Gesture Recognition Based on Deep Learning

Tingxuan Zhang*

International School, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding author e-mail: zhangtingxuan@bupt.edu.cn

Abstract. In the modern society, the technology of artificial intelligence is playing a more and more important role. It is well known that we can use the deep learning to do the human gesture recognition which is quite helpful for us. For example, if our system finds a student who is playing his cell-phone in the classroom, the system will give him a very low score in this lecture. For another example, if some students are taking notes carefully, our system will give them very high scores. In this project, I used the knowledge of the deep learning which is quite hot recently, and applied the knowledge of human key point detection to realize behavior recognition.

1. Introduction

Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields is a very hot topic. The core of this project is to propose a human body pose estimation algorithm that uses Part Affinity Fields (PAFs) bottom-up algorithm (getting the key point position to obtain the skeleton) instead of the traditional top-down algorithm (first detecting people and then return to the key point). I learned about OpenPose which is a real-time multi-person key point detection of CMU open source, including human key points, hand key points, face key point detection, and pose estimation. I decided to use OpenPose to solve the gesture recognition problem in the cloud classroom environment. After obtaining the key points of the human body, I could obtain some key angles by calculating the positional relationship between the key points, and use these angles and positions to estimate the gesture of the student in the video. The system could determine the six most common gestures in the classroom: listening, taking notes, playing cell phone, sleeping, raising hand, and standing. Although the code could run very fast on the server, it runs slower on my laptop because of the poor configuration of my laptop and there is no way to make real-time judgments on my laptop. I tried to combine Openpose with YOLO to make the system run faster, but there is still no way to solve the problem of slow speed on my laptop, and the average calculation time of one frame was more than 1 second. In order to achieve better results, I changed this real-time system to a system that does a recognition every 3 seconds. Finally, I combined the part of expression recognition with the part of gesture recognition. The system can do a recognition every 3 seconds, and record the results of each student's performance, and then send them to the teacher in real time. It is convenient for teachers to adjust their teaching styles for students' performance. What's more, the system also generates a student's classroom performance report and the score of the performance in the class in order to realize the interactive behavior recognition.

2. Background

Common Objects in Context (COCO): I use the COCO dataset to deal with the part of gesture recognition in the cloud classroom system. The COCO dataset is used for the OpenPose key points detection in the project. The COCO dataset prepared by Microsoft is a large image dataset designed for

object detection, segmentation, human key point detection, semantic segmentation and caption generation. The target of this dataset is scene understanding, which is mainly captured from complex daily scenes. The targets in the image are marked by precise segmentation. The COCO dataset includes a large number of images which are more than 200000 and 250000 person instances in this dataset are marked with key points. What's more, most of people in the COCO dataset are at medium scales and large scales. Some examples of COCO dataset are shown in Figure 1.[1]

Figure 1. Some examples of COCO dataset

OpenPose: OpenPose is a well-known open source library which is based on convolutional neural network and supervised learning. It can realize the tracking of human facial expressions, torso and limbs and even fingers. This algorithm is quite appropriate for both single person and multi-person detection. This algorithm comes from "Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields", a paper by CVPR 2017, written by Cao Zhe from the CMU Perceptual Computing Laboratory. [2]

In the OpenPose, we should enter an image, pass it through a backbone (such as VGG, Res-Net, Mobile-Net), and then go through 6 stages. Each stage has two branches, one for detecting heatmap and one for detecting vectmap. With heatmap and vectmap, you can know all the key points in the picture, and then mark the points to everyone through PAFs. The OpenPose framework is shown in Figure 2.

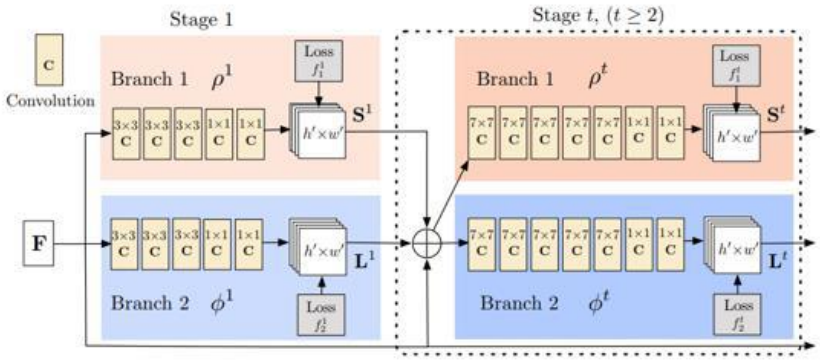

Figure 2. The OpenPose framework

3. Human gesture recognition

I need to use OpenPose to realize the key point recognition of the human body, and then estimate the posture of the person in the picture by the positional relationship between the marked key points. In addition, the system can send the detected gesture of the students to the teacher, and score the students' gestures at this time to reflect the performance of each student in the class. It can help teachers to have an intuitive understanding of the students' class performance.

OpenPose can estimate the pose of a single human target in the image, and can also handle the pose estimation of multiple people in the image. The input of this model is an image of h×w×3 and this model can output two arrays containing the confidence maps of the key points and the part affinity heatmaps of each key point pair. The top 10 layer of VGG19 is used to extract the feature maps of the input image. And then, 2-branch multi-stage CNN structure is used. The network structure of OpenPose is shown in Figure 3.

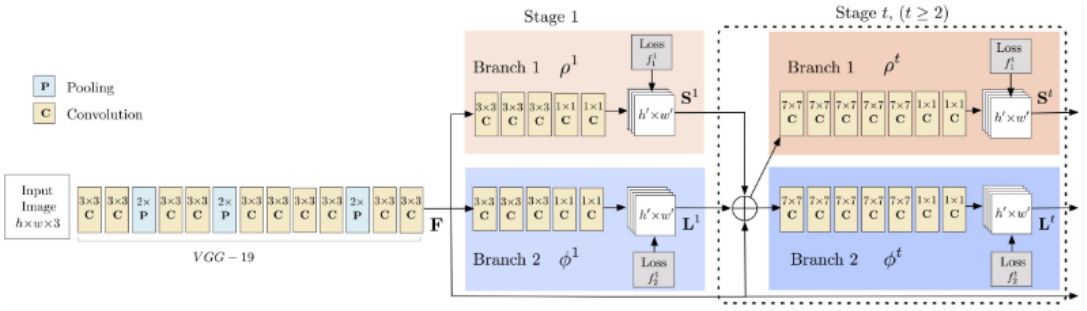

Figure 3. The network structure of OpenPose

After getting the input image, I used the first 10 layers of the VGG19 network to extract the features of the input image. One improvement of the VGG over AlexNet is the use of several consecutive 3×3 convolution kernels instead of the larger convolution kernels in AlexNet (11×11, 7×7, 5×5). The main purpose of it is to improve the depth of the network and to improve the effect of the neural network to a certain extent under the condition of ensuring the same perception field. For example, the layer superposition of three 3×3 convolution kernels with the stride of 1 can be regarded as a receptive field of size 7 which means that three 3×3 continuous convolutions are equivalent to a 7×7 convolution. [3] The number of total parameters of three 3×3 continuous convolutions is $3\times(3\times3\times C2)$. If the 3×3 convolution kernel is used directly, the number of total parameters is $7\times7\times C2$, where C refers to the number of channels. Obviously, $27\times C2$ is less than $49\times C2$, which means that three 3×3 continuous convolutions can reduce the parameters; and the 3×3 convolution kernel helps to better maintain the image properties. What's more, we used 3 nonlinear functions instead of 1 convolution, which increased the discriminating power of the function. The framework of VGG19 is shown in Figure 4. [4]

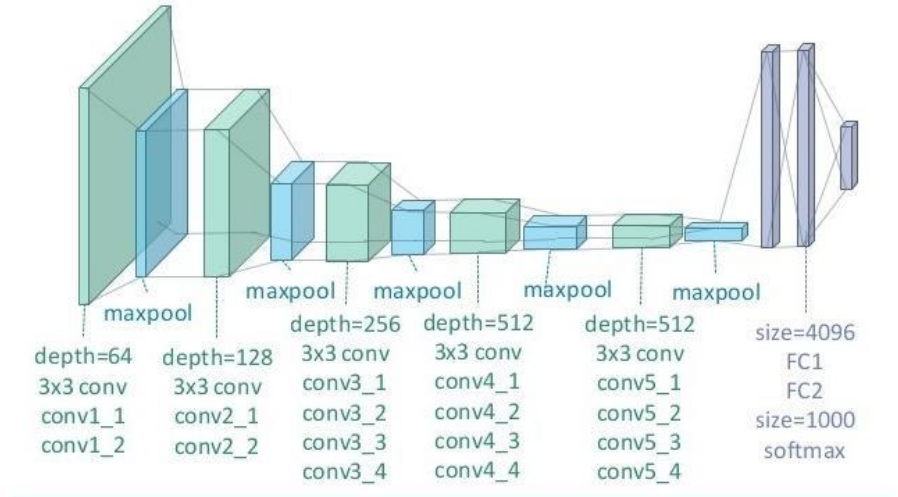

Figure 4. The framework of VGG19

After the first 10 layers of the VGG19 network, we can get the feature F. Feature F is processed through a continuous multi-stage network. Each phase (t) of the network contains two branches, and the

input results are S(t) (Part Confidence Map) and L(t) (Part Affinity Map). S(t) tells us where the head is and where the elbow is; L(t) tells us which places are definitely on which leg. With the help of L(t), the coordinate points of S(t) are connected to form a skeleton of the person's posture.

The input received in the first phase of the network is the feature F. The feature F is processed by the network to obtain S (1) and L (1) respectively. Starting from phase (2), the input to the phase (t) network consists of three parts: S(t−1), L(t−1) and feature F. The inputs to the network for each phase are:

$$St=\rho t\ (F, St-1, Lt-1), \forall t\geq 2 \quad (1)$$

$$Lt=\varphi t\ (F, St-1, Lt-1), \forall t\geq 2 \quad (2)$$

This network iterates until the network becomes converged. Figure 5 shows the results of two branch calculations for CNN.

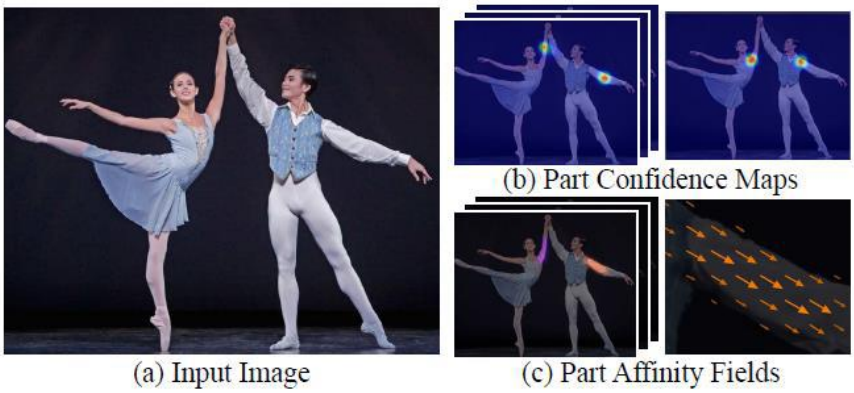

Figure 5. The results of two branch calculations for CNN

Branch one: This network branch predicts the 2D confidence maps of the position of human key points, such as elbow, knee and so on. Each confidence map is a grayscale image, and the position coordinate of its maximum value is the highest probability corresponding to a certain key point of the human body which is shown in Figure 6.

Figure 6. 2D confidence maps of the position of human key points

Branch two: This network branch predicts the 2D vector field of PAF, which represents the degree of association between two key points. For example, part affinity between key points neck and left shoulder is shown in Figure 7, the affinity value between the key points belonging to the same person is relatively large.

Figure 7. 2D vector field of PAF

In order to determine whether the network is convergent, we define the loss function of the network:

$$\mathrm{f} = \sum_{\mathrm{t}=1}^{\mathrm{N}} f_S^t + f_L^t \tag{3}$$

f_S^t and f_L^t represent the error conditions of the two output images respectively. Their formulas are

$$f_S^t = \sum_{j=1}^{J} \sum_P W(P) \cdot \left\| S_j^t(p) - S_j^*(p) \right\|_2^2 \tag{4}$$

$$f_L^t = \sum_{c=1}^{C} \sum_P W(P) \cdot \| L_c^t(p) - L_c^*(p) \|_2^2 \tag{5}$$

t represents the stage.

$S_j^*(p)$ represents the ground-truth of part confidence map and it can be obtained by:

$$S_{j,k}^*(p) = \exp\left(-\frac{\left\| p - x_{j,k} \right\|_2^2}{\sigma^2}\right) \tag{6}$$

$$S_j^*(p) = \max\left(S_{j,k}^*(p)\right) \tag{7}$$

$L_c^*(p)$ represents the ground-truth of part affinity vector field and it can be obtained by:

$$L_{c,k}^*(p) = \begin{cases} v & if\ p\ on\ limb\ c,k \\ 0 & others \end{cases} \tag{8}$$

$$L_c^*(p) = \frac{1}{n_c(p)} \sum_k L_{c,k}^*(p) \tag{9}$$

W represents a binary mask. If W(p)=0, it indicates that the current point p is missing (not visible or not in the image) in order to avoid wrong punishment during training.

Finally, we can splice the key points and the torsos to complete the required model. We can get all parts of the body and the torso, and the two adjacent torsos must have shared joint point. By combining all the torsos through the joint points, we can get the body skeleton of all people.

Through training for the ultimate network, I got the model: model.h5. I need to read the neural network, then adjust the size of the output to the same as the input, and then check the confidence map of the key point. We should save the (x, y) coordinates and the probability scores for each key point, and perform key point detection on the input image. And then use Heatmap to find the valid connection pair. Finally, combine all the key points belonging to the same person to draw the skeleton map.

In the part of gesture recognition, we need to recognize six kinds of gesture which are sitting, taking note, playing cell phone, sleeping, raising hand and standing. In order to realize the recognition of these

gestures, we can record the coordinates of the key points of the human body, and then calculate the positional relationship between the various parts of the human body through the coordinates of each key point. We only need to know the coordinates of the three points to find out the angle formed by the three sides, and then use the range of values of these angles to infer the pose of the person in the image. Formulas for calculating these angles are shown below:

$$\text{A } (x_1, y_1) \quad \text{B } (x_2, y_2) \quad \text{C } (x_3, y_3)$$

$$\text{Vector AB: } (x_2 - x_1, y_2 - y_1)$$

$$\text{Vector AC: } (x_3 - x_1, y_3 - y_1)$$

$$\text{Vector BC: } (x_3 - x_2, y_3 - y_2)$$

$$\cos \angle A = \frac{[(x_2 - x_1)(x_3 - x_1) + (y_2 - y_1)(y_3 - y_1)]}{|AB||AC|} \tag{10}$$

$$|AB| = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{0.5} \tag{11}$$

$$|AC| = [(x_3 - x_1)^2 + (y_3 - y_1)^2]^{0.5} \tag{12}$$

Finally, through OpenCV, I can use the camera to record the performance of the students in front of the screen in the cloud classroom, then send the student's gesture to the teacher, and record the score of student's gestures.

I use the OpenPose to do the human key points recognition. Through this model, we can get the coordinates of the 18 key points of the human body. We compared our approach with some of the methods proposed by previous people, and the result is shown in Table 1. Through the comparison of the experimental results in the table, we can find that the method we use has a very significant improvement in the accuracy of recognition compared to other methods.

Table 1. Comparison of some different methods

	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	mAP
Deepcut [18]	73.4	71.8	57.9	39.9	56.7	44.0	32.0	54.1
Iqbal et al. [19]	70.0	65.2	56.2	46.1	52.7	47.9	44.5	54.7
DeeperCut [20]	87.9	84.0	71.9	63.9	68.8	63.8	58.1	71.2
Our method	94.1	91.2	81.5	72.7	78.1	72.9	67.9	80.2

Our final system only needs to recognize six gestures. However, the COCO dataset does not have data specifically used to assess students' gestures in the classroom, so I took 200 photos with the camera to test the accuracy of these four poses. The test accuracy confusion matrix of the six gesture recognitions is shown in Table 2.

Table 2. The test accuracy confusion matrix of the six gesture recognitions

		Predicted value					
Actual value		Standing up	Raising hand	Taking notes	Listening	Playing cell phone	Sleeping
	Standing up	87.5	0	0	11	0	1.5
	Raising hand	0	92	1	6	0	1
	Taking notes	3.5	1	79	6	9.5	1
	Listening	6	0	1	90	2	1
	Playing cell phone	0	0	13	3.5	83	0.5
	Sleeping	5.5	0	0	0	0	94.5

4. Conclusion

I completed the pre-processing of the dataset. I originally planned to use the EgoHands dataset, but found that it did not meet my needs, and later switched to the COCO dataset. And then I used OpenPose

which is an open source library based on convolutional neural networks and supervised learning and developed with Caffe as the framework for human key points detection. After getting the key points of the human body, I could find the angle formed by each part of the human body through the coordinates of each key point, and then infer the posture of the person in the image. In the end, our system can recognize four human poses with an accuracy of 75%.

Reference

[1] COCO dataset, (2015). COCO 2018 Keypoint Detection Task. [online] Available from: http://cocodataset.org/#overview [Accessed 5th April 2019].
[2] Cao, Z., Simon, T., Wei, S. and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7291-7299.
[3] Andrew, NG. (2018). CS229: Machine Learning. [Stanford University]. San Francisco.
[4] Fang, H., Xie, S., Tai, Y. and Lu, C. (2017). RMPE: Regional multi-person pose estimation. 2017 IEEE International Conference on Computer Vision. pp. 2334-2343.