SittingPoseNet - Real time sitting pose estimation

Deep Learning (1002) - Group 3

Lin Xiaotao, Ma Zhuoheng, Xiao Dingwen, Zheng Yuhan

Abstract - In this project, a deep learning model for sitting position detection is proposed. We constructed a Transformer architecture and used Openpose as the encoder, training the decoder using dataset collected by ourselves and doing the finetuning for the encoder. By using the Openpose based SittingPoseNet, we build the camera-opsed sitting position detection model with a GUI.

Keywords: ***Camera, OpenPose, SittingPoseNet, Sitting Position Detection***

1 Introduction

Sitting position detection is a task of capturing kinematic parameters of the human upper part of the body. Nowadays, this task is implemented based on tree-structural graphical models, CNNs, ResNet, and so on. However, due to the light and background varieties in real practice, model performance may not be stable. Meanwhile, existing models running time are comparatively slow. In this project, a real-time OpenPose-based detection application with the GUI is accomplished. For the application, the computer camera keeps recording the target user's sitting position. Once any bad sitting position is caught, the screen of laptop would be locked by an additional layer which prevents the user to operate any move. The lock-down state is canceled until the user maintains a good sitting position again.

2 Related Work

Since the sitting position is a sub task under the domain of pose detection and estimation. The related work may not directly focus on sitting position detection. The traditional single human pose estimation approach is to perform reasoning on the combination of body's local observations and their spatial dependencies. This kind of models for articulated pose is either based on tree-structured graphical models, which encode the key points and pairs following a kinematic chain [7], [8], [9], [10], [11], [12], [13], or non-tree models which expand the tree structure with new-coming edges to capture symmetry, occlusion, and long-range relationships [14], [15], [16], [17], [18]. Besides, many studies devotes to solve problem by using Convolutional Neural Networks, which have significantly boosted the accuracy on body pose estimation [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32].

3 Methodology

3.1 SittingPoseNet

The goal of our work is to identify some key points position and further analyze whether they construct a bad sitting position. The framework of our model is encoder-decoder based, which the overall architecture is shown as follows:

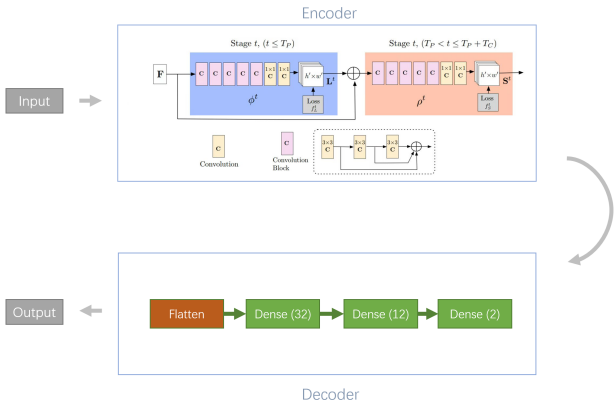

FIGURE 1. SittingPoseNet's whole architecture

3.1.1 *Encoder - Openpose*

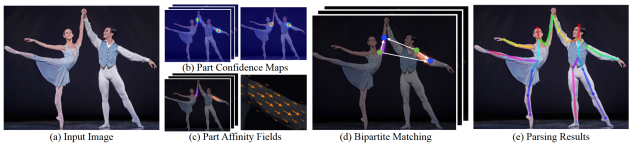

FIGURE 2. The total pipeline

The model receives a colorful picture of size $w \times h$ (Figure. 2a) as input and generates the two dimension locations of each key points of each person (Figure. 2e). Initially, a feed-forward network produces a two dimension confidence map S (Figure. 2b) and a two dimension vector fields L for part affinity fields (PAFs), which implies the angle between parts (Figure. 2c). The set $S = (S_1, S_2, ..., S_J)$ includes J confidence maps, one per part, where $S_j \in R^{w \times h}, j \in 1...J$. Another set $L = (L_1, L_2, ..., L_C)$ contains C vector fields, one per limb, where $L_c \in R^{w \times h \times 2, c} \in 1...C$. We refer to part pairs as limbs, however, some of them do not belong to human limbs, such as the face. Each image location in L_c represent a two dimension vector (Figure. 2c). Finally, the

confidence maps and the part affinity fields are parsed by a greedy inference to output the two dimension key points for everybody in the image (Figure. 2d).

3.1.2 *Network architecture*

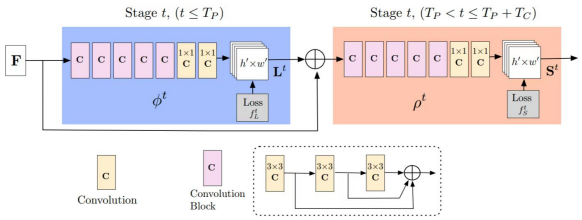

FIGURE 3. The Openpose architecture

Openpose are applied as the encoder module. For its detail architecture, it is shown in Figure 3. The yellow convolution blocks have different convolutional architectures. By making use of the residual block, the network depth increases. Meanwhile, shrinking the kernel size from 7×7 to 3×3 dramatically boosts the inference speed and while maintaining the inference performance.

3.1.3 *Simultaneous Detection and Association*

An image is processed by a CNN which initialized and fine-tuned by the first 10 layers of VGG-19 [34], producing feature maps set F, the input at the beginning stage. At this stage, the network generates part affinity fields (PAFs) $L^1 = \phi^1(F)$, where ϕ^1 denotes the CNNs for inference at the first stage. For each following stage, the concatenated result of the predictions from the last stage and the original image features F generates refined predictions,

$$L^t = \phi^t(F, L^{t-1}), \forall 2 \leq t \leq T_p \tag{1}$$

where ϕ^t denotes the CNNs for inference at stage t, and T_P represents the total number of PAF stages. After T_P iterations, the process is repeated for the confidence maps detection, starting in the most updated PAF prediction,

$$S^{T_P} = \rho^t(F, L^{T_P}), \forall t = T_P \tag{2}$$

$$S^t = \rho^t(F, L^{T_P}, S^{t-1}), \forall T_P < t \leq T_P + T_C \tag{3}$$

where ρ^t denotes the CNNs for inference at stage t, and T_C represents the total number of confidence map stages.

Such method is different from [3], where both the PAFs and confidence map branches are refined at each stage. Thus, the computation of each stage is decreased by a half. Then, confidence map results are forecast on top of the latest and most refined PAFs predictions, resulting in remarkable differences among confidence map stages. A loss function is packed at the end of each stage. It is also worth to mention that between the predictions and the ground truth maps fields is the L2 loss. For the PAF loss in stage t_i and the confidence map loss in stage t_k:

$$f_L^{t_i} = \sum_{c=1}^{C} \sum_{p} W(p) \bullet \|L_C^{t_i}(p) - L_C^*(p)\|_2^2 \tag{4}$$

$$f_S^{t_k} = \sum_{j=1}^{J} \sum_{p} W(p) \bullet \|S_j^{t_k}(p) - S_j^*(p)\|_2^2 \tag{5}$$

Then the total loss is:

$$f = \sum_{t=1}^{T_p} f_L^t + \sum_{t=T_P+1}^{T_p+T_C} f_S^t \tag{6}$$

3.1.4 *Confidence Maps for Part Detection*

The ground truth of confidence maps S^* is produced by the annotated two dimension key points. Each confidence map is a two dimension matrix of a specific body part located in the given image. For person k, let $x_{j,k} \in R^2$ be the ground truth position of body part j, the value at location $p \in R^2$ in $S_{j,k}^*$ is defined as:

$$S_{j,k}^*(p) = exp\left(-\frac{\|p - x_{j,k}\|_2^2}{\sigma^2}\right) \tag{7}$$

Where σ is a parameter that controls the peak spread. Further, we forecast the confidence maps using a max operator:

$$S_j^*(p) = \max_k S_{j,k}^*(p) \tag{8}$$

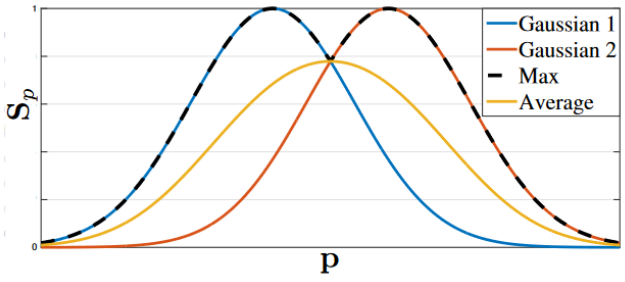

FIGURE 4. The confidence map

Taking maximum of the confidence maps instead of the average value make sure that the closed-by peaks precision keeps distinct. When performing testing, we predict confidence maps, and obtain body part candidates by conducting non-maximum suppression.

3.1.5 *Confidence Maps for Part Detection*

Each PAF is a two dimension vector which representing the direction from point $x_{j_1,k}$ to $x_{j_2,k}$ for a limb, where $x_{j_1,k}$ and $x_{j_2,k}$ are the ground truth position of body parts j_1 and j_2 from the limb c for person k. Different limbs maps to different PAF. An example is shown as below:

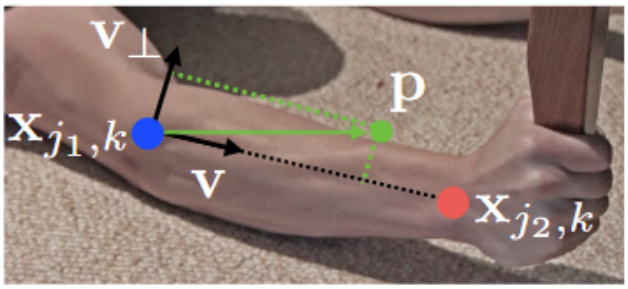

FIGURE 5. PAF (Part Affinity Fields for Part Association)

If a pixel lies on the limb, the value for $L^*_{c,k}(p)$ is a unit vector from j_1 to j_2; and for other points, the vector is zero:

$$L^*_{c,k}(p) = \begin{cases} v, & if\ p\ on\ limb\ c,k \\ 0, & otherwise \end{cases} \tag{9}$$

The unit vector $v = \frac{x_{j_2,k} - x_{j_1,k}}{\|x_{j_2,k} - x_{j_1,k}\|_2}$ is the PAF in the direction of limb. Any point in the limb is in the range of: $0 \leq v \bullet (p - x_{j_1,k}) \leq l_{c,k}$ and $|v_\perp \bullet (p - x_{j_1,k})| \leq \sigma_l$

3.1.6 *Decoder*

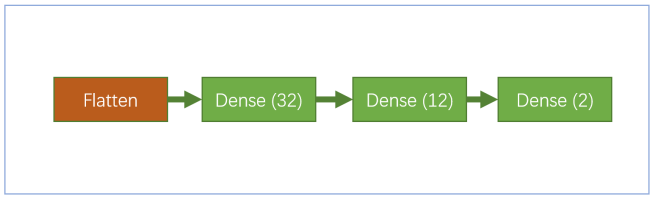

FIGURE 6. The total pipeline

The outputs of Encoder includes the PAFs and the confidence maps. These are the input of the decoder whose architecture is shown in Figure 4. The final layer of the decoder is a 2-neuron layer with the activation function SoftMax.

3.1.7 *GUI*

If an improper sitting posture is detected, we need to prevent the user continuing to use the computer until he has adjusted the sitting posture. The communication between the GUI and the model is bridged by a file. If the program detects a bad sitting position, then a flag True is written in the file, otherwise, a flag False is passed to the file. The screen would check the signal in the file frequently, if the signal True is checked, the obstructive window and warming window are moved to the top layer of the screen, preventing the user to conduct any operation. Only until the detected signal turns to False, then, the two windows would move the lowest layer, allowing the normal use of the computer.

4 Experiment

4.1 Dataset

Since there is not a public and widely-use dataset, and we try to implement a real-time detector application, data are collected by ourselves by a self-written python program. It utilizes the computer camera to capture users' sitting position. There are about 1800 pictures. Specifically, there are more than 1000 various bad sitting position pictures and about 750 good sitting position images. The data distribution are skewed because we assume that there is a great variety of bad sittings in reality.

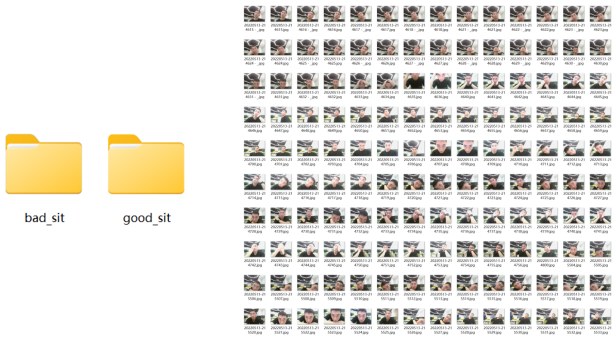

FIGURE 7. The Screenshot of Dataset

4.2 SittingPoseNet

We downloaded the Openpose pretrained network weight and compile the encoder. Then, we collected the positive and negative sitting position data by ourselves.

We used these data to perform fine tuning of our model. We chose "AdamW" as the optimizer, and cross entropy as the loss. The learning rate is set to be 0.001 initially. In order to increase accuracy, we leverage learning rate decay strategy and trained for 100 epochs. The accuracy of the sitting position is approximately 97%.

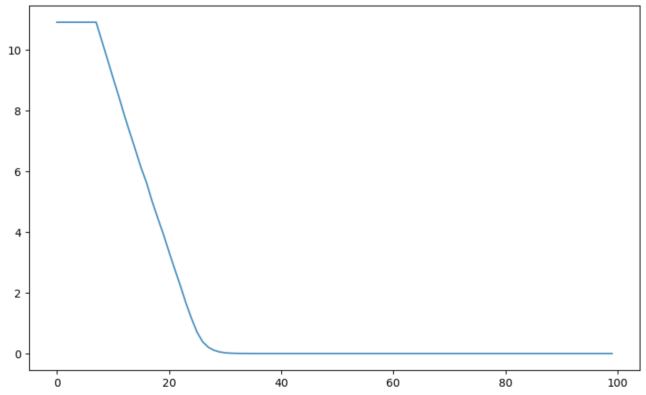

FIGURE 8. Training loss

5 Conclusion

In this project, we purpose an application with simple user graphic interface that allows the computer to detect the user's sitting position with the camera. Overall, the accuracy is pretty high (97%) in view of the size of the our dataset is not well large and all the data are self-captured.

FIGURE 9. TLE from Lin Xiaotao

6 Screenshot

Here are the four screenshots for teaching learning evaluation.

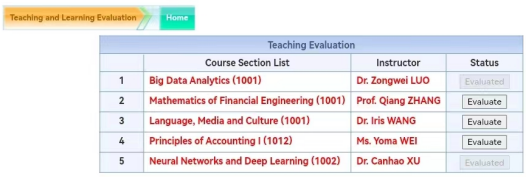

FIGURE 10. TLE from Ma Zhouheng

Teaching Evaluation			
	Course Section List	Instructor	Status
1	Principles of Photo Imaging (1002)	Mr. Rick Ruiling RAO	Evaluate
2	Big Data Analytics (1001)	Dr. Zongwei LUO	Evaluated
3	Neural Networks and Deep Learning (1002)	Dr. Canhao XU	Evaluated
4	Advanced Calculus (1001)	Dr. Xiaoyi CHEN	Evaluate

FIGURE 11. TLE from Xiao Dingwen

Here are the "yi$_j$ian$_s$an$_l$ian" $screenshots from group membl$

FIGURE 12. Lin Xiaotao's Screeshot

FIGURE 13. Ma Zhuoheng's Screenshot

FIGURE 14. Xiao Dingwen's Screenshot

FIGURE 15. Zheng Yuhan's Screenshot

7 Reference

1. L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele, "Deepcut: Joint subset partition and labeling for multi person pose estimation," in CVPR, 2016.

2. E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multi-person pose estimation model," in ECCV, 2016.

3. Kulikajevas A, Maskeliunas R, Damaševičius R. 2021. Detection of sitting posture using hierarchical image composition and deep learning. PeerJ Computer Science 7:e442
4. Arnold D, Li X, Lin Y, Wang Z, Yi W, Saniie J. 2020. Iot framework for 3d body posture visualization. IEEE International Conference on Electro Information Technology 2020:117-120
5. Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, Yaser Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. Cornell University arXiv:1812.08008 [cs.CV]
6. Ghadi Y, Akhter I, Alarfaj M, Jalal A, Kim K. 2021. Syntactic model-based human body 3D reconstruction and event classification via association based features mining and deep learning. PeerJ Computer Science 7:e764 https://doi.org/10.7717/peerj-cs.764
7. P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," in IJCV, 2005.
8. D. Ramanan, D. A. Forsyth, and A. Zisserman, "Strike a Pose: Tracking people by finding stylized poses," in CVPR, 2005.
9. M. Andriluka, S. Roth, and B. Schiele, "Monocular 3D pose estimation and tracking by detection," in CVPR, 2010.
10. ——, "Pictorial structures revisited: People detection and articulated pose estimation," in CVPR, 2009.
11. L. Pishchulin, M. Andriluka, P. Gehler, and B. Schiele, "Posele conditioned pictorial structures," in CVPR, 2013.
12. Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," in TPAMI, 2013.
13. S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in BMVC, 2010.
14. Y. Wang and G. Mori, "Multiple tree models for occlusion and spatial constraints in human pose estimation," in ECCV, 2008.
15. L. Sigal and M. J. Black, "Measure locally, reason globally: Occlusion-sensitive articulated pose estimation," in CVPR, 2006.
16. X. Lan and D. P. Huttenlocher, "Beyond trees: Common-factor models for 2d human pose recovery," in ICCV, 2005.
17. L. Karlinsky and S. Ullman, "Using linking features in learning non-parametric part models," in ECCV, 2012.
18. M. Dantone, J. Gall, C. Leistner, and L. Van Gool, "Human pose estimation using body parts dependent joint regressors," in CVPR, 2013.
19. A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in ECCV, 2016.
20. S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines," in CVPR, 2016.
21. W. Ouyang, X. Chu, and X. Wang, "Multi-source deep learning for human pose estimation," in CVPR, 2014.
22. J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," in CVPR, 2015.
23. J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in NIPS, 2014.
24. X. Chen and A. Yuille, "Articulated pose estimation by a graphical model with image dependent pairwise relations," in NIPS, 2014.
25. A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in CVPR, 2014.
26. V. Belagiannis and A. Zisserman, "Recurrent human pose estimation," in IEEE FG, 2017.
27. A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in ECCV, 2016.
28. A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in CVPR, 2014.
29. V. Belagiannis and A. Zisserman, "Recurrent human pose estimation," in IEEE FG, 2017.
30. A. Bulat and G. Tzimiropoulos, "Human pose estimation via convolutional part heatmap regression," in ECCV, 2016.
31. X. Chu, W. Yang, W. Ouyang, C. Ma, A. L. Yuille, and X. Wang, "Multi-context attention for human pose estimation," in CVPR, 2017.
32. W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," in ICCV, 2017.
33. Y. Chen, C. Shen, X.-S. Wei, L. Liu, and J. Yang, "Adversarial posenet: A structure-aware convolutional network for human pose estimation," in ICCV, 2017.
34. W. Tang, P. Yu, and Y. Wu, "Deeply learned compositional models for human pose estimation," in ECCV, 2018.
35. L. Ke, M.-C. Chang, H. Qi, and S. Lyu, "Multi-scale structure-aware network for human pose estimation," in ECCV, 2018.
36. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in CVPR, 2016.
37. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in ICLR, 2015.
38. Afza F, Khan MA, Sharif M, Kadry S, Manogaran G, Saba T, Ashraf I, Damaševičius R. 2021. A framework of human action recognition using length control features fusion and weighted entropy-variances based feature selection. Image and Vision Computing 106:104090