Real-time Exercise Posture Correction Using Human Pose Detection Technique

A project

presented to

Department of Computer and Information Science

SUNY Polytechnic Institute

Utica, New York.

In the partial fulfilment of

the requirements of the Master of

Science Degree

by

Nikhil Kumar Reddy Boyalla

(U00331309)

under guidance of

Prof. Bruno R. Andriamanalimanana

December 2021

DECLARATION

I declare that this project is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

[illegible]

Nikhil Kumar Reddy Boyalla

Exercise Posture Correction Using Human Pose Detection Technique

Department of Computer Science SUNY Polytechnic Institute

Approved and recommended for acceptance as
a project in partial fulfillment of the requirements
for the degree of Master of Science in
Computer and Information Sciences

Date

Dr. Bruno R. Andriamanalimanana
(Advisor)

Dr. Scott Spetka

Dr. Chen-Fu Chiang

ABSTRACT

Human pose detection is one of the fascinating research areas in the field of computer vision that has many unsolved challenges. Detecting and capturing human activity is advantageous in many fields like sport analysis, human co-ordination tracking, public surveillance etc. Due to the COVID-19 driven pandemic, it became hard for society to access exercising hubs and newbies into the fitness industry were left blank with almost no personal guidance that they usually get in gym in terms of exercising in the right way through one-on-one interactions. As these resources are not always available, human pose detection can be a medium to replace a human personal trainer by developing a real-time exercise posture correction system on recorded videos or real-time image stream that allows people to safely exercise at home avoiding injuries. This project uses a pre-trained OpenPose Caffe model with two datasets i.e., COCO and MPII to correct one of the exercises in the fitness industry. This project also discusses various pose estimation and key point detection techniques in detail and different deep learning models used for pose classification.

Keywords - Human Pose Detection, gym, COVID-19, OpenPose, deep learning

ACKNOWLEDGEMENT

I take this opportunity with immense pleasure to thank all the people who have helped me through the course of my journey towards this report.

I sincerely thank my project advisor Dr. Bruno R. Andriamanalimanana for his constant help, guidance and motivation. Being his student in every semester helped me pursue courses that are trending and important for the advancement of the current society. Apart from this project, I learned a lot from him, and I am sure it will be useful to me in different stages of my life. I would like to express my gratitude to him for his understanding, supervision, kind co-operation and in-valuable support in multiple aspects and in various tough situations.

I would like to express my sincere thanks to Dr. Scott Spetka and Dr. Chen-Fu Chiang for being on my project review committee. I always enjoyed your classes, especially RNN pursued under Dr. Chen-Fu and Database Management under Dr. Scott Spetka. I cannot thank you enough for all the knowledge you shared in making me an efficient engineer.

Finally, I would like to express my heartfelt thanks to my Parents, God, my teachers and colleagues for their blessings and constant support.

Nikhil Kumar Reddy Boyalla

Table of Contents

List Of Figures

CHAPTER 1

INTRODUCTION

1.1 What is Pose Detection?

Pose Detection of an object or a human is a computer vision technique which allows computers to predict and track the location and status by learning and understanding factors like posture, orientation etc. High-resolution image processing, reducing the consumption of computation power, tracing high-speed moving objects are major challenges and research areas in pose detection. Every object on this planet obtains a structural behavior by possessing a designated or forced pose which helps us in analyzing its status in any real-life situation, while computers exhibit the capability of following a strict set of algorithms inputted for better capturing and recognizing the patterns that objects display. With large-scale datasets and algorithmic trained models, pose detection enables machines to record and summarize the existing posture of any object at ample situations.

Object detection is used in diverse fields and in many technical and non-technical industries for even minute applications that proves to be critical. Tech giants like Tesla adopted object and human detection in developing their state-of-art self-driving vehicles. Food processing industries saw object pose detection as a solution for effective and safe packaging of their products. Human pose detection has significant scope in virtual reality augmentation fitness industry etc. Aspects like key points detection, dataset, model etc., are part of pose detection hierarchy. This project is a demonstration of one of the implementations of human pose detection in fitness industry using deep learning concepts and deep neural networks.

Human Pose Detection

Human pose detection is predicting poses of human joints and body parts through key point detection approach. In most of the HPD approaches, major joints of human are the key points. Imitating and replicating human pose has wide range of potential applications such as human/computer interfaces, videogames, or surveillance [1]. They are spatial locations, or points in the image that are independent of image rotation, shrinkage, translation, distortion etc., Key point detection consists of locating key object parts. For example, the key parts of our faces include eyebrows, nose tip, eye, eye corners and so on. These key points help to represent the underlying object in a feature-rich manner. Depending on the output expectations, key points are further used in mapping human into a skeleton, 2D or 3D representations. Fig 1 shows different types of human pose detection viz. Skeleton based kinetic model, 2D planar model and 3D volumetric model.

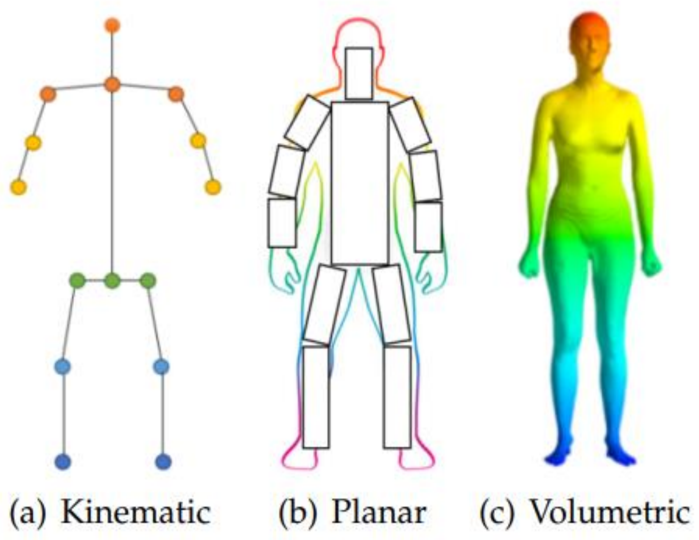


Figure 1. Three types of models for human body modelling. [2]

1.2 Deep Learning

Deep learning is a subset of machine learning in the family of artificial intelligence which is an algorithm driven neural network capable of processing metadata to produce desired output by involving some layers of hidden neural networks. These hidden layers are capable of fore driving the results to the next layer feeding output of one layer as input to the next layer. Deep learning algorithm has automatic feature extraction as its prime functionality which enables it to understand the relevant features required for delivering expected output, thus minimal developer involvement is supported in selection of features explicitly. Deep learning is used to solve real time problems with highest complexity and can perform exceptionally in solving supervised, unsupervised, and semi-supervised conditions.

In modern world, understanding and learning from the patterns and information that bulk data holds is where companies of all customer types are behind of. Deep Learning enables these companies to learn from the data and let them build futuristic products. There are several deep learning models deep neural networks, convolution neural network, recursive neural network etc., that are used for certain tasks like self-driving cars, natural language processing, image recognition, fraud detection respectively. The deep neural network is an artificial neural network that replicates human thinking by involving connected hidden layers with artificial neurons.

1.3 Deep Neural Network – DNN

Any neural network with more complex neuron arrangement with more than two hidden layers is called deep neural network. These layers are built up with various artificial neurons connected in a sophisticated pattern to replicate human brain. Each layer in DNN performs a specific type of sorting or ordering for realizing the features of provided input which is called feature hierarchy. Training these kinds of multi-layered neural networks is known to be hard [3]. Poor results are evident for networks with three or more hidden layers that are backed with the learning strategy of randomly initializing the weights of the network and applying gradient descent using backpropagation. But deep architectures can be much more efficient than shallow architectures according to complexity theory of circuits in terms of required parameters in function representation and computational elements [4]. Whereas it cannot be claimed that deep architectures are better than shallow ones in every problem [5]. There has been evidence of an advantage when the task is complex enough and there is enough data to capture that complexity. Using datasets like COCO and MPII which have enough data input for pose detection and pre-trained neural network based on regression can yield optimal results for this use case.

CHAPTER 2

BACKGROUND

2.1 Motivation and Initial Research

In this modern era of lifestyle, fitness is undoubtedly the at most priority of major population in each age category. People tend to have fitness goals based on the result they want to achieve and in achieving those goals, personal assistance is necessary in gaining exercise-based knowledge and avoiding injuries. My motivation to gym began back in 2016 where my initial goal was to lose weight and improve strength later. Performing over-weighted deadlift under no supervision led to a muscle tear near c4 spinal section. Warming up muscle groups before performing related exercises in something that I skipped that day which prone to an injury that threw me away from exercising for almost a year. When I fully recovered and decided to attain my fitness back by going to gym, pandemic due to novel corona virus has changed the circumstances for not only me but the population of this entire sphere. Hitting gym with a time gap urged a fitness trainer to help me regain the initial momentum to keep going and pursue my fitness goals. But isolation and staying at habitat becoming the new normal, has strictly restricted access to exercising hubs and one-on-one trainer interactions. This motivated me to re-imagine the concept of building a real-time AI engine with help of deep learning algorithms that can keep track of my exercising by correcting my postures and passing necessary suggestions at the very moment of exercising ensuring no failure of practice and reducing the chances of injury.

2.2 Related Work

A lot of work has been done in the past in building applications that are automated or semiautomated which help to analyze exercise and sports activities such as swimming [21], basketball [22] etc. Patil et al. [23], proposed a system for identifying yoga posture differences between an expert and a practitioner using speeded up robust features (SURF) which uses information of image contours. Result of this approach was not fruitful due to insufficient information.

Another approach proposed in [25] detected the human activity status using an android app that provides real-time voice commands as suggestions to the user but don't show any pictorial corrections of their posture. Three models were designed and trained on dataset with not more than 800 activity related images which when compared to huge datasets like COCO and MPII hold minimal throughput.

Deep learning is a promising domain where a lot of research is being done, enabling us to analyze tremendous data in a scalable manner. As compared to traditional machine learning models where feature extraction and engineering is a must, deep learning eliminates the necessity to do so by understanding complex patterns in the data and extracting features on its own.

2.3 Human Pose Detection – Overview

The idea of human pose detection has been an area of research for several decades. Initial HPD model was first applied to sign language translation where the sign language translation performed by human physical notation has been detected and decoded into symbols [3]. Since then, the development in the field of AI has benefited in creating more complex models on vast variety of data to understand and recognize human poses to the very deep end. Human pose detection is a problem of localizing the human joints in general. In order to predict pose of a human in any given input image or video, key point detection is the first known step. In this project, Lets discuss techniques and approaches that exist in human pose detection and in further sections we will see design, implementation, and result of proposed topic.

Generative Models

Generative type of methods adapts a procedure to predict the features from a given pose input. This approach begins with inaugurating the stance of the human body and rove it to the picture plane. consistency of extended picture and current picture is achieved by making changes. Generative based approaches offer easy generalization due to less constraint of a training pose dataset [7]. However, due to the high dimensional projection space search, this method is not considered computationally feasible, and is thus slower as compared to discriminative methods. [8] describes a generative Bayesian technique to analyze 3D segmented human body figures in continuous image stream. Even though this approach can track human in fuzzy complicated backgrounds, it holds the drawback of losing track of the detected objects.

Discriminative Models

Discriminative technique points to a set of models used in statistical classification. These models are even termed as conditional models as they improve from the limits bounding the classes or labels in a dataset. It restores the pose using the nonlinear regression based on the shape descriptor vectors fetched automatically from silhouettes of the image. It uses the relevant vector machine regressors and damped least squares for regression [9]. The thematic outcome of these models is simple separation of classes rather than generating new data points, as the name suggests, discriminative models bifurcate the classes instead of modelling the conditional probability. Discriminative models outperform the generative models when there exist the outliers in the dataset as they are more robust to outliers. One of the noticed setbacks of these models is the misclassification problem where the data points are wrongly classified. Learning methods and exemplar methods are the sub-categories of discriminative models. The model we use in this project is a kind of learning methods technique which will be elaborated further in this project.

Top-Down Approach

Most research refers generative methods as top-down methods

[10]. A module is used in top-down method to detect human stance in posting the key points to which the pose estimator can be applied. This task is broken into smaller task including object detection followed by pose detection. For efficient pose estimation, the detector must be precise enough to translate the minute objects, incomplete poses, and half visible human torso. One of the existing top-down method consists of a human candidate detector, a single person pose estimator and a human pose tracker [11]. With such modular implementation for human pose estimation and tracking, the precision achieved was 69.4% for pose estimation and 68.9 for pose tracking. These results ensure the research enthusiasts a scope for improvement.

Bottom-up Approach

This approach uses several data association mechanisms to map key points to human joints and limbs to detected appearances. It is more effective in multi-person detection out of any image, thus being cost effective. Like top-down method, bottom-up method is sometimes referred as discriminative methods [10]. One of the approaches proposed in [12] describes the combination of bottom up and top-down methods for multi-person pose estimation. In this approach, the parsing of poses along with bounding box constraints is performed in top-down fashion while feed forward into the network is done in bottom-up fashion. Using complex deep learning architecture like ResNet50, results are bought to be accurate.

2.4 Computer Vision

Computer vision is a subfield of AI which enables computers to see, detect, analyze, and learn from any images, videos, and other visual input streams. Computer vision provides human vision capability to a computer that uses machine learning models and algorithms to learn from the provided data in form of dataset. First ever computer vision was built to mimic the human vision system as a steppingstone to endowing robots with intelligent behavior [13]. This was further believed to be achieved by mounting a camera to a computer and let it "describe what it saw" [14][15]. Initial projects produced minimal results in terms of accuracy due to lack of processing power and limited its performance as no adequate data was available. What inspired the early researchers at that period was to extract a three-dimensional structure from images to create a full structural model which could help in understanding full scene. From then, computer vision has spread its wings is achieving multi-purpose application-based models including video tracking, scene reconstruction, image restoration, object recognition, event detection, 3D pose estimation etc. Current research is being carried in fields like cloud computing and supply chain warehouse management where computer vision is intersected with edge technologies and process the data where it is created rather than storing it in huge infrastructures. This project discusses one of the packages available for computer vision that has been used for HPD.

2.5 Key Point Detection models for HPD

Key points in any human pose detection are nothing but the reference points that are mapped to human parts or limb intersections that can be further used in applications like pose tracking, 2D/3D model generation etc. Key point detection is one of the major steps in locating key points in detected human subject. Below are few of the key point detection models for HPD:

OpenPose

OpenPose is first-in-class multi-person real-time key point detection model that was invented in Carnegie Mellon University (CMU) by the Perceptual Computing labs [14]. OpenPose uses CNN based architecture to recognize the human parts with key points which include eyes, ears, neck, nose, elbows, shoulders, knees, wrists, ankles, and hips. A total of 18 key points is generated from the given input and the input may be static i.e., an image or a recorded video or a real-time camera recording. Due to its real-time processing, its applications are vast ranging from activity detection, event detection to real-time exercise posture correction. The model proposed in [15] uses OpenPose for reviewing gym exercise by providing the recorded video of the exercise as input and let the model comment on the video. Outdrawn output is not real-time and doing exercise without real-time supervision could lead to injuries and even waste of time.

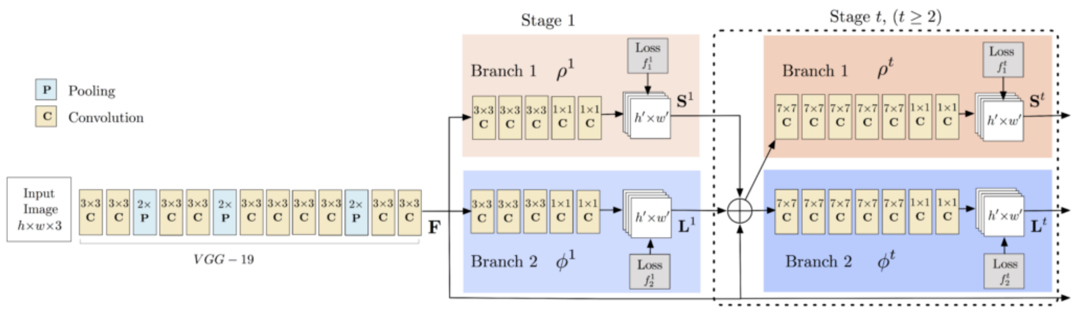


Figure 2 Architecture of OpenPose [14]

The process of OpenPose is divided into three parts i.e., stage 0, stage 1 and stage 2 shown in fig [2]. In stage 0, The 10-layer VGG Net creates feature maps for the given input image. The initial layer of the model architecture is shown in the fig [2]. These extracted features are then fore forwarded to stage 1 where two convolution layer networks which run in parallel. At this stage, one branch predicts 18 confidence maps, which represent major parts of human body in given image and another branch 38 Part Affinity Fields (PAF) which showcases the degree of association between the parts. In stage 2, the confidence and affinity maps are used to produce the 2D key points for any image.

PoseNet

PoseNet is another deep learning framework for human pose detection from images, video, or any continuous image stream by localizing the human joints in human body. PoseNet is like OpenPose but there are few architectural changes. The SoftMax layer in OpenPose is replaced by a sequence of fully connected layers. A detailed high-level architecture of PoseNet is shown in fig [3][16]. PoseNet considerably a lightweight architecture designed to work on mobiles and hardware with less processing power. PoseNet gives us a total of 17 key points that are labeled as the body part ID which is basically a confidence score that lies in the range of 0.0 to 1.0 with 1.0 as maximum value. As shown in the fig [3], the first stage in the architecture is an encoder which generates the encoding vector v, a 1024-dimensional vector that is an encoded representation of the features of the input image.

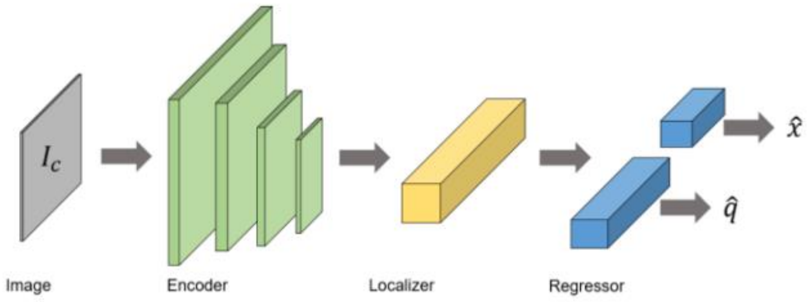


Figure 3 A Schematization of PoseNet Architecture [17]

2.6 HPD for Exercises and Activity

Performing any physical activity like sport, gymnasium, athletics etc., have some set of co-ordinations in between human body parts which creates a harmony of actions to attain a desired output. For example, efficient swimmers need to place their head in certain angle with the water level for not breaking the friction barrier resulting in a perfect swim. Likewise, weightlifting in gym demands a standard posture for different body types in invoking the muscle tension with proper contraction and expansion mechanisms at the optimal level resulting in muscle growth. Science behind any activity has left some imprints of measurements which could be taken as input to a deep learning model that can assume these inputs as thresholds and compare the user activity with the set values for each kind of posture in judging the correctness of the performed physical labor. This project is the result of such thought where posture correction in exercising is monitored using some set of necessary assumptions against which data created by user activity is compared and analyzed for better activity performance.

CHAPTER 3

DESIGN

In this project I have used a pre trained model developed in caffe framework for human pose detection and the entire project is coded in python programming language. Two datasets have been used with the same model and the results are compared for performance analysis.

3.1 Objective

The main objective of this project is to build a real-time application that assists users to perform exercise without any physical assistance. This application will monitor posture of the user while performing exercise and reverts the comments if the exercise is ill performed. Considering squat as the exercise of analysis in this project, factors that need to be considered are both the knees should be in align with the hip and the toe. The skeleton that is mapped onto the user image is colored green when the posture maintained is appropriate. Change in color to red occurs when the angle at the knee and alignment of knee with hip and toe are disturbed resulting in failure of activity. Application pops appropriate messages on tracking screen to either correct left leg if the left leg is positioned wrong or correct right leg when right leg is positioned wrong or both. Apart from this, a video representing the skeleton impression of the user activity is produced and saved in project file which can further be used to analyze the user performance in performing exercise. Main idea is to produce this application on hardware with minimal processing power assuming the availability of compatible system hardware in most of the users.

3.2 Tools and Technology Used

PyCharm

PyCharm is an IDE used for computer programming especially in python programming language. It is developed by the Czech Republic company JetBrains. It enables some smart features like and supports web development with Django. Any AI project is complex in nature. They have numerous files based on the application developed and to handle these files effectively we need a good IDE for files to be executed in command prompt enabled environment. PyCharm provides all the necessary features for project development in machine learning and deep learning and integration with database and support to Django framework will enable users to develop web applications as well. Few additional features of this IDE are code analysis, python intelli-sence, integrated unit tester, A graphical debugger, enables integration with version control systems etc.

PyPI

The python package index (PyPI) is an official third-party repository of software for the python programming language. PyPI helps to find and install software that are developed and shared by python community. It is often referred as cheese shop. PyPI primarily hosts python packages in the form of archives called sadists (source distributions) or pre-compiled "wheels"[17]. PyPI not only holds the recent releases but also store all previous versions of the release and enables to search for packages for key words or by filters against their metadata.

Requirements for installing packages

- Ensure python can be ran through command line in the machine.
- Ensure pip can be ran through command line in the machine
- Ensure pip, setup tools and wheel are up to date
- Optionally, create a virtual environment

Packages Installed

Python-Math

This module enables basic mathematical functions defined by C standard along with functions like number-theoretic and representation functions, power and logarithmic functions, trigonometric functions, Angular conversion Hyperbolic Functions and other special functions []. For this project, measuring angles between two key points is vital so this module helps in achieving it. Installation command on pip installer for this module is

pip install python-math

Argparse

This module is basically a parser for command-line options, arguments, and sub-commands. The argparse module makes it easy to write user-friendly command-line interfaces.
The program defines what arguments it requires, and argparse will figure out how to parse those out of *sys argv*. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments. Installation command on pip installer for this package is

pip install argparse

Time

This module provides time related functionalities. In this project, time taken for processing the collective of 24 frames into one frame is calculated for knowing the efficiency of the model. Installation command on pip installer for this package is

NumPy

Numerical Python (NumPy) is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. Installation command on pip installer for this package is

pip install numpy

OpenCV

OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision.[18] Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel. The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations [19]. It is free open-source library under Apache 2 License and a cross platform library. It features GPU acceleration for real-time operations. This package comes with functions like computer vision algorithms. Installation command on pip installer for this package is

pip install opencv-python

3.3 Datasets

COCO

Microsoft Common Object in Context also termed as COCO or MS-COCO [20] is an image dataset created with the goal of advancing image recognition. The coco dataset contains challenging high-quality visual datasets for computer vision, mostly for training state-of-the-art neural networks. Coco is often used as a benchmark algorithm to compare the performance of various models for real-time object detection.
Coco dataset is automatically interpreted by advanced neural network libraries. Features of coco dataset are

- Recognition in context
- Over 200000 images of the total 330000 images are labeled
- 5 captions per image

- Super pixel stuff segmentation
- Context recognition
- Object segmentation with detailed instance annotations

COCO recommends using the open-source tool Fifty-one to access the MS-COCO dataset for building computer vision models.

MPII

MPII Human pose dataset is a human pose estimation dataset. It consists of around 25K images extracted from online videos. Each image contains one or more people, with over 40K people annotated in total. Among 40K samples, nearly 28K samples are for training and the remaining are dedicated for testing. The complete dataset covers 410 human activities, and each image is provided with an activity label. The images were methodically collected using an established taxonomy of general human activities. The dataset covers 410 human activities, where each image is provided with its activity label. The images were extracted from a YouTube video, and each were provided with their respective preceding and succeeding un-annotated frames.

3.4 OpenPose pre-trained model

Training a model with datasets like COCO and MPII is a heavy operational task where the entire process can take too long and can be a significant load on the hardware. This project uses a state-of-art pre-trained model that is developed with caffe architecture and won COCO challenge in 2016[25] along with CMU-Pose model known as OpenPose. As it adapted caffe architecture, this model ensures speed with capacity of processing 60M images per day on a single NVIDIA K40 GPU. Compatibility with various hardware devices like CUDA GPUs, OpenCL GPUs, only CPU driven hardware etc., [25] made it to be one of most used caffe models in the market. Source code for invoking this caffe model needs a pre-trained model binary file (. caffemodel) that basically define weights, biases and gradients for each layer of the network and a model definition (. prototxt) file which holds the structure related data of the used network.

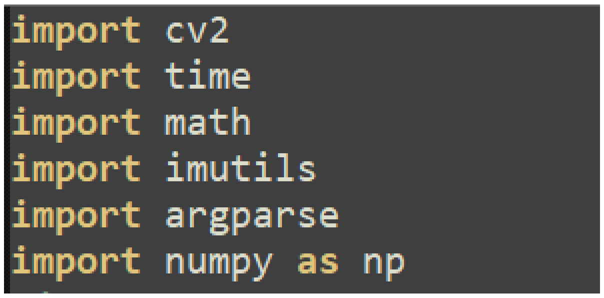
CHAPTER 4

IMPLEMENTATION AND RESULT

In this project, OpenPose model pre-trained on both COCO and MPII dataset is used for developing the proposed application. For better understanding and result separation two different files are created. In this section, implementation of code in each file is detailed along with their resulted output in monitoring a regular unweighted squat exercise.

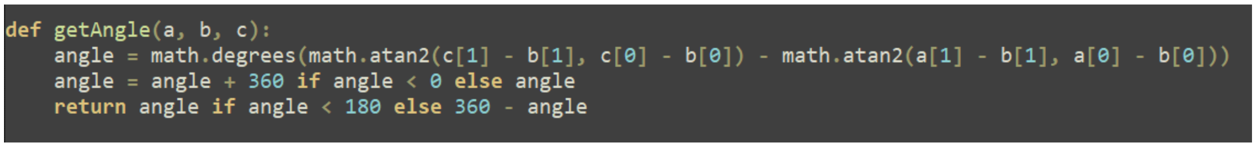
4.1 With COCO Dataset (coco_pose.py)

This file holds the code snippet relating the pre-trained OpenPose model using COCO dataset. The code starts with importing all the necessary packages as shown below



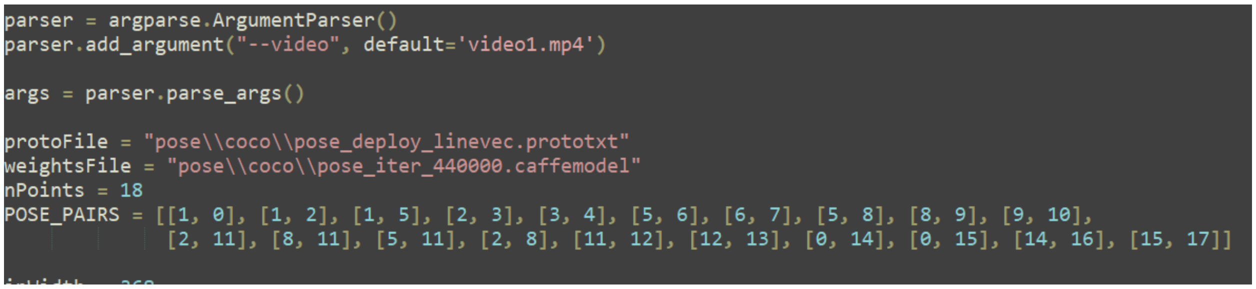
```
import cv2
import time
import math
import imutils
import argparse
import numpy as np
```

Below code snippet holds function *'getAngle()'* to calculate the necessary angle between the hip, knee and ankle co-ordinates



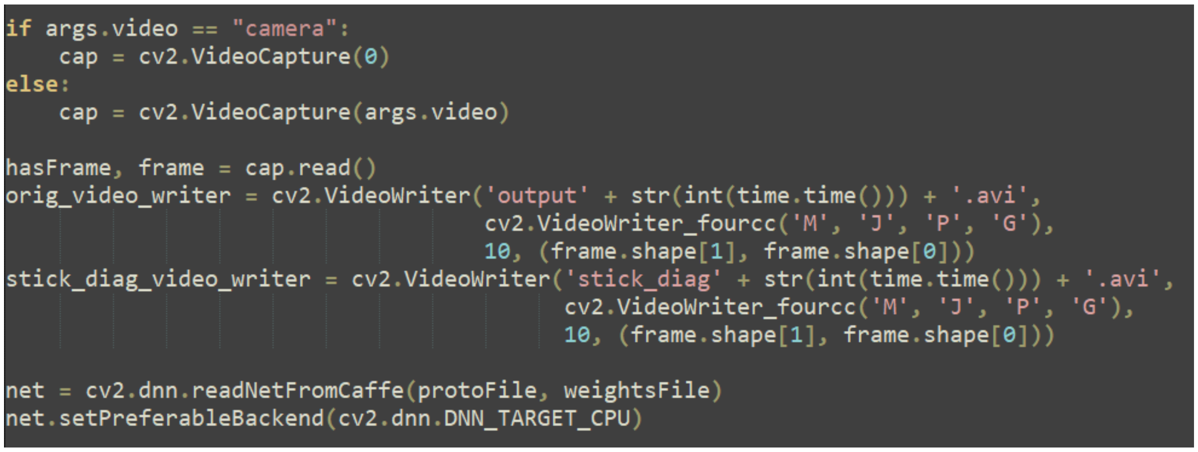
```
def getAngle(a, b, c):
    angle = math.degrees(math.atan2(c[1] - b[1], c[0] - b[0]) - math.atan2(a[1] - b[1], a[0] - b[0]))
    angle = angle + 360 if angle < 0 else angle
    return angle if angle < 180 else 360 - angle
```

Then, using argparse, default input to pointed to a sample video if camera is not chosen as the source of input video stream. Later in the below code snippet, the model related weights file and the architecture file are directed into the code file. The 18 co-ordinates are used to map the stick diagram between the major joints in the frame captured while performing exercises. This varies for MPII based model which will be further discussed in later sections.



```
parser = argparse.ArgumentParser()
parser.add_argument("--video", default='video1.mp4')

args = parser.parse_args()

protoFile = "pose\\coco\\pose_deploy_linevec.prototxt"
weightsFile = "pose\\coco\\pose_iter_440000.caffemodel"
nPoints = 18
POSE_PAIRS = [[1, 0], [1, 2], [1, 5], [2, 3], [3, 4], [5, 6], [6, 7], [5, 8], [8, 9], [9, 10],
              [2, 11], [8, 11], [5, 11], [2, 8], [11, 12], [12, 13], [0, 14], [0, 15], [14, 16], [15, 17]]
```

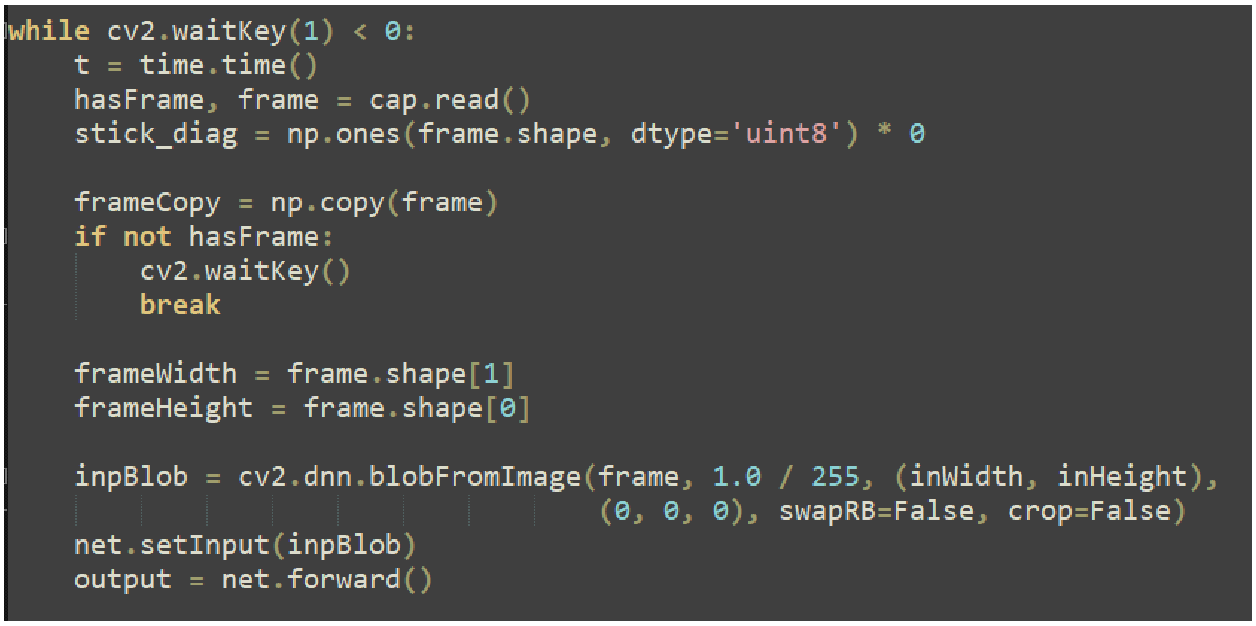
The camera capture object is created for which the possible inputs can only be either the video stream from camera or a pre- recorded video. Then, the captured frame is loaded into the frame object. This output of this code pops two result windows – one with the frame containing detected subject mapped with stick diagram and another canvas window with a recording of the stick diagram holding the action of the subject.



```
if args.video == "camera":
    cap = cv2.VideoCapture(0)
else:
    cap = cv2.VideoCapture(args.video)

hasFrame, frame = cap.read()
orig_video_writer = cv2.VideoWriter('output' + str(int(time.time())) + '.avi',
                                    cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
                                    10, (frame.shape[1], frame.shape[0]))
stick_diag_video_writer = cv2.VideoWriter('stick_diag' + str(int(time.time())) + '.avi',
                                          cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
                                          10, (frame.shape[1], frame.shape[0]))

net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)
net.setPreferableBackend(cv2.dnn.DNN_TARGET_CPU)
```

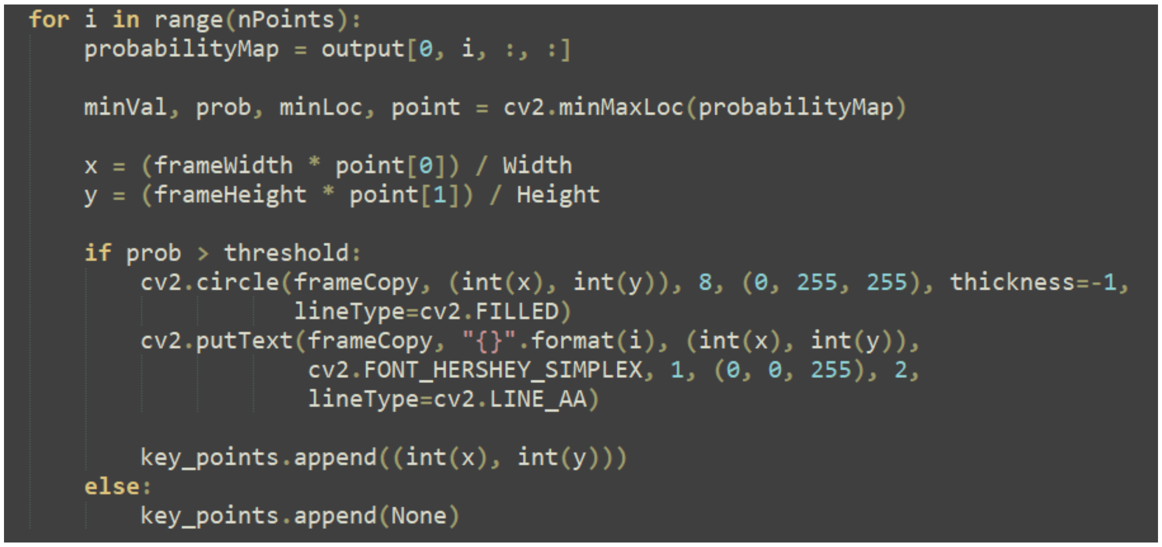
An object is loaded with files related to the pre-trained model and an inference for CPU is created through that object.

Followed with an infinite loop that reads each image frame captured from the input stream and holds the time at that read operation. Pre-processing of the read image frame is carried before feeding it to net object as the model is trained to identify one specific type of objects out of the provided input. The object holding the image vector is then fed as input to the net object which outputs the label object.



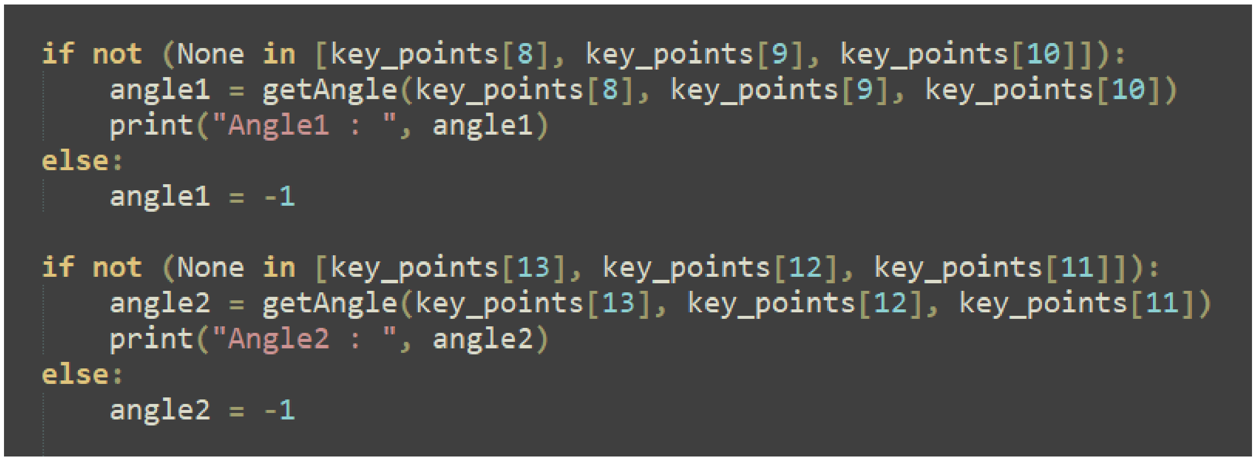
```
while cv2.waitKey(1) < 0:
    t = time.time()
    hasFrame, frame = cap.read()
    stick_diag = np.ones(frame.shape, dtype='uint8') * 0

    frameCopy = np.copy(frame)
    if not hasFrame:
        cv2.waitKey()
        break

    frameWidth = frame.shape[1]
    frameHeight = frame.shape[0]

    inpBlob = cv2.dnn.blobFromImage(frame, 1.0 / 255, (inWidth, inHeight),
                                    (0, 0, 0), swapRB=False, crop=False)
    net.setInput(inpBlob)
    output = net.forward()
```

As the labels for the key points are now available, the probability map co-ordinates are read from the confidence map knowing how confident is that co-ordinates marked. Then, key point co-ordinates obtained from confidence map are saved for connecting and giving shape to the stick skeleton.



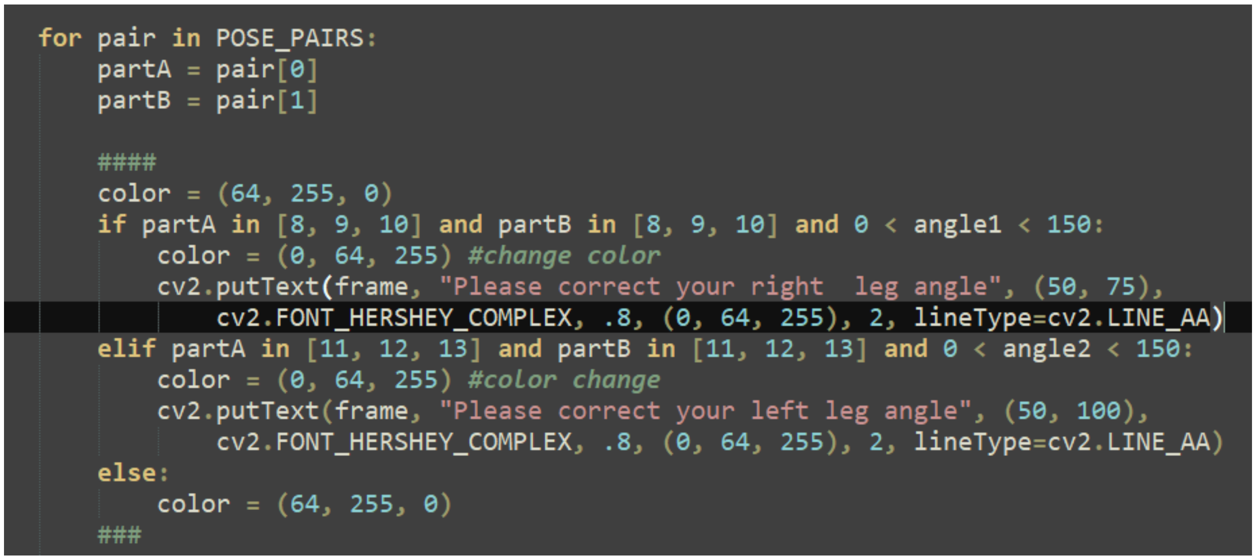
```python
for i in range(nPoints):
    probabilityMap = output[0, i, :, :]

    minVal, prob, minLoc, point = cv2.minMaxLoc(probabilityMap)

    x = (frameWidth * point[0]) / Width
    y = (frameHeight * point[1]) / Height

    if prob > threshold:
        cv2.circle(frameCopy, (int(x), int(y)), 8, (0, 255, 255), thickness=-1,
                   lineType=cv2.FILLED)
        cv2.putText(frameCopy, "{}".format(i), (int(x), int(y)),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2,
                    lineType=cv2.LINE_AA)

        key_points.append((int(x), int(y)))
    else:
        key_points.append(None)
```

Based on the key point cartesian co-ordinates at hip, knee and ankle, the angle is calculated for both left leg and right leg while performing a standard no-weighted squat. If the ankle is not visible due to any positional issues, then a default angle is shown as -1 which indicates that there is a visibility issue.



```python
if not (None in [key_points[8], key_points[9], key_points[10]]):
    angle1 = getAngle(key_points[8], key_points[9], key_points[10])
    print("Angle1 : ", angle1)
else:
    angle1 = -1

if not (None in [key_points[13], key_points[12], key_points[11]]):
    angle2 = getAngle(key_points[13], key_points[12], key_points[11])
    print("Angle2 : ", angle2)
else:
    angle2 = -1
```

The entire stick diagram is initially colored green while standing still which is considered the beginning of the squat and while performing the activity itself, change in color of the legs i.e., from hip to toe takes place if the exercise is performed wrong deforming the angle of 180 by more than 30. Correction instructions relating the wrong posture is passed onto the output subject frame and message can be relating to a single leg or both legs depending on the performance of the subject.

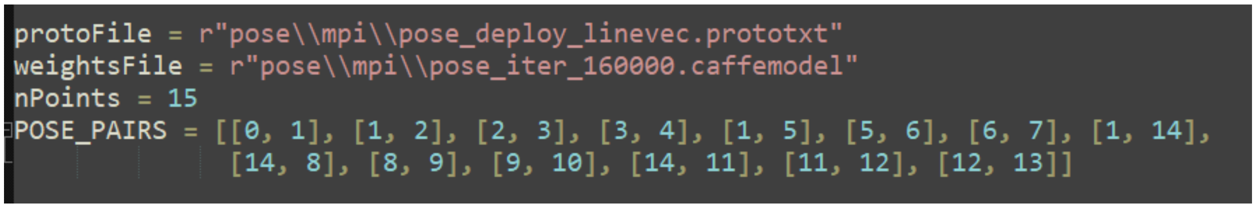


```
for pair in POSE_PAIRS:
    partA = pair[0]
    partB = pair[1]

    ####
    color = (64, 255, 0)
    if partA in [8, 9, 10] and partB in [8, 9, 10] and 0 < angle1 < 150:
        color = (0, 64, 255) #change color
        cv2.putText(frame, "Please correct your right  leg angle", (50, 75),
            cv2.FONT_HERSHEY_COMPLEX, .8, (0, 64, 255), 2, lineType=cv2.LINE_AA)
    elif partA in [11, 12, 13] and partB in [11, 12, 13] and 0 < angle2 < 150:
        color = (0, 64, 255) #color change
        cv2.putText(frame, "Please correct your left leg angle", (50, 100),
            cv2.FONT_HERSHEY_COMPLEX, .8, (0, 64, 255), 2, lineType=cv2.LINE_AA)
    else:
        color = (64, 255, 0)
    ###
```

The result of the above code showing all the discussed output features are shown further in this section.

4.2 With MPII Dataset (mpi_pose.py)

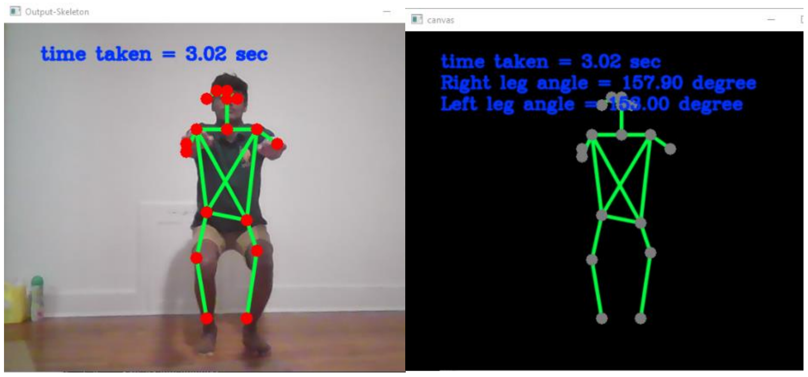
This file holds the same theme as the previous code file with minimal changes at the model file level and the number of key point co-ordinates that gets mapped onto the subject in output frame and the stick diagram canvas as well.



```
protoFile = r"pose\\mpi\\pose_deploy_linevec.prototxt"
weightsFile = r"pose\\mpi\\pose_iter_160000.caffemodel"
nPoints = 15
POSE_PAIRS = [[0, 1], [1, 2], [2, 3], [3, 4], [1, 5], [5, 6], [6, 7], [1, 14],
              [14, 8], [8, 9], [9, 10], [14, 11], [11, 12], [12, 13]]
```

4.3 Results

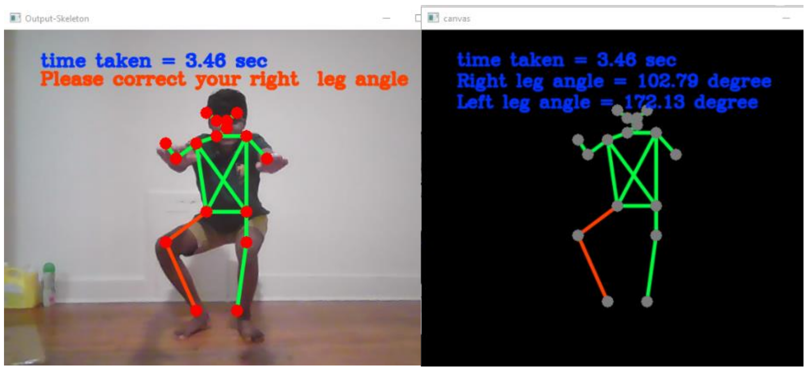
The outcome of this project is to check the correctness of performed un-weighted squat and to correct the posture if not practiced effectively by instructing the user in the form of a message on one of the output windows. Below screenshots of the result windows show an user performing a perfect squat, squat with partial posture negligence and squat with a total failure. Recommendation messages can be seen in the left window in every wrongly performed situation.



```
time taken = 3.02 sec
```

```
time taken = 3.02 sec
Right leg angle = 157.90 degree
Left leg angle = 153.00 degree
```

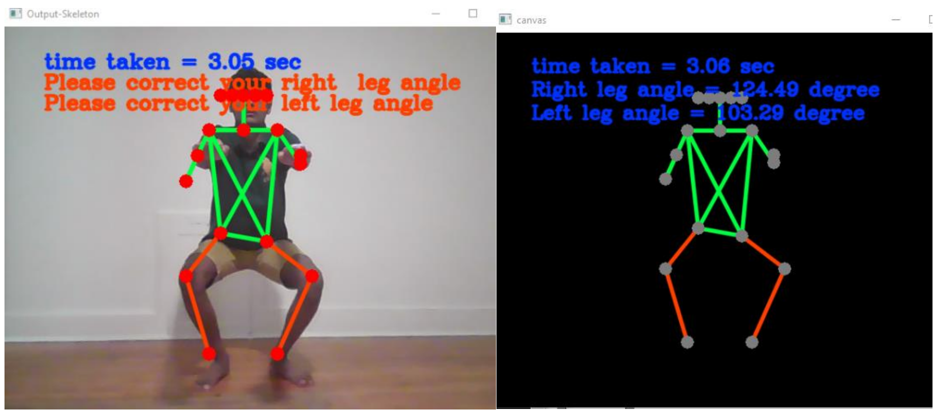
In the below result screenshot, posture of the user is indicating a flaw in performing the squat by bending the right leg more than suggested. Hence the applications suggest the user to correct his posture saying, '*Please correct your right leg angle*'.



```
time taken = 3.46 sec
Please correct your right  leg angle
```

```
time taken = 3.46 sec
Right leg angle = 102.79 degree
Left leg angle = 172.13 degree
```

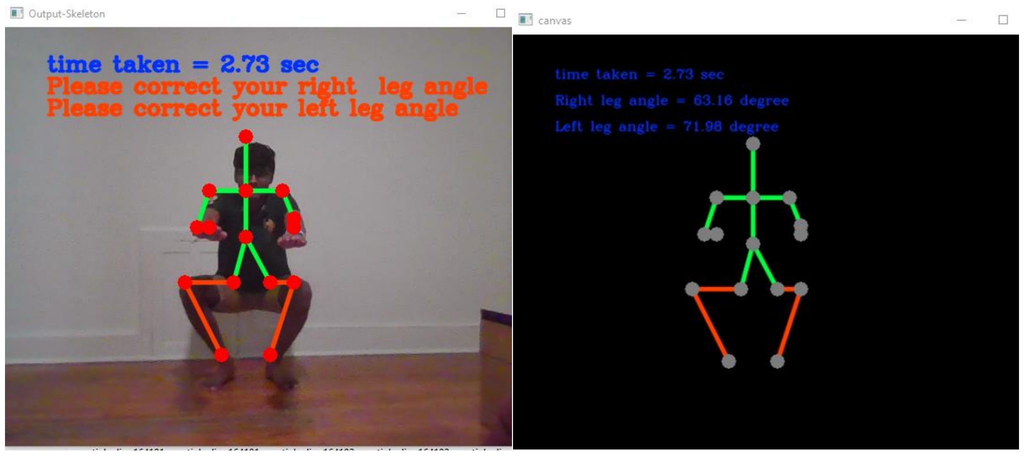
A total failure of practicing squat is shown in the image below where the position of both legs exceeds the suggesting angle range resulted in the correction message saying, ‘*Please correct your right leg Please correct your left leg*’. Instruction can be customized based on the user requirement in further carried research depending on the exercise and to what body part that suggestion refers to.



```
time taken = 3.05 sec
Please correct your right  leg angle
Please correct your left leg angle
```

```
time taken = 3.06 sec
Right leg angle = 124.49 degree
Left leg angle = 103.29 degree
```

Posture detection lies the same for both COCO and MPII based trainings but difference underly in the processing level in terms of time consumption. Below are the screenshots of output resulted using MPII trained model in total failure case with visible suggestions in the output skeleton window in the left.



```
time taken = 2.73 sec
Please correct your right  leg angle
Please correct your left leg angle
```

```
time taken = 2.73 sec
Right leg angle = 63.16 degree
Left leg angle = 71.98 degree
```

CHAPTER 5

CONCLUSION

5.1 Accomplishments and lessons learned

As part of the project research, I developed a deep understanding of the human pose detection models and requirements for developing a real-time exercise posture correction system. The implementation of the pre-trained model on COCO and MPII dataset produced lag results in terms of pose detection and where performance of MPII dataset trained model lied ahead than that of COCO trained model which had an upper hand on partially visible human torso. Due to lag in pose detection, the real-time supervision was affected, and exercise couldn't be corrected while performing it but had better results when image inputs are provided. Providing more processing power or GPU could improvise the performance in handling the situation better. In this process of materializing the idea, pose detection pre-trained models like VGG_Pose, Mediapipe etc., are explored. By implementing this idea, models build on caffe architecture are not the best fit to handle real-time applications proposed in this project at least with CPU driven hardware.

5.2 Future Work

The application developed in this project can only correct a partial factor of one exercise. Further research can be done in developing a full-fledged application or a portable device with modules that can correct all the known exercises in an activity and develop a cross-platform app where multiple activity categories like yoga, Zumba dance etc., can be accessed by users based on their choice. The performance of model depends on the OpenPose pose estimation model which may show some lag in CPU driven hardware but can perform lag-free on GPUs. The idea projected in this project can be applied in various fields like human co-ordination analysis where movements of multiple subjects be compared for better pattern performance or surveillance in remote areas where life of a human being needs immediate medical attention. There are a lot of scenarios where single person pose estimation would not suffice, for example pose estimation in crowded scenarios would have multiple persons who will involve tracking and identifying pose of everyone. A lot of factors such as background, lighting, overlapping figures etc. which have been discussed earlier in this survey would further make multi-person pose estimation challenging.

APPENDIX A

APPENDIX

Coco_pose.py code

```
import cv2
import time
import math
import imutils
import argparse
import numpy as np

def getAngle(a, b, c):
    angle = math.degrees(math.atan2(c[1] - b[1], c[0] - b[0]) - math.atan2(a[1] -
b[1], a[0] - b[0]))
    angle = angle + 360 if angle < 0 else angle
    return angle if angle < 180 else 360 - angle


parser = argparse.ArgumentParser()
parser.add_argument("--video", default='video1.mp4')

args = parser.parse_args()

protoFile = "pose\\coco\\pose_deploy_linevec.prototxt"
weightsFile = "pose\\coco\\pose_iter_440000.caffemodel"
nPoints = 18
POSE_PAIRS = [[1, 0], [1, 2], [1, 5], [2, 3], [3, 4], [5, 6], [6, 7], [5, 8], [8, 9], [9,
10],
              [2, 11], [8, 11], [5, 11], [2, 8], [11, 12], [12, 13], [0, 14], [0, 15], [14, 16],
[15, 17]]

inWidth = 368
inHeight = 368
threshold = 0.1
frame_count = 0
total_time = 0

if args.video == "camera":
    cap = cv2.VideoCapture(0)
else:
    cap = cv2.VideoCapture(args.video)

hasFrame, frame = cap.read()
orig_video_writer = cv2.VideoWriter('output' + str(int(time.time())) + '.avi',
                                    cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
                                    10, (frame.shape[1], frame.shape[0]))
stick_diag_video_writer = cv2.VideoWriter('stick_diag' + str(int(time.time())) +
'.avi',
                                          cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'),
                                          10, (frame.shape[1], frame.shape[0]))

net = cv2.dnn.readNetFromCaffe(protoFile, weightsFile)
net.setPreferableBackend(cv2.dnn.DNN_TARGET_CPU)

while cv2.waitKey(1) < 0:
    t = time.time()
    hasFrame, frame = cap.read()
    stick_diag = np.ones(frame.shape, dtype='uint8') * 0

    frameCopy = np.copy(frame)
    if not hasFrame:
        cv2.waitKey()
        break

    frameWidth = frame.shape[1]
    frameHeight = frame.shape[0]

    inpBlob = cv2.dnn.blobFromImage(frame, 1.0 / 255, (inWidth, inHeight),
                                    (0, 0, 0), swapRB=False, crop=False)
    net.setInput(inpBlob)
    output = net.forward()

    Height = output.shape[2]
    Width = output.shape[3]

    key_points = []

    for i in range(nPoints):
        probabilityMap = output[0, i, :, :]

        minVal, prob, minLoc, point = cv2.minMaxLoc(probabilityMap)

        x = (frameWidth * point[0]) / Width
        y = (frameHeight * point[1]) / Height
        if prob > threshold:
            cv2.circle(frameCopy, (int(x), int(y)), 8, (0, 255, 255), thickness=-1,
                       lineType=cv2.FILLED)
            cv2.putText(frameCopy, "{}".format(i), (int(x), int(y)),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2,
                        lineType=cv2.LINE_AA)

            key_points.append((int(x), int(y)))
        else:
            key_points.append(None)

    if not (None in [key_points[8], key_points[9], key_points[10]]):
        angle1 = getAngle(key_points[8], key_points[9], key_points[10])
        print("Angle1 : ", angle1)
    else:
        angle1 = -1

    if not (None in [key_points[13], key_points[12], key_points[11]]):
        angle2 = getAngle(key_points[13], key_points[12], key_points[11])
        print("Angle2 : ", angle2)
    else:
        angle2 = -1


    for pair in POSE_PAIRS:
        partA = pair[0]
        partB = pair[1]

        ####
        color = (64, 255, 0)
        if partA in [8, 9, 10] and partB in [8, 9, 10] and 0 < angle1 < 150:
            color = (0, 64, 255) #change color
            cv2.putText(frame, "Please correct your right  leg angle", (50, 75),
                cv2.FONT_HERSHEY_COMPLEX, .8, (0, 64, 255), 2,
lineType=cv2.LINE_AA)
        elif partA in [11, 12, 13] and partB in [11, 12, 13] and 0 < angle2 < 150:
            color = (0, 64, 255) #color change
            cv2.putText(frame, "Please correct your left leg angle", (50, 100),
                cv2.FONT_HERSHEY_COMPLEX, .8, (0, 64, 255), 2,
lineType=cv2.LINE_AA)
        else:
            color = (64, 255, 0)
        ###

        if key_points[partA] and key_points[partB]:
            cv2.line(frame, key_points[partA], key_points[partB], color, 3,
                    lineType=cv2.LINE_AA)
            cv2.circle(frame, key_points[partA], 8, (0, 0, 255), thickness=-1,
                      lineType=cv2.FILLED)
            cv2.circle(frame, key_points[partB], 8, (0, 0, 255), thickness=-1,
                      lineType=cv2.FILLED)

            cv2.line(stick_diag, key_points[partA], key_points[partB], color, 3,
                    lineType=cv2.LINE_AA)
            cv2.circle(stick_diag, key_points[partA], 8, (125, 125, 125), thickness=-1,
                      lineType=cv2.FILLED)
            cv2.circle(stick_diag, key_points[partB], 8, (125, 125, 125), thickness=-1,
                      lineType=cv2.FILLED)

    cv2.putText(frame, "time taken = {:.2f} sec".format(time.time() - t), (50, 50),
                cv2.FONT_HERSHEY_COMPLEX, .8, (255, 50, 0), 2,
lineType=cv2.LINE_AA)
    cv2.putText(stick_diag, "time taken = {:.2f} sec".format(time.time() - t), (50,
50),
                cv2.FONT_HERSHEY_COMPLEX, .8, (255, 50, 0), 2,
lineType=cv2.LINE_AA)

    cv2.putText(stick_diag, "Right leg angle = {:.2f} degree".format(angle1), (50,
80),
                cv2.FONT_HERSHEY_COMPLEX, .8, (255, 50, 0), 2,
lineType=cv2.LINE_AA)
    cv2.putText(stick_diag, "Left leg angle = {:.2f} degree".format(angle2), (50,
110),
                cv2.FONT_HERSHEY_COMPLEX, .8, (255, 50, 0), 2,
lineType=cv2.LINE_AA)

    cv2.imshow('Output-Skeleton', frame)
    cv2.imshow('canvas', imutils.resize(stick_diag, width=600))

    orig_video_writer.write(frame)
    stick_diag_video_writer.write(stick_diag)

    frame_count += 1
    total_time += (time.time() - t)

orig_video_writer.release()
stick_diag_video_writer.release()
cap.release()
cv2.destroyAllWindows()

print("Total number of frames: ", frame_count)
print("Total time spent: {:0.2f} sec".format(total_time))
print("The total time spent per frame : {:.2f} sec".format(total_time /
frame_count))
```

REFERENCES

[1] Deep Learning-Based Human Pose Estimation:
A Survey Ce Zheng∗, Wenhan Wu∗, Taojiannan Yang, Sijie Zhu, Chen Chen, Member, IEEE, Ruixu Liu, Ju Shen, Senior Member, IEEE, Nasser Kehtarnavaz Fellow, IEEE and Mubarak Shah, Fellow, IEEE arXiv:2012.13392v3 [cs.CV] 2 Jan 2021.

[2] DeepPose: Human Pose Estimation via Deep Neural Networks Alexander Toshev Christian Szegedy. arXiv:1312.4659v3 [cs.CV] 20 Aug 2014

[3] Exploring Strategies for Training Deep Neural NetworksHugo Larochelle,Yoshua Bengio , Jer´ ome ˆ Louradour Pascal Lamblin Departement ´ d'informatique et de recherche oper´ ationnelle, Universite´ de Montreal ´2920, chemin de la Tour Montreal, ´ Quebec, ´ Canada, H3T 1J8

[4] Scaling Learning Algorithms Towards AI Authors: Yoshua Bengio, Yann LeCun, 2007.

[5] On the Quantitative Analysis of Deep Belief Networks
Ruslan Salakhutdinov, Iain Murray, Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G4, Canada

[6] Greedy Layer-Wise Training of Deep Networks Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle Universit´e de Montr´eal Montr´eal, Qu´ebec

[7] A. Gupta, T. Chen, F. Chen, and D. Kimber, "Systems and methods for human body pose estimation", U.S. patent, 7,925,081 B2, 2011.

[8] H. Sidenbladh, M. Black, and D. Fleet, "Stochastic tracking of 3D human figures using 2D image motion", Proc 6th European Conf. Computer Vision, 2000.

[9] A. Agarwal and B. Triggs, "3D human pose from silhouettes by relevance vector regression", Intl Conf. on Computer Vision & Pattern Recogn.pp.882–888, 2004.

[10] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and H.

Zahzah, “Human pose estimation from monocular images: a comprehensive survey”, Sensors, Basel, Switzerland, vol. 16, 2016.

[11] G. Ning, P. Liu, X. Fan and C. Zhan, “A top-down approach to articulated human pose estimation and tracking”, ECCV Workshops, 2018.

[12] M. Li, Z. Zhou, J. Li and X. Liu, “Bottom-up pose estimation of multiple person with bounding box constraint”, 24th Intl. Conf. Pattern Recogn.,2018.

[13] Richard Szeliski (30 September 2010). Computer Vision: Algorithms and Applications. Springer Science & Business Media. pp. 10–16. ISBN 978-1-84882-935-0.

[14] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using part affinity fields”, Proc. 30th IEEE Conf. Computer Vision and Pattern Recogn,2017.

[15] Pose Trainer: Correcting Exercise Posture using Pose Estimation Steven Chen * Richard R. Yang ∗ Department of Computer Science, Stanford University arXiv:2006.11718v1 [cs.CV] 21 Jun 2020

[16] Y. Shavit, R. Ferens, “Introduction to camera pose estimation with deep learning”, [Online]. Available: https://arxiv.org/pdf/1907.05272.pdf.

[17] "PEP 427 -- The Wheel Binary Package Format 1.0". Python Software Foundation. 15 February 2013. Retrieved 28 October 2017.

[18] Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue. 10 (4): 40:40–40:56. doi:10.1145/2181796.2206309.

[19] https://en.wikipedia.org/wiki/OpenCV#cite_note-1

[20] Microsoft COCO: Common Objects in Context. Tsung-Yi Lin Michael Maire Serge Belongie Lubomir Bourdev Ross Girshick James Hays Pietro Perona Deva Ramanan C. Lawrence Zitnick Piotr Dollar arXiv:1405.0312v3 [cs.CV] 21 Feb 2015.

[21] N. Nordsborg, H. Espinosa, “Estimating energy expenditure during front crawl swimming using accelerometrics”, Procedia Eng., 2014.

[22] P. Pai, L. Changliao, K. Lin, “Analyzing basketball games by support vector machines with decision tree model”, Neural Comput. Appl., 2017.

[23] S. Patil, A. Pawar, A. Peshave, “Yoga tutor: visualization and analysis using

SURF algorithm”, Proc. IEEE Control Syst. Grad. Research Colloquium, 2011.

[24] A Portable Smart Fitness Suite for Real-Time Exercise Monitoring and Posture CorrectionAbdul Hannan 1,* , Muhammad Zohaib Shafiq 2 , Faisal Hussain 3,* and Ivan Miguel Pires 4,5,* published on 8 oct 2021.

[25] Physical Exercise Form Correction Using Neural Networks Cristian Militaru, Maria-Denisa Militaru, Kuderna-Iulian Benţa ICMI '20 Companion, October 25–29, 2020, Virtual Event, Netherlands.

[26] OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields Zhe Cao, Student Member, IEEE, Gines Hidalgo, Student Member, IEEE,
Tomas Simon, Shih-En Wei, and Yaser Sheikh arXiv:1812.08008v2 [cs.CV] 30 May 2019