International Journal of Computer

(IJC)

ISSN 2307-4523 (Print & Online)

© *Global Society of Scientific Research and Researchers*

http://ijcjournal.org/

AI-Based Yoga Pose Estimation And Correction

Mrs. Girija Gireesh Chiddarwar[a], Abhishek Ranjane[b], Mugdha Chinde[c], Rachana Deodhar[d], Palash Gangamwar[e]

Computer Department, SCOE, Sinhgad Road, Pune-411041, India

[a]*Email: ggchiddarwar@sinhgad.edu*

[b]*Email: abhishekranjane33@gmail.com*

[c]*Email: mugdhab6@gmail.com*

Abstract

The importance of yoga is renowned worldwide and its health benefits, which were preached by ancient sages, have stood the test of time. Even though yoga is becoming preeminent, there are important challenges faced while doing yoga such as performing it with incorrect form, the classes being expensive and the shortage of time in our busy lives. Computer vision techniques exhibit promising solutions for human pose estimation. However, these techniques are seldom applied in the domain of health or exercise, with no literature or projects cited specifically for yoga. This paper surveys the various technologies that can be used for pose estimation and concludes the best method based on the usability for an android application. The paper then discusses the methodology that will be used to deploy the yoga pose estimation on an android application.

1. Introduction

Deep learning techniques have proved their importance for object detection tasks. The same models can be effectively used to detect the various essential body parts and estimate the pose of the user in real-time. The paper surveys the different options that can be used for pose estimation such as OpenPose, Posenet, DeepPose, and concludes which technique should be used for deploying it for an android application for doing yoga.

There are several methods that can be used for pose estimation. We shall discuss some of the most important ones. In this paper we shall discuss the evolution of human pose estimation over the years and how Posenet is the most suitable for our project (yoga pose estimation on android). The project provides real-time pose estimation for yoga pose estimation and correction on the client-side. This is achieved with Tensorflow. The library allows the inference of models on Android, which makes it faster and more privacy respecting. Posenet

is an open-sourced technology that allows us to extract the 17 essential points natively. A skeleton of the human pose is drawn with the help of these points, which is then used to derive angles between these points thus enabling us to effectively correct the user's yoga poses.

2. Related Work

2.1. *DeepPose*

DeepPose was the first major paper[1], published in CVPR 2014 that applied Deep Learning to Human pose estimation. It achieved SOTA performance and beat existing models back in the year 2014. The model has an AlexNet backend and estimated pose in a holistic fashion, i.e certain poses can be estimated even if some joints are hidden when the pose is reasoned holistically. The paper applies Deep Learning (CNN) to pose estimation and kicked off research in this direction. The model used regression for XY locations for certain regions. This added complexity and weakened generalization hence performing poorly.

2.2. *Efficient Object Localization Using Convolutional Networks*

This approach in this paper[2] uses heatmaps by sliding through multiple subsets of the image (windows) in parallel to simultaneously capture features at a variety of scales. A heatmap predicts the probability of one of the essential points. This model is very successful for pose detection and is used in current models as well.

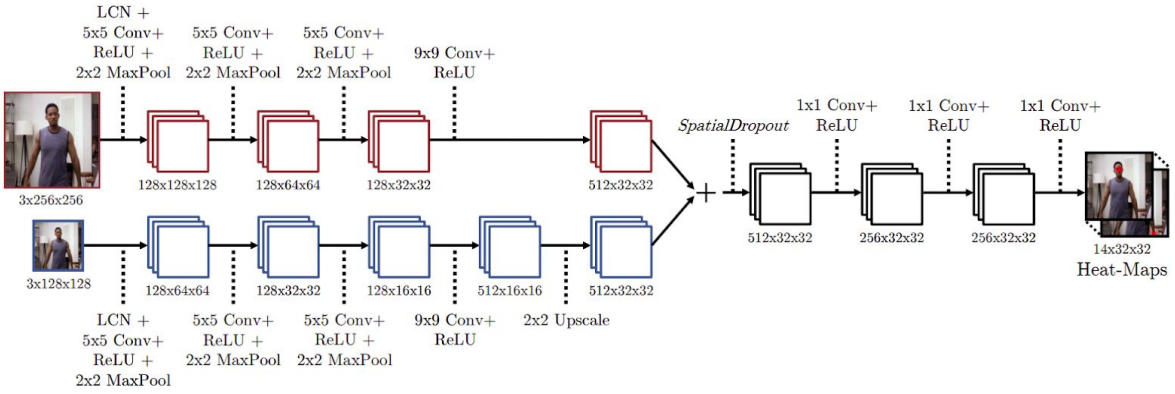


Fig. 1. Working of Sliding window detector

Heatmaps work better than direct joint regression. This model, however, does not include structure modeling, that is, taking into consideration the human body structure, such as body part proportion, symmetries, joint limits, among others.

2.3. *Convolutional Pose Machines*

This interesting paper[3] proposes the serial usage of stages to predict. a CPM (Convolutional Pose Machine) consists of an image feature computation module followed by a prediction module. The multiple stages can be trained end to end. Stage 1 generates heatmaps and image evidence as input for stage 2. Stages > 2 are just repetitions of the second stage.

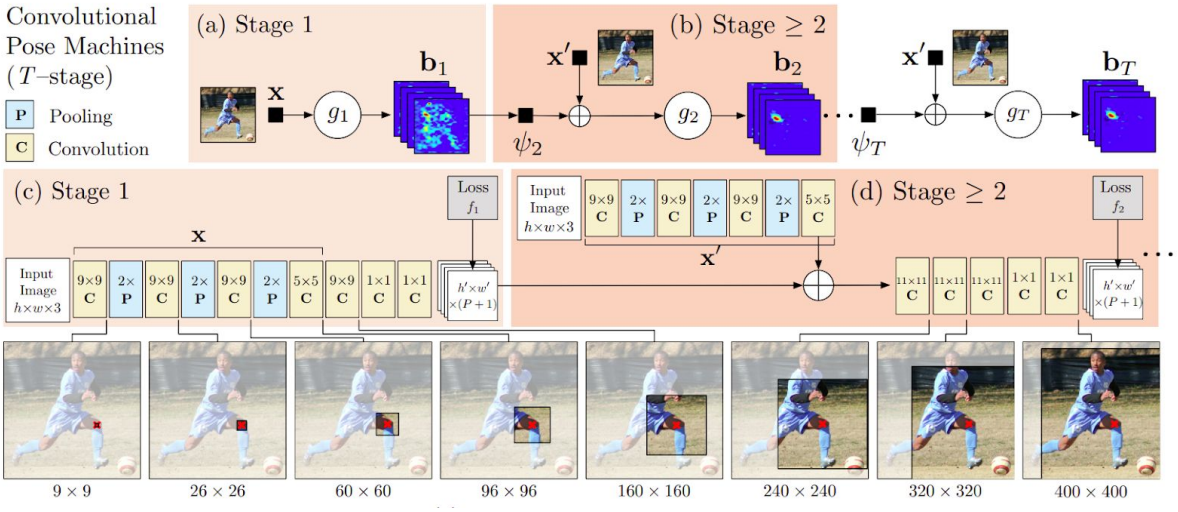


Fig. 2. Working of Sliding window detector

2.4. *OpenPose*

Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields[4] proposes the detection of multiple people in an image. It uses an approach called non-parametric representation, also known as Part Affinity Fields (PAFs). The architecture encodes a global context, allowing a greedy bottom-up parsing step that maintains high accuracy while achieving real-time performance, irrespective of the number of people in the image. The method got SOTA performance on the MPII Multi-Person benchmark.

The bottom-up approach decouples runtime complexity from the number of people in the image. It uses Part Affinity Fields(PAFs), a set of 2D vector fields that encode the location and orientation of limbs over the image domain.

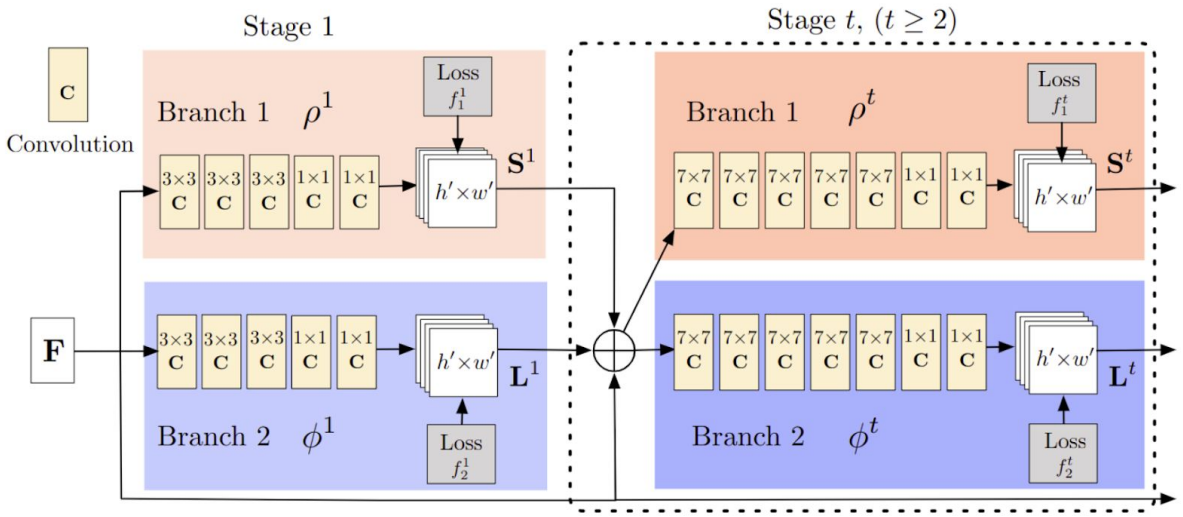


Fig. 3. Architecture of the Two-branch Multi-stage CNN

The problem with OpenPose is that it requires special hardware and can not perform well on mobile devices.

2.5. *Deep High-Resolution Representation Learning for Human Pose Estimation*

The HRNet (High-Resolution Network) model [5] maintains a high-resolution representation throughout the whole process, instead of high → low → high-resolution representation, and this works very well. The architecture starts from a high-resolution subnetwork as the first stage, and gradually adds high-to-low resolution subnetworks one by one to form more stages and connect the multi-resolution subnetworks in parallel.

Repeated multi-scale fusions are conducted by exchanging information across parallel multi-resolution subnetworks over and over through the whole process. Another pro is that this architecture does not use intermediate heatmap supervision. Heatmaps are regressed using an MSE loss.

2.6. *Deep High-Resolution Representation Learning for Human Pose Estimation*

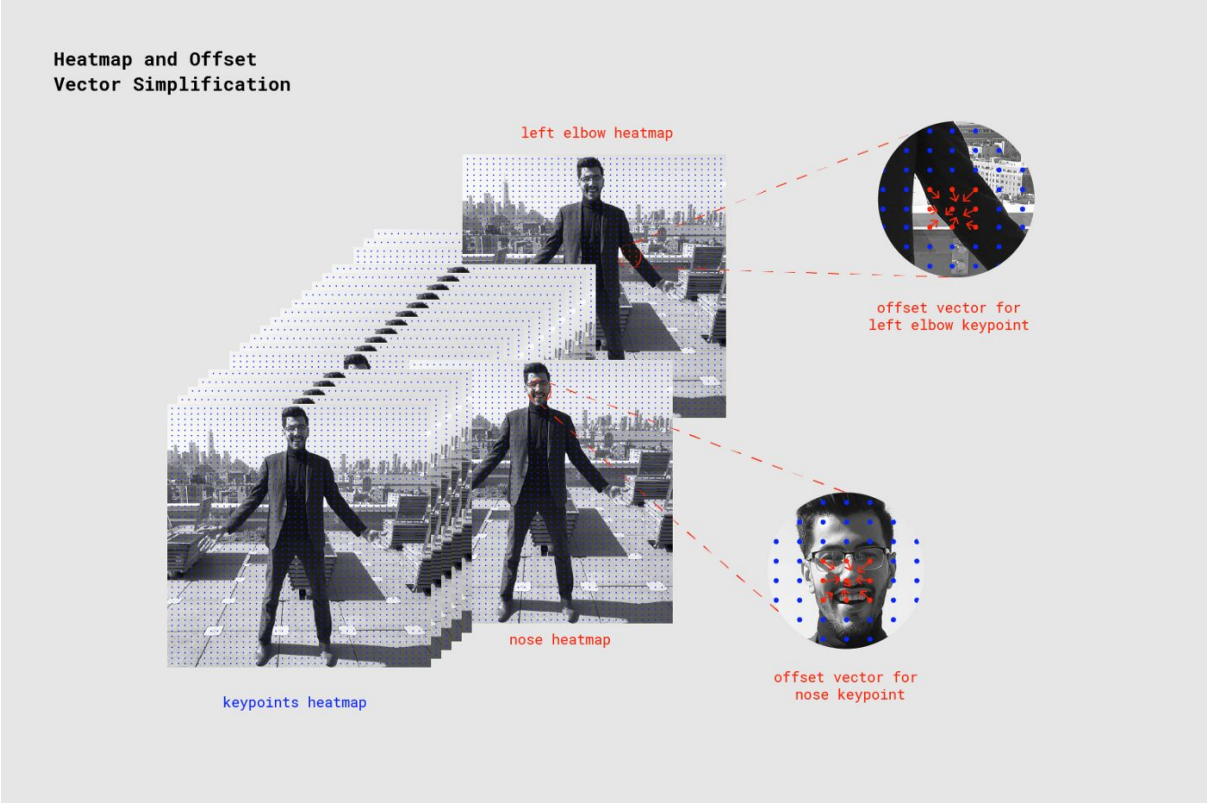
The HRNet (High-Resolution Network) model [5] maintains a high-resolution representation throughout the whole process, instead of high → low → high-resolution representation, and this works very well. The architecture starts from a high-resolution subnetwork as the first stage, and gradually adds high-to-low resolution subnetworks one by one to form more stages and connect the multi-resolution subnetworks in parallel.

Repeated multi-scale fusions are conducted by exchanging information across parallel multi-resolution subnetworks over and over through the whole process. Another pro is that this architecture does not use

intermediate heatmap supervision. Heatmaps are regressed using an MSE loss.

2.7. *PoseNet*

The paper [6] by George Papandreou et al. presented a box-free bottom-up approach that is used in PoseNet. PoseNet is an open-sourced application that can be deployed using Tensorflow. PoseNet gives an output of poses, pose confidence scores, keypoint positions, and keypoint confidence scores from the model outputs. It uses the best practices from some of the previous papers and gives very high accuracy for the minimal cost it requires.



PoseNet can be configured with the *output stride*. PoseNet is invariant of the size of the image presented, by downscaling the image by a factor of this output stride. The higher the output stride the lower the accuracy, but the faster the speed, which is key in the android implementation. Altrous convolution [7] is then used to enable the convolution filters in the subsequent layers to have a wider field of view. PoseNet reapplies the concept of Heatmaps from Efficient Object Localization Using Convolutional Networks [2]. It also uses *offset vectors* that correspond in location to the heatmaps points, and are used to predict the exact locations of the keypoints by traveling along the vector from the corresponding heatmap point.

PoseNet offers both single and multiple pose estimations. Both of these techniques used different approaches. Single pose estimation is much faster and efficient, thus making it suitable to use on an android device. PoseNet is a 2D pose estimation technique, which only does the job of giving the XY coordinates as opposed to XYZ

coordinates present in 3D methods. PoseNet also does not require special hardware, unlike OpenPose[4]. Thus, we can conclude that PoseNet is the most suitable method that can be used for pose estimation in a mobile application.

3. Methodology

3.1. *Model*

In the Android implementation of the PoseNet model is used. PoseNet is an open-sourced project by google, that is excellent for pose estimation. While there are various architectures in PoseNet, the MobileNet architecture is used. This is because, while the ResNet50 has the most accurate results, the MobileNet is smaller and faster, thus making it more suitable for mobile device implementation. The model is stored in the Android project in a separate module. TFlite is a version of TensorFlow that can be implemented on mobile devices. TFlite is the format used to store the module and the TFlite framework will be used to derive inferences from this, as we will discuss later.

3.2. *Android*

The Android app implementation consists of two parts: first, the inference of the PoseNet model to get the 17 essential points such as left shoulder, right shoulder, left knee, right knee, etc. Secondly, the derivation of angles from these points and correcting the pose.

For the first part, the trained model is stored in a PoseNet module as a ".TFlite" file. The various hyperparameters for the model are stored in private variables stored in the app. The app does the job of asking permission for camera capture. When the pose estimation process begins, the app uses the *Imageview* to display the captured video frame by frame. The coordinates for the essential points are generated by inferring to model, by referring to the hyperparameters. The app then uses a *Surfaceview* to draw the points and the user's skeleton using the *draw()* function.

For the second part, the app needs the correct angles to compare the calculated angles. For this, we used a single image for each yoga pose. This the coordinates for these poses were calculated by using OpenCV. The formula used to calculate the angles from the extracted coordinates is:

Angle_B=Math.acos ((Math.pow (len (b, c), 2) + Math.pow (len (a, b),2) - Math.pow (len (a, c),2))/ (2*len (b, c) *len (a, b)))*180/Math.PI

Therefore the correct angles are now calculated and are stored in private variables defined inside the app. These

angles are compared with the correct ones to get the correctness of the pose.

All of these functions are done natively, on the client-side. Not only does this preserve user privacy but it also makes faster inferences. The rest of the app consists of UI elements and various views. The UI and working of the app get deployed on the model-view-model framework, present in android, which allows for easy implementation of the backend logic.

4. Future Direction

The Android application will have the following features:

- A voice-controlled system: This feature is currently in development. It will be very difficult to perform yoga and look at the corrections at the same time. Thus it is essential to convert the commands given by the device to be heard by the user. For that, we have used google's text-to-speech API. Thus, the commands corrections, if any, given by the system will be heard by the user and corrected immediately. The user should also be able to navigate the app by using their voice. Thus, we use google's speech-to-text android functionality to convert the user's voice instructions to enable them to navigate the system efficiently. These features will let the user do yoga without interruption.
- Gesture recognition: This feature has not been implemented yet. However, since PoseNet already gives the coordinates of various body parts, it will be fairly easy to implement a gesture recognition system to navigate the app.
- Progress tracking: This feature will store the measures of the correctness of the user's yoga poses. Giving users the ability to monitor their progress will not only be informative to them of their improvement but it will also keep the user motivated to keep on doing yoga.

5. Conclusion

Deep Learning methods have proved to be extremely useful for pose estimation, compared to any other methods. The diverse applications of pose estimation have provoked many advancements in this field, both in terms of speed and accuracy. We have concluded that PoseNet is, by the current standards, the best technique for implementing mobile applications, specifically for yoga. This method can be easily implemented by using the TensorFlow-Lite framework. The trained model is stored in a ".tflite" format and can be inferred to get the 17 essential points. We use these points to calculate the angles and compare them to the correct pose angles calculated by OpenCV. The skeleton and wrong angles are displayed. Functionality for effectively controlling and navigating the app with voice and gestures can be implemented. Thus, This paper substantially provided a glimpse of how the methods of pose estimation have evolved over the years, and how they can be effectively Acknowledgmentspplications, such as yoga pose correction.

Acknowledgements

We would also like to take this opportunity to thank our guide, Mrs. G. G. Chiddarwar, for helping us throughout and bring patient with us. Also, to our review committee member Mrs. J. B. Kulkarni for invaluable advice and critique. Furthermore, we would like to express gratitude and earnestly acknowledge our parents for supporting us and friends for encouraging us. To Prof. M. P. Wankhade our Head of Department and our principal Dr. S. D. Lokhande for backing us.

Print References

Articles from Conference Proceedings (published)

[1] "DeepPose: Human Pose Estimation via Deep Neural Networks" The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1653-1660

[2] "Efficient Object Localization Using Convolutional Networks" The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 648-656

[3] "Convolutional Pose Machines" The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4724-4732

[4] "Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields" The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7291-7299

[5] "Deep High-Resolution Representation Learning for Human Pose Estimation" The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 5693-5703

[6] "PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model" The European Conference on Computer Vision (ECCV), 2018, pp. 269-286

[7] "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation" The European Conference on Computer Vision (ECCV), 2018, pp. 801-818

Electronic References

World Wide Web

Dan Oved, Irene Alvarado, and Alexis Gallo. "Real-time Human Pose Estimation in the Browser with TensorFlow.js" Internet: https://medium.com/tensorflow/real-time-human-pose-estimation-in-the-browser-with-tensorflow-js-7dd0bc881cd5, May 7, 2018 [Sept. 13, 2019].