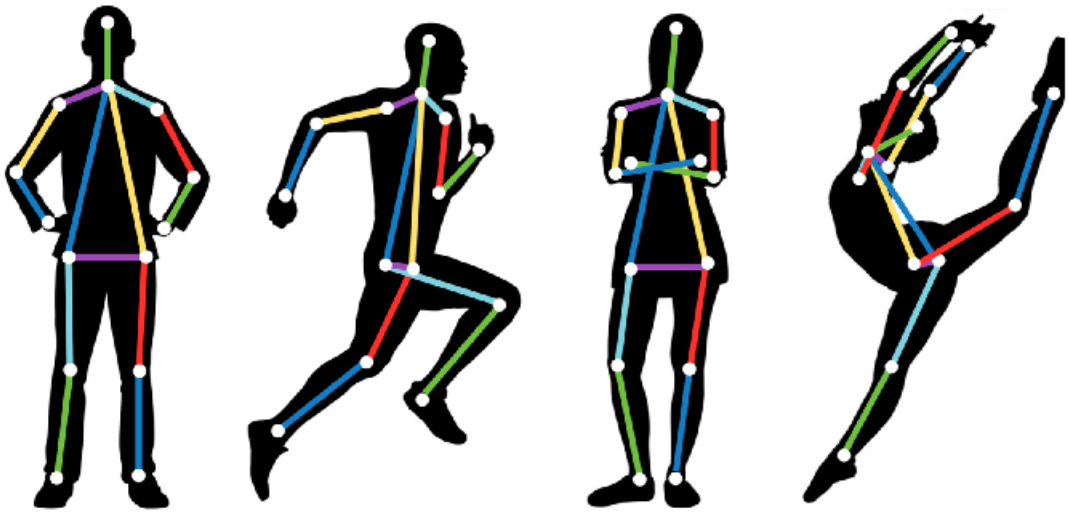
Human Pose Estimation Using Machine Learning in Python

Pose detection is an active field of study in the field of computer vision. You can literally find hundreds of research papers and several models that try to solve the problem of pose detection. The reason why so many machine learning enthusiasts are attracted to pose estimations is because of its wide variety of applications and usefulness. In this article, we are going to cover one such application of pose detection and estimation using machine learning and some of the very useful libraries in python.

What is Pose estimation?



Pose estimation is a computer vision technique to track the movements of a person or an object. This is usually performed by finding the location of key points for the given objects. Based on these key points we can compare various movements and postures

and draw insights. Pose estimation is actively used in the field of augmented reality, animation, gaming, and robotics.

There are several models present today to perform pose estimation. Some of the methods for pose estimation are given below:

1. Open pose
2. Pose net
3. Blaze pose
4. Deep Pose
5. Dense pose
6. Deep cut

Choosing any one model over another may totally depend upon the application. Also, the factors like running time, size of the model, and ease of implementation can be various reasons to choose a specific model. So, it is better to know your requirements from starting and choose the model accordingly.

For this article, we will be using the Blaze pose for detecting human pose and extracting key points. The model can be easily implemented through a very helpful library, well known as media pipe.

Media Pipe – Media pipe is an open-source cross-platform framework for building multimodel machine learning pipelines. It can be used to implement cutting-edge models like human face detection, multi-hand tracking, hair segmentation, object detection and tracking, and so on.

Blaze Pose Detector – Where most of the pose detection relies on COCO topology consisting of 17 key points, the blaze pose detector predicts 33 human key points including torso, arms, leg, and face. The inclusion of more key points is necessary for

succeeding applications of domain-specific pose estimation models, like for hands, face, and feet. Each key point is predicted with three degrees of freedom along with the visibility score. The blaze pose is a sub-millisecond model and can be used for real-time applications with an accuracy better than most of the existing models. The model is available in two versions Blaze pose lite and Blaze pose fully to provide a balance between speed and accuracy.

Blaze pose offers several applications including fitness and yoga trackers. These applications can be implemented by using an additional classifier like the one we are going to build in this article itself.

You can learn more about the blaze pose detector here.

2D vs 3D pose estimation

Pose estimation can be done either in 2D or in 3D. 2D pose estimation predicts the key points from the image through pixel values. Whereas 3D pose estimation refers to predicting the three-dimensional spatial arrangement of the key points as its output.

Preparing Dataset for Pose Estimation

We learned in the previous section that key points of the human pose can be used to compare different postures. In this section, we are going to prepare the dataset by using the media pipe library itself. We are going to take images of two yoga poses, extract key points from them and store them in a CSV file.

You can download the dataset from Kaggle through this link. The dataset consists of 5 yoga poses, however, in this article I am taking only two poses. You can use all of them if you want, the procedure will remain the same

```
import mediapipe as mp
import cv2
import time
import numpy as np
import pandas as pd
import os
mpPose = mp.solutions.pose
pose = mpPose.Pose()
mpDraw = mp.solutions.drawing_utils # For drawing keypoints
points = mpPose.PoseLandmark # Landmarks
path = "DATASET/TRAIN/plank" # enter dataset path
data = []
for p in points:
        x = str(p)[13:]
        data.append(x + "_x")
        data.append(x + "_y")
        data.append(x + "_z")
        data.append(x + "_vis")
data = pd.DataFrame(columns = data) # Empty dataset
```

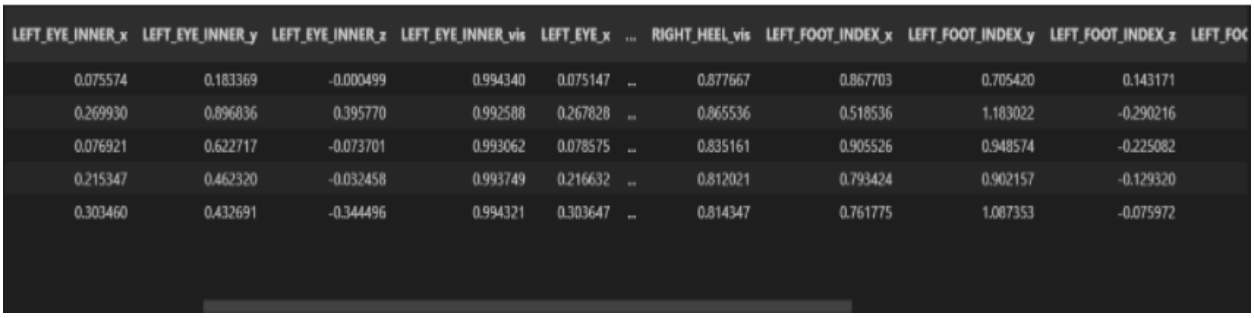
In the above snippet of code, we have first imported the necessary libraries that will help in creating the dataset. Then in the next four lines, we are importing the modules required to extract key points and their draw utils. Next, we create an empty pandas data frame and enter the columns. Here the columns include the thirty-three key points that will be detected by the blaze pose detector. Each keypoint contains four attributes that are x and y coordinates of the keypoint(normalized from 0 to 1), z coordinate that represents landmark depth with hips as the origin and same scale as that of x, and lastly the visibility score. The visibility score represents the probability that the landmark is either visible in the image or not.

```
count = 0

for img in os.listdir(path):

        temp = []

        img = cv2.imread(path + "/" + img)

        imageWidth, imageHeight = img.shape[:2]

        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        blackie = np.zeros(img.shape) # Blank image

        results = pose.process(imgRGB)

        if results.pose_landmarks:

                # mpDraw.draw_landmarks(img, results.pose_landmarks, mpPose.POSE_CONNECTIONS) #d

                mpDraw.draw_landmarks(blackie, results.pose_landmarks, mpPose.POSE_CONNECTIONS)

                landmarks = results.pose_landmarks.landmark

                for i,j in zip(points,landmarks):

                        temp = temp + [j.x, j.y, j.z, j.visibility]

                data.loc[count] = temp

                count +=1

        cv2.imshow("Image", img)

        cv2.imshow("blackie",blackie)

        cv2.waitKey(100)

data.to_csv("dataset3.csv") # save the data as a csv file
```

In the above code, we are iterating through the pose images individually, extracting the key points using the blaze pose model and storing them in temporary array ‘temp’. After the iteration is completed, we append this temporary array as a new record in our dataset.

You can also see these landmarks by using the drawing utils present in the media pipe itself. In the above code, I have drawn these landmarks on the image as well as on a blank image ‘blackie’ to focus on the results of the blaze pose model only. The blank image ‘blackie’ has the same shape as that of the given image. One thing that should be noticed is that the blaze pose model takes RGB images instead of BGR (read by OpenCV).

After getting the key points of all the images we have to add a target value that will act as a label for our machine learning model. You can make the target value for 1st pose as 0 and the other as 1. After that, we can just save this data to a CSV file which we will use for creating a machine learning model in the later steps.



LEFT_EYE_INNER_x	LEFT_EYE_INNER_y	LEFT_EYE_INNER_z	LEFT_EYE_INNER_vis	LEFT_EYE_x	...	RIGHT_HEEL_vis	LEFT_FOOT_INDEX_x	LEFT_FOOT_INDEX_y	LEFT_FOOT_INDEX_z	LEFT_FOO
0.075574	0.183369	-0.000499	0.994340	0.075147	...	0.877667	0.867703	0.705420	0.143171	
0.269930	0.896836	0.395770	0.992588	0.267828	...	0.865536	0.518536	1.183022	-0.290216	
0.076921	0.622717	-0.073701	0.993062	0.078575	...	0.835161	0.905526	0.948574	-0.225082	
0.215347	0.462320	-0.032458	0.993749	0.216632	...	0.812021	0.793424	0.902157	-0.129320	
0.303460	0.432691	-0.344496	0.994321	0.303647	...	0.814347	0.761775	1.087353	-0.075972	

You can observe how the dataset looks like from the above image.

Creating the Pose Estimation model

Now we have created our dataset, we just have to pick a machine-learning algorithm to classify the poses. In this step, we will take an image, run the blaze pose model (that we used earlier for creating the dataset) to get the key points of the person present in that image, and run our model on that test case. The model is expected to give the correct results with a high confidence score. In this article, I am going to use the SVC(Support Vector Classifier) from the sklearn library to perform the classification task.

```
from sklearn.svm import SVC
data = pd.read_csv("dataset3.csv")
X,Y = data.iloc[:,:132],data['target']
model = SVC(kernel = 'poly')
model.fit(X,Y)
mpPose = mp.solutions.pose
pose = mpPose.Pose()
mpDraw = mp.solutions.drawing_utils
path = "enter image path"
img = cv2.imread(path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
results = pose.process(imgRGB)
if results.pose_landmarks:
        landmarks = results.pose_landmarks.landmark
        for j in landmarks:
                temp = temp + [j.x, j.y, j.z, j.visibility]
        y = model.predict([temp])
        if y == 0:
            asan = "plank"
        else:
            asan = "goddess"
        print(asan)
        cv2.putText(img, asan, (50,50), cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),3)
        cv2.imshow("image",img)
```

In the above lines of code, we have first imported the SVC (Support Vector Classifier) from the sklearn library. We have trained the dataset that we build earlier on SVC with the target variable as the Y label. Then we read the input image and extract the key points, the same way we did while creating the dataset. Lastly, we input the temporary variable and use the model to make the prediction. The pose can now be detected using simple if-else conditions.

Results of the Model

From the above images, you can observe that the model has correctly classified the pose. You can also see the pose detected by the blaze pose model on the right side. In the first image, if you observe closely, some of the key points aren't visible, still, the pose is

classified correctly. This could be possible because of the visibility of the key points attribute given by the blaze pose model.

Conclusion

Pose detection is an active area of research in the field of machine learning and offers several real-life applications. In this article, we tried to work on one such application and get out hands dirty with pose detection. We learned about pose detection and several models that can be used for pose detection. We selected the blaze pose model for our purpose and learned about its pros and cons over other models. In the end, we built a classifier to classify yoga poses using the support vector classifier from the sklearn library. We also built our own dataset for this purpose which could further be extended easily using more images.