Online Human Action Detection Using Joint Classification-Regression Recurrent Neural Networks

Yanghao Li[1], Cuiling Lan[2(✉)], Junliang Xing[3], Wenjun Zeng[2], Chunfeng Yuan[3], and Jiaying Liu[1(✉)]

[1] Institute of Computer Science and Technology, Peking University, Beijing, China
{lyttonhao,liujiaying}@pku.edu.cn
[2] Microsoft Research Asia, Beijing, China
{culan,wezeng}@microsoft.com
[3] Institute of Automation, Chinese Academy of Sciences, Beijing, China
{jlxing,cfyuan}@nlpr.ia.ac.cn

Abstract. Human action recognition from well-segmented 3D skeleton data has been intensively studied and has been attracting an increasing attention. Online action detection goes one step further and is more challenging, which identifies the action type and localizes the action positions on the fly from the untrimmed stream data. In this paper, we study the problem of online action detection from streaming skeleton data. We propose a multi-task end-to-end Joint Classification-Regression Recurrent Neural Network to better explore the action type and temporal localization information. By employing a joint classification and regression optimization objective, this network is capable of automatically localizing the start and end points of actions more accurately. Specifically, by leveraging the merits of the deep Long Short-Term Memory (LSTM) subnetwork, the proposed model automatically captures the complex long-range temporal dynamics, which naturally avoids the typical sliding window design and thus ensures high computational efficiency. Furthermore, the subtask of regression optimization provides the ability to forecast the action prior to its occurrence. To evaluate our proposed model, we build a large streaming video dataset with annotations. Experimental results on our dataset and the public G3D dataset both demonstrate very promising performance of our scheme.

Keywords: Action detection · Recurrent neural network · Joint classification-regression

This work was done at Microsoft Research Asia.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-3-319-46478-7_13) contains supplementary material, which is available to authorized users.

B. Leibe et al. (Eds.): ECCV 2016, Part VII, LNCS 9911, pp. 203–220, 2016.
DOI: 10.1007/978-3-319-46478-7_13

1 Introduction

Human action detection is an important problem in computer vision, which has broad practical applications like visual surveillance, human-computer interaction and intelligent robot navigation. Unlike action recognition and offline action detection, which determine the action after it is fully observed, online action detection aims to detect the action on the fly, as early as possible. It is much desirable to accurately and timely localize the start point and end point of an action along the time and determine the action type as illustrated in Fig. 1. Besides, it is also desirable to forecast the start and end of the actions prior to their occurrence. For example, for intelligent robot system, in addition to the accurate detection of actions, it would also be appreciated if it can predict the start of the impending action or the end of the ongoing actions and then get something ready for the person it serves, e.g., passing towels when he/she finishes washing hands. Therefore, the detection and forecast system could respond to impending or ongoing events accurately and as soon as possible, to provide better user experiences.

For human action recognition and detection, many research works have been designed for RGB videos recorded by 2D cameras in the past couple of decades [1]. In recent years, with the prevalence of the affordable color-depth sensing cameras, such as the Microsoft Kinect [2], it is much easier and cheaper to obtain depth data and thus the 3D skeleton of human body (see skeleton examples in Fig. 1). Biological observations suggest that skeleton, as an intrinsic high level representation, is very valuable information for recognizing actions by humans [3]. In comparison to RGB video, such high level human representation by skeleton is robust to illumination and clustered background [4], but may not be appropriate for recognizing fine-grained actions with marginal differences. Taking the advantages of skeleton representation, in this paper, we investigate skeleton based human action detection. The addition of RGB information may result in better performance and will be addressed in the future work.

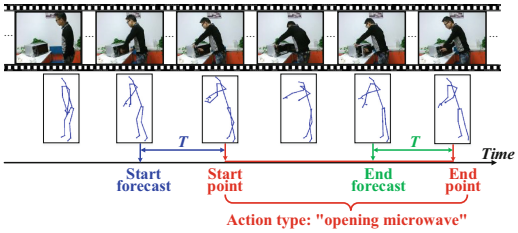


Fig. 1. Illustration of online action detection. It aims to determine the action type and the localization on the fly. It is also desirable to forecast the start and end points (e.g., T frames ahead).

Although online action detection is of great importance, there are very few works specially designed for it [5,6]. Moreover, efficient exploitation of the

advanced recurrent neural network (RNN) has not been well studied for the efficient temporal localization of actions. Most published methods are designed for offline detection [7], which performs detection after fully observing the sequence. To localize the action, most of previous works employ a sliding window design [5,8–10], which divides the sequence into overlapped clips before action recognition/classification is performed on each clip. Such sliding window design has low computational efficiency. A method that divides the continuous sequence into short clips with the shot boundary detected by computing the color histogram and motion histogram was proposed in [11]. However, indirect modeling of action localization in such an unsupervised manner does not provide satisfactory performance. An algorithm which can intelligently localize the actions on the fly is much expected, being suitable for the streaming sequence with actions of uncertain length. For action recognition on a segmented clip, deep learning methods, such as convolutional neural networks and recurrent neural networks, have been shown to have superior performances on feature representation and temporal dynamics modeling [12–15]. However, how to design an efficient online action detection system that leverages the neural network for the untrimmed streaming data is not well studied.

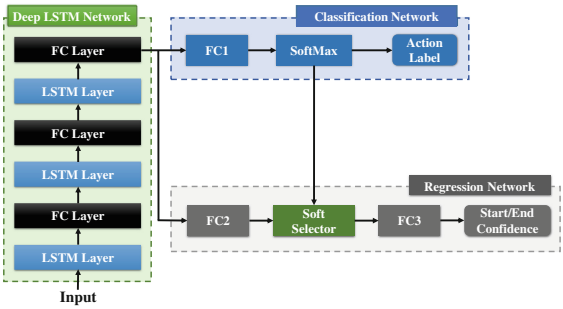


Fig. 2. Architecture of the proposed joint classification-regression RNN framework for online action detection and forecasting.

In this paper, we propose a Joint Classification-Regression Recurrent Neural Network to accurately detect the actions and localize the start and end positions of the actions on the fly from the streaming data. Figure 2 shows the architecture of the proposed framework. Specifically, we use LSTM [16] as the recurrent layers to perform automatic feature learning and long-range temporal dynamics modeling. Our network is end-to-end trainable by optimizing a joint objective function of frame-wise action classification and temporal localization regression. On one hand, we perform frame-wise action classification, which aims to detect the actions timely. On the other hand, to better localize the start and end of actions, we incorporate the regression of the start and end points of actions into the network. We can forecast their occurrences in advance based on the regressed curve. We train this classification and regression network jointly to obtain high

detection accuracy. Note that the detection is performed frame-by-frame and the temporal information is automatically learnt by the deep LSTM network without requiring a sliding window design, which is time efficient.

The main contributions of this paper are summarized as follows:

- We investigate the new problem of online action detection for streaming skeleton data by leveraging recurrent neural network.
- We propose an end-to-end Joint Classification-Regression RNN to address our target problem. Our method leverages the advantages of RNNs for frame-wise action detection and forecasting without requiring a sliding window design and explicit looking forward or backward.
- We build a large action dataset for the task of online action detection from streaming sequence.

2 Related Work

2.1 Action Recognition and Action Detection

Action recognition and detection have attracted a lot of research interests in recent years. Most methods are designed for action recognition [13,14,17], i.e., to recognize the action type from a well-segmented sequence, or offline action detection [8,10,18,19]. However, in many applications it is desirable to recognize the action on the fly, without waiting for the completion of the action, e.g., in human computer interaction to reduce the response delay. In [5], a learning formulation based on a structural SVM is proposed to recognize partial events, enabling early detection. To reduce the observational latency of human action recognition, a non-parametric moving pose framework [6] and a dynamic integral bag-of-words approach [20] are proposed respectively to detect actions earlier. Our model goes beyond early detection. Besides providing frame-wise class information, it forecasts the occurrence of start and end of actions.

To localize actions in streaming video sequence, existing detection methods utilize either sliding-window scheme [5,8–10], or action proposal approaches [11,21,22]. These methods usually have low computational efficiency or unsatisfactory localization accuracy due to the overlapping design and unsupervised localization approach. Besides, it is not easy to determine the sliding-window size.

Our framework aims to address the online action detection in such a way that it can predict the action at each time slot efficiently without requiring a sliding window design. We use the regression design to determine the start/end points learned in a supervised manner during the training, enabling the localization being more accurate. Furthermore, it forecasts the start of the impending or end of the ongoing actions.

2.2 Deep Learning

Recently, deep learning has been exploited for action recognition [17]. Instead of using hand-crafted features, deep learning can automatically learn robust

feature representations from raw data. To model temporal dynamics, RNNs have also been used for action recognition. A Long-term Recurrent Convolutional Network (LRCN) [13] is proposed for activity recognition, where the LRCN model contains several Convolutional Neural Network (CNN) layers to extract visual features followed by LSTM layers to handle temporal dynamics. A hybrid deep learning framework is proposed for video classification [12], where LSTM networks are applied on top of the two types of CNN-based features related to the spatial and the short-term motion information. For skeleton data, hierarchical RNN [14] and fully connected LSTM [15] are investigated to model the temporal dynamics for skeleton based action recognition.

Despite a lot of efforts in action recognition, which uses pre-segmented sequences, there are few works on applying RNNs for the action detection and forecasting tasks. Motivated by the advantages of RNNs in sequence learning and some other online detection tasks (e.g. audio onset [23] and driver distraction [24] detection), we propose a Joint Classification and Regression RNN to automatically localize the action location and determine the action type on the fly. In our framework, the designed LSTM network simultaneously plays the role of feature extraction and temporal dynamic modeling. Thanks to the long-short term memorizing function of LSTM, we do not need to assign an observation window as in the sliding window based approaches for the action type determination and avoid the repeat calculation. This enables our design to have superior detection performance with low computation complexity.

3 Problem Formulation

In this section, we formulate the online action detection problem. To help clarify the differences, offline action detection is first discussed.

3.1 Offline Action Detection

Given a video observation $V = \{v_0, \ldots, v_{N-1}\}$ composed of frames from time 0 to $N-1$, the goal of action detection is to determine whether a frame v_t at time t belongs to an action among the predefined M action classes.

Without loss of generality, the target classes for the frame v_t are denoted by a label vector $\mathbf{y}_t \in R^{1\times(M+1)}$, where $y_{t,j} = 1$ means the presence of an action of class j at this frame and $y_{t,j} = 0$ means absence of this action. Besides the M classes of actions, a blank class is added to represent the situation in which the current frame does not belong to any predefined actions. Since the entire sequence is known, the determination of the classes at each time slot is to maximize the posterior probability

$$\mathbf{y}_t^* = \underset{\mathbf{y}_t}{\operatorname{argmax}}\, P(\mathbf{y}_t|V), \tag{1}$$

where $\mathbf{y}_t$ is the possible action label vector for frame v_t. Therefore, conditioned on the entire sequence V, the action label with the maximum probability $P(\mathbf{y}_t|V)$ is chosen to be the status of frame v_t in the sequence.

According to the action label of each frame, an occurring action i can be represented in the form $d_i = \{g_i, t_{i,start}, t_{i,end}\}$, where g_i denotes the class type of the action i, $t_{i,start}$ and $t_{i,end}$ correspond to the starting and ending time of the action, respectively.

3.2 Online Action Detection

In contrast to offline action detection, which makes use of the whole video to make decisions, online detection is required to determine which actions the current frame belongs to without using future information. Thus, the method must automatically estimate the start time and status of the current action. The problem can be formulated as

$$\mathbf{y}_t^* = \underset{\mathbf{y}_t}{\operatorname{argmax}} P(\mathbf{y}_t | v_0, ..., v_t). \tag{2}$$

Besides determining the action label, an online action detection system for streaming data is also expected to predict the starting and ending time of an action. We should be aware of the occurrence of the action as early as possible and be able to predict the end of the action. For example, for an action $d_i = \{g_i, t_{i,start}, t_{i,end}\}$, the system is expected to forecast the start and end of the action during $[t_{i,start} - T, t_{i,start}]$ and $[t_{i,end} - T, t_{i,end}]$, respectively, ahead its occurrence. T could be considered as the expected forecasting time in statistic. We define the optimization problem as

$$(\mathbf{y}_t^*, \mathbf{a}_t^*, \mathbf{b}_t^*) = \underset{\mathbf{y}_t, \mathbf{a}_t, \mathbf{b}_t}{\operatorname{argmax}} P(\mathbf{y}_t, \mathbf{a}_t, \mathbf{b}_t | v_0, \ldots, v_t), \tag{3}$$

where $\mathbf{a}_t$ and $\mathbf{b}_t$ are two vectors, denoting whether actions are to start or to stop within the following T frames, respectively. For example, $a_{t,g_i} = 1$ means the action of class g_i will start within T frames.

4 Joint Classification-Regression RNN for Online Action Detection

We propose an end-to-end Joint Classification-Regression framework based on RNN to address the online action detection problem. Figure 2 shows the architecture of the proposed network, which has a shared deep LSTM network for feature extraction and temporal dynamic modeling, a classification subnetwork and a regression subnetwork. Note that we construct the deep LSTM network by stacking three LSTM layers and three non-linear fully-connected (FC) layers to have powerful learning capability. We first train the classification network for the frame-wise action classification. Then under the guidance of the classification results through the Soft Selector, we train the regressor to obtain more accurate localization of the start and end time points.

In the following, we first briefly review the RNNs and LSTM to make the paper self-contained. Then we introduce our proposed joint classification-regression network for online action detection.

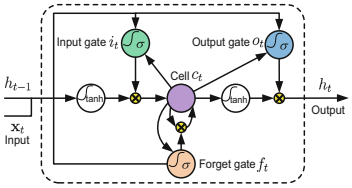


Fig. 3. The structure of an LSTM neuron, which contains an input gate i_t, a forget gate f_t, and an output gate i_t. Information is saved in the cell c_t.

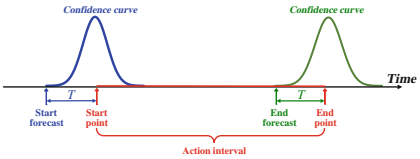


Fig. 4. Illustration of the confidence values around the start point and end point, which follow Gaussian-like curves with the confidence value 1 at the start and end point.

4.1 Overview of RNN and LSTM

In contrast to traditional feedforward neural networks, RNNs have self-connected recurrent connections which model the temporal evolution. The output response $\mathbf{h}_t$ of a recurrent hidden layer can be formulated as follows [25]

$$\mathbf{h}_t = \theta_h(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{4}$$

where $\mathbf{W}_{xh}$ and $\mathbf{W}_{hh}$ are mapping matrices from the current inputs $\mathbf{x}_t$ to the hidden layer h and the hidden layer to itself. $\mathbf{b}_h$ denotes the bias vector. θ_h is the activation function in the hidden layer.

The above RNNs have difficulty in learning long range dependencies [26], due to the vanishing gradient effect. To overcome this limitation, recurrent neural networks using LSTM [14,16,25] has been designed to mitigate the vanishing gradient problem and learn the long-range contextual information of a temporal sequence. Figure 3 illustrates a typical LSTM neuron. In addition to a hidden output h_t, an LSTM neuron contains an input gate i_t, a forget gate f_t, a memory cell c_t, and an output gate o_t. At each timestep, it can choose to read, write or reset the memory cell through the three gates. This strategy allows LSTM to memorize and access information many timesteps ago.

4.2 Subnetwork for Classification Task

We first train an end-to-end classification subnetwork for frame-wise action recognition. The structure of this classification subnetwork is shown in the upper part of Fig. 2. The frame first goes through the deep LSTM network, which is responsible for modeling the spatial structure and temporal dynamics. Then a fully-connected layer FC1 and a SoftMax layer are added for the classification of the current frame. The output of the SoftMax layer is the probability distribution of the action classes $\mathbf{y}_t$. Following the problem formulation as described in Sect. 3, the objective function of this classification task is to minimize the cross-entropy loss function

$$\mathcal{L}_c(V) = -\frac{1}{N}\sum_{t=0}^{N-1}\sum_{k=0}^{M} z_{t,k} \ln P(y_{t,k}|v_0, \ldots v_t), \tag{5}$$

where $z_{t,k}$ corresponds to the groundtruth label of frame v_t for class k, $z_{t,k} = 1$ means the groundtruth class is the k^{th} class, $P(y_{t,k}|v_0, \ldots v_t)$ denotes the estimated probability of being action classes k of frame v_t.

We train this network with Back Propagation Through Time (BPTT) [27] and use stochastic gradient descent with momentum to compute the derivatives of the objective function with respect to all parameters. To prevent over-fitting, we have utilized dropout at the three fully-connected layers.

4.3 Joint Classification and Regression

We fine-tune this network on the initialized classification model by jointly optimizing the classification and regression. Inspired by the Joint Classification-Regression models used in Random Forest [28,29] for other tasks (e.g. segmentation [28] and object detection [29]), we propose our Joint learning to simultaneously make frame-wise classification, localize the start and end time points of actions, and to forecast them.

We define a confidence factor for each frame to measure the possibility of the current frame to be the start or end point of some action. To better localize the start or end point, we use a Gaussian-like curve to describe the confidences, which centralizes at the actual start (or end) point as illustrated in Fig. 4. Taking the start point as an example, the confidence of the frame v_t with respect to the start point of action j is defined as

$$c_t^s = e^{-(t-s_j)^2/2\sigma^2}, \tag{6}$$

where s_j is the start point of the nearest (along time) action j to the frame v_t, and σ is the parameter which controls the shape of the confidence curve. Note that at the start point time, i.e., $t = s_j$, the confidence value is 1. Similarly, we denote the confidence of being the end point of one action as c_t^e. For the Gaussian-like curve, a lower confidence value suggests the current frame has larger distance from the start point and the peak point indicates the start point.

Such design has two benefits. First, it is easy to localize the start/end point by checking the regressed peak points. Second, this makes the designed system have the ability of forecasting. We can forecast the start (or end) of actions according to the current confidence response. We set a confidence threshold θ_s (or θ_e) according to the sensitivity requirement of the system to predict the start (or end) point. When the current confidence value is larger than θ_s (or θ_e), we consider that one action may start (or end) soon. Usually, larger threshold corresponds to a later response but a more accurate forecast.

Using the confidence as the target values, we include this regression problem as another task in our RNN model, as shown in the lower part of Fig. 2.

This regression subnetwork consists of a non-linear fully-connected layer FC2, a Soft Selector layer, and a non-linear fully-connected layer FC3. Since we regress one type of confidence values for all the start points of different actions, we need to use the output of the action classification to guide the regression task. Therefore, we design a Soft Selector module to generate more specific features by fusing the output of SoftMax layer which describes the probabilities of classification together with the output of the FC2 layer.

We achieve this by using class specific element-wise multiplication of the outputs of SoftMax and FC2 layer. The information from the SoftMax layer for the classification task plays the role of class-based feature selection over the output features of FC2 for the regression task. A simplified illustration about the Soft Selector model is shown in Fig. 5. Assume we have 5 action classes and the dimension of the FC2 layer output is reshaped to 7×5. The vector (marked by circles) with the dimension of 5 from the SoftMax output denotes the probabilities of the current frame belonging to the 5 classes respectively. Element-wise multiplication is performed for each row of features and then integrating the SoftMax output plays the role of feature selection for different classes.

The final objective function of the Joint Classification-Regression is formulated as

$$\begin{aligned}\mathcal{L}(V) =& \mathcal{L}_c(V) + \lambda \mathcal{L}_r(V) \\ =& -\frac{1}{N}\sum_{t=0}^{N-1}\left[\left(\sum_{k=0}^{M} z_{t,k} \ln P(y_{t,k}|v_0, \dots v_t)\right)\right. \\ &\left. + \lambda \cdot \left(\ell(c_t^s, p_t^s) + \ell(c_t^e, p_t^e)\right)\right],\end{aligned} \tag{7}$$

where p_t^s and p_t^e are the predicted confidence values as start and end points, λ is the weight for the regression task, ℓ is the regression loss function, which is defined as $\ell(x, y) = (x - y)^2$. In the training, the overall loss is a summarization of the loss from each frame v_t, where $0 \leq t < N$. For a frame v_t, its loss consists of the classification loss represented by the cross-entropy for the $M + 1$ classes and the regression loss for identifying the start and end of the nearest action.

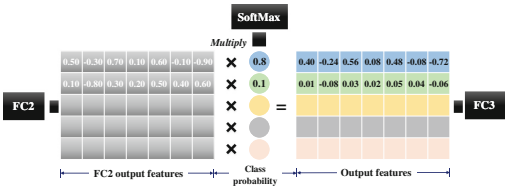


Fig. 5. Soft Selector for the fusion of classification output and features from FC2. Element-wise multiplication is performed for each row of features (we only show the first two rows here).

We fine-tune this entire network over the initialized classification model by minimizing the object function of the joint classification and regression optimization. Note that to enable the classification result indicating which action will begin soon, we set the groundtruth label $z^s_{t,k} = 1$ in the training where $t_{k,start} - T \leq t < t_{k,start}$ for all actions, according to the expected forecast-forward value T as defined in Sect. 3. Then, for each frame, the classification output indicates the impending or ongoing action class while the two confidence outputs show the probability to be the start or end point. We set the peak positions of confidences to be the predicted action start (or end) time. Note that since the classification and regression results of the current frame are correlated with the current input and the previous memorized information for the LSTM network, the system does not need to explicitly look back, avoiding sliding window design.

5 Experiments

In this section, we evaluate the detection and forecast performance of the proposed method on two different skeleton-based datasets. The reason why we choose skeleton-based datasets for experiments is three-fold. First, skeleton joints are captured in the full 3D space, which can provide more comprehensive information for action detection compared to 2D images. Second, the skeleton joints can well represent the posture of human which provide accurate positions and capture human motions. Finally, the dimension of skeleton is low, i.e., $25 \times 3 = 75$ values for each frame from Kinect V2. This makes the skeleton based online action detection much attractive for real applications.

5.1 Datasets and Settings

Most published RGB-D datasets were generated for the classification task where actions were already pre-segmented [30,31]. They are only suitable for action recognition. Thus, besides using an existing skeleton-based detection dataset, the Gaming Action Dataset (G3D) [32], we collect a new online streaming dataset following similar rules of previous action recognition datasets, which is much more appropriate for the online action detection problem. In this work, being similar to that in [15], the normalization processing on each skeleton frame is performed to be invariant to position.

Gaming Action Dataset (G3D). The G3D dataset contains 20 gaming actions captured by Kinect, which are grouped into seven categories, such as fighting, tennis and golf. Some limitations of this dataset are that the number and occurrence order of actions in the videos are unchanged and the actors are motionless between performing different actions, which make the dataset a little unrealistic.

Online Action Detection Dataset (OAD). This is our newly collected action dataset with long sequences for our online action detection problem.

The dataset was captured using the Kinect v2 sensor, which collects color images, depth images and human skeleton joints synchronously. It was captured in a daily-life indoor environment. Different actors freely performed 10 actions, including *drinking, eating, writing, opening cupboard, washing hands, opening microwave, sweeping, gargling, throwing trash, and wiping*. We collected 59 long video sequences at 8 fps (in total 103,347 frames of 216 min). Note that our low recording frame rate is due to the speed limitation of writing large amount of data (i.e., skeleton, high resolution RGB-D) to the disk of our laptop.

Since the Kinect v2 sensor is capable of providing more accurate depth, our dataset has more accurate tracked skeleton positions compared to previous skeleton datasets. In addition, the acting orders and duration of the actions are arbitrary, which approach the real-life scenarios. The length of each sequence is very long and there are variable idle periods between different actions, which meets the requirements of realistic online action detection from streaming videos.

Network and Parameter Settings. We show the architecture of our network in Fig. 2. The number of neurons in the deep LSTM network is 100, 100, 110, 110, 100, 100 for the first six layers respectively, including three LSTM layers and three FC layers. The design choice (i.e., LSTM architecture) is motivated by some previous works [14,15]. The number of neurons in the FC1 layer corresponds to the number of action classes $M+1$ and the number of neurons in the FC2 layer is set to $10 \times (M+1)$. For the FC3 layer, there are two neurons corresponding to the start and end confidences respectively. The forecast response threshold T can be set based on the requirement of the applications. In this paper, we set $T = 10$ (around one second) for the following experiments. The parameter σ in (6) is set to 5. The weight λ in the final loss function (7) is increased gradually from 0 to 10 during the fine-tuning of the entire network. Note that we use the same parameter settings for both OAD and G3D datasets.

For our OAD dataset, we randomly select 30 sequences for training and 20 sequences for testing. The remaining 9 long videos are used for the evaluation of the running speed. For the G3D dataset, we use the same setting as used in [32].

5.2 Action Detection Performance Evaluation

Evaluation Criterions. We use three different evaluation protocols to measure the detection results.

1. $F1$-Score. Similar to the protocols used in object detection from images [33], we define a criterion to determine a correct detection. A detection is correct when the overlapping ratio α between the predicted action interval I and the groundtruth interval I^* exceeds a threshold, e.g., 60 %. α is defined as

$$\alpha = \frac{|I \cap I^*|}{|I \cup I^*|}, \tag{8}$$

where $I \cap I^*$ denotes the intersection of the predicted and groundtruth intervals and $I \cup I^*$ denotes their union. With the above criterion to determine a

correction detection, the $F1$-Score is defined as

$$F1 = 2\frac{Precision * Recall}{Precision + Recall}. \tag{9}$$

2. SL-Score. To evaluate the accuracy of the localization of the start point for an action, we define a Start Localization Score (SL-Score) based on the relative distance between the predicted and the groundtruth start time. Suppose that the detector predicts that an action will start at time t and the corresponding groundtruth action interval is $[t_{start}, t_{end}]$, the score is calculated as $e^{-|t-t_{start}|/(t_{end}-t_{start})}$. For false positive or false negative samples, the score is set to 0.
3. EL-Score. Similarly, the End Localization Score (EL-Score) is defined based on the relative distance between the predicted and the groundtruth end time.

Baselines. We have implemented several baselines for comparison to evaluate the performance of our proposed Joint Classification-Regression RNN model (JCR-RNN), (i) SVM-SW. We train a SVM detector to detect the action with sliding window design (SW). (ii) RNN-SW. This is based on the baseline method Deep LSTM in [15], which employs a deep LSTM network that achieves good results on many skeleton-based action recognition datasets. We train the classifiers and perform the detection based on sliding window design. We set the window size to 10 with step of 5 for both RNN-SW and SVM-SW. We experimentally tried different window sizes and found 10 gives relatively good average performance. (iii) CA-RNN. This is a degenerated version of our model that only consists of the LSTM and classification network, without the regression network involved. We denote it as Classification Alone RNN model (CA-RNN).

Dectection Performance. Table 1 shows the $F1$-Score of each action class and the average $F1$-Score of all actions on our OAD Dataset. From Table 1,

Table 1. $F1$-Score on OAD dataset.

Actions	SVM-SW	RNN-SW [15]	CA-RNN	JCR-RNN
Drinking	0.146	0.441	**0.584**	0.574
Eating	0.465	0.550	**0.558**	0.523
Writing	0.645	**0.859**	0.749	0.822
Opening cupboard	0.308	0.321	0.490	**0.495**
Washing hands	0.562	0.668	0.672	**0.718**
Opening microwave	0.607	0.665	0.468	**0.703**
Sweeping	0.461	0.590	0.597	**0.643**
Gargling	0.437	0.550	0.579	**0.623**
Throwing trash	0.554	**0.674**	0.430	0.459
Wiping	**0.857**	0.747	0.761	0.780
Average	0.540	0.600	0.596	**0.653**

we have the following observations. (i) The RNN-SW method achieves 6 % higher $F1$-Score than the SVM-SW method. This demonstrates that RNNs can better model the temporal dynamics. (ii) Our JCR-RNN outperforms the RNN-SW method by 5.3 %. Despite RNN-SW, CA-RNN and JCR-RNN methods all use RNNs for feature learning, one difference is that our schemes are end-to-end trainable without the involvement of sliding window. Therefore, the improvements clearly demonstrate that our end-to-end schemes are more efficient than the classical sliding window scheme. (iii) Our JCR-RNN further improves over the CA-RNN and achieves the best performance. It can be seen that incorporating the regression task into the network and jointly optimizing classification-regression make the localization more accurate and enhance the detection accuracy.

To further evaluate the localization accuracy, we calculate the SL- and EL-Scores on the Online Action Dataset. The average scores of all actions are shown in Table 2. We can see the proposed scheme achieves the best localization accuracy.

Table 2. SL- and EL-Score on the OAD dataset.

Scores	SVM-SW	RNN-SW [15]	CA-RNN	JCR-RNN
SL-	0.316	0.366	0.378	**0.418**
EL-	0.325	0.376	0.382	**0.443**

For the G3D dataset, we evaluate the performance in terms of the three types of scores for the seven categories of sequences. To save space, we only show the results for the first two categories *Fighting* and *Golf* in Tables 3 and 4, and more results which have the similar trends can be found in the supplementary material. The results are consistent with the experiments on our own dataset. We also compare these methods using the evaluation metric action-based $F1$ as defined in [32], which treats the detection of an action as correct when the predicted start point is within 4 frames of the groundtruth start point for that action. Note that the action-based $F1$ only considers the accuracy of the start point. The results are shown in Table 5. The method in [32] uses a traditional boosting algorithm [34] and its scores are significantly lower than other methods.

5.3 Action Forecast Performance Evaluation

Evaluation Criterion. As explained in Sect. 3, the system is expected to forecast whether the action will start or end within T frames prior to its occurrence. To be considered as a true positive start forecast, the forecast should not only predict the impending action class, but also do so within a reasonable interval, i.e., $[t_{start} - T, t_{start}]$ for an action starting at t_{start}. This rule is also applied to end forecast. We use the Precision-Recall Curve to evaluate the performance of

Table 3. *SL*- and *EL*-Score on the G3D Dataset.

Action category	Scores	SVM-SW	RNN-SW [15]	CA-RNN	JCR-RNN
Fighting	*SL*-	0.318	0.412	0.512	**0.528**
	EL-	0.328	0.419	0.525	**0.557**
Golf	*SL*-	0.553	0.635	0.789	**0.793**
	EL-	0.524	0.656	0.791	**0.836**

Table 4. $F1$-Score on G3D.

Action category	SVM-SW	RNN-SW [15]	CA-RNN	JCR-RNN
Fighting	0.486	0.613	0.700	**0.735**
Golf	0.680	0.745	0.900	**0.967**

Table 5. Action-based $F1$ [32] on G3D.

Action category	G3D [32]	SVM-SW	RNN-SW [15]	CA-RNN	JCR-RNN
Fighting	58.54	76.72	83.28	94.00	**96.18**
Golf	11.88	45.00	55.00	50.00	**70.00**

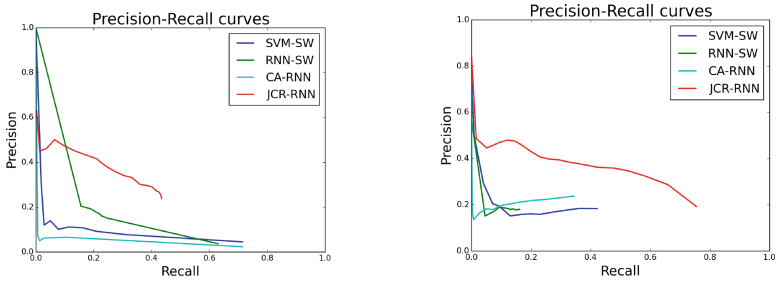
Table 6. Average running time (seconds per sequence).

SVM-SW	RNN-SW [15]	JCR-RNN
1.05	3.14	2.60

the action forecast methods. Note that both precision and recall are calculated on the frame-level for all frames.

Baselines. Since there is no previous method proposed for the action forecast problem, we use a simple strategy to do the forecast based on the above detection baseline methods. For SVM-SW, RNN-SW, CA-RNN, they will output the probability $q_{t,j}$ for each action class j at each time step t. At time t, when the probability $q_{t,j}$ of action class j is larger than a predefined threshold β_s, we consider that the action of class j will start soon. Similarly, during an ongoing period of the action of class j, when the probability $q_{t,j}$ is smaller than another threshold β_e, we consider this action to end soon.

Forecast Performance. The peak point of the regressed confidence curve is considered as the start/end point in the test. When the current confidence value



(a) Forecast of start. (b) Forecast of end.

Fig. 6. The Precision-Recall curves of the start and end time forecast with different methods on the OAD dataset. Overall JCR-RNN outperforms other baselines by a large margin. This figure is best seen in color. (Color figure online)

is higher than θ_s but ahead of the peak, this frame forecasts that the action will start soon. By adjusting the confidence thresholds θ_s and θ_e in our method, we draw the Precision-Recall curves for our method. Similarly, we draw the curves for the baselines by adjusting β_s and β_e. We show them in Fig. 6. The performance of the baselines is significantly inferior to our method JCR-RNN. This suggests that only using the traditional detection probability is not suitable for forecasting. One important reason is that the frames before the start time are simply treated as background samples in the baselines but actually they contain evidences. While in our regression task, we deal with these frames using different confidence values to guide the network to explore the hidden starting or ending patterns of actions. In addition, we note that the forecast precision of all the methods are not very high even though our method is much better, e.g., precision is 28 % for start forecast and 37 % for end forecast when recall is 40 %. This is because the forecast problem itself is a difficult problem. For example, when a person is writing on the board, it is difficult to forecast whether he will finish writing soon.

Figure 7 shows the confusion matrix of the start forecast by our proposed method. This confusion matrix represents the relationships between the predicted start action class and the groundtruth class. The shown confusion matrix is obtained when the recall rate equals to 40 %. From this matrix, although there are some missed or wrong forecasts, most of the forecasts are correct. In addition, there are a few interesting observations. For example, the action *eating* and *drinking* may have similar poses before they start. Action *gargling* and *washing hands* are also easy to be mixed up when forecasting since the two actions both need to turn on the tap before starting. Taking into account human-object interaction should help reduce the ambiguity and we will leave it for future work.

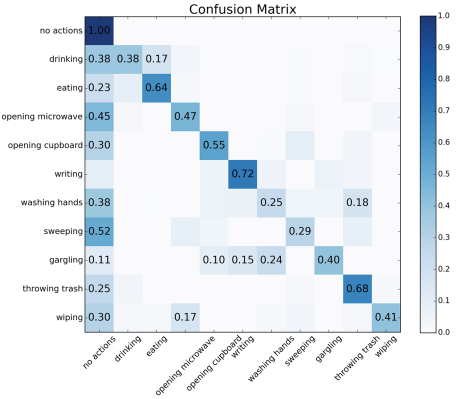


Fig. 7. Confusion Matrix of start forecast on the OAD dataset. Vertical axis: groundtruth class; Horizontal axis: predicted class.

5.4 Comparison of Running Speeds

In this section, we compare the running speeds of different methods. Table 6 shows the average running time on 9 long sequences, which has 3200 frames on average. SVM-SW has the fastest speed because of its small model compared with the deep learning methods. The RNN-SW runs slower than our methods due to its sliding window design. We can notice the running speed for the action detection based on skeleton input is rather fast, being 1230 fps for the JCR-RNN approach. This is because the dimension of skeleton is low ($25 \times 3 = 75$ values for each frame) in comparison with RGB input. This makes the skeleton based online action detection much attractive for real applications.

6 Conclusion and Future Work

In this paper, we propose an end-to-end Joint Classification-Regression RNN to explore the action type and better localize the start and end points on the fly. We leverage the merits of the deep LSTM network to capture the complex long-range temporal dynamics and avoid the typical sliding window design. We first pretrain the classification network for the frame-wise action classification. Then with the incorporation of the regression network, our joint model is capable of not only localizing the start and end time of actions more accurately but also forecasting their occurrence in advance. Experiments on two datasets demonstrate the effectiveness of our method. In the future work, we will introduce more features, such as appearance and human-object interaction information, into our model to further improve the detection and forecast performance.

Acknowledgement. This work was supported by National High-tech Technology R&D Program (863 Program) of China under Grant 2014AA015205, National Natural Science Foundation of China under contract No. 61472011 and No. 61303178, and Beijing Natural Science Foundation under contract No. 4142021.

References

1. Weinland, D., Ronfard, R., Boyerc, E.: A survey of vision-based methods for action representation, segmentation and recognition. Comput. Vis. Image Underst. **115**(2), 224–241 (2011)
2. Microsoft Kinect. https://dev.windows.com/en-us/kinect
3. Johansson, G.: Visual perception of biological motion and a model for it is analysis. Percept. Psychophys. **14**(2), 201–211 (1973)
4. Han, F., Reily, B., Hoff, W., Zhang, H.: Space-time representation of people based on 3D skeletal data: a review, pp. 1–20 (2016). arXiv:1601.01006
5. Hoai, M., De la Torre, F.: Max-margin early event detectors. Int. J. Comput. Vis. **107**(2), 191–202 (2014)
6. Zanfir, M., Leordeanu, M., Sminchisescu, C.: The moving pose: an efficient 3D kinematics descriptor for low-latency action recognition and detection. In: Proceedings of IEEE International Conference on Computer Vision, pp. 2752–2759 (2013)

7. Oneata, D., Verbeek, J., Schmid, C.: The LEAR submission at THUMOS 2014 (2014)
8. Siva, P., Xiang, T.: Weakly supervised action detection. In: British Machine Vision Conference, Citeseer, vol. 2, p. 6 (2011)
9. Wang, L., Qiao, Y., Tang, X.: Action recognition and detection by combining motion and appearance feature (2014)
10. Sharaf, A., Torki, M., Hussein, M.E., El-Saban, M.: Real-time multi-scale action detection from 3D skeleton data. In: Proceedings of IEEE Winter Conference on Applications of Computer Vision, pp. 998–1005 (2015)
11. Wang, L., Wang, Z., Xiong, Y., Qiao, Y.: CUHK&SIAT submission for THUMOS15 action recognition challenge (2015)
12. Wu, Z., Wang, X., Jiang, Y.G., Ye, H., Xue, X.: Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In: Proceedings of ACM International Conference on Multimedia (2015)
13. Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 2625–2634 (2015)
14. Du, Y., Wang, W., Wang, L.: Hierarchical recurrent neural network for skeleton based action recognition. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1110–1118 (2015)
15. Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., Xie, X.: Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In: AAAI Conference on Artificial Intelligence (2016)
16. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
17. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems, pp. 568–576 (2014)
18. Wei, P., Zheng, N., Zhao, Y., Zhu, S.C.: Concurrent action detection with structural prediction. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 3136–3143 (2013)
19. Tian, Y., Sukthankar, R., Shah, M.: Spatiotemporal deformable part models for action detection. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 2642–2649 (2013)
20. Ryoo, M.S.: Human activity prediction: early recognition of ongoing activities from streaming videos. In: Proceedings of IEEE International Conference on Computer Vision, pp. 1036–1043 (2011)
21. Jain, M., Van Gemert, J., Jégou, H., Bouthemy, P., Snoek, C.G.: Action localization with tubelets from motion. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 740–747 (2014)
22. Yu, G., Yuan, J.: Fast action proposals for human action detection and search. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1302–1311 (2015)
23. Böck, S., Arzt, A., Krebs, F., Schedl, M.: Online real-time onset detection with recurrent neural networks. In: Proceedings of IEEE International Conference on Digital Audio Effects (2012)
24. Wollmer, M., Blaschke, C., Schindl, T., Schuller, B., Farber, B., Mayer, S., Trefflich, B.: Online driver distraction detection using long short-term memory. IEEE Trans. Intell. Transp. Syst. **12**(2), 574–582 (2011)

25. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks. SCI, vol. 385. Springer, Heidelberg (2012)
26. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: Kremer, S.C., Kolen, J.F. (eds.) A Field Guide to Dynamical Recurrent Neural Networks. IEEE Press, Los Alamitos (2001)
27. Werbos, P.J.: Backpropagation through time: what it does and how to do it. Proc. IEEE **78**(10), 1550–1560 (1990)
28. Glocker, B., Pauly, O., Konukoglu, E., Criminisi, A.: Joint classification-regression forests for spatially structured multi-object segmentation. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7575, pp. 870–881. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33765-9_62
29. Schulter, S., Leistner, C., Wohlhart, P., Roth, P.M., Bischof, H.: Accurate object detection with joint classification-regression random forests. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition, pp. 923–930 (2014)
30. Li, W., Zhang, Z., Liu, Z.: Action recognition based on a bag of 3D points. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition Workshops, pp. 9–14 (2010)
31. Yun, K., Honorio, J., Chattopadhyay, D., Berg, T.L., Samaras, D.: Two-person interaction detection using body pose features and multiple instance learning. In: Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition Workshops, pp. 28–35 (2012)
32. Bloom, V., Makris, D., Argyriou, V.: G3D: a gaming action dataset and real time action recognition evaluation framework. In: Proceedings of International Conference on Computer Vision and Pattern Recognition Workshops, pp. 7–12 (2012)
33. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)
34. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: Proceedings of International Conference on Machine Learning, vol. 96, pp. 148–156 (1996)