Skeleton Based Temporal Action Detection with YOLO

Jun Wu¹, Yu Li¹, Liuqing Wang², Ke Wang¹, Ruifeng Li¹, Tianxiang Zhou¹
¹Harbin Institute of Technology, Harbin 150001, China
²China Mobile (Hangzhou) Information Technology Co., Ltd, Hangzhou, 311100, China
wangke@hit.edu.cn

Abstract. Detecting actions in untrimmed sequences is an important yet challenging task. In this paper, we innovatively transform the temporal action detection issue into the object detection issue. Our method allows for real-time detection and end-to-end training. It consists of two stages. Firstly, we propose an idea to represent action sequences as images as well as preserving the original temporal dynamics and spatial structure information. Secondly, based on such description, we design a one-dimensional YOLO network to detect human action. In addition, we make a dataset for skeleton based temporal action detection. Experiments on our dataset demonstrate the superiority of our method.

1. Introduction
Due to the huge demand in many fields such as human-computer interaction, game control and intelligent monitoring, the action recognition of the human body has been increasingly important in the field of computer vision. In order to recognize action from different scenes, people mainly use discriminant feature to describe and classify action.

In general, the current common methods to recognize human action are based on RGB and depth data. With the great success of CNNs [1] in the field of image classification, people try to process original image with multiple convolution layers and pooling layers such that computer can learn action features automatically. Compared with image classification, action has features with respect to time. Also, CNNs for action recognition is always complex. Most action recognition methods [2, 3] based on RGB data are implemented in two steps. Firstly, create special CNNs with static images. Secondly, merge them in time domain. However, RGB data is easily affected by complex lighting conditions and cluttered backgrounds, so human action recognition systems based on RGB data lack ability to be generalized when given new scenes.

With the rapid development of imaging technology in extracting depth information, people are increasingly concerned with using depth data generated by depth sensors to solve these problems. Compared with RGB data, depth data generated by depth sensors is more robust to changes of lighting conditions. Also, depth data makes it easier to extract valid information from cluttered backgrounds. In addition, the depth image generated by the depth camera provides three-dimensional information about the object. The human body can be described as a system with joints and rigid bones. The action of the human body can be described as the movement of the bones. As Kinect being able to capture skeleton information based on depth data [4], human action recognition [5] is mainly divided into two steps. Firstly, recognize skeleton with depth data. Secondly, recognize the action based on skeleton.

At present, however, many human action recognition methods have the following problem. Existing methods usually test their own recognition algorithms on some human action datasets, such
as Two-stream CNNs put forward by Simonyan et al. [6], which achieves 88% accuracy rate on the UCF101 dataset. There are 3783 videos and 19 frames are sampled in each video. The final prediction is defined as the voting result of all 19 predictions. However, there is a very important problem in practice, that is, temporal action detection. Since data sets are labelled and segmented manually, every video or skeleton sequence contains a certain action. Nevertheless, in practice, it is hard to determine which period to analyse.

Fig. 1. Pipeline of our method.

Pipeline of our method is shown in Fig. 1. In order to solve the problem mentioned above, this paper novelly put forward a temporal action detection algorithm based on YOLO [7, 8]. The method has following contributions:

1. Innovatively transform the temporal action detection issue into the object detection issue. Firstly, action sequences are encoded as RGB images while retaining the temporal dynamic information and spatial structure information of the original action. Secondly, based on such description, design a one-dimensional YOLO network to detect skeleton-based action sequence.

2. A dataset for validating the temporal action detection algorithm was built with Kinect V2. The dataset contains 100 human action sequences, including 7 actions. The data is in the form of RGB sequence and skeleton sequence. Every sequence contains 0~10 actions which are used to test the algorithm proposed by this paper.

2. From Skeleton Sequences to RGB Images
Since the position information, which is \((x, y, z)\), of each joint is corresponding to RGB, we can transform the spatial position of the joints into RGB. That is to say \(R^f = [x_1^f, x_2^f, \ldots, x_N^f]\), \(G^f = [y_1^f, y_2^f, \ldots, y_N^f]\), \(B^f = [z_1^f, z_2^f, \ldots, z_N^f]\).

Fig. 2. Pipeline of our data encoding method.
As shown in Fig. 2, all frames are finally arranged in the order of time to represent the entire sequence of actions. The horizontal axis of the picture represents the time information of the action sequence, and the vertical axis represents the spatial position information of the action sequence.

In this case, the global discrimination is pretty obvious and spatial distribution of action features of each part is very clear. The next thing we do is to create an affine map that converts the spatial coordinates into RGB components:

\[
\begin{align*}
R_n^f &= floor(255 \times \frac{x_n^f - x_{\min}}{x_{\max} - x_{\min}}) \quad (1) \\
G_n^f &= floor(255 \times \frac{y_n^f - y_{\min}}{y_{\max} - y_{\min}}) \quad (2) \\
B_n^f &= floor(255 \times \frac{z_n^f - z_{\min}}{z_{\max} - z_{\min}}) \quad (3)
\end{align*}
\]

Where, \(R_n^f, G_n^f, B_n^f\) represent the pixel values of the \(n\)th column of the \(f\)th row. And the \(floor\) function represents the rounding down. \(x_{\max}, y_{\max}, z_{\max}\) are the maximum coordinate values of all joint coordinates in the training set. \(x_{\min}, y_{\min}, z_{\min}\) are the minimum coordinate values of all joint coordinates in the training set.

3. Our YOLO Network

Since spatio-temporal skeleton sequences are encoded into RGB images, the time action detection problem can be handled by object detection method. Compared with target detection of two-dimensional images, our problem is simpler since action detection is a one-dimensional problem and the only thing to do is to detect the start time of action.

At present, most detection algorithms are mainly based on deep learning models, which can be divided into two categories: (1) Two-stage detection algorithm, which divides the detection problem into two stages. First, it generates region proposals, and then classify the candidate regions (generally also requires position refinement). The typical example of such algorithms is R-CNN algorithms [9] based on region proposals. (2) One-stage detection algorithm, which does not require the region proposals and generates the probability and position of the object directly. Typical algorithms are like YOLO and SSD [10]. YOLO are convolutional neural networks that can predict multiple box positions and categories at one time. It can realize end-to-end target detection and recognition. Its advantage is that it is fast. YOLO does not choose the sliding window or extracts proposals to train the network, but selects the whole image training model. The advantage of doing this is that you can better distinguish between the target and the background area.

Our system divides the input image into \(S \times 1\) grid. If the center of an action falls into a grid cell, that grid cell is responsible for detecting that action. Each grid cell predicts \(A\) bounding boxes at each cell according to \(A\) anchor box in the output feature map. The network predicts 3 coordinates \((x, w, o)\) and \(C\) class probabilities for each bounding box.

Given the anchor box has with \(p_w\), in the case that the cell is offset from left corner of image by \(c_x\), the predictions is:

\[
\begin{align*}
b_x &= \sigma(x) + c_x \quad (4) \\
b_w &= p_w e^w \quad (5) \\
Pr(action) * IOU(box, action) &= \sigma(o) \quad (6)
\end{align*}
\]

Where \(b_x\) and \(b_w\) is the center coordinate of anchor box and width. The confidence score \(o\) effect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally, \(Pr(action) \in \{0,1\}\). The confidence score would be zero if there is no action in the cell. \(\sigma\) is the sigmoid function.

In summary, we divide the picture into \(S \times 1\) grids where each of them predict \(A\) bounding boxes. Thus, the output is a tensor with size \(S \times 1 \times A \times (3 + C)\).
3.1. Network Design

As shown in Fig. 3, we implement this model as a Fully Convolutional Network (FCN), which reduces the amount of parameters. Our network has 5 convolutional layers and 4 max pooling layers. We add batch normalization on all of the convolutional layers in our network. Batch normalization leads to significant improvements in convergence while eliminating the need for other forms of regularization. Batch normalization also helps regularize the model. We choose LeakyReLU as the activation function:

\[
y = \max(0, x) + \alpha \min(0, x)
\]  

In the equation above, \( \alpha \) is a very small positive number to preserve the information of negative axis. Fig. 3 shows the full network. We set \( S = 12 \) and \( A = 5 \). And the action category of our dataset is 7. The final output is the \( 12 \times 1 \times 5 \times (3 + 7) \) tensor of predictions.

3.2. Loss Function

YOLO algorithm regards target detection as a regression problem and uses the mean square error loss function. We divide the loss into positioning error and classification error.

For the positioning error, the prediction error of box position, a larger weight is used, where \( \lambda_{\text{coord}} = 5 \). Then, it distinguishes the bounding box that contains the target from the bounding box that does not contain the target. For the latter, a smaller weight value \( \lambda_{\text{noact}} = 0.5 \) is used. All other weight values are set to 1. Then we use the mean square error to treat the bounding boxes of different sizes equally. However, the coordinate error of smaller bounding box should be more sensitive than those of larger ones. With this in mind, the prediction of the width of the bounding box is changed to its square root. That is to say, the predicted value becomes \((x, \sqrt{w})\).

For a bounding box without action, the only error is confidence, because coordinate errors cannot be calculated. For classification errors, the classification errors are calculated only if there is an action in the cell. Final loss function:

\[
L = \lambda_{\text{coord}} \sum_{i=0}^{S} \sum_{j=0}^{A} \frac{1}{N_{ij}} (x_i - \hat{x}_i)^2 + \lambda_{\text{coord}} \sum_{i=0}^{S} \sum_{j=0}^{A} \frac{1}{N_{ij}} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2
\]
\[
+ \sum_{i=0}^{S} \sum_{j=0}^{A} \text{act}_{ij} (o_i - \hat{o}_i)^2 \\
+ \lambda_{\text{noact}} \sum_{i=0}^{S} \sum_{j=0}^{A} \text{noact}_{ij} (a_i - \hat{a}_i)^2 \\
+ \sum_{i=0}^{S} \sum_{j=0}^{A} \text{act}_{ij} (p_i(c) - \hat{p}_i(c))^2 
\]  

(8)

The first item is the center error of the bounding box, where \( \text{act}_{ij} \) means that the \( i \)th cell has a target and the \( j \)th bounding box in the cell is to predict the target. The second item is the error term of the width of the bounding box. The third is to consider the confidence error term of the target bounding box. The fourth term is the confidence error term that does not consider the bounding box of the target. The last item is the classification error item of the cell containing the target.

4. **Experiments**

4.1. **Our Action Dataset**

Since the current data set used to verify the temporal action detection algorithm is usually based on RGB image sequences, we collected dataset for 7 actions (Sit to Stand, Stand to Sit, Lying Down, Get Up, Eat, Read and Drink). In order to improve the ability of generalization, 3 subjects were selected in 2 scenes, which take factor of environment and people into consideration. The device to collect data is Kinect V2. Examples of our dataset is shown in Fig. 4.

![Fig 4. Our action dataset examples.](image)

The dataset contains 100 human skeleton sequences. Each subject in the sequence performs 0~10 actions according to the instructions. Since the Kinect V2 can provide information of 25 joints, we set the network input image size to 25×200. The labelling and segmentation of the dataset is done manually. After the data enhancement, our final data contains 2000 RGB images converted from the skeleton sequence, and the image size is 25×200. They are divided into a training set and a test set.

4.2. **Anchor boxes**

We use the K-means clustering algorithm to select the anchor box. The algorithm is as follows:

Input: \( D = \{l_1, l_2, ..., l_M\} \), where \( l \) represent the duration of action in training set.

Iteration threshold \( \delta \).

Output: Result of clustering.

1. Randomly select \( k \) cluster centers
   \[ \{\text{Center}_1, \text{Center}_2, ..., \text{Center}_k\} \]
2. For \( t = 1, 2, ..., T \)
3. For every \( l_i \)
4. \[ \text{dist}(l_i, \text{Center}_k) = (l_i - \text{Center}_k)^2 \]
5. Make \( l_i \) into the nearest cluster
6. **End for**

   **Update cluster centers:**

   \[ \mu_k = \frac{1}{|C_k|} \sum_{l_t \in C_k} l_t \], where \( C_k \) represents \( k \)th cluster, \(|C_k|\) represents the number of data in \( k \)th cluster

7. **Calculate difference between two iterations:**

   \[ \Delta J = J_t - J_{t-1} \]

8. **If** \( \Delta J < \delta \)

9. **Then output clusters**

10. **Break**

11. **End if**

12. **End for**

According to the above algorithm, we set \( K = 5 \). There are two anchor boxes with width \( p_w \in \{0.6, 1.2, 2.0, 2.8, 3.75\} \).

### 4.3. Training

The Adam gradient descent method is used for training, and the mini-batch size is set to 128. The training process is shown in Fig. 5.

![Fig 5. Convergence curves on our action dataset.](image)

Since the size of input image is small, the size of network is small. Also, because our network is FCN, which reduces a large number of parameters of the fully connected layer, the training process is very fast, and it converged in 60 epochs of iteration. The model can converge quickly. According to the data volume of our dataset, our model can converge within 1 hour on a normal computer.

### 4.4. Predict

![Fig 6. mAP-tIOU curve on our dataset.](image)
The predict phase is the same as YOLO. We set confidence threshold = 0.5, max boxes = 10, tIoU (temporal intersection over union) threshold = 0.2 and use the Non-Maximum Suppression (NMS) algorithm. The final mAP@tIOU=0.5 is 81.88% for our dataset. Fig. 6 shows the tIOU-mAP curve for our method.

Two examples of the final test result are shown in Fig. 7. The black box is the label and the blue dotted box is our prediction. The number in parentheses is confidence score. The experimental results show that our algorithm can effectively effective actions from the skeleton sequence. Our algorithm not only categorizes actions, but also determines the start and end times of actions. What’s more, our algorithm is fast and can be used for real-time temporal action detection.

5. Conclusion
In this paper, we have proposed a skeleton-based temporal action detection algorithm, detecting in real time and training from end to end. Firstly, action sequences are encoded as RGB images while retaining the temporal dynamic information and spatial structure information of the original action. Secondly, based on such description, we design a one-dimensional YOLO network to detect human action. Finally, experiments on our dataset demonstrate the superiority of our method.

Although human skeleton recognition technology of Kinect is relatively mature, its performance is not stable in the case of occlusion. And this will directly affect our algorithm. So next we will continue to optimize the human skeleton recognition algorithm and strive to obtain stable and accurate human skeleton data. Another problem with our algorithm is that it can only be used to detect one person's actions. When some actions are done by multiple people, it can not recognize and segment the action well.

Acknowledgments
This work is supported by the Joint Research Fund for The Ministry of Education of China and China Mobile (MCM20170208) and the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No.51521003).

References
[1] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks[C]// International Conference on Neural Information Processing Systems. Curran Associates Inc. 2012:1097-1105.
[2] Wang L, Qiao Y, Tang X. Action recognition with trajectory-pooled deep-convolutional descriptors[C]// IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society, 2015:4305-4314.
[3] Wang P, Cao Y, Shen C, et al. Temporal Pyramid Pooling Based Convolutional Neural Network for Action Recognition[J]. IEEE Transactions on Circuits & Systems for Video Technology, 2015, PP(99):1-1.
[4] Zhang Z. Microsoft Kinect Sensor and Its Effect[J]. IEEE Multimedia, 2012, 19(2):4-10.
[5] Du Y, Fu Y, Wang L. Skeleton based action recognition with convolutional neural network[C]// Pattern Recognition. IEEE, 2016:579-583.
[6] Simonyan K, Zisserman A. Two-stream convolutional networks for action recognition in videos[J]. 2014, 1(4):568-576.
[7] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[J]. 2015.
[8] Redmon J, Farhadi A. [IEEE 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - Honolulu, HI (2017.7.21-2017.7.26)] 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) - YOLO9000: Better, Faster, Stronger[J]. 2017:6517-6525.
[9] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[J]. 2013:580-587.
[10] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[C]// European Conference on Computer Vision. Springer International Publishing, 2016:21-37.