Early Action Recognition with Category Exclusion using Policy-based Reinforcement Learning

Junwu Weng, *Member, IEEE,* Xudong Jiang, *Senior Member, IEEE,* Wei-Long Zheng, *Member, IEEE* and Junsong Yuan, *Senior Member, IEEE*

***Abstract*—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 paper, 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.**

***Index Terms*—Category Exclusion, Early Action Recognition, Policy-based Reinforcement Learning**

I. 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.

Junwu Weng, Xudong Jiang are with School of Electrical & Electronics Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. (e-mail: we0001wu@e.ntu.edu.sg, exdjiang@ntu.edu.sg).

Wei-Long Zheng is with the Department of Neurology, Massachusetts General Hospital, Harvard Medical School, Boston, MA 02114, USA. (e-mail: wzheng8@mgh.harvard.edu).

Junsong Yuan is with Department of Computer Science & Engineering, The State University of New York, Buffalo, NY 14260-2500, USA. (e-mail: jsyuan@buffalo.edu).

Manuscript received – –, 2019; revised – –, 2019.

To date, the existing methods [16, 2, 3, 15, 1, 13, 14, 4] 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 paper is one step, and the first attempt, towards this direction.

In this paper, 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.1 illustrates

the framework of the proposed method. During testing, the pre-trained classifier outputs classification probability of all categories based on its observation. At each observation ratio, the agent checks its input *state* and generates binary masks to mask out the classification probabilities of the interfering negative categories, i.e., category exclusion.

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.

II. Related Work

Human action analysis is an important topic in computer vision community. To well understand human behaviors through videos, basic tasks like action recognition, early action recognition need to be considered and solved.

A. 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 category label. This task has been widely explored in the community. Considering that action recognition is a basic classification problem, the visual representation of videos becomes a critical part in research, and it attracts heated attention from researchers.

Traditional methods focus on how to design discriminative hand-crafted features for action description. In image-based action recognition, the representative approaches include Cuboids [19, 20], 3D Historgram of Gradient (3D HoG) [22], Space-Time Interest Point (STIP) [21], 3D SIFT [23], and Dense Trajectory [24, 25] *etc.*, which model the spatio-temporal property of action sequence into the representation. With the development of deep learning, a few deep-learning-based models are proposed for action recognition. Three dimensional convolution neural network (3D CNN) [65, 27, 28, 68] is one of the representative models. In 3D CNN, the two dimensional convolution operator in 2D CNN is extended to extract spatio-temporal information for action sequences, which achieves satisfactory recognition performance. Another promising action recognition structure is the Two-streams model [31, 30], in which two CNNs are involved to extract the appearance and motion features of action sequences respectively. Another aspect of action recognition in deep learning is the Temporal Convolution Network [58] (TCN) for temporal modeling of action sequence. After the TCN, a few variants of it are proposed. The idea of deformable convolution kernel is introduced in Deformable TCN [60] to model optimal temporal receptive field. The Dense TCN is proposed in [57] to capture human actions in hierarchical views.

In skeleton-based action recognition, the input data is a sequence of human 3D poses. Each pose consists of the three-dimensional coordinates of body joints. The state-of-the-arts can be separated into two groups, the traditional models [33, 34, 36, 37, 38, 9, 40, 39, 41, 42, 43], and the deep-learning-based models [44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]. 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. Given the limited complexity of the classifier, these methods 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. Due to the tremendous amount of these works, we limit the review to a few representative works.

In SMIJ [39], 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 an action. Weng *et. al.* proposed ST-NBNN [38] to discover the key joints and key frames of pose sequences, in which the well-known NBNN in image classification is extended to solve sequence classification task. The ST-NBNN model is further developed in ST-NBMIM [9] to achieve more discriminative action representation. The rolling rotation of pose motion dynamics is a variant feature for skeleton-based action recognition. In [41], the a rolling rotation representation is proposed for 3D skeletal data. Before the Graph Convolution Neural Network (G-CNN) is widely discussed in this community, Wang *et. al.* [43] proposed a traditional graph-based pose motion representation for 3D action recognition, which fully utilizes the topological structure of 3D human pose as a graph.

The deep-learning-based models can be further categorized as the RNN-based models, and the CNN-based models. The RNN-based models mainly use the recurrent neural network to model the temporal dynamics of action sequence. Song *et. al.* proposed a Spatio-Temporal Attention LSTM (STA-LSTM) [48] for 3D action recognition, in which two sub-LSTMs are involved for spatial and temporal attention dynamics on temporal domain. However, in [44], the LSTM is re-designed to construct the spatio-temporal dependency of pose joints in action sequence. The convolution neural network is applied in action recognition task in three different ways. In Deformable Pose Traversal Convolution Network [32], one-dimensional convolutional operator is used for pose feature extraction, which is spatial modeling. In [18], one-dimensional convolution network [18] is involved to extract temporal features of partial action sequence for temporal modeling. In [54], 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. G-CNN based models like [56] fully utilize the property that human body pose is a undirected acyclic graph to extract features from pose-based action sequence, and these models achieve promising results.

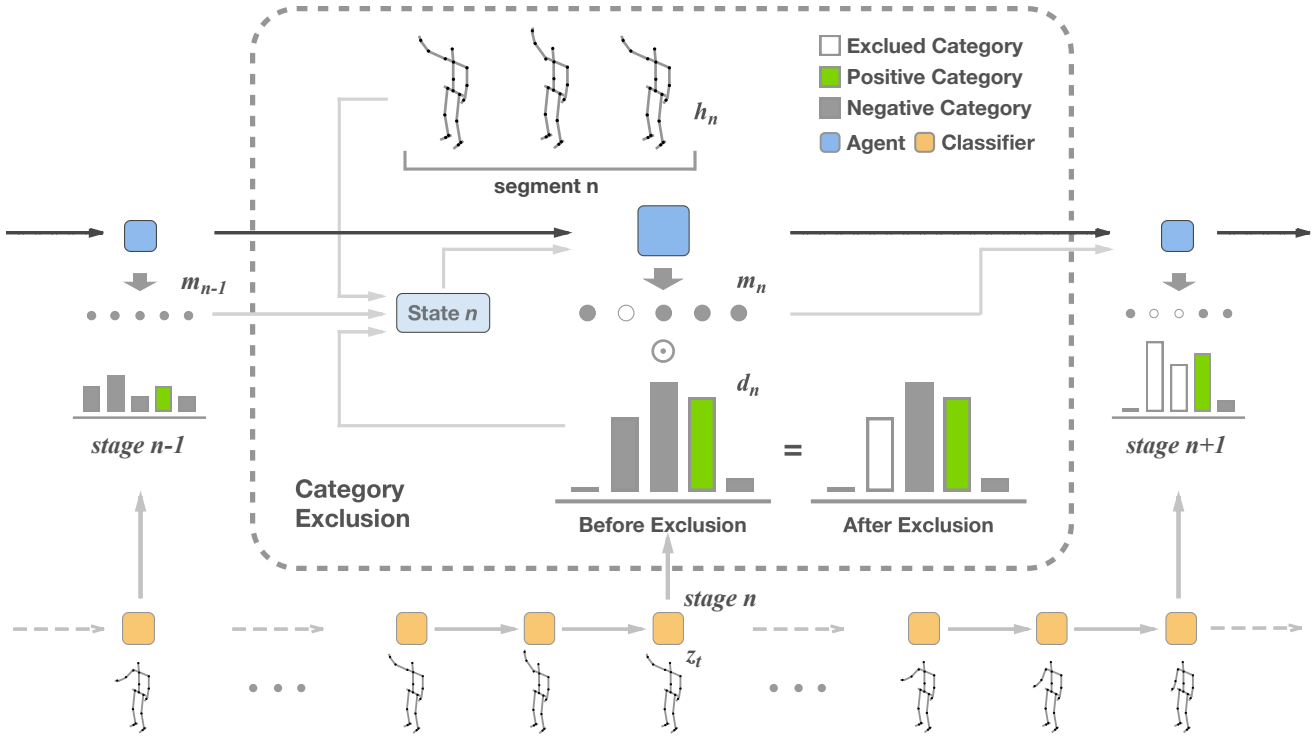


Fig. 1. Illustration of category exclusion in early action recognition. Frames of the action sequence are fed into the LSTM classifier (orange squares) one by one to extract features and output action classification probability (gray, white, and green bars). The action classification probability, the mask from previous stage, and the features from local segment are further fed into the agent to generate masks for category exclusion.

B. Early Action Recognition

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.* [4] propose to involve the well-known integral bag-of-words and dynamic bag-of-words into the action prediction task. Kong et al. [16] and Lan *et al.* [17] introduce the Support Vector Machine (SVM) into the early recognition. A soft regression framework is proposed in [1] 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 [13], 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) [3]. In Hierarchical LSTM [61], an adaptive clip mining strategy is introduced for better online visual representation learning. The Long Short-Term Memory (LSTM) model is re-designed in [15] to memorize hard-to-predict samples for early action recognition. Considering the changing scale of partial actions, a Scale Selection Network is proposed in [18] 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.

III. 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.1. The formulation of this proposed method for the early action recognition task is first introduced in Sec.III-A. Then we describe how to model the category exclusion in action recognition in Sec.III-B. The training of the category exclusion agent is detailed in Sec.III-C. 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.

A. Problem Formulation

Given a pre-trimmed sequence $V = \{z_t\}_{t=1}^{T}$ with its sequence-level label $c \in \mathbb{C} = \{1, 2, ..., C\}$, 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 $z_{1:t}$, and provide the accurate recognition as early as possible. Following the existing setting in [2, 3, 1, 4], 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 $\boldsymbol{x}_n$ as the feature extracted from the first n segments, and therefore $\boldsymbol{x}_N$ is the representation for the whole sequence. With these notations, our task can be formulated based on maximum a posteriori (MAP) rule,

$$c_n^* = \arg\max_{c \in \mathbb{C}(\boldsymbol{s}_n)} p(c|\boldsymbol{x}_n). \tag{1}$$

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* $\boldsymbol{s}_n$ observed by the agent. As defined in Eq.(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* $\boldsymbol{s}_n$ as input and decides which categories to be excluded, and the classifier will predict action category from the modified category set $\mathbb{C}(\boldsymbol{s}_n)$.

B. 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 $f : \boldsymbol{x}_n \rightarrow \boldsymbol{d}_n \in [0,1]^C$ involved to estimate the probability of classification in Eq.(1). $\boldsymbol{d}_n$ consists of the estimated probability of each action category, namely $d_n^c = p(c|\boldsymbol{x}_n)$. To exclude categories, a corresponding binary vector $\boldsymbol{m}_n \in \{0,1\}^C$ is generated at each observation ratio o_n to mask out some negative categories (dimensions) of $\boldsymbol{d}_n$. Each element of $\boldsymbol{m}_n$ 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.(1) can be rewritten as

$$c_n^* = \arg\max_{c \in \mathbb{C}} \; p(c|\boldsymbol{x}_n)\, m_n^c = \arg\max_{c \in \mathbb{C}} \; d_n^c\, m_n^c. \tag{2}$$

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 $\boldsymbol{h}_n$ extracted from the partially observed video, the current classification probability $\boldsymbol{d}_n$, and the previous generated exclusion mask $\boldsymbol{m}_{n-1}$, i.e., $\boldsymbol{s}_n = (\boldsymbol{h}_n,\ \boldsymbol{d}_n,\ \boldsymbol{m}_{n-1})$. 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 $g : \boldsymbol{s} \rightarrow \boldsymbol{u} \in \{0,1\}^C$, which is approximated by a neural network $\hat{g} : \boldsymbol{s} \rightarrow \boldsymbol{v} \in [0,1]^C$. 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 $\hat{g}^c(\boldsymbol{s})$ 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 $\boldsymbol{u}$ is given by

$$\pi(\boldsymbol{s}) = \prod_{c=1}^{C} \hat{g}^c(\boldsymbol{s})^{u^c} \left(1 - \hat{g}^c(\boldsymbol{s})\right)^{1-u^c}. \tag{3}$$

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

$$\boldsymbol{m}_n = \boldsymbol{m}_{n-1} \odot \boldsymbol{u}_n, \tag{4}$$

where $\odot$ is an element-wise multiplication operation. Since $\boldsymbol{u}_n$ is a binary mask, the accumulated mask $\boldsymbol{m}_n$ 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 $\boldsymbol{r} \in \mathbb{R}^C$ is defined as

$$r_n^c = \begin{cases} -\operatorname{sign}(m_n^c)\ \eta, & if \quad c = c^+ \\ \operatorname{sign}(m_n^c)\ d_n^c, & if \quad c \neq c^+, \end{cases} \tag{5}$$

where η is a constant encouraging the agent to maintain the positive category c^+. The function sign(x) is a sign function introduced to steer the direction between encouragement and punishment of chosen *action*. The function sign(x) returns 1 when x equals to 0, and returns -1 when x equals to 1. With the immediate reward defined in Eq.(5), the accumulated reward $\boldsymbol{R} \in \mathbb{R}^C$ is defined as $\boldsymbol{R}_n = (\sum_{i=n}^{N} \boldsymbol{r}_i)/(N-n+1)$, which is the average value of the immediate rewards from current stage n to the end.

C. Agent Training

The goal of the agent training in our proposed method is to learn a 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 alogrithm REINFORCE [6] to train the agent and maximize the accumulated rewards. During agent training, the agent generates binary masks $\boldsymbol{u}_n$ 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

$$\begin{aligned}\mathcal{L}_n^r &= -\log\Big(\pi(\boldsymbol{s}_n)\Big)\mathcal{R}_n \\ &= -\sum_{c=1}^{C}\Big[u_n^c \log(\hat{g}_n^c) + (1-u_n^c)\log(1-\hat{g}_n^c)\Big]\mathcal{R}_n, \end{aligned} \tag{6}$$

where the $\hat{g}_n^c$ is short for $\hat{g}^c(\boldsymbol{s}_n)$. 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 $\pi(\boldsymbol{s}_n)$ to the related *action* $\boldsymbol{u}_n$.

While in our method, as we described in Sec.III-B, 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.(7).

$$\mathcal{L}_n^r = -\sum_{c=1}^{C}\Big[u_n^c \log(\hat{g}_n^c) + (1-u_n^c)\log(1-\hat{g}_n^c)\Big]\tilde{R}_n^c, \tag{7}$$

We normalize $\boldsymbol{R}$ to be $\tilde{\boldsymbol{R}}$ here to strengthen the gradient descent. For the reward set $\{\boldsymbol{R}_n\}_{n=1}^N$ 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 $\{\boldsymbol{d}_n\}_{n=1}^N$ of every sample
2) Feature $\{\boldsymbol{h}_n\}_{n=1}^N$ 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 $\boldsymbol{m}_0 \leftarrow \mathbf{1}$
    for $n \leftarrow 1$ **to** N **do**
      get $\boldsymbol{d}_n$ and $\boldsymbol{h}_n$ from sample V_k
      explore $\boldsymbol{u}_n$ with $\hat{g}(\boldsymbol{d}_n, \boldsymbol{m}_{n-1}, \boldsymbol{h}_n)$
      update $\boldsymbol{m}_n$ using $\boldsymbol{u}_n$ by Eq.(4)
      obtain reward $\boldsymbol{r}_n$ using $\boldsymbol{m}_n$ and $\boldsymbol{d}_n$ by Eq.(5)
    end
    compute the normalized reward $\tilde{\boldsymbol{R}}$ by $\{\boldsymbol{r}_n\}_{n=1}^N$
    compute the loss $\mathcal{L}^r$ by Eq.(7)
    update agent $\hat{g}(\cdot)$
  end
end

IV. Experiment

In this section, we evaluate the proposed method on three action benchmark datasets, the NTU-RGBD dataset [5], the First-Person Hand Action (FPHA) dataset [7] as well as the UCF-101 dataset [8], and compare its performance to the existing methods. We briefly introduce the datasets in Sec.IV-B. The implementation details are described in Sec.IV-A. Comparison experiments and analysis on the three datasets are discussed in Sec.IV-C. The evaluation results show the efficiency of the proposed method.

A. Implementation

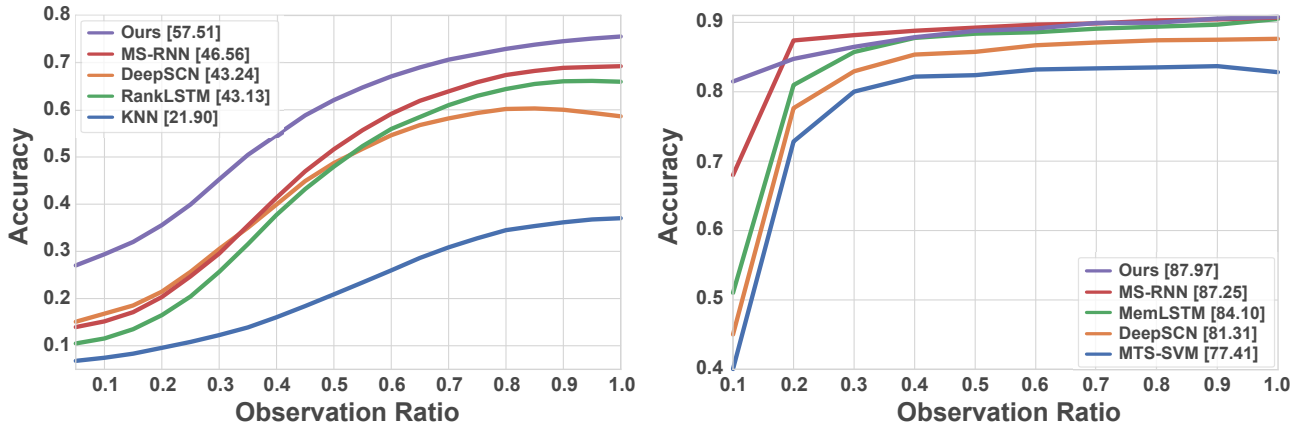
Features: There are two models in the proposed method, the baseline classifier and the agent for category exclusion. We use the 3D pose data in the NTU-RGBD and the First-Person Hand Action dataset. We follow the pre-processing method described in [9, 7] to process the pose data. For the training of $f(\cdot)$ on the 3D datasets, we directly use the raw pose as the input feature $\boldsymbol{z}_t$. The accumulated hidden feature bear in LSTM is regarded as feature $\boldsymbol{x}_n$ 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 $\boldsymbol{z}_t$ in w_n are concatenated together and fed into 3D ResNeXt-101 network [68] pre-trained on Kinetics dataset [10] for visual features $\bar{\boldsymbol{z}}_n$.

The trained classifier $f(\cdot)$ is further used to generate classification probability $\boldsymbol{d}_n$ 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, $\boldsymbol{h}_n$ is the concatenated raw pose data from local window w_n. For UCF-101, the feature $\boldsymbol{h}_n$ for agent is the same as $\bar{\boldsymbol{z}}_n$.

To inform the agent to be aware of the exclusion situation in the previous one stage, we add one more information $\bar{\boldsymbol{d}}_n = \boldsymbol{d}_n \odot \boldsymbol{m}_{n-1}$ to the classification probability part of the input *state*. Considering that the input to the agent consists of

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

TABLE I
EARLY RECOGNITION ACCURACY COMPARISON WITH BASELINE ON TWO SETTINGS (CS & CV) OF NTU-RGBD, FPHA AND UCF-101 DATASETS (%)



(a) NTU-RGBD (CS) (%) (b) UCF-101 (%)

Fig. 2. Comparison of early recognition curves. The recognition curve of our proposed method is marked by Purple.

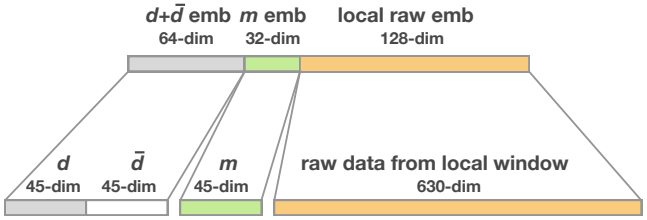


Fig. 3. Illustration of linear feature embedding layers (FPHA)

multiple parts, ($\boldsymbol{h}_n$, $\boldsymbol{d}_n$, $\boldsymbol{m}_{n-1}$, $\bar{\boldsymbol{d}}_n$), 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.3, 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.3. The embedded version of different parts are then concatenated together and fed into the agent.

Training: Without loss of generality, we use LSTM [11] as the sequential classifier the agent cooperates with. The sequential classifier can be any model including Graph Convolutional Neural Network [56, 64, 63], 3D Convolutional Neural Network [65, 67, 66, 68] once these models are well-designed for sequential classification. The LSTM classifier is trained with frame-level label supervision, and optimized by Adam [12]. The output of $f(\cdot)$ is further processed by Softmax function to have classification probability. For NTU dataset, $f(\cdot)$ is a three-layers 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-layers 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 $\boldsymbol{u}$ defined in Sec.III-B. 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.

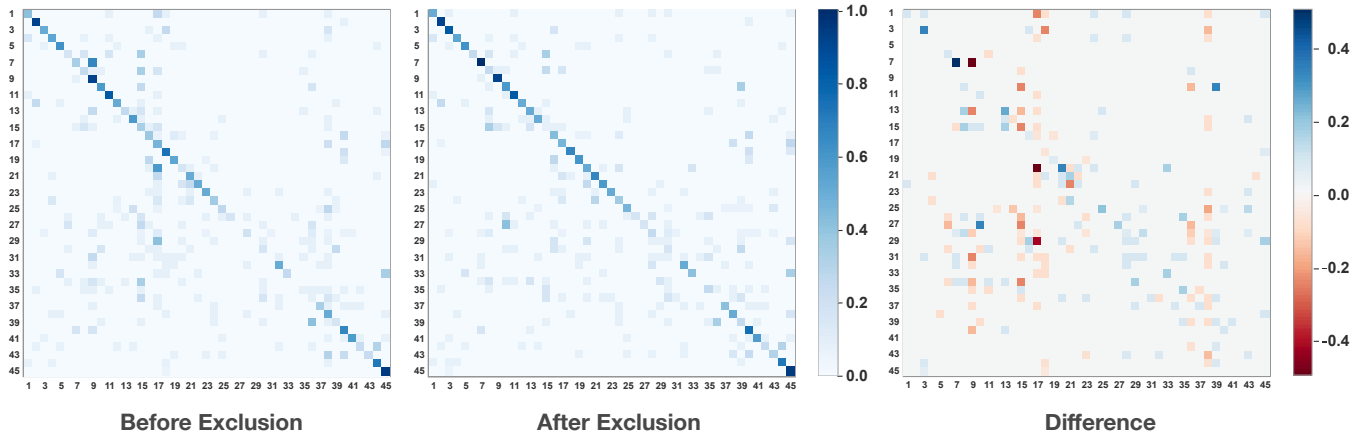


Fig. 4. The confusion matrices before and after exclusion and their difference at the observation ratio 5% (FPHA). The difference matrix is the difference between the confusion matrices after and before exclusion. The negative values are marked by red and the positive values are marked by blue in this difference matrix.

B. 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 [5] 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 [7] 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 [3] 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.

C. Results and Analysis

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

We compare the early action recognition performance of the proposed method with the baseline on the three datasets, the NTU-RGBD dataset, the FPHA dataset and the UCF-101 dataset. The comparison results are shown in TableI with different observation ratios. The notations 'w/o Exc.' and 'w/ Exc.' are classification without Category Exclusion and classification with Category Exclusion respectively. The 'w/o Exc.' is the baseline. 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 (w/o Exc.) 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.

We also compare the performance of the proposed method with the existing methods, MS-RNN [13], DeepSCN [3], RankLSTM [14], MemLSTM [15], and MTS-SVM [16] 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.2, 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

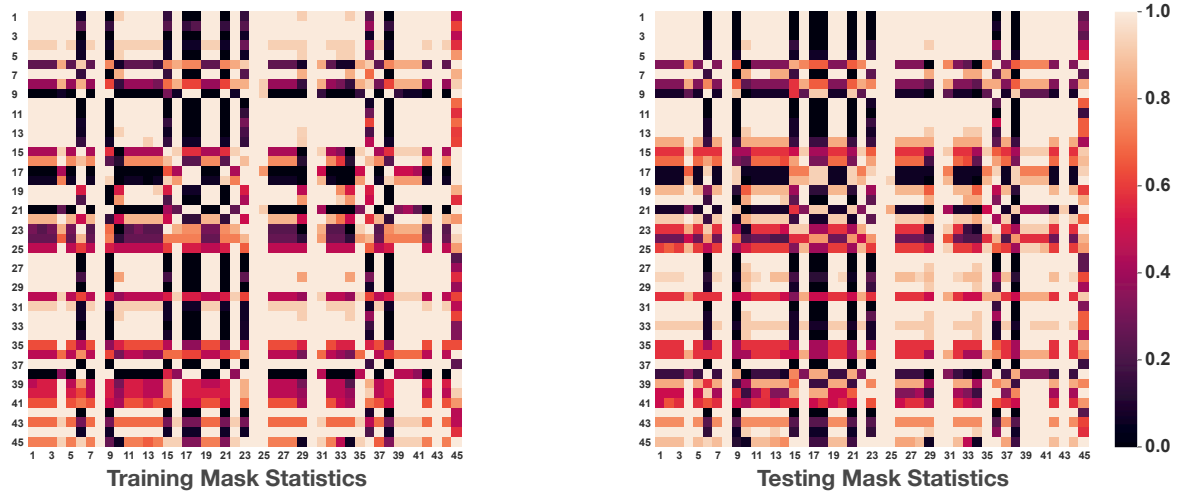


Fig. 5. Mask Statistics on FPHA. Each row of the matrix is the average mask of the binary mask over all input videos of the corresponding category. Each dimension of each row shows the exclusion situation of that related category.

Observation Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	AUC
w/o Exc.	45.91	54.26	61.39	63.30	65.74	69.22	70.61	72.17	73.39	74.43	64.11
w/ Exc.-None	46.43	54.61	61.74	63.13	65.22	68.35	69.74	71.83	72.87	74.09	63.85
w/ Exc.-Hid	51.83	58.61	64.00	64.87	66.78	69.22	70.43	71.65	72.35	73.22	65.46
w/ Exc.-Local	52.00	59.65	64.52	65.91	67.65	70.43	72.00	**73.57**	**74.26**	**74.96**	66.66
w/ Exc.-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 II
EARLY RECOGNITION ABLATION STUDY ON FPHA (%)

and after exclusion as well as their difference in Fig.4. The 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.5. 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.(4), we use the accumulated mask from the final stage as the exclusion mask to show the exclusion pattern of a action sequence. Each row of the matrix is the mean mask over all the input videos of the corresponding 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.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.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 $\boldsymbol{d}_n$ and the generated mask $\boldsymbol{m}_{n-1}$, the agent also need information from observation $\boldsymbol{h}_n$ 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 are set to the same, *e.g.*, 128. There are four different experiments conducted. Here, we provide the early recognition accuracy of the proposed method (w/ Exc.) without observation $\boldsymbol{h}_n$ (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 TableII. The highest accuracies are emboldened.

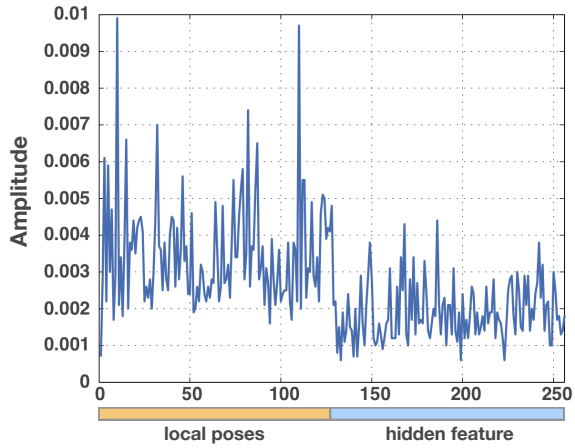


Fig. 6. The learned weights distribution of Agent's input layer on input embeddings. The color bar under the x-axis marks the range of each related feature. For example the orange bar marks the range of the embedded local pose feature.

Phase	#Sample	Time (s)
Training	600	206.57
Testing	575	181.12

TABLE III
EPOCH TIME EVALUATION OF THE AGENT IN TRAINING AND TESTING PHASE ON FHPA

As can be seen, without the information from sequence observation $\boldsymbol{h}$, the proposed method cannot achieve the accuracy as good as the baseline (w/o Exc.). This shows that there are limited informative patterns from the classification probability $\boldsymbol{d}$ 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 $\boldsymbol{h}_n$ 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 help achieve better classification performance.

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 $\boldsymbol{h}$ in the input layer is a $D_e \times D_h$ matrix $\boldsymbol{W}_{ih}$, where D_e is the dimensionality of the embedded $\boldsymbol{h}$, and D_h is the dimensionality of the hidden feature in LSTM classifier. Fig.6 shows the contribution of the two parts, *Hid* and *Local*, of the embedded $\boldsymbol{h}$ to the exclusion mask generation. This curve is recorded in a one-dimensional vector $\boldsymbol{a} \in \mathbb{R}^{D_e}$ obtained by,

$$\boldsymbol{a} = \frac{1}{D_h} \sum_{j=1}^{D_h} \frac{1}{Z_j} p\left(\boldsymbol{W}_{ih}^{j}\right), \tag{8}$$

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

Complexity and Time Cost

We follow the definition of the Floating Point Operations (FLOPs) in the appendix of [62] to estimate the computing complexity of our model. We denote the dimension of input feature as N_i, the number of layers as N_l, the number of hidden neural as N_n and the number of category as N_c. Therefore the FLOPs of the RNN input layer is $(2N_i + 1)N_n$. The FLOPs of the hidden layers is $N_n N_l + [(2N_n + 1)N_n](2N_l - 1)$. The FLOPs of the output layer is $(2N_n + 1)N_c$. In the case of the FPHA experiment, N_i is 224, N_l is 2, N_n is 100 and N_c is 45. Hence the FLOPs the agent takes is around 114K.

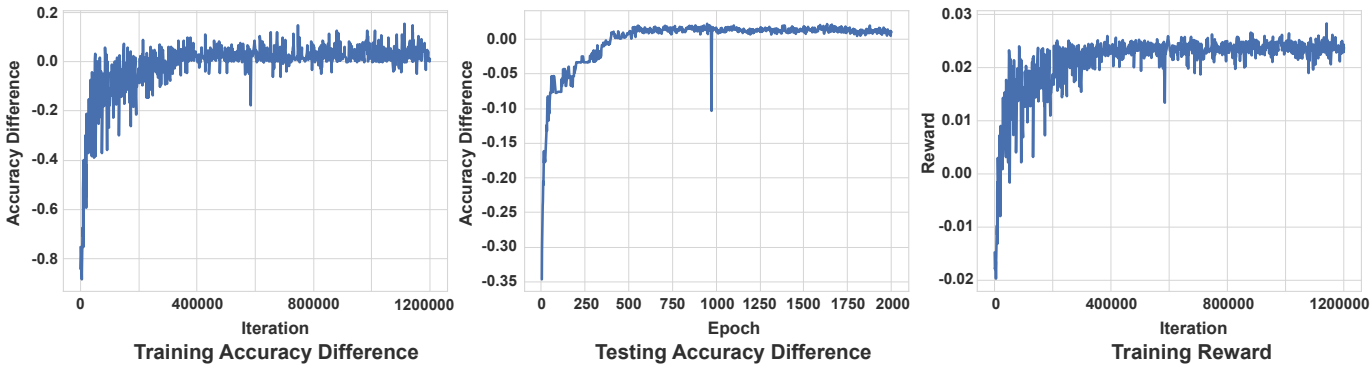


Fig. 7. Accuracy Difference and Reward Curve of FPHA. The accuracy difference is defined as the recognition accuracy difference between early recognition before and after category exclusion. The training accuracy and reward curves are drawn sample by sample, and we denote the index of the x-axis as *iteration*. The testing accuracy curve is drawn epoch by epoch, and therefore we set the index of the x-axis as *epoch*.

Observation Ratio	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%	AUC
w/o Exc.	45.91	54.26	61.39	63.30	65.74	69.22	70.61	72.17	73.39	74.43	64.11
w/ Exc.-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 IV
PARAMETER SENSITIVITY ANALYSIS OF THE POSITIVE REWARD η ON FPHA (%).

We further evaluate the training time and testing time of the proposed agent model and the results are shown in Table III. The experiment is conducted on an Intel Xeon Platinum 8255C CPU with 2.50GHz clock frequency and a NVIDIA Tesla V100 GPU.

Depth	1	2	3	4	5	w/o Exc.
AUC	63.72	66.66	66.20	63.84	63.20	64.11

TABLE V
IMPACT OF MODEL DEPTH ON MODEL PERFORMANCE - FHPA (%)

Neuron	50	100	150	200	250	w/o Exc.
AUC	66.13	66.66	65.14	65.74	64.11	64.11

TABLE VI
IMPACT OF THE NUMBER OF NEURON ON MODEL PERFORMANCE - FHPA (%)

We further conduct experiments of model capacity sensitivity on the FHPA dataset and the results are shown in Table V and Table VI. In these two tables, the criterion Area under Curve (AUC) is reported for different network parameter settings, and the AUC of the baseline model, namely the classifier without category exclusion (w/o Exc.), is also reported. We first fix the number of neuron in each layer as 100 and vary the depth of the agent from 1 to 5. As can be seen from Table V, when the model is too deep (with more than 4 layers) or too shallow (with only one layer), the agent cannot either well generalize or well learn the category exclusion strategy. We then fix the number of model depth as 2 and vary the number of neuron in each layer from 50 to 250. As can be seen in Table VI, when there are too many neurons in each layer, for example 250 in this experiment, the agent cannot well generalize the exclusion strategy due to over-fitting.

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.7. From Fig.7 we can see

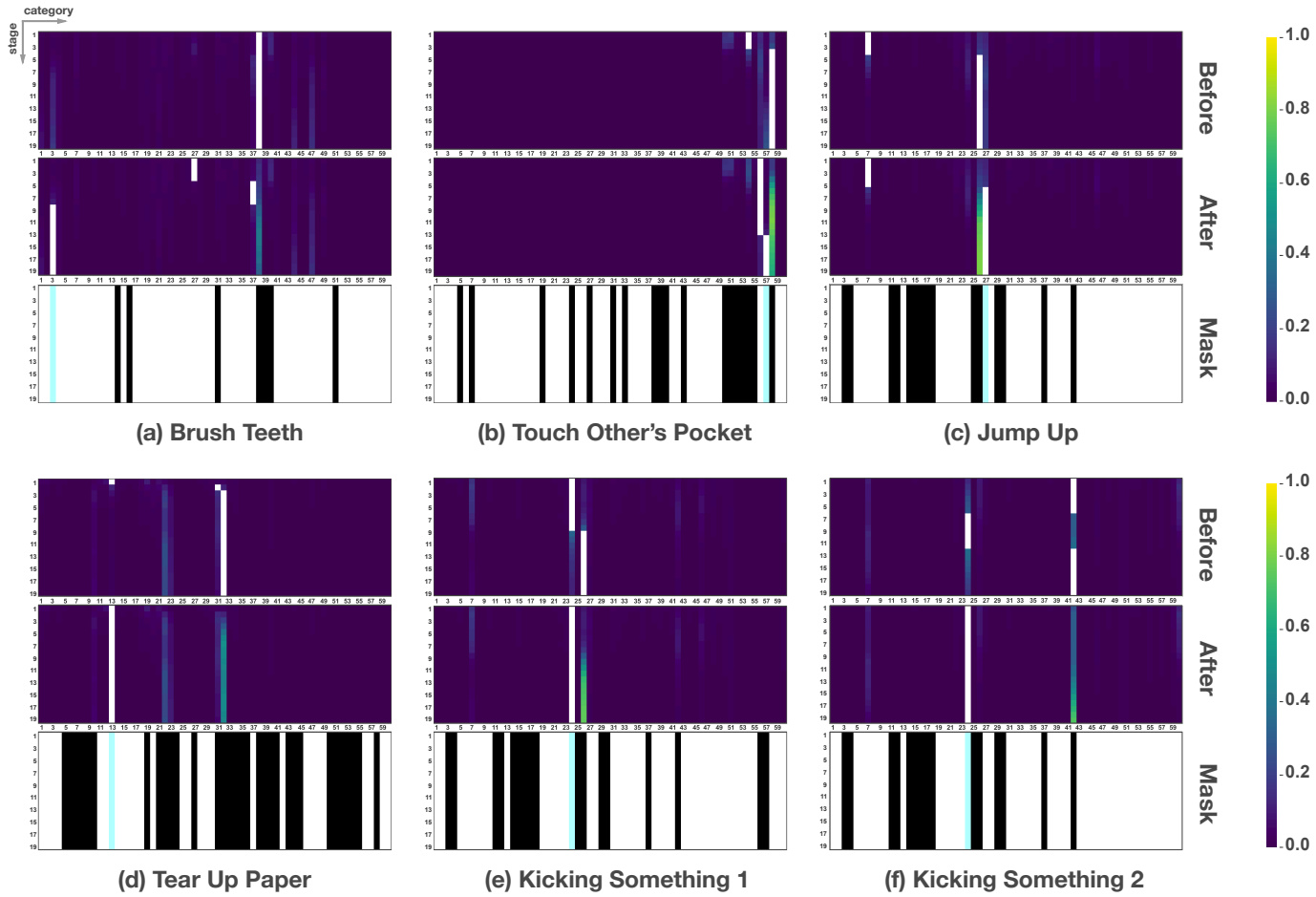


Fig. 8. Visualization of the generated mask and the recognition before and after exclusion. Each visualization of a sample consists of three matrices. The size of each matrix is 20 by 60, namely $N * C$. The numbers on the vertical direction of each matrix indicate the stage indices, and the ones on 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 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.

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.

One of the important indicators in Reinforcement Learning is the *reward*, as defined in Eq.(5). Here the reward we draw is defined as $\sum_{n,c} r_n^c/(NC)$. As can be seen from Fig.7, the *reward* increases gradually during training, and tend to be stable after 400,000 iterations, which indicates that the training based on Reinforcement Learning converges.

Parameter Sensitivity Analysis

The positive reward η, as defined in Eq.(5), is a critical parameter balancing 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 TableIV. In this table, the rows with the overall accuracies lower than or close to the baseline (w/o Exc.) 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 TableIV, 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 baseline (w/o Exc.). 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 TableIV 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.

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.8 shows the visualizations of six samples from the NTU-RGBD (CS). The detailed explanation of the figure is

shown in the figure caption. 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.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 avoid 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.

V. Conclusion

In this paper, 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.

References

[1] J. Hu, W. Zheng, L. Ma, G. Wang, J. Lai. "Real-time RGB-D activity prediction by soft regression," in *ECCV*, 2016.

[2] Y. Kong, Y. Fu. "Max-margin action prediction machine," in *TPAMI*, 2016.

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

[4] M. Ryoo. "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *ICCV*, 2011.

[5] A. Shahroudy, J. Liu, T. Ng, G. Wang. "NTU RGB+ D: A large scale dataset for 3D human activity analysis," in *CVPR*, 2016.

[6] R. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning," in *Machine learning*, 1992.

[7] G. O, S. Yuan, S. Baek, T. Kim. "First-person hand action benchmark with RGB-D videos and 3D hand pose annotations," in *CVPR*, 2018.

[8] K. Soomro, A. Zamir, M. Shah. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild," in *CRCV-TR-12-01*, 2012.

[9] J. Weng, C. Weng, J. Yuan, Z. Liu. "Discriminative spatio-temporal pattern discovery for 3D action recognition," in *TCSVT*, 2018.

[10] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev. "The kinetics human action video dataset," in *arXiv preprint arXiv:170506950*, 2017.

[11] S. Hochreiter, J. Schmidhuber. "Long short-term memory," in *Neural computation*, 1997.

[12] D. Kingma, J. Ba. "Adam: A method for stochastic optimization," in *ICLR*, 2014.

[13] J. Hu, W. Zheng, L. Ma, G. Wang, J. Lai, J. Zhang. "Early Action Prediction by Soft Regression," in *TPAMI*, 2018.

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

[15] Y. Kong, S. Gao, B. Sun, Y. Fu. "Action prediction from videos via memorizing hard-to-predict samples," in *AAAI*, 2018.

[16] Y. Kong, D. Kit, Y. Fu. "A discriminative model with multiple temporal scales for action prediction," in *ECCV*, 2014.

[17] T. Lan, T. Chen, S. Savarese. "A hierarchical representation for future action prediction," in *ECCV*, 2014.

[18] J. Liu, A. Shahroudy, G. Wang, L. Duan, A. Kot. "SSNet: scale selection network for online 3D action prediction," in *CVPR*, 2018.

[19] A. Fathi, G. Mori. "Action recognition by learning mid-level motion features," in *CVPR*, 2008.

[20] Z. Zhang, D. Tao. "Slow feature analysis for human action recognition," in *TPAMI*, 2012.

[21] M. Bregonzio, S. Gong, T. Xiang. "Recognising action as clouds of space-time interest points.," in *CVPR*, 2009.

[22] A. Klaser, M. Marszałek, C. Schmid. "A spatio-temporal descriptor based on 3d-gradients," in *BMVC*, 2008.

[23] P. Scovanner, S. Ali, M. Shah. "A 3-dimensional sift descriptor and its application to action recognition," in *ACM MM*, 2007.

[24] H. Wang, A. Kläser, C. Schmid, L. Cheng-lin. "Action Recognition by Dense Trajectories," in *CVPR*, 2011.

[25] H. Wang, C. Schmid. "Action recognition with improved trajectories," in *ICCV*, 2013.

[26] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri. "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[27] J. Carreira, A. Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, 2017.

[28] K. Hara, H. Kataoka, Y. Satoh. "Learning spatio-temporal features with 3D residual networks for action recognition," in *ICCV*, 2017.

[29] K. Hara, H. Kataoka, Y. Satoh. "Can spatiotemporal 3d

cnns retrace the history of 2d cnns and imagenet?," in *CVPR*, 2018.

[30] C. Feichtenhofer, A. Pinz, A. Zisserman. "Convolutional two-stream network fusion for video action recognition," in *CVPR*, 2016.

[31] K. Simonyan, A. Zisserman. "Two-stream convolutional networks for action recognition in videos," in *NeurIPS*, 2014.

[32] J. Weng, M. Liu, X. Jiang, J. Yuan. "Deformable pose traversal convolution for 3d action and gesture recognition," in *ECCV*, 2018.

[33] Z. Ren, J. Yuan, Z. Zhang. "Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera," in *ACM MM*, 2011.

[34] J. Wang, Z. Liu, Y. Wu, J. Yuan. "Mining actionlet ensemble for action recognition with depth cameras," in *CVPR*, 2012.

[35] J. Wang, Z. Liu, Y. Wu, J. Yuan. "Learning actionlet ensemble for 3D human action recognition," in *TPAMI*, 2013.

[36] H. Liang, J. Yuan, D. Thalmann, N. Thalmann. "Ar in hand: Egocentric palm pose tracking and gesture recognition for augmented reality applications," in *ACM MM*, 2015.

[37] Z. Ren, J. Yuan, J. Meng, Z. Zhang. "Robust part-based hand gesture recognition using kinect sensor," in *TMM*, 2013.

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

[39] F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, R. Bajcsy. "Sequence of the most informative joints (smij): A new representation for human skeletal action recognition," in *JVCI*, 2014.

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

[41] R. Vemulapalli, R. Chellapa. "Rolling rotations for recognizing human actions from 3d skeletal data," in *CVPR*, 2016.

[42] G. O, T. Kim. "Transition forests: Learning discriminative temporal transitions for action recognition and detection," in *CVPR*, 2017.

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

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

[45] J. Liu, G. Wang, P. Hu, L. Duan, A. Kot. "Global context-aware attention LSTM networks for 3D action recognition," in *CVPR*, 2017.

[46] M. Liu, J. Yuan. "Recognizing human actions as the evolution of pose estimation maps," in *CVPR*, 2018.

[47] H. Wang, L. Wang. "Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks," in *CVPR*, 2017.

[48] S. Song, C. Lan, J. Xing, W. Zeng, J. Liu. "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[49] I. Lee, D. Kim, S. Kang, S. Lee. "Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks," in *ICCV*, 2017.

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

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

[52] W. Li, L. Wen, M. Chang, S. Namlim, S. Lyu. "Adaptive RNN tree for large-scale human action recognition," in *ICCV*, 2017.

[53] Q. Ke, M. Bennamoun, S. An, F. Sohel, F. Boussaid. "A new representation of skeleton sequences for 3d action recognition," in *CVPR*, 2017.

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

[55] Y. Du, W. Wang, L. Wang. "Hierarchical recurrent neural network for skeleton based action recognition," in *CVPR*, 2015.

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

[57] D. Guo, S. Wang, Q. Tian, M. Wang. "Dense temporal convolution network for sign language translation," in *IJCAI*, 2019.

[58] C. Lea, M. Flynn, R. Vidal, A. Reiter, G. Hager. "Temporal convolutional networks for action segmentation and detection," in *CVPR*, 2017.

[59] C. Lea, R. Vidal, A. Reiter, G. Hager. "Temporal convolutional networks: A unified approach to action segmentation," in *ECCV*, 2016.

[60] P. Lei, S. Todorovic. "Temporal deformable residual networks for action segmentation in videos," in *CVPR*, 2018.

[61] D. Guo, W. Zhou, H. Li, M. Wang. "Hierarchical lstm for sign language translation," in *AAAI*, 2018.

[62] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz. "Pruning convolutional neural networks for resource efficient inference," in *ICLR*, 2017.

[63] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, Q. Tian. "Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition," in *CVPR*, 2019.

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

[65] D. Tran, L. Bourdev, R. Fergus, L. Torresani, M. Paluri. "Learning spatiotemporal features with 3d convolutional networks," in *ICCV*, 2015.

[66] D. Tran, H. Wang, L. Torresani, J. Ray, Y. Lecun, M. Paluri. "A closer look at spatiotemporal convolutions for action recognition," in *CVPR*, 2018.

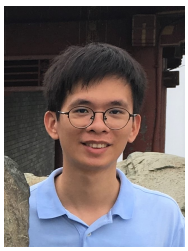
[67] Z. Qiu, T. Yao, T. Mei. "Learning spatio-temporal

representation with pseudo-3d residual networks," in *ICCV*, 2017.

[68] K. Hara, H. Kataoka, Y. Satoh. "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?," in *CVPR*, 2018.

Junwu Weng (S'17) received his M.Eng. degree from South China University of Technology (SCUT), Guangdong, China, in 2015. Before that he graduated from the Talented Student Program of School of Electronics & Information, SCUT. He is currently pursuing the Ph.D. degree at the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. His current research interests include computer vision, machine learning, as well as action and gesture analysis.

Xudong Jiang (M'02-SM'06) received the B.Eng. and M.Eng. degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, and the Ph.D. degree from Helmut Schmidt University, Hamburg, Germany. After several years' teaching as a Lecturer in UESTC, he was with the Institute for Infocomm Research, A*STAR, Singapore, as a Lead Scientist, and the Head of the Biometrics Laboratory from 1998 to 2004, where he developed a system that achieved the most efficiency and the second most accuracy at the International Fingerprint Verification Competition in 2000. He joined Nanyang Technological University (NTU), Singapore, as a Faculty Member, in 2004, where he has also served as the Director of the Centre for Information Security from 2005 to 2011. He is currently a tenured Associate Professor with the School of EEE, NTU. He holds 7 patents and has authored over 150 papers with 39 papers in the IEEE journals, including 11 papers in IEEE TIP and 6 papers in IEEE TPAMI. His research interests include signal/image processing, pattern recognition, computer vision, machine learning, and biometrics. He has been an IFS Technical Committee Member of the IEEE Signal Processing Society, and he has served as Associate Editors for IEEE TIP, IEEE SPL, and Editor-in-Chief for IET Biometrics.



Wei-Long Zheng (S'14-M'19) received the bachelor's degree in information engineering with the Department of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, in 2012. He received the Ph.D. degree in computer science with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2018. He is a research fellow in the Department of Neurology, Massachusetts General Hospital, Harvard Medical School, Boston, MA, USA. He received the IEEE Transactions on Autonomous Mental Development Outstanding Paper Award from IEEE Computational Intelligence Society in 2018. His research focuses on affective computing, brain-computer interaction, machine learning and pattern recognition.

Junsong Yuan (M'08-SM'14) is currently an Associate Professor and Director of Visual Computing Lab at Department of Computer Science and Engineering (CSE), State University of New York at Buffalo, USA. Before that he was an Associate Professor at Nanyang Technological University (NTU), Singapore. He obtained his Ph.D. from Northwestern University, M.Eng. from National University of Singapore and B.Eng from the Special Program for the Gifted Young of Huazhong University of Science and Technology (HUST), China. His research interests include computer vision, pattern recognition, video analytics, gesture and action analysis, large-scale visual search and mining. He received Best Paper Award from IEEE Trans. on Multimedia, Nanyang Assistant Professorship from NTU, and Outstanding EECS Ph.D. Thesis award from Northwestern University. He is currently Senior Area Editor of Journal of Visual Communications and Image Representation (JVCI), Associate Editor of IEEE Trans. on Image Processing (T-IP) and IEEE Trans. on Circuits and Systems for Video Technology (T-CSVT), and served as Guest Editor of International Journal of Computer Vision (IJCV). He is Program Co-Chair of IEEE Conf. on Multimedia Expo (ICME'18) and Steering Committee Member of ICME (2018-2019). He also served as Area Chair for CVPR, ICIP, ICPR, ACCV, ACM MM, WACV etc. He is a Fellow of International Association of Pattern Recognition (IAPR).