Loading Dataset [LoadKaggleDataset.py]

Dataset overview

TRAIN/

- Downdog/ 223 images
- Goddess/ 180 images
- Plank/ 266 images
- Tree/ 160 images
- Warrior/ 252 images

TEST/

- Downdog/ 97 images
- Goddess/ 80 images
- Plank/ 115 images
- Tree/ 69 images
- Warrior/ 109 images

Preprocessing Dataset [*PreProcessDataset.py*]

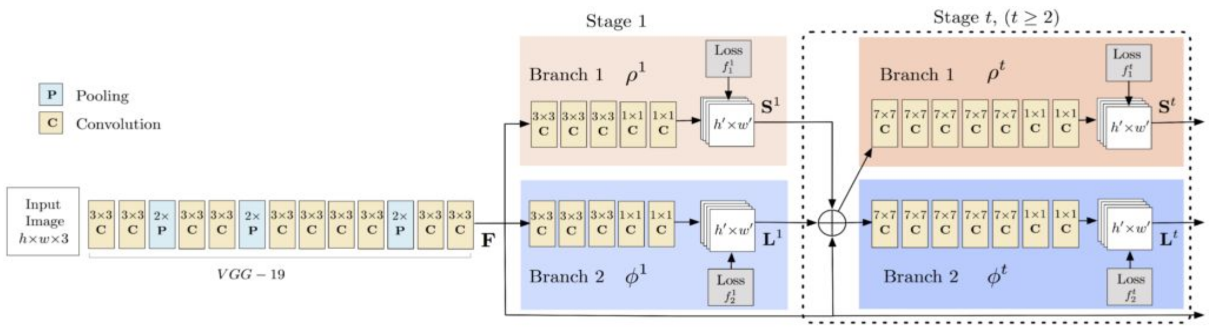
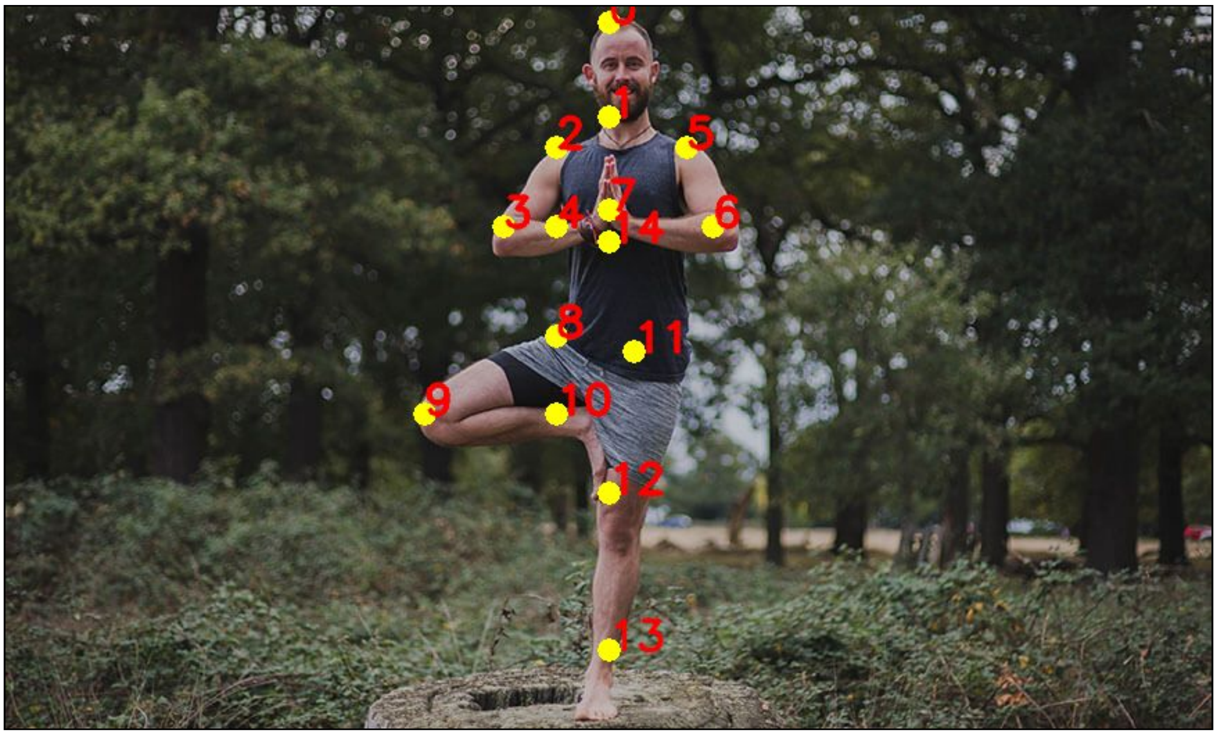


Figure 1. Multi-Person (MPII) Pose Estimation model architecture. Source: CMU Perceptual Computing Lab

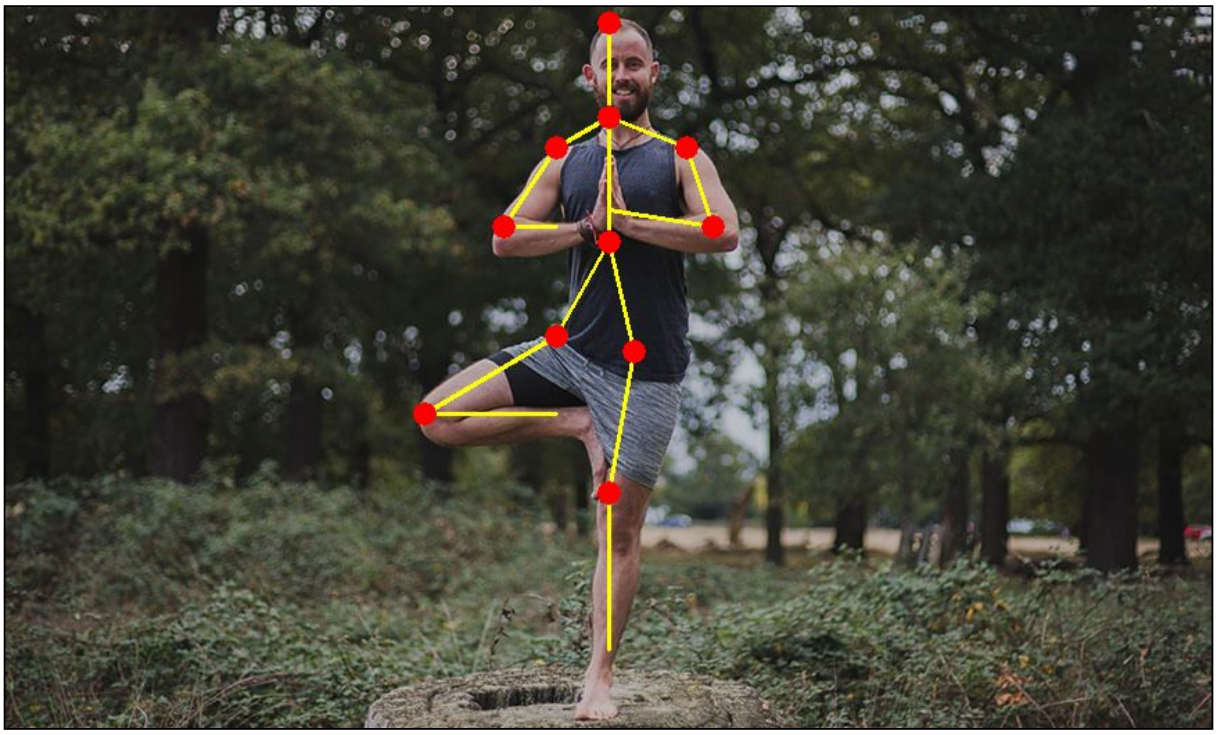
How does preprocessing work?

Sample Image (tree pose)

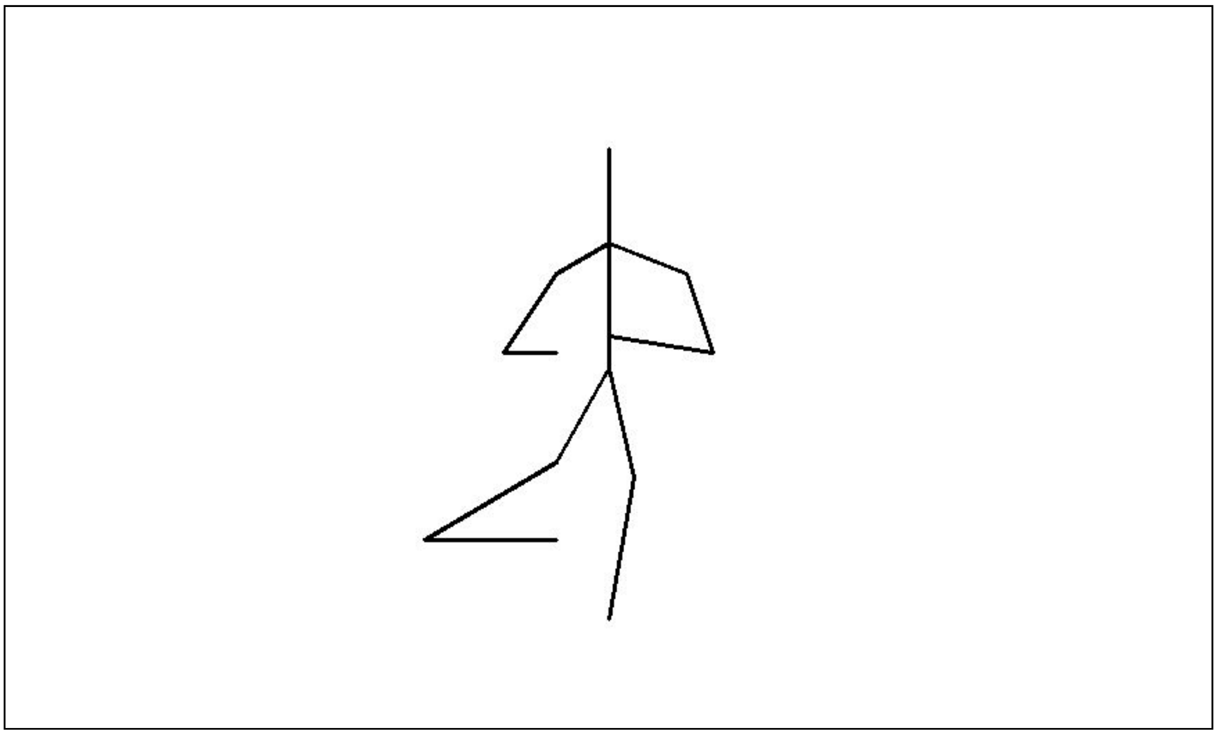
Using OpenPose MPII we obtain body key points



Then, we connect *pose_pairs,* to obtain a skeleton figure

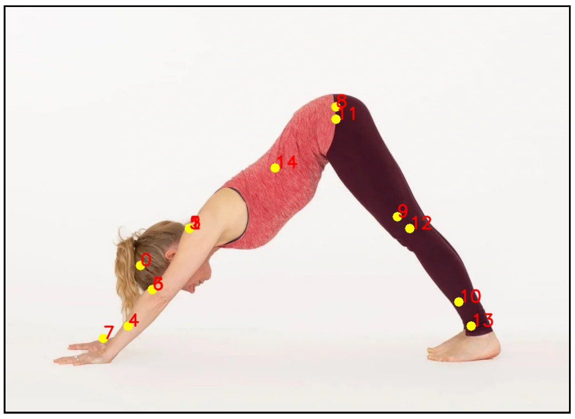


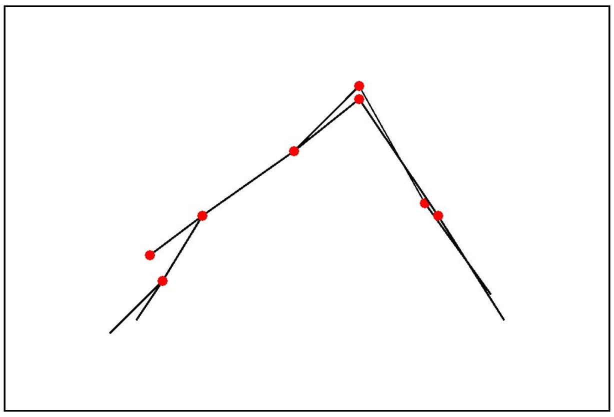
Then, taking the skeleton figure and grabbing by its body center which is point 14, listed as the chest in MPII output, and moving the body center to the center of the image and normalize it. And also getting rid of the key points and only keeping the skeleton figure. Images for which body center could not be detected by OpenPose were discarded for training.



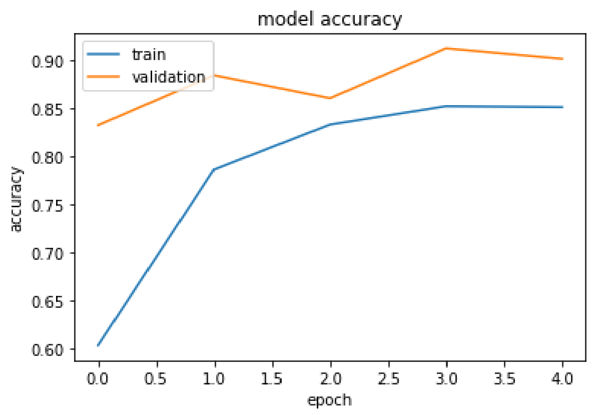
Preprocessing example on down dog pose.

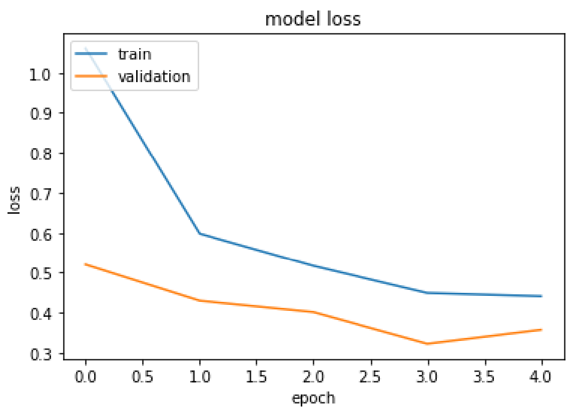
Detected key points on the real image and skeleton. The key points are visible in the skeleton image here just for reference.





Deep Learning Training Model [ModelTraining.py]





~ 90% accuracy

When I incrementally fitted my model with 10 epoch after already fitting it with 5 epochs, accuracy did not improve, it was revolving around 90% accuracy. Even though this is good accuracy, the reason the model wasn't able to perform well is due to the dataset. I had taken only taken a glance at Kaggle but then when I was figuring out the reason behind accuracy hitting plateau, is because there are images in the dataset which are just noise (irrelevant pics) and incorrectly assigned. This makes sense as I later read that the dataset publisher had just scraped images from bing using an API. So, it is acceptable if the model isn't going above 90% accuracy. This model's performance can also be affected by the poor performance of OpenPose in preprocessing images.

Extension

Using OpenCV to capture real-time frames via the device camera, preprocessing them, and then predicting them using the model would be a great application of this model. If I were to make this model *stricter* in terms of classification, I think the potential way would be to train the model on only perfect poses from one side with some angle variations. Then the user could be informed to pose facing a particular direction and could place their device camera at any angle.

References

Brownlee, J. (2019). Transfer Learning in Keras with Computer Vision Models. In Deep Learning for Computer Vision.

Cao, Z., Simon, T., Wei, S. E., & 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).

OpenPose / Github:: CMU-Perceptual-Computing-Lab/openpose: OpenPose: Real-time multi-person keypoint detection library for body, face, hands, and foot estimation

OpenCV with OpenPose / Github: opencv/openpose.py at master · opencv/opencv · GitHub