%, 5.74% and 0.87% respectively.

Observation Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	AUC
N-CS : LSTM	26.41	32.67	42.48	52.33	60.49	65.86	69.78	72.21	73.99	75.20	55.83
N-CS : Ours	29.39	35.56	45.25	54.63	62.07	67.08	70.63	72.91	74.54	75.53	57.51
N-CV : LSTM	27.39	33.86	43.55	54.70	63.25	68.85	72.34	74.86	76.56	77.84	57.93
N-CV : Ours	30.55	37.22	46.95	57.18	64.97	69.92	73.13	75.41	76.88	77.99	59.71
FPHA : LSTM	45.91	54.26	61.39	63.30	65.74	69.22	70.61	72.17	73.39	74.43	64.11
FPHA : Ours	52.00	59.65	64.52	65.91	67.65	70.43	72.00	73.57	74.26	74.96	66.66
UCF : LSTM	80.61	84.11	85.96	87.37	88.38	88.76	89.60	89.68	90.47	90.66	87.56
UCF : Ours	81.48	84.74	86.47	87.86	88.78	89.11	89.92	89.95	90.55	90.79	87.97

TABLE 6.1: Early recognition accuracy comparison with baseline on two settings (CS & CV) of NTU-RGBD, FPHA and UCF-101 datasets (%)

We also compare the performance of the proposed method with the existing methods, MS-RNN [91], DeepSCN [136], RankLSTM [137], MemLSTM [92], and MTS-SVM [87] on the NTU and UCF-101 dataset. Here for the NTU dataset, we only use the 3D pose data for the evaluation. However, those methods included in the comparison use not only the 3D pose data, but the RGB and Depth data as well. As can be seen from Fig. 6.3, with the feature used in our model, the proposed method can achieve comparable performance with the state-of-the-arts or beats some of them, which indicates the efficiency of our method.

Mask Exclusion Analysis

To be detailed, we show the confusion matrices before and after exclusion as well as their difference in Fig. 6.4. The difference matrix is the difference between the confusion matrices after and before exclusion. The negative values are marked by red and the positive values are marked by blue in this difference matrix. The

FIGURE 6.4: The Confusion Matrices before and after Exclusion and Their Difference at the Observation Ratio 5% (FPHA)

evaluation is conducted on the FPHA dataset, and the confusion matrix is based on the classification situation of observation ratio 5%. As can be seen from the figure, after exclusion, the diagonal elements of confusion matrix are strengthened, while the off diagonal elements are weakened for many categories. In the difference confusion matrix, the blue squares on the diagonal line indicate the accuracy improvements after exclusion, and the red ones off diagonal line indicate the drops of misclassification rates after exclusion. Two phenomena can be summarized from the difference confusion matrix. 1) The squares on the diagonal are the classification accuracies of different categories. We can see that for most of the categories, there are accuracy improvements after exclusion; 2) The off diagonal squares are misclassification rates of different categories. There are many red squares in this area, which indicates the misclassification rates of most categories decreases. There are also small number of blue squares located in this off diagonal area, showing a little price of the proposed scheme.

To further check the exclusion mask distribution among different samples, we visualize the statistics of the exclusion mask generated from training and testing samples of FPHA dataset. The visualizations are shown in Fig. 6.5. Each row of the matrix is the average mask of the binary mask over all input videos of the corresponding category. Each dimension of each row shows the exclusion situation of that related category. Considering that for each action sequence, the exclusion mask generated at the final stage is the accumulation of the ones from the previous stages, as defined in Eq. 6.4, we use the accumulated mask from the final stage as the exclusion mask to show the exclusion pattern of an action sequence. Each row of the matrix is the mean mask over all the input videos of the corresponding

FIGURE 6.5: Mask Statistics on FPHA

category. The value 0.0 of the square of each row indicates that all the samples in the corresponding category (row) choose to exclude the category the column represents. The value 1.0 indicates that all the samples in the corresponding category (row) choose not to exclude the category the column represents.

As can be seen from Fig. 6.5, for most categories of the input videos, they tend to exclude the categories 6. *Prick Fork*, 9. *Put Sugar*, 15. *Put Tea Bag*, 17. *Open Dish Soap*, 18. *Close Dish Soap*, 21. *Flip Sponge*, 23. *Squeeze Sponge*, 36. *Unfold Glasses* and 38. *Open Wallet*, which means that these categories are very similar to other categories, especially at the early stages. If we refer back to Fig. 6.4, we can clearly see that most of the category are wrongly recognized as 9, 15, 17 and 38, which shows that the agent has correctly learned the interference categories that should be excluded for better classification decision. If further checking the squares on the diagonal of the matrix, we can see that most of the values are 1.0, which means that the agent learns not to exclude the positive categories. Besides, the exclusion patterns of the training and testing samples are very similar to each other. This similarity shows that the exclusion patterns learned from the training data are well generalized to the testing data. This shows good generalization capability of our proposed agent model for category exclusion learning.

Ablation Study for Agent State

During the progression of category exclusion, besides the classification probability d_n and the generated mask m_{n-1} , the agent also needs information from observation h_n to make decisions. There are two sources of information to be considered,

the feature of the data directly from the sequence, and the hidden data accumulated in LSTM classifier. In this section, we conduct experiments to evaluate the efficiency of these two sources of information on the FPHA dataset. To see the contribution of the raw data from local window, and the hidden representation from LSTM in exclusion mask estimation, the dimension of their embedded version is set to the same, *e.g.*, 128. There are four different experiments conducted. Here, we provide the early recognition accuracy of the proposed method without observation $\mathbf{h}_n$ (None), with only the hidden feature from the classifier (Hid), with only the feature from local temporal window w_n (Local), and with both the hidden feature and the feature from local window (Hid+Local). The results are shown in Table. 6.2. The highest accuracies are emboldened.

Observation Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	AUC
LSTM	45.91	54.26	61.39	63.30	65.74	69.22	70.61	72.17	73.39	74.43	64.11
None	46.43	54.61	61.74	63.13	65.22	68.35	69.74	71.83	72.87	74.09	63.85
Hid	51.83	58.61	64.00	64.87	66.78	69.22	70.43	71.65	72.35	73.22	65.46
Local	52.00	59.65	64.52	65.91	67.65	70.43	72.00	73.57	74.26	74.96	66.66
Hid+Local	53.91	60.17	65.39	66.78	68.35	70.61	72.17	73.22	73.74	74.61	67.04

TABLE 6.2: Early recognition ablation study on FPHA (%)

As can be seen, without the information from sequence observation $\mathbf{h}$, the proposed method cannot achieve the accuracy as good as the baseline. This shows that there are limited informative patterns from the classification probability $\mathbf{d}$ for category exclusion. With only the hidden feature from the classifier, the proposed method can achieve better performance than the baseline at the early stages. However, the accuracy drops after observation ratio 60%. We consider the reason is that the hidden feature is learned by positive label supervision. The information bear in the hidden feature is to distinguish the positive category from all negative categories. While here the observation $\mathbf{h}_n$ is designed to search for interfering negative categories, which is eliminative induction. It is not easy to extract useful information from the hidden feature for the exclusion task. With only the feature from local temporal window, we can see there is great improvement over the baseline. Although the hidden feature comes from the raw data, the information of the diverse negative categories is suppressed in hidden feature. The local feature itself is the raw data, and we can see the agent learns good patterns from it. In the case of *Hid+Local*, we can observe that with the help of *Hid*, the improvement is limited. The comparison shows that 1) learning classification and learning category exclusion are two tasks that are complementary; 2) During classification training, there

is some information thrown away which is helpful for the exclusion task. Here the exclusion task collects this information and helps achieve better classification performance.

FIGURE 6.6: The Learned Weights Distribution of Agent's Input Layer on Input Embeddings

We further check the contribution of these two sources by visualizing the parameters of the agent's input layer. This experiment is conducted on the FPHA dataset. The parameters that related to the embedded $\mathbf{h}$ in the input layer is a $D_e \times D_h$ matrix $\mathbf{W}_{ih}$, where D_e is the dimensionality of the embedded $\mathbf{h}$, and D_h is the dimensionality of the hidden feature in LSTM classifier. Fig. 6.6 shows the contribution of the two parts, *Hid* and *Local*, of the embedded $\mathbf{h}$ to the exclusion mask generation. The color bar under the x -axis marks the range of each related feature. This curve is recorded in a one-dimensional vector $\mathbf{a} \in \mathbb{R}^{D_e}$ obtained by,

$$\mathbf{a} = \frac{1}{D_h} \sum_{j=1}^{D_h} \frac{1}{Z_j} p(\mathbf{W}_{ih}^j), \quad (6.8)$$

in which $\mathbf{W}_{ih}^j$ denotes the j -th column of $\mathbf{W}_{ih}$. $p(\cdot)$ is the element-wise square function. Z_j is a normalization factor for each column vector of $\mathbf{W}_{ih}$. Z_j here is the maximum of $\mathbf{W}_{ih}^j$.

From Fig. 6.6 we can see that the weights corresponding to local poses embedding feature bear most of the high weights, while the weights of the hidden feature part

are less than the weights of local poses feature. We further calculate the mean weights of each part, and the results are $3.5 * 10^{-3}$ and $1.9 * 10^{-3}$ respectively. The results indicate that the local poses feature contributes the most to the generation of the exclusion mask, which further verify the conclusion we made in the early part of this section.

Convergence Analysis

We record during training phase the classification accuracy difference before and after the category exclusion on the training and testing data, as well as the reward of training data, and draw the curve in Fig. 6.7. From Fig. 6.7 we can see that the agent tends to converge after 500 epochs. After convergence, the average difference of recognition accuracy is above 0, which means that the average recognition accuracy after exclusion is better than the accuracy before exclusion. The similar trend happens for the testing samples.

FIGURE 6.7: Accuracy Difference and Reward Curve of FPHA

One of the important indicators in Reinforcement Learning is the *reward*, as defined in Eq. 6.5. Considering that the *reward* designed in our proposed method is not a single scalar, here the reward we draw is defined as $\sum_{n,c} r_n^c / (NC)$. The accuracy difference is defined as the recognition accuracy difference between early recognition before and after category exclusion. The accuracy and reward curves are drawn step/iteration by step/iteration, and we denote the index of the x -axis as *step*. As can be seen from Fig. 6.7, the *reward* increases gradually during training, and tends to be stable after 400 steps, which indicates that the training based on Reinforcement Learning converges. The agent has already found the optimal strategies for category exclusion after the training.

Parameter Sensitivity Analysis

The positive reward η , as defined in Eq. 6.5, is a critical parameter balancing

Observation Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	AUC
LSTM	45.91	54.26	61.39	63.30	65.74	69.22	70.61	72.17	73.39	74.43	64.11
Ours-1.0	52.00	59.65	64.52	65.91	67.65	70.43	72.00	73.57	74.26	74.96	66.66
$\eta = 0.1$	24.52	27.13	28.70	28.70	29.04	29.74	29.22	29.22	29.22	29.39	28.37
$\eta = 0.3$	48.52	51.65	54.26	56.17	58.09	58.78	59.30	60.52	61.04	61.57	56.34
$\eta = 0.5$	53.39	60.35	65.22	67.30	68.87	71.13	72.00	74.09	74.61	75.13	67.43
$\eta = 0.7$	51.13	58.43	63.48	65.04	66.09	69.22	70.78	72.52	73.22	73.74	65.54
$\eta = 0.9$	53.04	60.17	65.22	66.78	68.52	71.13	72.35	73.91	74.43	75.13	67.20
$\eta = 1.1$	51.83	58.96	64.17	65.57	67.48	70.43	71.65	73.22	73.74	74.43	66.24
$\eta = 1.3$	53.57	59.83	64.87	66.26	68.00	70.61	72.00	73.22	74.09	74.61	66.89
$\eta = 1.5$	50.26	57.22	63.30	64.52	66.43	69.74	70.96	72.52	73.39	74.78	65.40
$\eta = 1.7$	51.65	58.43	64.35	65.91	68.00	70.43	71.83	73.22	73.74	74.43	66.33
$\eta = 1.9$	51.13	58.09	64.00	65.22	67.13	70.43	71.48	73.04	73.91	74.43	66.00
$\eta = 2.1$	51.65	58.43	64.17	65.39	67.13	70.26	71.65	73.39	74.26	74.78	66.25
$\eta = 3.1$	45.91	54.09	61.39	62.96	65.57	68.70	70.09	71.83	73.04	73.91	63.82
$\eta = 4.1$	46.61	54.43	61.04	62.61	65.04	68.35	69.74	71.48	72.70	73.74	63.65
$\eta = 5.1$	47.48	55.65	62.26	64.52	66.96	69.91	70.78	72.87	73.57	74.43	64.96

TABLE 6.3: Parameter Sensitivity Analysis of the Reward η on FPFA (%).

the encouragement and punishment of the agent’s behavior. In this section, we evaluate the sensitivity of our proposed method to this parameter η . The results are recorded in Table. 6.3. In this table, the rows with the overall accuracies lower than or close to the LSTM are marked by red. We set the value of η in two ranges, from 0 to 2 (step set as 0.2) and higher than 2.0 (step set as 1). As can be seen from Table. 6.3, in the first range, when η is set lower than 0.5, the agent cannot learn the exclusion patterns well, and hence achieves worse performance than the LSTM. When η is larger than 2.1, the encouragement of exclusion of the negative categories is much stronger than that of maintaining the positive category to learn the exclusion properly. From Table. 6.3 we can see that when $0.5 < \eta < 2$, the agent has a good balance between maintaining the positive category and excluding the interfering negative categories, and hence performs better than the baseline LSTM.

Visualization

In this part, we visualize the classification probability outputs of the classifier, the prediction decision of maximum a posteriori (MAP), and the exclusion mask in our model. Fig. 6.8 shows the visualizations of six samples from the NTU-RGBD (CS). Each visualization of a sample consists of three matrices. The size of each matrix is 20 by 60, namely $N * C$. The numbers on the vertical direction of each matrix indicate the stage indices, and the ones on the horizontal direction indicate the category indices. The matrix in the first row of each visualization sample shows the prediction outputs before exclusion. The colors in the matrix indicate the value of the prediction probability from low (purple) to high (yellow). The white

FIGURE 6.8: Visualization of the Generated Mask and the Recognition before and after Exclusion

lines in the matrix indicate the prediction decision based on maximum a posteriori (MAP). In each row of the visualization matrix, the square with the highest value is marked as white. The matrix in the second row of each individual visualization is the probability outputs after exclusion with the marked prediction decision. The masks are shown in the third row. The black lines in the mask indicate that the related categories are excluded. The white lines in the mask indicate that the related category is kept for MAP prediction. The light blue line in the mask indicates the positive category. As can be seen from the figure, the generated mask helps exclude the interfering negative categories, and helps the classifier search the correct category for prediction. Fig. 6.8 also shows that the learnt agent excludes interfering negative categories at the very early observation ratio. Let's take the (c) *Jump Up* for example. As can be seen from the visualization, the action *27-Jump Up* and the action *26-Hopping (one foot jumping)* are very wrongly distinguished by the classifier, especially at the late stages. However, at the early stage, the agent learns to exclude the category *26-Hopping (one foot jumping)*, and therefore avoids confusion at the late stages.

Except the samples (*e*) and (*f*), all other illustrations are from different categories. As can be seen from visualizations (*a*) to (*e*), samples from different categories generate different exclusion patterns. Even samples from the same category, namely sample (*e*) and (*f*), do not share exactly the same exclusion pattern. The exclusion mask is generated differently from sample to sample. Also, from samples like (*c*) and (*f*), we see that two samples from different categories may share the same exclusion pattern.

6.5 Conclusion

In this chapter, we propose a new approach to the task of early action recognition with a novel strategy: learning category exclusion. Different from the existing methods considering early action recognition as sequential classification problems, we introduce the idea of eliminative induction in this task, which utilizes the complementary information to that utilized in sequential classification, and hence category exclusion helps achieve good performance. We train an agent to exclude negative interfering categories by reinforcement learning. During action execution, the agent cooperates with the pre-trained classifier and generates masks to exclude interfering categories based on the partial input sequence it observes. The experimental studies show some insights of the proposed framework for category exclusion. Performance evaluations at different observation ratios on three benchmark datasets demonstrate the effectiveness and consistency of our proposed model on early action recognition task.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

This thesis mainly focuses on the *Action Recognition* and *Action Prediction* tasks to further analyze human behaviors. To analyze human behaviors by computer vision, there are two basic questions to answer, “what is action?” and “how to differentiate actions?”. This thesis attempts to provide answers to these two questions.

To the question “what is action?”, the answer in this thesis is that an action can be represented by the key spatio-temporal patterns in the sequence. These spatio-temporal patterns can be discovered from the training data and further utilized to recognize actions and help understand human behaviors. Therefore, two main works are proposed to discover those informative spatio-temporal patterns. The ST-NBNN in Chapter 3 and the ST-NBMIM in Chapter 4 introduce the STM to learn the spatial weights and temporal weights, which can fully utilize the spatial temporal structure of 3D human actions. With the mined spatial weights and temporal weights, the proposed ST-NBNN and ST-NBMIM are able to not only achieve better recognition performance than the baseline without key pattern discovery, and provide physical interpretation of the action. These two methods are evaluated on five benchmark datasets including the MSR-Action3D dataset [107], the UT-Kinect [98], the Berkeley MHAD dataset [108], the SBU-Interaction dataset [116], and the NTU RGB+D dataset [61]. The experimental results show that these two proposed methods are able to achieve state-of-the-art performance. Besides, extensive experiments demonstrate that these two proposed methods can well handle

the case of noisy joints, and the visualization of the key joints and the key frames verify the interpretability of the discovered spatio-temporal action patterns.

However, the spatial and temporal patterns learned in ST-NBNN and ST-NBMIM are class-related, and they are learned off-line, which is not suitable for the action sequence data with large temporal dynamic variation. Hence, the Deformable Pose Traversal Convolution is further proposed in Chapter 5 to solve this problem. The deformable convolution network [122] for image recognition is introduced in this method to process pose data. The deformation parameters for the convolution kernel vary for different input samples, temporal steps and convolution anchors. These variations improve the flexibility of the attention mechanism and spatio-temporal pattern discovery. Evaluations on three benchmark datasets the Dynamic Hand Gesture 14/28 dataset (DHG) [129], the NTU-RGB+D dataset (NTU) [61], and the Berkeley Multi-modal Human Action dataset (MHAD) [108] show that the proposed DPTC can well handle the task of 3D gesture recognition and 3D action recognition. The visualization shows the learned deformation parameters are also interpretable and that the kernel bears the ability to sample those informative coordinations between joints.

To the question “how to differentiate actions?”, this thesis explores two classification directions, the non-parametric model and the idea of category exclusion. The ST-NBNN and ST-NBMIM in Chapter 3 and Chapter 4 utilize the non-parametric model NBNN as the basic classifier, which makes the classification decision by the label distribution in local neighborhood. Considering that the pose data consists of only tens of joints, the recognition based on it is a light-weight problem and the nearest neighbor based methods like NBNN is quite suitable for it. Besides, the STM is involved to reshape the complex classification boundary the NN-based method constructs, which further improve the recognition accuracy. The evaluations in these two chapters demonstrate the potential of using non-parametric method for skeleton-based action recognition.

To date, most of the classification problems are modeled as distinguishing the positive category from all negative categories, which ignores the diverse information among negative clusters. However, the truth is the positive cluster may be very close to some negative clusters, while at the same time be far away from the others. The negative categories can not be treated identically. This thesis therefore

introduces the *Category Exclusion* into the early action recognition task in Chapter 6. The proposed reinforcement-learning-based method can well distinguish the hard-to-differentiate (from positive category) negative categories, and those easy-to-differentiate ones. During action performing, the learned agent adaptively chooses which negative categories to exclude to reduce the confusion of classification decision making, and therefore it can help achieve better action prediction accuracy. The proposed method is evaluated on three benchmark datasets the NTU-RGBD dataset [61], the First-Person Hand Action (FPHA) dataset [139] as well as the UCF-101 dataset [140], and the results demonstrate the effectiveness and consistency of our proposed model on early action recognition task.

7.2 Future Work

To well understand human behaviors, the next step can be the *Online Action Recognition* and the *Action Recognition under Ego-centric View*. Both of these two tasks have very different input data from the ones introduced in this thesis.

The methods introduced in Chapter 3 to Chapter 6 are proposed for off-line action recognition, in which the video only includes one action instance, and the whole sequence is observed in classification. While online action detection aims to estimate the frame-level label of the ongoing activity from a video stream in real-time. The streaming video may include multiple action instances. Owing to the requirement of responsiveness, the classification of action label is performed almost in every frame, and thus the input action is always partially observed for the recognition. To solve the online action recognition task, three problems may be considered, 1) the model must be simple enough so that it can be highly responsive; 2) the starting and ending time of the action should be detected online; 3) it is not easy to recognize the background actions (non-target actions) as these actions are not easy to cluster.

As the development of the techniques for mobile devices, the research focusing on the wearable devices becomes a popular trend. How to recognize people's daily activity from the egocentric perspective has high requirement under this trend, and it will be an interesting research topic to focus on. The data of this task is recorded by the camera on the user's head. Compared with the third-person

perspective, with which the videos are always recorded in fixed view, the view under the egocentric perspective often changes, and this view changing may cause the motion blur and sudden appearance variety, which results in a very challenging computer vision task. However, the benefit would be there is no occlusion for hand-object interaction since the hand and the object are always located at the center of the image, and therefore this perspective can be quite suitable for human-object interaction recognition.

List of Author's Awards, Patents, and Publications

Journal Articles

- **Weng, Junwu** and Weng, Chaoqun and Yuan, Junsong and Liu, Zicheng, “Discriminative Spatio-Temporal Pattern Discovery for 3D Action Recognition,” *IEEE Transactions on Circuits and Systems for Video Technology (T-CSVT)*, Mar. 2018.
- **Weng, Junwu** and Jiang, Xudong and Zheng, Weilong and Yuan, Junsong, “Early Action Recognition with Category Exclusion using Policy-based Reinforcement Learning,” *IEEE Transactions on Circuits and Systems for Video Technology (T-CSVT)*, Feb. 2020.

Conference Proceedings

- **Weng, Junwu** and Weng, Chaoqun and Yuan, Junsong, “Spatio-Temporal Naive-Bayes Nearest-Neighbor (ST-NBNN) for Skeleton-based Action Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- **Weng, Junwu** and Liu, Mengyuan and Jiang, Xudong and Yuan, Junsong, “Deformable Pose Traversal Convolution for 3D Action and Gesture Recognition,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

- Ge, Liuhao and Cai, Yujun and **Weng, Junwu** and Yuan, Junsong, “Hand PointNet: 3D Hand Pose Estimation using Point Sets,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Bibliography

- [1] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding (CVIU)*, 104(2-3):90–126, 2006. [1](#)
- [2] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010. [1](#)
- [3] Gang Yu, Junsong Yuan, and Zicheng Liu. Introduction to human action analysis. In *Human Action Analysis with Randomized Trees*, pages 1–8. Springer, 2015. [2](#)
- [4] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2204–2212, 2014. [5](#)
- [5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017. [5](#)
- [6] Jamie Shotton, Andrew W Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 3, 2011. [11](#), [14](#), [65](#)
- [7] Lihao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. Hand pointnet: 3d hand pose estimation using point sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8417–8426. IEEE, 2018.

- [8] Liuhao Ge, Zhou Ren, and Junsong Yuan. Point-to-point regression pointnet for 3d hand pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 475–491, 2018. [14](#), [65](#)
- [9] Zhe Cao, Tomas Simon, ShihEn Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1310. IEEE, 2017. [14](#), [78](#)
- [10] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–682, 2018. [11](#), [14](#), [78](#)
- [11] Ivan Laptev. On space-time interest points. *International Journal of Computer Vision (IJCV)*, 64(2-3):107–123, 2005. [12](#)
- [12] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 650–663. Springer, 2008. [12](#)
- [13] Alireza Fathi and Greg Mori. Action recognition by learning mid-level motion features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. [12](#)
- [14] Zhang Zhang and Dacheng Tao. Slow feature analysis for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, (3):436–450, 2012. [12](#)
- [15] Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action recognition by dense trajectories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176. IEEE, 2011. [12](#)
- [16] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013.

- [17] Heng Wang, Alexander Kläser, Cordelia Schmid, and Cheng-Lin Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision (IJCV)*, 103(1):60–79, 2013. [12](#)
- [18] Alexander Klaser, Marcin Marszałek, and Cordelia Schmid. A spatio-temporal descriptor based on 3d-gradients. In *British Machine Vision Conference (BMVC)*, pages 275–1. British Machine Vision Association, 2008. [12](#)
- [19] Navneet Dalal, Bill Triggs, and Cordelia Schmid. Human detection using oriented histograms of flow and appearance. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 428–441. Springer, 2006. [12](#)
- [20] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM International Conference on Multimedia (ACM MM)*, pages 357–360. ACM, 2007. [12](#)
- [21] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. [12](#)
- [22] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1470–1477, 2003. [12](#)
- [23] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3304–3311. IEEE, 2010. [13](#)
- [24] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2007. [13](#)
- [25] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 143–156. Springer, 2010. [13](#)
- [26] Xinxiao Wu, Dong Xu, Lixin Duan, and Jiebo Luo. Action recognition using context and appearance distribution features. In *Proceedings of the IEEE*

- Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 489–496. IEEE, 2011. [13](#)
- [27] Piotr Bilinski and Francois Bremond. Video covariance matrix logarithm for human action recognition in videos. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015. [13](#)
- [28] Basura Fernando, Efstratios Gavves, Jose M Oramas, Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5378–5387, 2015. [13](#)
- [29] Bing Su, Jiahuan Zhou, Xiaoqing Ding, Hao Wang, and Ying Wu. Hierarchical dynamic parsing and encoding for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 202–217. Springer, 2016.
- [30] Kuan-Ting Lai, Felix X Yu, Ming-Syan Chen, and Shih-Fu Chang. Video event detection by inferring temporal instance labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2243–2250, 2014. [13](#)
- [31] Roberto Vezzani, Davide Baltieri, and Rita Cucchiara. Hmm based action recognition with projection histogram features. In *International Conference on Pattern Recognition (ICPR)*, pages 286–293. Springer, 2010. [13](#)
- [32] Cristian Sminchisescu, Atul Kanaujia, and Dimitris Metaxas. Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding (CVIU)*, 104(2-3):210–220, 2006. [13](#)
- [33] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. [13](#)
- [34] Amlan Kar, Nishant Rai, Karan Sikka, and Gaurav Sharma. Adascan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3376–3385, 2017. [13](#)

- [35] Basura Fernando, Peter Anderson, Marcus Hutter, and Stephen Gould. Discriminative hierarchical rank pooling for activity recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [14](#)
- [36] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 568–576, 2014. [14](#), [17](#)
- [37] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016. [14](#), [17](#)
- [38] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, 2015. [14](#)
- [39] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015. [14](#)
- [40] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015. [14](#)
- [41] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6299–6308, 2017.
- [42] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning spatio-temporal features with 3d residual networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3154–3160, 2017.

- [43] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6546–6555, 2018. [14](#), [92](#)
- [44] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1297–1304. IEEE, 2011. [14](#)
- [45] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1290–1297. IEEE, 2012. [15](#), [24](#), [41](#), [68](#)
- [46] Chunyu Wang, Yizhou Wang, and Alan L Yuille. An approach to pose-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 915–922. IEEE, 2013. [15](#), [41](#)
- [47] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. Sequence of the most informative joints (smij): A new representation for human skeletal action recognition. *Journal of Visual Communication and Image Representation (JVCI)*, 25(1):24–38, 2014. [15](#), [23](#), [33](#), [36](#), [41](#), [51](#), [68](#), [76](#)
- [48] Xiaodong Yang and Ying Li Tian. Eigenjoints-based action recognition using naive-bayes-nearest-neighbor. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 14–19. IEEE, 2012. [15](#)
- [49] Xiaodong Yang and YingLi Tian. Effective 3d action recognition using eigen-joints. *Journal of Visual Communication and Image Representation (JVCI)*, 25(1):2–11, 2014. [15](#)
- [50] Gang Yu, Zicheng Liu, and Junsong Yuan. Discriminative orderlet mining for real-time recognition of human-object interaction. In *Asian Conference on Computer Vision (ACCV)*, pages 50–65. Springer, 2014. [15](#), [41](#), [65](#), [68](#)
- [51] Chunyu Wang, Yizhou Wang, and Alan L Yuille. Mining 3d key-pose-motifs for action recognition. In *Proceedings of the IEEE Conference on Computer*

- Vision and Pattern Recognition (CVPR)*, pages 2639–2647, 2016. [15](#), [24](#), [33](#), [35](#), [41](#), [51](#)
- [52] Junwu Weng, Chaoqun Weng, and Junsong Yuan. Spatio-temporal naive-bayes nearest-neighbor (st-nbnn) for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4171–4180, 2017. [15](#), [41](#), [42](#), [61](#), [65](#), [68](#), [76](#)
- [53] Junwu Weng, Chaoqun Weng, Junsong Yuan, and Zicheng Liu. Discriminative spatio-temporal pattern discovery for 3d action recognition. *IEEE Transactions on Circuits and Systems for Video Technology (T-CSVT)*, 2018. [15](#), [68](#), [92](#)
- [54] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 588–595, 2014. [15](#), [16](#), [32](#), [33](#), [35](#), [51](#), [53](#)
- [55] Raviteja Vemulapalli and Rama Chellappa. Rolling rotations for recognizing human actions from 3d skeletal data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4471–4479, 2016. [15](#), [68](#)
- [56] Pei Wang, Chunfeng Yuan, Weiming Hu, Bing Li, and Yanning Zhang. Graph based skeleton motion representation and similarity measurement for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 370–385. Springer, 2016. [16](#), [32](#), [33](#), [35](#), [51](#), [68](#)
- [57] Piotr Koniusz, Anoop Cherian, and Fatih Porikli. Tensor representations via kernel linearization for action recognition from 3d skeletons. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–53. Springer, 2016. [16](#), [32](#), [33](#), [51](#)
- [58] Chunyu Wang, John Flynn, Yizhou Wang, and Alan L Yuille. Recognizing actions in 3d using action-snippets and activated simplices. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2016. [16](#), [26](#), [33](#), [35](#), [51](#)
- [59] Guillermo Garcia-Hernando and Tae-Kyun Kim. Transition forests: Learning discriminative temporal transitions for action recognition and detection.

- In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 432–440, 2017. [16](#), [68](#)
- [60] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118, 2015. [16](#), [23](#), [32](#), [33](#), [36](#), [40](#), [51](#), [52](#), [53](#), [57](#), [66](#), [68](#), [76](#)
- [61] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. [16](#), [23](#), [40](#), [49](#), [53](#), [56](#), [57](#), [66](#), [74](#), [75](#), [91](#), [93](#), [105](#), [106](#), [107](#)
- [62] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, Xiaohui Xie, et al. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 2, page 8, 2016. [16](#), [23](#), [40](#), [52](#), [65](#), [68](#)
- [63] Yanghao Li, Cuiling Lan, Junliang Xing, Wenjun Zeng, Chunfeng Yuan, and Jiaying Liu. Online human action detection using joint classification-regression recurrent neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 203–220. Springer, 2016. [16](#)
- [64] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 816–833. Springer, 2016. [17](#), [23](#), [24](#), [32](#), [33](#), [35](#), [36](#), [41](#), [51](#), [52](#), [53](#), [57](#), [66](#), [68](#), [69](#), [76](#), [78](#)
- [65] Jun Liu, Gang Wang, Ping Hu, Ling-Yu Duan, and Alex C. Kot. Global context-aware attention lstm networks for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [17](#), [53](#), [65](#), [68](#), [76](#)
- [66] Sijie Song, Cuiling Lan, Junliang Xing, Wenjun Zeng, and Jiaying Liu. An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 1, page 7, 2017. [17](#), [65](#)
- [67] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In

- Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 499–508, 2017. [17](#), [53](#), [57](#), [66](#), [68](#), [76](#), [78](#)
- [68] Wenbo Li, Longyin Wen, Ming-Ching Chang, Ser Nam Lim, and Siwei Lyu. Adaptive rnn tree for large-scale human action recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1444–1452, 2017. [17](#), [65](#), [76](#)
- [69] Inwoong Lee, Doyoung Kim, Seoungyoon Kang, and Sanghoon Lee. Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1012–1020, 2017. [17](#), [65](#), [68](#)
- [70] Junwu Weng, Mengyuan Liu, Xudong Jiang, and Junsong Yuan. Deformable pose traversal convolution for 3d action and gesture recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 142–157. Springer, 2018. [17](#)
- [71] Jun Liu, Amir Shahroudy, Gang Wang, Ling-Yu Duan, and Alex C Kot. Ss-net: Scale selection network for online 3d action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8349–8358, 2018. [17](#), [86](#)
- [72] Aäron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, 2016. [17](#)
- [73] Pichao Wang, Zhaoyang Li, Yonghong Hou, and Wanqing Li. Action recognition based on joint trajectory maps using convolutional neural networks. In *ACM International Conference on Multimedia (ACM MM)*, pages 102–106. ACM, 2016. [17](#), [41](#), [68](#)
- [74] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. A new representation of skeleton sequences for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [17](#), [53](#), [57](#), [68](#), [76](#)
- [75] Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *Pattern Recognition (PR)*, 68: 346–362, 2017. [17](#), [41](#), [53](#)

- [76] Mengyuan Liu and Junsong Yuan. Recognizing human actions as the evolution of pose estimation maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1159–1168, 2018. [17](#), [68](#)
- [77] Chaolong Li, Zhen Cui, Wenming Zheng, Chunyan Xu, and Jian Yang. Spatio-temporal graph convolution for skeleton based action recognition. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. [18](#)
- [78] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. [18](#)
- [79] Yansong Tang, Yi Tian, Jiwen Lu, Peiyang Li, and Jie Zhou. Deep progressive reinforcement learning for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5323–5332, 2018. [18](#)
- [80] Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. An attention enhanced graph convolutional lstm network for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1227–1236, 2019. [18](#)
- [81] Michael S Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1036–1043. IEEE, 2011. [19](#), [83](#), [86](#), [87](#)
- [82] Yu Cao, Daniel Barrett, Andrei Barbu, Siddharth Narayanaswamy, Haonan Yu, Aaron Michaux, Yuewei Lin, Sven Dickinson, Jeffrey Siskind, and Song Wang. Recognize human activities from partially observed videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2658–2665, 2013. [19](#)
- [83] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–704. Springer, 2014. [19](#), [86](#)

- [84] Kang Li and Yun Fu. Prediction of human activity by discovering temporal sequence patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36(8):1644–1657, 2014. [19](#)
- [85] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2863–2870. IEEE, 2012. [19](#)
- [86] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. *International Journal of Computer Vision (IJCV)*, 107(2):191–202, 2014. [19](#)
- [87] Yu Kong, Dmitry Kit, and Yun Fu. A discriminative model with multiple temporal scales for action prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 596–611. Springer, 2014. [19](#), [83](#), [86](#), [95](#)
- [88] Zhen Xu, Laiyun Qing, and Jun Miao. Activity auto-completion: Predicting human activities from partial videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3191–3199, 2015. [19](#)
- [89] Jian-Fang Hu, Wei-Shi Zheng, Lianyang Ma, Gang Wang, and Jianhuang Lai. Real-time rgb-d activity prediction by soft regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 280–296. Springer, 2016. [19](#), [20](#), [83](#), [86](#), [87](#)
- [90] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. [19](#)
- [91] Jian-Fang Hu, Wei-Shi Zheng, Lianyang Ma, Gang Wang, Jianhuang Lai, and Jianguo Zhang. Early action prediction by soft regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 2018. [20](#), [83](#), [86](#), [95](#)
- [92] Yu Kong, Shangqian Gao, Bin Sun, and Yun Fu. Action prediction from videos via memorizing hard-to-predict samples. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018. [20](#), [83](#), [86](#), [95](#)
- [93] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *Proceedings of the IEEE Conference*

- on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008. [21](#), [24](#), [25](#), [27](#), [40](#)
- [94] Hamed Pirsiavash, Deva Ramanan, and Charless C Fowlkes. Bilinear classifiers for visual recognition. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1482–1490, 2009. [22](#), [23](#), [27](#), [39](#), [46](#), [47](#)
- [95] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. Differential recurrent neural networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4041–4049, 2015. [23](#), [65](#)
- [96] Gang Yu, Zicheng Liu, and Junsong Yuan. Discriminative orderlet mining for realtime recognition of human-object interaction. In *Asian Conference on Computer Vision (ACCV)*, 2014. [24](#)
- [97] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Expandable data-driven graphical modeling of human actions based on salient postures. *IEEE Transactions on Circuits and Systems for Video Technology (T-CSVT)*, 18(11): 1499–1510, 2008. [24](#), [41](#)
- [98] Lu Xia, Chia-Chih Chen, and JK Aggarwal. View invariant human action recognition using histograms of 3d joints. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 20–27. IEEE, 2012. [24](#), [30](#), [34](#), [41](#), [49](#), [50](#), [105](#)
- [99] Xiaodong Yang and YingLi Tian. Super normal vector for activity recognition using depth sequences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 804–811. IEEE, 2014. [24](#), [41](#)
- [100] Samsu Sempena, Nur Ulfa Maulidevi, and Peb Ruswono Aryan. Human action recognition using dynamic time warping. In *International Conference on Electrical Engineering and Informatics (ICEEI)*, pages 1–5. IEEE, 2011. [24](#), [41](#)
- [101] Régis Behmo, Paul Marcombes, Arnak Dalalyan, and Véronique Prinet. Towards optimal naive bayes nearest neighbor. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 171–184. Springer, 2010. [24](#)
- [102] Tinne Tuytelaars, Mario Fritz, Kate Saenko, and Trevor Darrell. The nbnn kernel. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1824–1831. IEEE, 2011. [24](#)

- [103] Junsong Yuan, Zicheng Liu, and Ying Wu. Discriminative subvolume search for efficient action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2442–2449. IEEE, 2009. [24](#), [39](#), [40](#), [42](#), [44](#)
- [104] I. Kuzborskij, F. M. Carlucci, and B. Caputo. When Naive Bayes Nearest Neighbours Meet Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [24](#)
- [105] Matt Kusner, Stephen Tyree, Kilian Q Weinberger, and Kunal Agrawal. Stochastic neighbor compression. In *International Conference on Machine Learning (ICML)*, pages 622–630, 2014. [24](#), [59](#)
- [106] Deng Cai, Xiaofei He, Ji-Rong Wen, Jiawei Han, and Wei-Ying Ma. Support tensor machines for text categorization. 2006. [29](#), [47](#)
- [107] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. Action recognition based on a bag of 3d points. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 9–14. IEEE, 2010. [30](#), [32](#), [49](#), [50](#), [105](#)
- [108] Ferda Ofli, Rizwan Chaudhry, Gregorij Kurillo, René Vidal, and Ruzena Bajcsy. Berkeley mhad: A comprehensive multimodal human action database. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 53–60. IEEE, 2013. [30](#), [36](#), [49](#), [51](#), [74](#), [76](#), [105](#), [106](#)
- [109] David M Mount and Sunil Arya. Ann: library for approximate nearest neighbour searching. 1998. [31](#), [50](#)
- [110] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. [32](#), [50](#)
- [111] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. <http://cvxr.com/cvx>, March 2014. [32](#), [50](#)
- [112] Suraj Vantigodi and Venkatesh Babu Radhakrishnan. Action recognition from motion capture data using meta-cognitive rbf network classifier. In

- IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6. IEEE, 2014. [33](#), [36](#), [51](#), [76](#)
- [113] Ioannis Kapsouras and Nikos Nikolaidis. Action recognition on motion capture data using a dynemes and forward differences representation. *Journal of Visual Communication and Image Representation (JVCI)*, 25(6):1432–1445, 2014. [33](#), [36](#), [51](#), [76](#)
- [114] Ivan Lillo, Juan Carlos Niebles, and Alvaro Soto. A hierarchical pose-based approach to complex action understanding using dictionaries of actionlets and motion poselets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1981–1990, 2016. [41](#)
- [115] Chuankun Li, Yonghong Hou, Pichao Wang, and Wanqing Li. Joint distance maps based action recognition with convolutional neural networks. *IEEE Signal Processing Letters (SPL)*, 24(5):624–628, 2017. [41](#)
- [116] Kiwon Yun, Jean Honorio, Debaleena Chattopadhyay, Tamara L Berg, and Dimitris Samaras. Two-person interaction detection using body-pose features and multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 28–35. IEEE, 2012. [43](#), [49](#), [52](#), [105](#)
- [117] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2(331-340):2, 2009. [50](#)
- [118] Wenbo Li, Longyin Wen, Mooi Choo Chuah, and Siwei Lyu. Category-blind human action recognition: A practical recognition system. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4444–4452, 2015. [52](#)
- [119] Georgios Evangelidis, Gurkirt Singh, and Radu Horaud. Skeletal quads: Human action recognition using joint quadruples. In *International Conference on Pattern Recognition (ICPR)*, pages 4513–4518, 2014. [53](#), [75](#)
- [120] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. [55](#), [57](#)

- [121] Yong Du, Yun Fu, and Liang Wang. Skeleton based action recognition with convolutional neural network. In *IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 579–583. IEEE, 2015. [55](#)
- [122] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. [67](#), [71](#), [106](#)
- [123] SHI Xingjian, Zhouong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 802–810, 2015. [67](#), [71](#), [72](#)
- [124] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. [67](#), [93](#)
- [125] Zhou Ren, Junsong Yuan, and Zhengyou Zhang. Robust hand gesture recognition based on finger-earth mover’s distance with a commodity depth camera. In *ACM International Conference on Multimedia (ACM MM)*, pages 1093–1096. ACM, 2011. [68](#)
- [126] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Learning actionlet ensemble for 3d human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 36(5):914–927, 2014.
- [127] Hui Liang, Junsong Yuan, Daniel Thalmann, and Nadia Magnenat Thalmann. Ar in hand: Egocentric palm pose tracking and gesture recognition for augmented reality applications. In *ACM International Conference on Multimedia (ACM MM)*, pages 743–744. ACM, 2015.
- [128] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. Robust part-based hand gesture recognition using kinect sensor. *IEEE Transactions on Multimedia (T-MM)*, 15(5):1110–1120, 2013.
- [129] Quentin De Smedt, Hazem Wannous, and Jean-Philippe Vandeboer. Skeleton-based dynamic hand gesture recognition. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–9, 2016. [68](#), [74](#), [75](#), [106](#)

- [130] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. [69](#), [75](#)
- [131] Alex Graves. Supervised sequence labelling. In *Ph.D. Thesis*, pages 5–13. Springer, 2012. [74](#)
- [132] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014. [75](#), [93](#)
- [133] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3104–3112, 2014. [75](#)
- [134] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Skeleton-based action recognition with convolutional neural networks. In *International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 597–600. IEEE, 1997. [76](#)
- [135] Yu Kong and Yun Fu. Max-margin action prediction machine. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 38(9):1844–1858, 2016. [83](#), [87](#)
- [136] Yu Kong, Zhiqiang Tao, and Yun Fu. Deep sequential context networks for action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1473–1481, 2017. [86](#), [87](#), [94](#), [95](#)
- [137] Shugao Ma, Leonid Sigal, and Stan Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1942–1950, 2016. [83](#), [95](#)
- [138] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. [90](#)

-
- [139] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 409–419, 2018. [91](#), [92](#), [94](#), [107](#)
- [140] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *CRCV-TR-12-01*, 2012. [91](#), [107](#)
- [141] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. [92](#)