Deep Progressive Reinforcement Learning for Skeleton-based Action Recognition

Yansong Tang[1,2,3,*]   Yi Tian[1,*]   Jiwen Lu[1,2,3]   Peiyang Li[1]   Jie Zhou[1,2,3]

[1]Department of Automation, Tsinghua University, China
[2]State Key Lab of Intelligent Technologies and Systems, Tsinghua University, China
[3]Beijing National Research Center for Information Science and Technology, China

{tys15,tianyi15,lipy15}@mails.tsinghua.edu.cn   {lujiwen,jzhou}@mail.tsinghua.edu.cn

Abstract

In this paper, we propose a deep progressive reinforcement learning (DPRL) method for action recognition in skeleton-based videos, which aims to distil the most informative frames and discard ambiguous frames in sequences for recognizing actions. Since the choices of selecting representative frames are multitudinous for each video, we model the frame selection as a progressive process through deep reinforcement learning, during which we progressively adjust the chosen frames by taking two important factors into account: (1) the quality of the selected frames and (2) the relationship between the selected frames to the whole video. Moreover, considering the topology of human body inherently lies in a graph-based structure, where the vertices and edges represent the hinged joints and rigid bones respectively, we employ the graph-based convolutional neural network to capture the dependency between the joints for action recognition. Our approach achieves very competitive performance on three widely used benchmarks.

1. Introduction

Action recognition is an important research direction in computer vision, which has worldwide applications, such as video surveillance, human-robot interaction and so on. Compared with the conventional RGB videos, the skeleton-based sequences contain compact 3D positions of the major body joints, which are robust to variations of viewpoints, body scales and motion speeds [1]. Thus, skeleton-based action recognition has attracted more and more attention in recent years [2–8].

With the development of the cost-effective depth sensors (e.g. Kinect) and pose estimation algorithms [9], the amount of skeleton-based data is growing rapidly [10, 11]. Therefore data-driven methods have been increasingly proposed for skeleton-based action recognition, by training
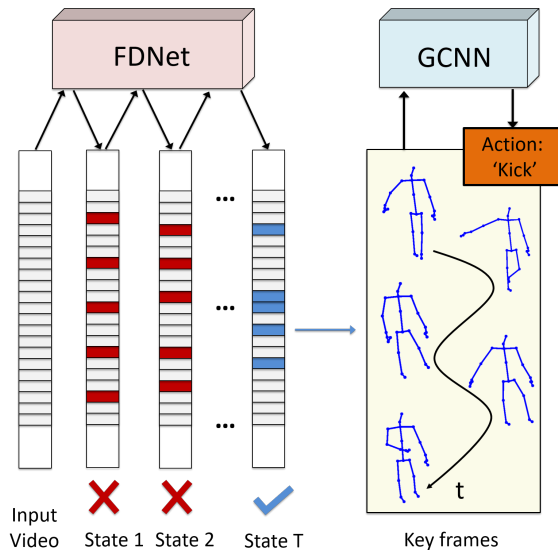

Figure 1. The pipeline of our proposed method for skeleton-based action recognition in the testing period. Given a video of human body joints, we first select key frames with a frame distillation network (FDNet), which is trained by the proposed deep progressive reinforcement learning method. Then, we employ a graph-based convolutional neural network (GCNN), which retains the dependency between human joints, to deal with the selected key frames for action recognition. (Best viewed in color)

deep models like recurrent neural networks (RNN) [12–14] and convolutional neural networks (CNN) [15, 16]. The RNN-based model has the capability to model the temporal dependency, but it is difficult to train the stacked RNN in practice [15, 17]. On the other hand, the CNN-based model, which captures the relationship of neighboring frames at lower layers and long-term dependency at higher layers [18], is more effective and obtains promising performance recently [15, 16]. However, most CNN-based methods for skeleton-based action recognition consider all of the frames in a sequence as equally important, which fails to focus on the most representative frames. Take a video of action

---

'kick' as an example, there are some frames in which the subject stands upright, as well as other frames showing the subject kicks out the leg. The latter are more informative for recognizing this action.

In order to seek the most informative frames of a sequence, we propose a deep progressive reinforcement learning (DPRL) method. Since the choices of selecting different frames are multitudinous for each video, we model the procedure of selecting frames as a progressive process. Specifically, given the initialised frames which are uniformly sampled from an input sequence, we progressively adjust the chosen frames at each state according to two important factors. One is the discriminative power of the selected frames for action recognition. The other is the relationship of the selected frames to the whole action sequences. The final selected frames are considered as the distillation of the video, and are employed to recognize actions. Moreover, most CNN-based methods adopt the Euclidean structure to model the articulated joints, which ignore the intrinsic topology of human bodies. To address this, we model the joints and their dependency as a graph. The vertices of the graph contain the 3D coordinates of the body joints, while the adjacency matrix captures their relationship. Since the graph of joints lies in a non-Euclidean space, we leverage the graph-based convolutional neural network (GCNN) to learn the spatial dependency between the joints. We evaluate our approach on three skeleton-based action recognition datasets, where the competitive experimental results demonstrate the effectiveness of our approach.

2. Related Work

Skeleton-based Action Recognition: There have been a number of skeleton-based action recognition methods in recent years [12–15, 19–24], and they can be mainly classified into two categories: *hand-crafted feature based* and *deep learning feature based*. For the first category, Vemulapalli *et al.* [6] represented the human skeleton as a point in the Lie group, and implemented temporal modelling and classification in the Lie algebra. Weng *et al.* [8] extended Naive-Bayes Nearest-Neighbor (NBNN) method [25] to Spatio-Temporal-NBNN, and employed the stage-to-class distance to classify actions. Koniusz *et al.* [26] presented two kernel-based tensor representations to capture the compatibility between two action sequences and the dynamic information of a single action. Wang *et al.* [5] proposed an undirected complete graph representation, and presented a new graph kernel to measure the similarity between graphs. However, the graph representation in [5] is used to model the video, while the graph in our work is adopted to capture the topology of human body. The *Deep learning feature based methods* can be further divided into CNN-based model and RNN-based model. For the CNN-based model, Ke *et al.* [15] presented a new representation of skele-

ton sequences based on the cylindrical coordinates. Liu *et al.* [16] transformed the skeletons into a series of color images and fed them into the CNN architecture to classify action category. Motivated by [27], Li *et al.* [20] employed a two-stream CNN architecture to combine the position and velocity information of human joints. Different from these CNN-based methods in which all the frames are treated equally, our method aims to find the most informative frames of the video for action recognition. For the RNN-based model, Zhu *et al.* [12] introduced a regularized LSTM model for co-occurrence feature learning. Song *et al.* [13] proposed a spatio-temporal attention model to allocate different weights to different frames and joints in the video. Liu *et al.* [14] presented a trust gate module to address the noise in skeletal data. Recently, Jain *et al.* [28] combined RNN with the spatio-temporal graph, modelling the relationship of three parts (*i.e.* spine, arm and leg) for human motion. Different from [28], our graph model takes every joint of human body as a vertex, which is a finer way for utilizing the skeleton-based data.

Deep Reinforcement Learning: Reinforcement learning [29] is originated from the psychological and neuro-scientific understandings of how humans learn to optimize their behaviors in an environment. It can be mathematically formulated as a Markov decision process (MDP) [30]. With a person being generalized to an agent, the behaviors being generalized to a set of **actions***, a typical reinforcement learning problem can be formulated as an agent optimizes its policy of **actions** by maximizing the numerical rewards it receives from an environment. As a pioneering work, Mnih *et al.* [31] combined recent advances in deep neural networks. They proposed deep reinforcement learning (DRL) to bridge the divide between high-dimensional sensory inputs and **actions**, and achieved human-level control in Atari games. Computer vision has also benefited from DRL in recent years. For example, Mnih *et al.* [32] proposed the Recurrent Attention Model, in which the visual fixations of an image is modelled as a sequential MDP. Haque *et al.* [33] applied DRL to person identification, Yeung *et al.* [34] to action detection, and Jie *et al.* [35] to object detection. More recently, Yun *et al.* [36] employed DRL for visual tracking and Rao *et al.* [37] for face recognition. To date, little progress has been made in DRL for action recognition, especially skeleton-based action recognition. [34,37] are similar to our work in the purpose of DRL, *i.e.* selecting key frames in videos. However, in both works the **actions** affected only one single frame. More specifically, [37] decided whether to drop a frame or not and [34] selected one frame at each step, while we deal with the adjustments of all the selected frames at one time.

---

*There are two types of actions in this paper, *i.e.,* the actions to be recognized and the **actions** in the Markov decision process (MDP). For clarity, We use the bold-sized word to represent the **actions** of MDP.
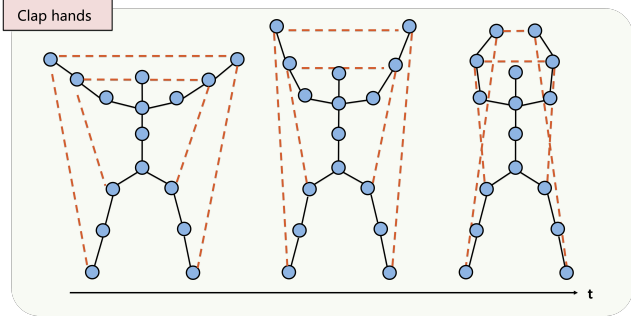
Figure 2. **Modelling human body as a graph.** The vertices of the graph are presented as the blue dots, which contain the 3D coordinates of human joints, while the edges reflect the relationships between joints, which can be categorized as *intrinsic* dependencies (*i.e.* physical connection) and *extrinsic* dependencies (*i.e.* physical disconnection). Take the action 'clap hand' as an example, the *intrinsic* dependency is suggested as black solid lines, while the *extrinsic* dependency is represented as orange dashed lines. We set different parameters in the weighted adjacency matrix to distinguish these two types of dependencies. For simplicity, we only draw several important lines.

3. Approach

Figure 1 illustrates the pipeline of our proposed model. Specifically, there are two sub-networks in our method: frame distillation network (FDNet) and graph-based convolutional network (GCNN). The FDNet aims to distil a fixed number of key frames from the input sequences with a deep progressive reinforcement learning method. Then, we organize the outputs of the FDNet into a graphical structure based on the dependencies between human joints, and feed them into the GCNN to recognize the action label. As the GCNN provides rewards for the FDNet during the training process, we first introduce the GCNN part as follows.

3.1. Graph-based Representation Learning

Graph Construction: Since the human body can be considered as an articulated system consisting of hinged joints and rigid bones, which inherently lies in a graphed-based structure, we construct a graph $\mathcal{G}(x, W)$ to model the human body for each single frame, where $x \in \mathbb{R}^{N \times 3}$ contains the 3D coordinates of the N joints and W is a $N \times N$ weighted adjacency matrix:

$$w_{ij} = \begin{cases} 0, & \text{if } i = j \\ \alpha, & \text{if joint } i \text{ and joint } j \text{ are connected} \\ \beta, & \text{if joint } i \text{ and joint } j \text{ are disconnected} \end{cases} \quad (1)$$

Here, we set $w_{ii} = 0$ to discard the self connection of each joint. Moreover, we distinguish the relationship between joints as *intrinsic* dependency and *extrinsic* dependency. The intrinsic dependency, which is described as α in the weighted matrix W and suggested as the black solid

lines in Figure 2, refers to the physical connection of joints. As an important property, the distance between each pair of connected joints keeps invariant during the action process. The extrinsic dependency, as the orange dashed lines show in Figure 2, refers to the disconnected relationship of two joints, which is also an important factor during the action process. For example, the left hands and right hands are disconnected physically, but their relationship presents significant importance for recognizing the action 'clap hands'. Here, we use the parameter β in W to model the extrinsic relationship.

Graph-based Architecture: The GCNN can be regarded as a modification of the conventional CNN, aiming to deal with the graph-based data which lies in non-Euclidean space. Given a video with T frames, we first construct each frame into a graph according to Eqn.1 as $[\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T]$. For each graph $\mathcal{G}_t$ at tth frame, we first feed it into the graph-based convolutional layer as:

$$z_t = y(\eta, W) * x_t \quad (2)$$

where $y(\eta, W)$ and * are the kernel and operator of the graph-based convolutional [38] respectively and will be detailed later. We then feed z_t into a fully-connected layer, the output of which is denoted as g_t. For $t = 1, 2, ..., T$, we concatenate g_t in the time axis and obtain a feature map G for the input video:

$$G = concat[g_1, g_2, .., g_T] \quad (3)$$

where G is a 3D tensor, which is finally sent into a conventional CNN for action recognition. We adopt the categorical cross-entropy loss to train the GCNN.

Graph-based Convolution: The graph-based convolutional layer is the core module in this network. We consider the graph Laplacian [39] on the spectral domain with the normalized definition: $L = I_n - D^{-1/2} W D^{-1/2}$, where D is the diagonal degree matrix with $d_{ii} = \sum_j w_{ij}$. We scale L as $\widetilde{L} = 2L/\lambda_{max} - I_n$ and denote $\overline{x}_k = T_k(\widetilde{L}) * x$, where λ_{max} is the maximised eigen value of L and T_k is the Chebyshev polynomial [40]. Then, the convolutional operator can be formulated as [38]:

$$y(\eta, W) * x = \eta[\overline{x}_0, \overline{x}_1 ..., \overline{x}_{K-1}]^T \quad (4)$$

Here, $\eta \in [\eta_0, \eta_1 ..., \eta_{K-1}]$ are the parameters to be trained, and K is the size of the graph-based convolutional kernel.

3.2. Deep Progressive Reinforcement Learning

For the task of action recognition in skeletal videos, not every frame is of equal temporal importance. This is the key insight to our application of reinforcement learning-based attention. The selection of key frames is formulated as a Markov decision process (MDP) [41], based on which we
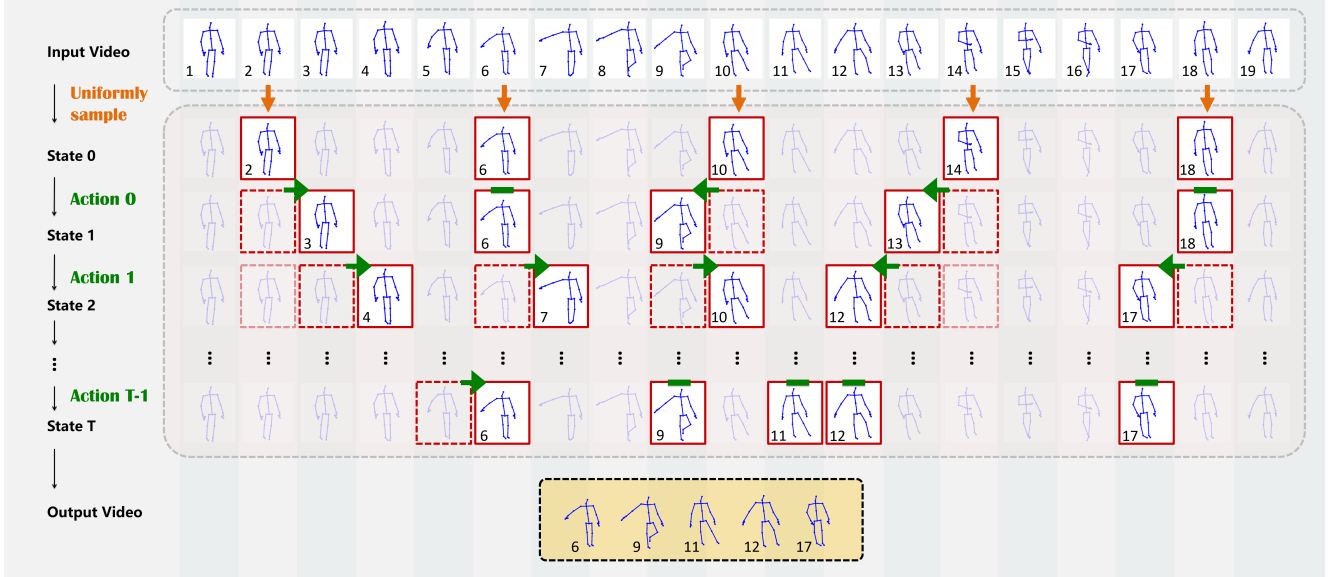
Figure 3. **Process of selecting key frames in skeleton-based videos progressively.** Given a skeleton-based sequence, we first uniformly sample several frames. After the progressive adjustment, we obtain the most informative frames of the videos. Each state contains the information of selected frames, their indices and the whole sequence. Actions, which are obtained by the states and FDNet, denote the direction of 'shifting to left', 'staying the same' or 'shifting to right' at the next step for each selected frame.

use reinforcement learning to refine the frames at each iteration. Figure 3 provides a sketch map of this process, which is implemented based on FDNet as shown in Figure 4. The agent, interacting with an environment that provides rewards and updates its state, learns by maximizing the total discounted reward to adjust the selected frames, finally resulting in a given number m of the most distinguishable frames. The states, **actions** and rewards of this MDP are elaborated below.

States: The state S of the MDP consists of two separate parts $\{S_a, S_b\}$. $S_a = [F, M]$, which is the concatenation of two tensors F and M. F consists of the global information of a skeletal video, which is a tensor with the shape of $f \times N \times 3$. Here, f, N and 3 denote the numbers of frames, joints and axes respectively. For the videos that are not exactly f frames long, we use bicubic interpolation [42] to derive a video of f frames in which the first and last frame are the same as the original one. Similar to F, M is a $m \times N \times 3$ tensor, representing the information of the m selected frames. M is introduced to *implicitly* provide the FDNet with knowledge about which frames of the video are selected. S_b, the binary mask of the selected indices, is designed to *explicitly* make the FDNet aware of the selection. It is an f-dimensional vector with m elements being 1 and the rest being 0. Here we set f to be 100 and m to be 30.

Actions: The **action**, *i.e.* the output of the FDNet, is the adjustment direction of each selected frame. We define 3 types of **action** as 'shifting to left' (**action** 0), 'staying the same' (**action** 1) and 'shifting to right' (**action** 2), and

shifting step is set to be 1 frame. As shown in Figure 4, the FDNet emits a vector $A \in R^{m \times 3}$ at each iteration, where $A_{i,j} \in [0, 1]$ represents the probability of choosing action j for the ith selected frame. To ensure the order of the m frames, for example the 1st selected frame should always be temporally earlier than the 2nd selected one, we set the upper bound of the frame adjustment Υ_i $(i = 1, 2, ..., m)$ to be the middle between one frame and the frame next to it in the selected frame set:

$$\Upsilon_i = \begin{cases} \lceil (M_i + M_{i+1})/2 \rceil & , 1 \le i \le m-1 \\ f & , i = m \end{cases} \quad (5)$$

where '$\lceil \rceil$' represents the ceil function. Similarly, the lower bound Λ_i $(i = 1, 2, ..., m)$ is set to be the middle between the current frame and the previous one.

$$\Lambda_i = \begin{cases} \lceil (M_{i-1} + M_i)/2 \rceil & , 2 \le i \le m \\ 0 & , i = 1 \end{cases} \quad (6)$$

Here Υ and Λ are two arrays of size m. The adjustment of a frame i is executed within the bound $[\Lambda_i, \Upsilon_i)$, or otherwise invalidated. Then, the frame adjustments can be written as:

$$M_i' = M_i + \delta_i \quad (7)$$

where

$$\delta_i = \begin{cases} -\min\{1, (M_i - \Lambda_i)\} & , \text{if } \textbf{action } 0 \\ 0 & , \text{if } \textbf{action } 1 \\ \min\{1, (\Upsilon_i - M_i - 1)\} & , \text{if } \textbf{action } 2 \end{cases} \quad (8)$$
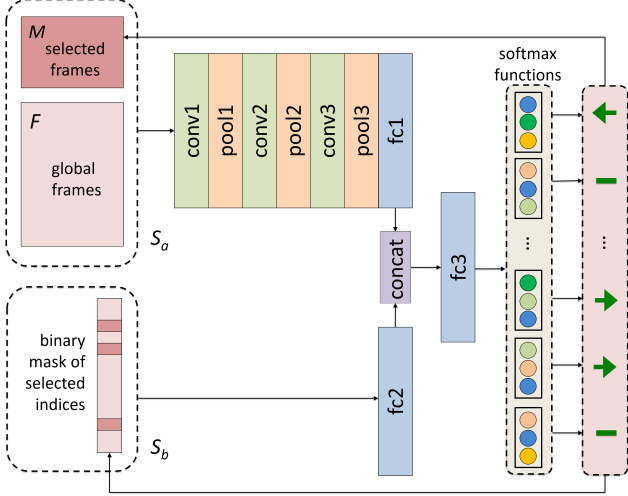
Figure 4. **The FDNet architecture for adjusting the key frames in skeleton-based videos.** The FDNet takes the input of S_a and S_b separately, where S_a contains the information of the whole video F as well as the selected frames M, and S_b is an f - dimension binary mask of the selected indices with f elements being 1 and the rest being 0. Then, S_a is processed by a CNN of 3 convolutional layers with the kernel size 3×3 and a fully connected layer (fc1), while S_b is passed through fc2. The extracted features of these two parts are concatenated before they are fed into fc3. Softmax functions are then employed to regularize the output of fc3. The output is a set of **actions**, which direct the refining process at the next step.

In this way, the action will make influence on the state transition.

Rewards: The reward, as a function $r(S, A)$, reflects how good the **action** taken by the agent is with regard to the state S. We generate the reward with the pre-trained GCNN, which takes the m selected frames of a video as input (we set $T = m$). For the first iteration, r is set to be 1 if the prediction is correct, and -1 otherwise. For the nth $(n > 1)$ iteration, we first define the r_0 reward as follows:

$$r_0 = sgn(P_{n,c} - P_{n-1,c}) \tag{9}$$

where c is the ground truth label of the video, and $P_{n,c}$ represents the probability of predicting the video as class c at the nth iteration. The reward r_0 takes value in $\{-1, 1\}$, reflecting the predicted possibility improvement of the ground-truth action, *i.e.* the aggregated predicted possibility fall of the other actions. We choose this function to enhance the rewards by probability change and it is shown better than numeric values from experimental results. Besides, a strong stimulation of $r = \Omega$ is enforced when the predicted action turns from incorrect to correct after one iteration, and a strong punishment of $r = -\Omega$ if the turning goes otherwise. Thus, the final form of the reward r when

$n > 1$ can be written as:

$$r = \begin{cases} \Omega & \text{, if stimulation} \\ -\Omega & \text{, if punishment} \\ r_0 & \text{, otherwise} \end{cases} \tag{10}$$

Progressive Reinforcement: Figure 4 presents the architecture of our FDNet $\mathbf{FD}(S; \theta)$, which contains three convolutional layers and three fully connected layers. It predicts the optimal **action** when fed with the state in the form of S, which is initialized by uniformly sampling. The two parts of S are sent into the FDNet separately, as S_a is fed into a convolutional network followed by one fully connected layer and S_b is fed into a fully connected layer. Then, the outputs of the two fully connected layers are concatenated and fed through the third fully connected layer. Finally, m softmax layers are adopted to produce A for **actions**.

In order to maximize the discounted reward $R = \sum_{t \geq 0} \gamma^t r_t$, we compute the cross-entropy loss as follows:

$$l(\theta) = -\frac{1}{m} \sum_{t=1}^{m} log(\pi_\theta(S_t, A_t)) \tag{11}$$

This loss term gives the direction of updating the parameters θ. We normalize R to be $\widetilde{R}$, which plays the role of strengthening this gradient descent. Thus, θ is updated by

$$\theta_{i+1} = \theta_i + \nabla l(\theta) \widetilde{R} \tag{12}$$

The pipeline of our DPRL is summarized in **Algorithm 1**.

The training of DRL problems can generally be categorized into two branches: deep Q-learning and policy gradient. As defined above, our **action** set consists of the different choices of adjusting the m selected frames. There are 3 **actions** for each selected frame, the *exponential* size 3^m is computationally infeasible for deep Q-learning. Thus, we employ the policy gradient method which requires only a *linear* increase of the output dimension.

3.3. Combination of GCNN and FDNet

For all of the skeleton-based videos in the training set, we first sample their frames uniformly to obtain the sequences in fixed size. These sequences are used to train the GCNN to capture joint dependencies in the *spatial* domain. Then, we fix the parameters in GCNN to train FDNet and update the selected frames for each video in the *temporal* domain, which are used to refine the GCNN. These two models promote each other mutually, as GCNN provides rewards for FDNet and FDNet selects key frames for refining GCNN. The better GCNN is, the more accurate rewards will be provided. The higher quality the selected frames have, the better GCNN can be refined. At the test time, each video goes through the FDNet to produce its corresponding sequence with the informative frames, which will be finally sent into the GCNN to provide the action label.

Algorithm 1: DPRL

Input: Training videos V, label l, GCNN model $\mathbf{G}$.
Output: Weights θ of FDNet $\mathbf{FD}$
initialise θ
for $epoch \leftarrow 1, 2, ..., E$ **do**
 for V_i *in* V **do**
  uniformly select frames M_1 from V_i
  initialise S_1 with M_1
  for $t \leftarrow 1,2, ... , \tau$ **do**
   use S_t to generate $A_t = \mathbf{FD}(S_t; \theta)$
   choose the **action** w.r.t. A_t
   update the selected frames to M_{t+1} by (7)
   update the state to S_{t+1}
   compute the reward r_t using $\mathbf{G}, l$ by (10)
  end
  compute the loss $l(\theta)$ by (11)
  compute the normalized total reward $\widetilde{R}$
  update θ by (12)
 end
end
return θ

4. Experiments

We conducted experiments on three widely used datasets to evaluate our proposed DPRL method, and compared it with state-of-the-art skeleton-based action recognition approaches as well as the baseline methods. The following describes the details of the experiments and results.

4.1. Datasets and Experiment Settings

NTU+RGBD Dataset (NTU) [22]: This is the currently largest dataset for action recognition with more than 56 thousand sequences and 4 million frames. The dataset was captured from 40 different human subjects and has 60 classes actions. We use the 3D skeleton data of 25 major body joints. The benchmark evaluations include Cross-Subject (CS) and Cross-View (CV) setting. In the Cross-Subject evaluation, 40320 samples from 20 subjects were used for training and the other 16540 samples were for testing. In Cross-View evaluation, the 37,920 samples captured from camera 2 and 3 were used for training, while the other 18960 samples from camera 1 were for testing.

SYSU-3D Dataset (SYSU) [43]: The SYSU-3D dataset contains 480 sequences and 12 different actions performed by 40 persons. The 3D coordinates of 20 joints are associated with each frame of the sequence. We employed the videos performed by 20 subjects for training, and the sequences captured from the rest 20 subjects for testing. We adopted 30-fold cross-validation and show the mean accuracy on this dataset.

UT-Kinect Dataset (UT) [44]: This dataset includes 200 skeleton sequences with 20 skeleton joints per frame. There are 10 types of actions and each of them is performed by 10 subjects twice. We adopted the leave-one-out cross-validation protocol to evaluate our method on this dataset [45].

Baseline Methods: We organized each video as a $T \times N \times 3$ tensor, where T represents the uniformly sampled frames, N is the number of the body joints, and 3 denotes the 3D coordinates of joints. We empirically set T to be 30, and N is equal to 25, 20 and 20 for NTU, SYSU and UT respectivly. Then, a CNN-based model with 3 convolutional layers and 3 fully-connected layers is employed to recognize the actions. The kernel sizes of the 3 convolutional layers were 3×3 and the numbers of channels were 32, 64 and 128. We adopted 3 max pooling layers with the size of 2×2 after each convolutional layer. The dimension of 3 fully connected layers were 256, 128 and C (the number of action category). In order to demonstrate the effectiveness of DPRL and module of graph-based representation learning, we report the results on the baseline model (*i.e. Ours-CNN*) as well as our proposed methods (*i.e. Ours-GCNN*, *Ours-DPRL* and *Ours-DPRL+GCNN*) for each dataset respectively. Here, *Ours-DPRL* stands for adopting the DPRL to select frames rather than uniformly sampling frames, while *Ours-DPRL+GCNN* represents replacing the baseline model with GCNN architecture while DPRL is employed.

Implementation Details: Our proposed method was implemented with the Tensorflow [46] and Keras [47] toolbox, while the network architecture was built on two Nvidia GTX 1080 GPUs. The two sub-networks were both trained from scratch. For the GCNN, we chose ELUs [48] as the activation functions and set the dropout rate to 0.5. The kernel size of the graph-based convolutional layer was set to be 5, and the batchsize was set to be 64, 16, 8 for NTU, SYSU and UT dataset respectively. In terms of constructing the adjacent weight matrix, we set $\alpha = 5$ and $\beta = 1$, which highlights the intrinsic dependency and retained the extrinsic dependency. We employed Adam [49] to train the whole network with the initial learning rate 10^{-3}. In order to deal with the condition of two people in NTU dataset, we adopted the maxout scheme [50] as suggested in [20]. We did not perform any rotations and normalization for skeleton data during pre-processing.

For the FDNet model, the structure of which is shown in Figure 4, we set the dropout rate to be 0.5, chose ReLUs as the activation functions, and utilized Adam optimizer to train the FDNet with the learning rate 10^{-5}. The **actions** were selected stochastically with the corresponding probability $A_{i,j}$. We empirically set the number of adjustment iterations τ of a video to be 7 and the parameter Ω in Eqn.10 to be 25, so that Ω was greater than $\tau \times |r_0|$ to perform the strong simulation/punishment.
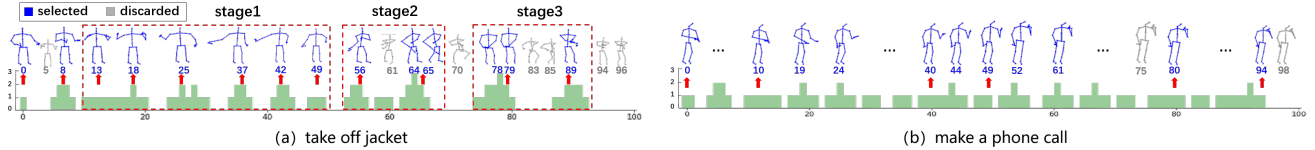
Figure 5. **Visualizations of the selected results.** The horizontal axis denotes the frame index, while the vertical axis represents the number of frames selected in the neighbourhood corresponding to the index.

Table 1. Comparisons of action recognition accuracy (%) on the NTU dataset. The GCNN, GCNN[1] and GCNN[2] stand for different adjacency matrices used for graph construction: GCNN for $\alpha = 5$, $\beta = 1$, GCNN[1] for $\alpha = 1$, $\beta = 0$, GCNN[2] for $\alpha = 1$, $\beta = 1$.

Method	CS	CV	Year
Dynamic Skeletons [43]	60.2	65.2	2015
HBRNN-L [24]	59.1	64.0	2015
Part-aware LSTM [22]	62.9	70.3	2016
ST-LSTM+Trust Gate [51]	69.2	77.7	2016
STA-LSTM [13]	73.4	81.2	2017
LieNet-3Blocks [21]	61.4	67.0	2017
Two-Stream RNN [19]	71.3	79.5	2017
Clips+CNN+MTLN [15]	79.6	84.8	2017
VA-LSTM [23]	79.2	87.7	2017
View invariant [16]	80.0	87.2	2017
Two-Stream CNN [20]	83.2	89.3	2017
LSTM-CNN [52]	82.9	91.0	2017
Ours-CNN	79.7	84.9	
Ours-GCNN	81.1	87.0	
Ours-DPRL	82.3	87.7	
Ours-DPRL+GCNN[1]	82.5	88.1	
Ours-DPRL+GCNN[2]	82.8	88.9	
Ours-DPRL+GCNN	**83.5**	**89.8**	

4.2. Results on NTU+RGBD Dataset

The results on the NTU dataset show that, our method achieves the performance of 83.5% (CS) and 89.8% (CV) respectively. For the DPRL, our model achieves 2.6% (CS) and 2.8% (CV) improvement over the baseline method (*ours-CNN*), while the graph-based module (*ours-GCNN*) brings 1.4% (CS) and 2.1% (CV) improvement, which shows the effectiveness of our proposed method.

Comparison with the State-of-the-arts: Table 1 presents the comparison performance with the state-of-the-arts. We see that, our method is superior to other state-of-the-art approaches except LSTM-CNN method [52]. This is because [52] combines 3 LSTM and 7 CNN to reach the higher performance, while our model only requires training the two CNN-based models and is easier to implement. Compared with the soft attention model [13], our method achieves 10.1% (CS) and 8.6% (CV) improvement.

Analysis on DPRL: We analyze the results of the selected frames in Figure 5. As mentioned, our key insight is that the different temporal significances of the frames in
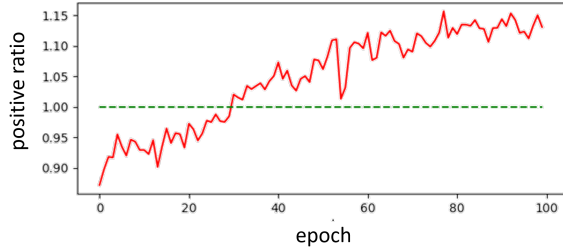

Figure 6. **DPRL training process.** This figure demonstrates that, with training, the positive ratio gradually becomes stably above the level of 1, which demarcates the effectiveness of the progressive refinement process.

a given video can be estimated progressively by deep reinforcement learning. Thus, in the climax of a video, the frames should be selected more frequently than the trivial parts. In Figure 5, the horizontal axis represents the frame index and the vertical axis represents the number of frames selected in the neighbourhood of the index, different actions should correspond to different shapes of the curve. We also present several selected skeleton frames in blue and discarded frames in grey, while their corresponding indices are under the skeletons. The action of (a) is 'take off jacket', which mainly contains three stages: 1) pulling the jacket off the back, 2) pulling the jacket off the forearm, and 3) putting the jacket on the forearm. The first stage has a larger range of motion and lasts for a longer period, while the second and the third stages are relatively gentle and quick. Note that, though the jacket may cause some occlusion and noise, our DPRL can discard these inferior frames. The action label of (b) is 'make a phone call' and there is no part of particular significance. The selected frames first show the process of taking on the phone, after which the subject remains the pose.

Moreover, Figure 6 shows the process of DPRL training, where the horizontal axis is the number of training epochs, the vertical axis refers to the positive ratio, which is defined as follows. After an iteration of progressive refinement, the prediction of a video can go from 'incorrect' to 'correct', which is called a positive result, the opposite being called a negative result. In an epoch, the number of positive results is denoted by n_{pos} and the number of negative results is denoted by n_{neg}. We define the positive ratio as the ratio of n_{pos} to n_{neg}, *i.e.* $\frac{n_{pos}}{n_{neg}}$. Obviously, the positive ratio being

Table 2. Comparisons of action recognition accuracy (%) on the SYSU dataset.

Method	Acc.	Year
LAFF(SKL) [53]	54.2	2016
Dynamic Skeletons [43]	75.5	2015
ST-LSTM(Tree) [51]	73.4	2017
ST-LSTM(Tree)+Trust Gate [51]	76.5	2017
Ours-CNN	75.5	
Ours-GCNN	76.0	
Ours-DPRL	76.7	
Ours-DPRL+GCNN	**76.9**	

Table 3. Comparisons of action recognition accuracy (%) on the UT dataset.

Method	Acc.	Year
Grassmann Manifold [54]	88.5	2015
Histogram of 3D Joints [44]	90.9	2012
Riemannian Manifold [55]	91.5	2015
ST-LSTM+Trust Gate [51]	97.0	2016
GMSM [5]	97.4	2016
SCK+DCK [26]	98.2	2016
ST-NBNN [8]	98.0	2017
VA-LSTM [23]	99.5	2017
Ours-CNN	96.0	
Ours-GCNN	97.0	
Ours-DPRL	98.0	
Ours-DPRL+GCNN	**98.5**	

1 represents no change of total classification accuracy. Figure 6 demonstrates the effectiveness of our progressive refinement method. Typically, there are approximately 3,000 positive results in an epoch.

Analysis on GCNN: The graph-based representation learning mines the dependency between body joints. Experimental results in Table 1 demonstrate that the graph-based module improves the performance. We tried different parameters for α and β in our experiments, which are regarded as different weights we allocate to the intrinsic dependency and extrinsic dependency. In the table, $GCNN^1$ ignores the extrinsic dependency, $GCNN^2$ attaches equal importance to these two types of dependencies, and $GCNN$ highlights the intrinsic dependency while retains the extrinsic dependency in the meantime. As the results demonstrate, $GCNN$ performs the best and $GCNN^2$ performs better than $GCNN^1$. We can conclude from the results that, both types of dependencies make contributions to action recognition, and the intrinsic dependency is more crucial. However, there is still room for further improving the results by carefully and dynamically adjusting the allocated weights.

4.3. Results on SYSU-3D Dataset

We compare our method with the state-of-the-art skeleton-based action recognition methods, which are presented in Table 2. As is seen, our proposed method outperforms all the other state-of-the-art methods on this dataset. We also find that the proposed DPRL method can help to improve the baseline classification accuracy by 1.2%, while combining the graph-based representation learning model can lead to another 0.2% improvement. This shows the effectiveness of selecting the key frames and learning the spatial dependency in the two modules.

4.4. Results on UT-Kinect Dataset

The performance comparison with the state-of-the-art methods on the UT dataset is presented in Table 3. We also find that the proposed DPRL and graph-based model can lead to 2% and 1% improvement respectively, which demonstrate the effectiveness of our proposed methods again. Further more, we discover that our proposed method

outperforms all the other state-of-the-art methods except VA-LSTM [23]. The reason is that the VA-LSTM benefits a lot from the view adaptation sub-network, which is specially designed to recognize actions in variant views. In the UT dataset, such conditions are common. On the other hand, we outperform GMSM [5] by 1.1%, where another graphical model is also employed.

5. Conclusion

In this paper, we have proposed a deep progressive reinforcement learning (DPRL) method for action recognition in skeleton-based videos, which aims to select the most informative frames of the input sequences. Moreover, we employ a graph-based deep learning model to capture both the intrinsic and extrinsic dependencies between human joints. Our approach achieves very competitive performance on the widely used NTU, SYSU and UT datasets. In the future, it is promising to apply our method to other related computer vision tasks like video summarization and event detection. Moreover, as our GCNN employs hand-crafted parameters for the graph adjacency matrix, it is desirable to explore some learning-based methods to design the weights.

Acknowledgement

References

[1] Fei Han, Brian Reily, William Hoff, and Hao Zhang. Space-time representation of people based on 3d skeletal data: A review. *CVIU*, 158:85–105, 2017. 1

[2] Lu Xia, Chia-Chih Chen, and J. K. Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPR*, pages 20–27, 2012. 1

[3] Jiang Wang, Zicheng Liu, Ying Wu, and Junsong Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, pages 1290–1297, 2012. 1

[4] Mohammad Abdelaziz Gowayyed, Marwan Torki, Mohamed Elsayed Hussein, and Motaz El-Saban. Histogram of oriented displacements (HOD): describing trajectories of human joints for action recognition. In *IJCAI*, pages 1351–1357, 2013. 1

[5] Pei Wang, Chunfeng Yuan, Weiming Hu, Bing Li, and Yanning Zhang. Graph based skeleton motion representation and similarity measurement for action recognition. In *ECCV*, pages 370–385, 2016. 1, 2, 8

[6] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *CVPR*, pages 588–595, 2014. 1, 2

[7] Chun-yu Wang, Yizhou Wang, and Alan L. Yuille. Mining 3d key-pose-motifs for action recognition. In *CVPR*, pages 2639–2647, 2016. 1

[8] Junwu Weng, Chaoqun Weng, and Junsong Yuan. Spatio-temporal naive-bayes nearest-neighbor for skeleton-based action recognition. In *CVPR*, pages 4171–4180, 2017. 1, 2, 8

[9] 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 *CVPR*, pages 1297–1304, 2011. 1

[10] Michael Firman. RGBD datasets: Past, present and future. In *CVPRW*, pages 661–673, 2016. 1

[11] Jing Zhang, Wanqing Li, Philip O. Ogunbona, Pichao Wang, and Chang Tang. Rgb-d-based action recognition datasets: A survey. *PR*, 60:86–105, 2016. 1

[12] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *AAAI*, pages 3697–3703, 2016. 1, 2

[13] 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*, pages 4263–4270, 2017. 1, 2, 7

[14] Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *ECCV*, pages 816–833, 2016. 1, 2

[15] Qiuhong Ke, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. A new representation of skeleton sequences for 3d action recognition. In *CVPR*, 2017. 1, 2, 7

[16] Mengyuan Liu, Hong Liu, and Chen Chen. Enhanced skeleton visualization for view invariant human action recognition. *PR*, 68:346–362, 2017. 1, 2, 7

[17] Razvan Pascanu, Caglar Gulehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. In *ICLR*, 2014. 1

[18] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. Convolutional sequence to sequence learning. In *ICML*, pages 1243–1252, 2017. 1

[19] Hongsong Wang and Liang Wang. Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks. In *CVPR*, pages 499–508, 2017. 2, 7

[20] Chao Li, Qiaoyong Zhong, Di Xie, and Shiliang Pu. Skeleton-based action recognition with convolutional neural networks. In *ICMEW*, 2017. 2, 6, 7

[21] Zhiwu Huang, Chengde Wan, Thomas Probst, and Luc Van Gool. Deep learning on lie groups for skeleton-based action recognition. In *CVPR*, pages 16–145, 2017. 2, 7

[22] Amir Shahroudy, Jun Liu, Tian Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *CVPR*, pages 1010–1019, 2016. 2, 6, 7

[23] Pengfei Zhang, Cuiling Lan, Junliang Xing, Wenjun Zeng, Jianru Xue, and Nanning Zheng. View adaptive recurrent neural networks for high performance human action recognition from skeleton data. In *ICCV*, pages 1110–1118, 2017. 2, 7, 8

[24] Yong Du, Wei Wang, and Liang Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, pages 1110–1118, 2015. 2, 7

[25] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *CVPR*, pages 1–8, 2008. 2

[26] Piotr Koniusz, Anoop Cherian, and Fatih Porikli. Tensor representations via kernel linearization for action recognition from 3d skeletons. In *ECCV*, pages 37–53, 2016. 2, 8

[27] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576, 2014. 2

[28] Ashesh Jain, Amir Roshan Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, pages 5308–5317, 2016. 2

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 2

[30] Michael L Littman. Reinforcement learning improves behaviour from evaluative feedback. *Nature*, 521(7553):445–451, 2015. 2

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 2

[32] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *NIPS*, pages 2204–2212, 2014. 2

[33] Albert Haque, Alexandre Alahi, and Li Fei-Fei. Recurrent attention models for depth-based person identification. In *CVPR*, pages 1229–1238, 2016. 2

[34] Serena Yeung, Olga Russakovsky, Greg Mori, and Li Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, pages 2678–2687, 2016. 2

[35] Zequn Jie, Xiaodan Liang, Jiashi Feng, Xiaojie Jin, Wen Lu, and Shuicheng Yan. Tree-structured reinforcement learning for sequential object localization. In *NIPS*, pages 127–135, 2016. 2

[36] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. Action-decision networks for visual tracking with deep reinforcement learning. In *CVPR*, pages 2711–2720, 2017. 2

[37] Yongming Rao, Jiwen Lu, and Jie Zhou. Attention-aware deep reinforcement learning for video face recognition. In *ICCV*, pages 3931–3940, 2017. 2

[38] Michal Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016. 3

[39] Fan Chung. Spectral graph theory. page 212, 1997. 3

[40] David K. Hammond, Pierre Vandergheynst, and Rmi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. 3

[41] Richard Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 6(4):15, 1957. 3

[42] Rafael C Gonzalez and Richard E Woods. Book on digital image processing, 2005. 4

[43] Jian Fang Hu, Wei Shi Zheng, Jianhuang Lai, and Jianguo Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. In *CVPR*, pages 5344–5352, 2015. 6, 7, 8

[44] Lu Xia, Chia Chih Chen, and J. K. Aggarwal. View invariant human action recognition using histograms of 3d joints. In *CVPRW*, pages 20–27, 2012. 6, 8

[45] Mohamed Hussein, Marwan Torki, Mohammad Gowayyed, and Motaz El-Saban. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. In *IJCAI*, pages 2466–2472, Beijing, China, August 2013. 6

[46] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 6

[47] Franois Chollet. Keras. https://github.com/fchollet/keras, 2015. 6

[48] Djork-Arn Clevert, Thomas Unterthiner, Sepp Hochreiter, Djork-Arn Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *ICLR*, 2016. 6

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Computer Science*, 2014. 6

[50] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. Maxout networks. In *ICML*, pages 1319–1327, 2013. 6

[51] Jun Liu, Amir Shahroudy, Dong Xu, Alex C. Kot, and Gang Wang. Skeleton-based action recognition using spatio-temporal lstm network with trust gates. *TPAMI*, 2017. 7, 8

[52] Chuankun Li, Pichao Wang, Shuang Wang, Yonghong Hou, and Wanqing Li. Skeleton-based action recognition using lstm and cnn. In *ICMEW*, 2017. 7

[53] Jianfang Hu, Wei-Shi Zheng, Lianyang Ma, Gang Wang, and Jian-Huang Lai. Real-time RGB-D activity prediction by soft regression. In *ECCV*, pages 280–296, 2016. 8

[54] Rim Slama, Mohamed Daoudi, Mohamed Daoudi, and Anuj Srivastava. Accurate 3d action recognition using learning on the grassmann manifold. *PR*, 48(2):556–567, 2015. 8

[55] M Devanne, H Wannous, S Berretti, P Pala, M Daoudi, and Bimbo A Del. 3-d human action recognition by shape analysis of motion trajectories on riemannian manifold. *TCYB*, 45(7):1340, 2015. 8