TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

A

FINAL YEAR PROJECT REPORT

ON

FEEDBACK BASED POSTURE CORRECTION SYSTEM FOR GYM EXERCISES USING HUMAN POSE ESTIMATION

By:

Rabindra Regmi 073BCT531

Sabal K.C. 073BCT537

Subash Tiwari 073BCT544

Sujeet Silwal 073BCT546

Lalitpur,Nepal

March,2021

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

FEEDBACK BASED POSTURE CORRECTION SYSTEM FOR GYM EXERCISES USING HUMAN POSE ESTIMATION

By:

Rabindra Regmi 073BCT531

Sabal K.C. 073BCT537

Subash Tiwari 073BCT544

Sujeet Silwal 073BCT546

A PROJECT WAS SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR THE BACHELOR'S DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

LALITPUR, NEPAL

March,2021

LETTER OF APPROVAL

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project report entitled FEEDBACK BASED POSTURE CORRECTION SYSTEM FOR GYM BASED EXERCISE USING HUMAN POSE ESTIMATION submitted by Rabindra Regmi, Sabal K.C., Subash Tiwari, Sujeet Silwal in partial fulfillment of the requirements for the Bachelor's degree in Computer Engineering.

Supervisor: Mrs. Bibha Sthapit
Deputy Head of Department of Electronics and Computer Engineering
Institute of Engineering, Pulchowk Campus

Internal Examiner: Er. Anil Verma
Lecturer
Department of Electronics and Computer Engineering

External Examiner: Bishesh Khanal, PhD
President
NAAMI- Nepal Applied Mathematics and Informatics Institute

DATE OF APPROVAL:

COPYRIGHT

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to the **Department of Electronics and Computer Engineering,** Pulchowk Campus, Institute of Engineering for providing us with this wonderful opportunity and the great platform to apply and extend our knowledge gained in past years performing a real world project. Our special thanks goes to our supervisor **Bibha Sthapit** whose contribution in stimulating suggestions and encouragement helped us a lot for initiating this project.

We would like to express our deep gratitude to **Mr. Rahul Amatya**, instructor of Balaju Gym, Balaju, Kathmandu for providing us the details rule about the exercises so we could integrate those rules into our project.

Our special thanks goes to all the teachers who taught us in the past four years and made us capable to do this project. We would also like to express our gratitude to our classmates for giving feedback on our ideas.

ABSTRACT

Fitness exercises are good for personal health. However doing 'wrong' or ineffectively can be no good and may lead to injury. There are certain exercises that experts want people to stop due to complications and risk involved in incorrectly performing them. Exercise mistakes are often introduced when users lack understanding about the pose and proper form of the exercise.

In our work, we introduce an application that detects the user's exercise, evaluates their pose according to the exercise and provides feedback about the exercise along with the additional feature of exercise tracking at real time. In this way, users can get feedback from the system and work out accordingly to improve exercise efficiency using the feedback.

We use OpenPose, a machine learning based solution for robust pose detection which is used by our system to evaluate the correctness of the exercise based on the rule defined from various dataset, heuristics and feedback from expert trainers. Our system supports the evaluation and feedback for common exercises at real time which can greatly enhance the efficiency of learning and improving the common mistakes that occur while performing the fitness exercise.

Keywords: fitness, pose, pose estimation, machine learning, exercise, rules, feedback, heuristics, learning

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI Artificial Intelligence

ML Machine Learning

CNN Convolution Neural Network

NN Neural Network

UML Unified Modeling Language

PDF Portable Document Format

TOC Table of Contents

UI User Interface

GPU Graphics Processing Unit

SPPE Single Person Pose Estimation

MPPE Multi-Person Pose Estimation

DTW Dynamic Time Warping

COCO Common Objects in Context

OKS Object Keypoint Similarity

CPU Center Processing Unit

API Application Programming Interface

PCP Percentage of Correct Parts

PCK Percentage of Correct Key-points

PDJ Percentage of Detected Joints

IR Infra-red

RGB Red Green Blue

AUC Area Under the Curve

1. INTRODUCTION

1.1. Background

The process of predicting the location of body part and joints from an image is known as pose estimation. It involves tracing the skeleton system of human body in any visual data. It is also important to note that pose estimation has various sub-tasks such as single pose estimation, estimating poses in an image with many people, estimating poses in crowded places, and estimating poses in videos. It can be used in various areas but our goal is to use in posture correction during gym exercise.

Exercise are very beneficial to the health if done correctly but also can be very dangerous if posture and form is incorrect. Some exercises such as deadlift can cause big accident if form is incorrect. Our idea for this project is to develop a software app which can identify whether the person is performing exercise correctly and give feedback on how to improve form. There is always much more people in gym than trainer so trainer can't keep eye on everyone. That might cause problem for those person who are not being focused by the trainer at that moment. Our software,SmartTrainer, will try to minimize that problem by monitoring those people and giving them proper feedback.

First part of SmartTrainer is Human Pose Estimation, which is very difficult but highly applicable field in Computer Vision. Pose estimation is a crucial part for many applications like motion capture, AR/VR and Gait analysis(impact of human walking behavior on their health).

Second part of SmartTrainer is to monitor individual person and give proper feedback on the exercise they are doing. For this, we have implemented rule based system using pose and instruction from personal trainers and other qualified persons as the ground truth for proper form. We have a complete software which consists of feature of recording video to monitor and keep track of user's exercise routine. It also has feature of counting numbers of correct reps and sets and will also record user progress on exercise.

1.2. Motivation

Exercise has various advantages. They help to reduce the risk of several diseases like cancer, diabetes, heart disease and chronic disease.These diseases are the major causes

of death around the world. However, they need to be done correctly with good posture and form otherwise it can be dangerous. Some exercises such as deadlift can cause big accidents if form is not correct.

A beginner with no knowledge of exercise and workouts can make various mistakes when performing even the simplest of exercise. In the case of home workouts, there are information sources, sites or apps which can provide users with the steps to do the exercise. But knowing whether the steps are actually being followed correctly is a challenge. For example, a person could go through an exercise app, read the rules, watch some videos and perform the exercises, but he has no way of knowing whether he is performing it correctly, even in the presence of a mirror. Certain exercises are hard to evaluate individually. A person may unknowingly make unidentifiable mistakes which could hamper his progress. In the case of gym based workouts, a trainer is present to provide feedback and to guide the user. However, the number of trainers to the number of people are not one to one. It is impossible for a trainer to give full attention to a particular person.

So, the main motivation for the project came from these situations. These problems could be tackled by using a virtual AI-powered trainer. If a person can perform exercises at home with just a webcam and a computer, we can use the power of advanced image processing (pose estimation) and exercise analysis to build a system which can provide real time feedback to the user.

Technologies have transformed the world and the field of fitness is no exception. Tech giant Apple has introduced the feature of workout tracking in their new Apple Watch. Google has introduced similar app called Google Fit. Different digital technologies along with artificial intelligence are present as a mobile app, wearable to help user track user's fitness in the form of fitness assistant. However, they are limited to tracking exercises, and progress, generating the summary for particular day or week. They don't provide the feedback to the user regarding the correctness of exercise.

The recent outbreak of COVID-19 this year has shut gyms and outdoor exercises. As a result interest on indoor exercises and workout at home has increased drastically. This has increased demand for a good and productive fitness assistant app. Although various forms of learning from internet like interactive classes, on-demand classes they come with additional monetary cost and finding right schedule.

A data on Google trends shows that during the month of March, a peak time of the lockdown there was an increase in search query on two common fitness keyword "home gym" and "home workouts" with estimated rise by 66% and 55% respectively. This shows that people's interest has increased on home workout due to the lockdown and this might also be the future of the fitness exercises. So instead of going to gyms people may adapt to the fitness assistant, fitness application in the near future.

1.3. Problem Definition

Exercises are great for health if performed correctly. People all over the world perform the exercise either in gyms or at the home. Although many exercises can be performed by watching tutorials videos on YouTube and other sites some exercises can be very difficult as well. Lack of proper knowledge and the trainer at the home leads to a physical injury and damage to the person. Proposed system can solve this issue by detecting the human pose of the exercising video and giving proper feedback to the user which is similar to having a personal trainer. Proposed system can be run with a single computer with a webcam on it, so it can capture the exercising video and after running the pose estimation model, give proper feedback to the user.

1.4. Report Organization

The material presented at the report is organized into 9 sections. Chapter 1 is an introductory section which describes the background, objectives, scope and application of the project. Chapter 2 provides brief summaries of all existing works that have already been carried out in the related field. It describes the activities related to a project that has been carried by some specific people. Chapter 3 explain about the theory of things which has been implemented in this project which includes NN, CNN, OpenPose, Pose Estimation etc. Chapter 4 provides the information on working of system. It also explains the mathematical models required in the project and explains the diagram, dataset and system application of the project and how application works. Chapter 5 describes about the system flow of the application which includes diagram like sequence, state chart, activity and Use case. Chapter 6 focuses on final output and results of the system. Tasks like data collection, visualization and implementation of

rules is explained in this section. Chapter 7 describes the problem that has been faced during this project. Chapter 8 describes the whole project application and final chapter concludes the whole report and also mention about future enhancements.

1.5. Objectives

This project main aim is single pose estimation of human from video and correcting posture of person while doing exercise and giving proper feedback to the users.
The main objectives of this project can be listed as:

1. Estimate key joint of human body using Pose Estimation almost in realtime to detect quality of exercise of a person

2. Provide proper feedback to the user

1.6. Scope

The project mainly focuses on pose estimation of human body. This project explores one of the possible uses of machine learning to improve the exercise efficiency. By the use of the latest pose estimation technique to estimate the human pose in real-time, we have used it to infer the correctness of the exercise pose and provide necessary feedback to improve it. This project aids people by providing a software application that detects user pose and provides useful feedback. The main goal of the project is to improve the quality of workouts with the help of a PC, webcam and AI based personal trainer apps. It is by no means a general system that works for all types of exercise. These exercises included in the project are extensively studied to generate rules and feedback with the help of trainers. They are highly specific to the exercise. The objective of this project is to prevent the side effects due to wrong pose rather than improving upon the correct pose.Hence, this project is targeted to the beginner fitness enthusiast who is planning to learn common fitness exercise.

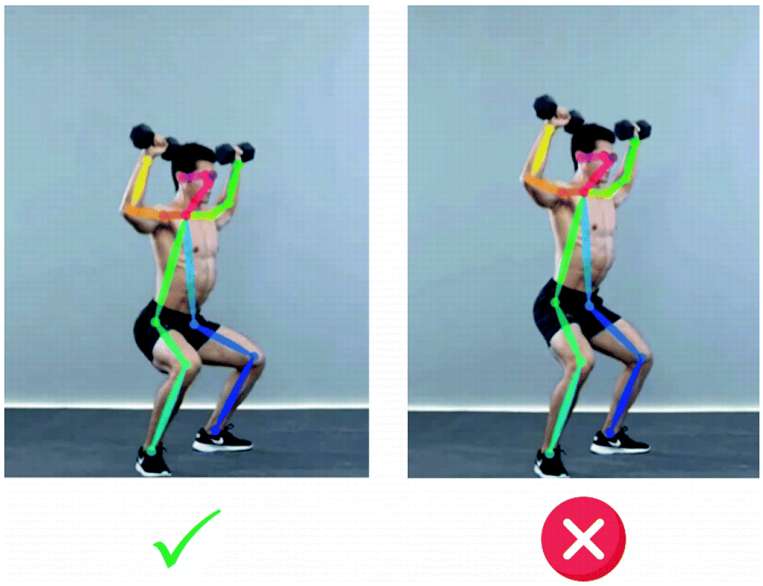


Figure 1.1: Training analysis

2. LITERATURE REVIEW

Numerous studies have been conducted in the area of human pose estimation. There are a number of state-of-the-art methods for estimating human poses that work with a variety of sensor settings, shots, and individual counts per frame. Cao et al. [3] used Part Affinity Fields to estimate poses of multiple people in a scene in real time without the need to identify individual persons first, Cao et al. [4] have open sourced their work as a project called OpenPose, which we have utilized in our project for real time pose estimation. Bazarevsky et al. [11] proposed Blazepose model for Human Pose Estimation which is a very lightweight model that uses convolutional neural network architecture and is more suitable for mobile devices.

In the case of exercise posture correction, Chan et al. [1] proposed a model PoseTrainer which uses OpenPose model for human pose estimation and provides feedback to the user. Also, Ouyang [12] proposed a system called Poze which basically provides live feedback to young women exercising to fitness videos. Talal and Cheng [26] have also proposed exercise recognition by using neural network and counting repetitions of exercise using major joints and angle ranges, both in real time.

In some of the previous works which laid the foundation for pose estimation, Toshev et al. [16] were the first to use a deep neural network to improve pose detection , finding the location of the body joint using regression on CNN features. Newell et al. [15] introduced a stacked hourglass neural architecture that uses repeated bottom-up and top-down processes to achieve accurate single pose predictions. Wei et al. [19] proposes a different architecture, using multiple convolution networks to refine joint estimates over sequential passes. Bogo et al. [14] estimated 3D pose, as well as 3D mesh shape, using just single RGB images.

A significant area of research has also focused on detecting the poses of multiple people in one shot. Papandreou et al. [16] detected multiple poses through a two-stage process, first identifying possible bounding boxes for people and then detecting pose key points in each bounding box. In contrast,

Also, Novel architecture for Human Pose Estimation has also been proposed by researchers from New York University. This includes an efficient positions refinement

model that is trained to estimate the join offset location within a small region of the image. This refinement model is jointly trained in cascade with a state-of-the-art ConvNet model to achieve improved accuracy in human joint location estimation.

3. THEORETICAL BACKGROUND

3.1. Exercise Description

3.1.1. Bicep Curl

The Biceps Curl is highly popular weight exercise done for muscle of upper arm and lower arm. This exercise can be done by using dumbbells, barbells etc. Muscle of upper and lower arm is used whenever you pick up something, so picking up dumbbell and carrying this exercise puts significant pressure in muscle building it's strength. Single arm bicep curl is performed using dumbbell where muscle in upper arm bends and twists. So, only the rotation around the elbow should be done without significant movement of the body. Common mistakes when performing a bicep curl is moving your hand too fast, changing the position of elbow while carrying the exercise and moving your entire body.

Figure 3.1: Bicep curl

Basic Instructions to perform Biceps Curl are:

1. Standing with a dumbbell in each hand and elbows touching the side of the torso.
2. Holding the upper arm stationary, curl the right weight with the palm facing forward. Lift the dumbbell toward the shoulder until the bicep is fully contracted. Hold this position for a second and squeeze the bicep.
3. Lower the right dumbbell to the starting position and repeat the same movement with the left dumbbell.
4. Repeat for reps.

3.1.2. Front Raise

Front raise is exercise performed by the dumbbell which is mostly targeted for beginners. This is also used as therapy for those who are having shoulder problems. Two dumbbells is taken at both hands vertically which are parallel to the legs.Then the dumbbell is raised until hands are parallel to the floor and then is returned to initial position. Then these steps are performed repeatedly. This exercise strengthens the shoulder also strengthens the upper chest muscles to some extent. Mistakes while performing these exercise are moving your whole body, taking heavy weights and having wrong wrist positions.

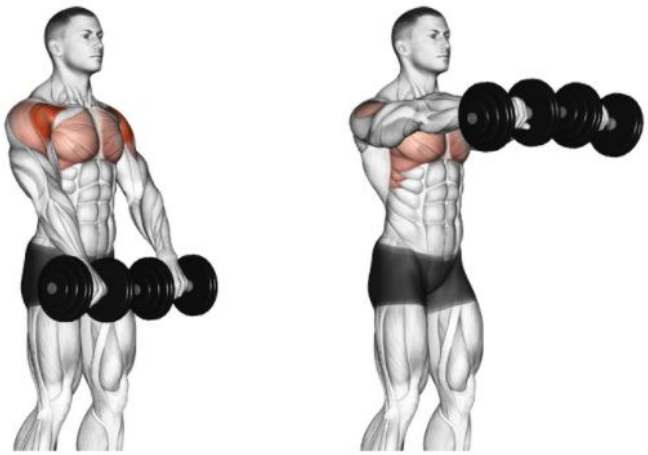


Figure 3.2: Front raise

Basic Instructions to perform Front Raise are:

1. Pick up the dumbbell with both hands and position them in front of your legs with your elbows straight.
2. Raise the dumbbells in front of you until your arms are at least the parallel to the floor or slightly beyond.
3. Lower and repeat for reps.

3.1.3. Shoulder Press

Shoulder press also known as overhead press is the weight lifting exercise done for the shoulder muscle. It can be performed both by sitting and standing. Exercise performed by sitting helps the back, while standing helps the muscle of shoulder. Two dumbbells are first taken in horizontal position and dumbbells are raised and brought back to initial position. Only shoulder should be moved while performing the exercise. Moving whole body, pressing too fast, lowering dumbbell to low are common mistakes while performing shoulder press.

Figure 3.3: Shoulder press

Basic Instructions to perform Shoulder Press are:

1. Grab the dumbbells and swing the dumbbells upwards to raise them to shoulder height.

2. Align the elbows with the dumbbells so that your elbows are directly underneath the weights.

3. With a braced core and your glutes tight, press the dumbbell upwards, then exhale.

4. Lower the dumbbells to starting position.

5. Repeat for reps.

3.2. Neural Networks

Neural networks are set of algorithm that models human brain.As human brain consists of several neurons interconnected to each other, neural network try to simulate the similar behaviour. It consists of various algorithms to recognize the pattern in data. When an input is passed through the network, various neurons are fired. These neurons are trained to recognize the pattern from training data. Unlike traditional system where an output of system is explicitly programmed by user to perform a specific function, neural network attempts to learn this function by minimizing the error from given labelled train data. They don't need to be programmed explicitly.The neural network are also used as bits and pieces for various complex machine learning algorithms.

Neural networks are used in various problems that we encounter in real world like speech recognition, spam filtering, image recognition, credit card fraud detection. Each neural networks are first supplied with training data. The network iteratively process training data and update weight and bias to adapt to the input data.This is also known as learning process.This process of learning is called learning by analogy. After the learning phase is complete, the network is supplied with data that need to be labelled. By using the knowledge of training data, the network tries to classify the unknown data. The accuracy of the output depends upon various parameters like input data-sets size and quality, complexity of network and so.

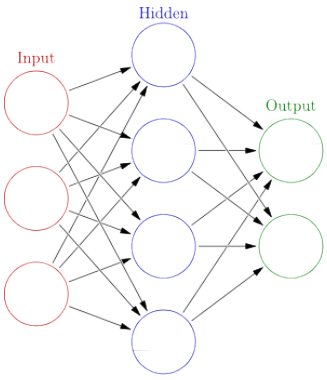


Figure 3.4: A simple neural network

The power of neural networks comes from the fact that they can work on different types of input like images, videos, files etc. Different variations of neural networks are present for processing different types of input data. Convolutional Neural Networks are designed to handle images and temporal data. Recurrent Neural Networks are suitable for text.

3.3. Convolutional Neural Networks

CNN attempts to represent and find the features by exploiting the spatial and temporal coherence by the use of relevance filter. The main goal is capture most features as possible by reducing the data required for representation.CNN attempts to reduce the image in smaller form that are easier to process without losing important features.This is important to make the learning algorithm scalable to large datasets. CNN consists of various layers. They are:

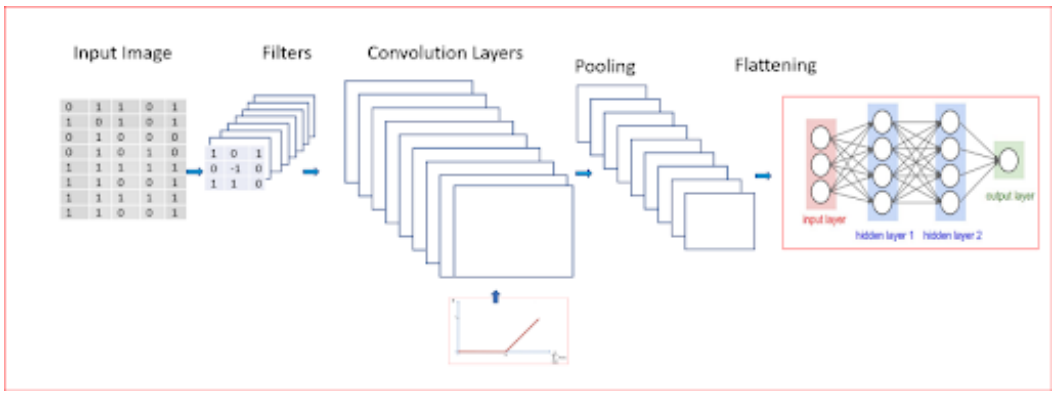


Figure 3.5: Convolutional neural network

Convolution Layer: Convolution layer receives input images and a number of kernels or filters. Kernels are also known as feature detectors or filters. They are convoluted with input images to produce the output for a certain pixel. Typically used kernel sizes are 3x3, 5x5 and 7x7. The kernel moves with a certain stride until it reaches the complete width and height. The stride of the kernel determines the size of the output image. Padding are introduced at the edge of image if kernel lies outside the boundary of image.

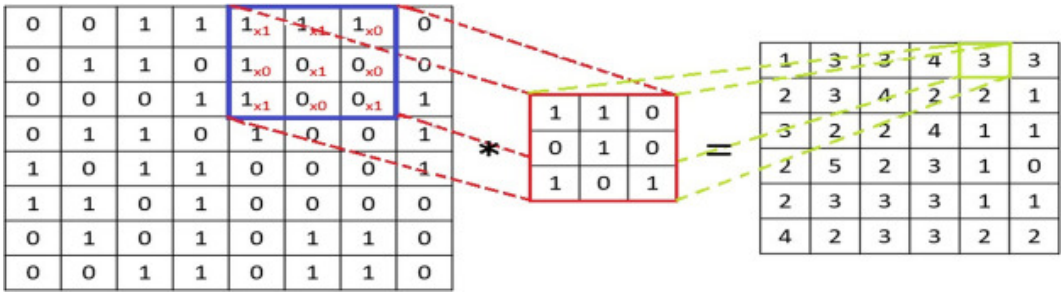


Figure 3.6: Convolution of image with filter to get feature map

Pooling Layer: The main role of pooling layer is reduce the size of convolved features. This is done to reduce the computation required to process the data. It is a process of dimensionality reduction that extracts the dominant data. Two common types of Pooling are: Max Pooling and Average Pooling. Max Pooling returns the maximum value whereas Average Pooling returns the average of all the value from the portion of image that are covered by the Kernel.

The convolution layer together with the pooling layer forms the ith layer of the network. Depending upon the complexity of the network, numbers of such layers are stacked on top of each other to capture low level details even further.

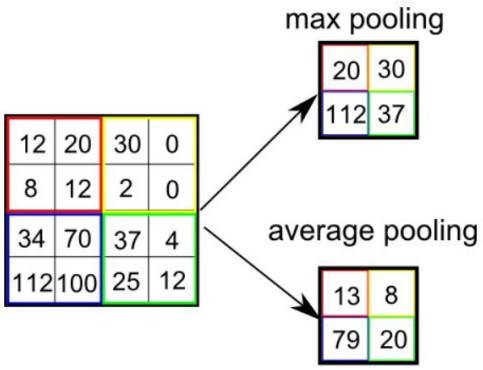


Figure 3.7: Pooling

After going through the above process, we create a model that is able to understand the features. Moving on, we flatten the final output and feed it to a regular Neural Network for classification purposes.

Classification-Fully Connected Layer: Fully-Connected layer is the process of learning a possibly non-linear function in that space. It is a cheap way of learning non-linear combination of high-level features as represented by the output of convolution layer. The output is converted into a column vector which is also called flatten. This enables us to convert the input image to Multi-Level perceptron. The flattened output is fed to the feed-forward network and backpropagation is applied in each iteration for training over a series of epochs. Then they are classified using Softmax Classification techniques.

3.4. Classification:

Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories a.k.a "sub-populations." With the help of these pre-categorized training datasets, classification in machine learning programs provides a wide range of algorithms to classify future datasets into respective and relevant Categories. Classification algorithms used in machine learning utilize input training data for the purpose of predicting the likelihood or probability that the data that follows will fall into one of the predetermined categories.

Classification can be classified as – single label classification and multi label classification.

3.4.1. Single Label Classification:

Single label classification, labels (category) are mutually exclusive and each instance is assigned to only one category. Formally, a single-label classifier is concerned with learning from a set of examples that are associated with a single label l from a set of disjoint labels L. The single-label classification problem can be further divided into two categories: Binary and multi-class classification.

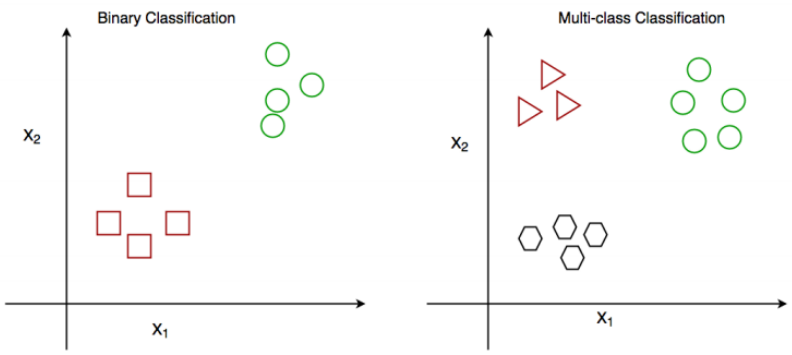


Figure 3.8: Single label classification

Binary Classification: Binary classification is the task of classifying the elements of a set into two groups. Examples include: Email spam detection (spam or not). Churn prediction (churn or not).

Multiclass Classification: Multiclass classification is the problem of classifying instances into one of three or more classes.While many classification algorithms such as: neural networks, decision trees, k-nearest neighbors, naive Bayes, support vector machines and extreme learning machines permit the use of more than two classes, some are by nature binary algorithms; these can, however, be turned into multi-class classifiers by a variety of strategies. We can view a multiclass classifier in terms of many independent binary classifiers. In this process we transform a multi class classification problem into binary classifications.

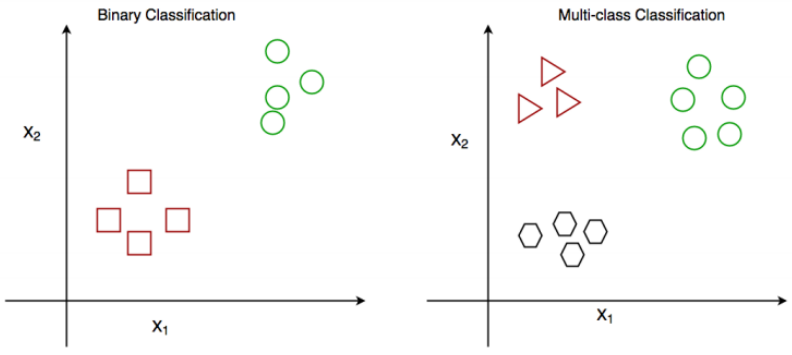


Figure 3.9: Multiclass classification

3.4.2. Multi Label Classification:

In multi-label classification multiple labels may be assigned to each instance. It is a

generalization of multiclass classification where there is no constraint on how many of the classes the instance can be assigned to.

3.5. Dynamic Time Warping

Dynamic Time Warping is used to compare the similarity of two arrays or time series of different lengths or to calculate the distance between them. For example, even if one person was walking faster than the other or if there were accelerations and declarations during the observation, DTW could detect walking similarities. DTW has been used to analyze video, audio, and graphics data in temporal sequences — in fact, any data that can be converted into a linear sequence can be analyzed with DTW. Automatic speech recognition is a well-known application for dealing with different speaking speeds. Speaker recognition and online signature recognition are two other applications. It can also be used in applications that require partial shape matching.

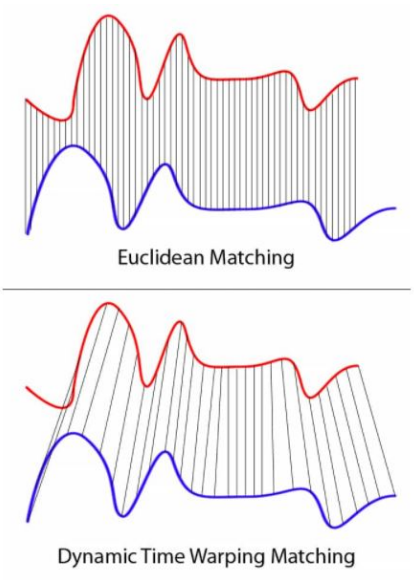


Figure 3.10: Euclidean distance and DTW between two time series

When two time series (the base time series and the new time series) can be mapped with function f(x) according to the following rules to match the magnitudes using an optimal (warping) path, they are considered similar.

1. $f(x_i)$ maps to $f(x_j)$ when $i <= j$

2. $f(x_i)$ maps to $f(x_j)$ only when $(j - i)$ is within fixed range

DTW distance between two time series is calculated using following formula

$$DTW[i, j] := cost + minimum(DTW[i-1, j], DTW[i, j-1], DTW[i-1, j-1]) \quad (1)$$

Which is saying that the cost of between two arrays with length i and j equals the distance between the tails + the minimum of cost in arrays with length i-1 j, i j-1, and i-1 j-1.

3.6. Pose Estimation

The problem of locating human joints (also known as keypoints - elbows, wrists, etc.) in images or videos is known as human pose estimation. It's also known as the search for a specific pose among all articulated poses in space. Human Pose Estimation has a wide range of applications, including action recognition, animation, gaming, and more. The major of parameters in pose estimation are:

3.6.1. Number of People Being Tracked

The number of people being tracked is one of the parameters in pose estimation. Pose estimation can be divided into two types based on this parameter: single-person and multi-person pose estimation. Only one person is present in the frame when single-person pose estimation (SPPE) is used. Multi-person pose estimation (MPPE), on the other hand, must deal with the additional issue of inter-person occlusion.When an object you are tracking is obscured (occluded) by another object, it is called occlusion. Initial approaches to pose estimation focused primarily on SPPE; however, with the availability of large multi-person datasets, the MPPE problem has recently gotten more attention.

3.6.2. Images/Videos in Pose Detection

Pose detection can be done with either an image or a video. A video is nothing more than a sequence of images in which every two frames share a large portion of the information in them (which is the basis of most of the video compression techniques). Pose estimation can take advantage of this temporal (time-based) dependence in videos. For a video, a series of poses need to be produced for the input video sequence. It is expected that the estimated poses should ideally be consistent across successive frames

of video, and the algorithm needs to be computationally efficient to handle large number of frames. The problem of occlusion might be easier to solve for a video due to the availability of past or future frames where the body part is not occluded. If temporal features are not a part of the pipeline, it is possible to apply static pose estimation for each frame in a video. However, the results are generally not as good as desired due to jitters and inconsistency problems. Hence, image or video representations can be used. For animation purposes, video is preferred as it can provide real time and continuous representation of a movement to be represented.

3.6.3. Input Modality

The various types of inputs available are referred to as modality. Different parameters may be present in the input video/image. The following are some examples of input modes:

3.6.4. RGB image/video

The most common type of input for Pose Estimation is images that we see around us on a daily basis. Models that work with RGB-only input have a significant advantage over others in terms of input source mobility. This is because common cameras (which capture RGB images) are widely available, making them models that can be used on a wide range of devices.

3.6.5. Depth information included image/video

The value of a pixel in a Depth image corresponds to the distance from the camera as measured by time-of-flight. The introduction and popularity of low-cost devices such as the Microsoft Kinect have made obtaining Depth data easier. Depth images can be used in conjunction with RGB images to create more complex and accurate Computer Vision models, whereas Depth-only models are commonly used when privacy is an issue.

3.6.6. Infra-red (IR) image/video

The amount of infrared light reflected back to the camera determines the value of a pixel in an IR image. When compared to RGB and Depth images, there is less experimentation in Computer Vision based on IR images. While recording, Microsoft

Kinect also provides an infrared image. However, there are currently no datasets that include IR images.

3.6.7. 2D vs 3D Pose Estimation

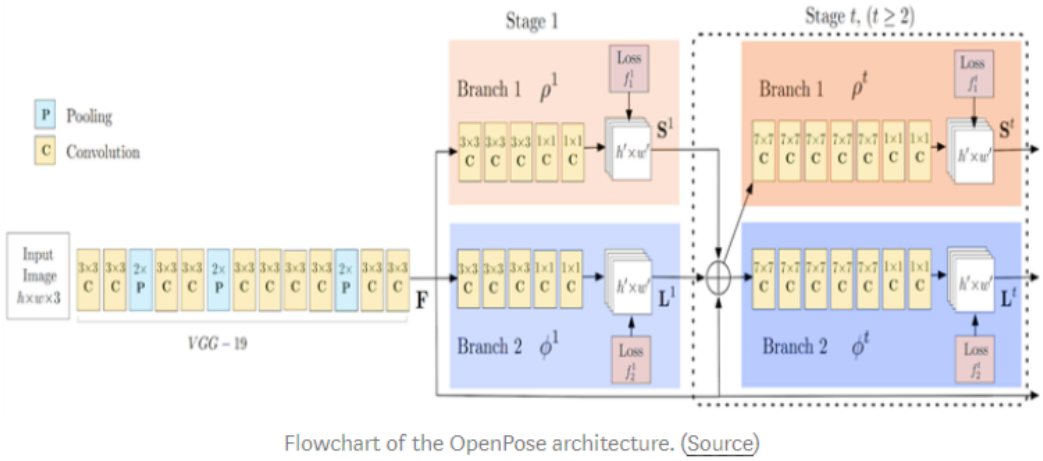
Pose estimation's output can be expressed in two or three dimensions. The Pose Estimation problem can be divided into two categories based on the output dimension requirement: 2D Pose Estimation and 3D Pose Estimation. In 2D Pose Estimation, the location of body joints I in the image is predicted (in terms of pixel values).3D Pose Estimation, on the other hand, forecasts a three-dimensional spatial arrangement of all body joints as its final output. It also includes the prediction of the z coordinate. The majority of 3D Pose Estimation models predict 2D Pose first, then attempt to lift it to 3D Pose. However, there are some end-to-end 3D Pose Estimation techniques that can predict 3D Pose directly.

3.7. Methods of Pose Estimation

3.7.1. Openpose

It is one of the most popular and widely used bottom-up approaches for multi-person human pose estimation is OpenPose. OpenPose, like many other bottom-up approaches, begins by detecting parts (keypoints) that belong to each person in the image, then assigning parts to distinct individuals.

The OpenPose network extracts features from an image using the first few layers. Following that, the features are fed into two convolutional layer branches that run parallel to one another. The first branch generates 18 confidence maps, each representing a different aspect of the human pose skeleton. The second branch predicts 38 Part Affinity Fields (PAFs), which reflect the degree of part association. The forecasts of each branch are fine-tuned in stages. Component confidence maps are used to construct bipartite graphs between pairs of components. PAF values are used to prune weaker ties in bipartite graphs.



Flowchart of the OpenPose architecture. (Source)

Figure 3.11: Open pose architecture

The system takes, as input, a color image of size w × h and produces the 2D locations of anatomical keypoints for each person in the image . First, a feedforward network predicts a set of 2D confidence maps S of body part locations and a set of 2D vector fields L of part affinity fields (PAFs), which encode the degree of association between parts. The set S = (S1, S2, ..., SJ) has J confidence maps, one per part, where Sj Rw×h ,j 1 . . . J. The set L = (L1,L2, ...,LC) has C vector fields, one per limb, where Lc Rw×h×2, c 1 . . . C. Each image location in Lc encodes a 2D vector. Finally, the confidence maps and the PAFs are parsed by greedy inference to output the 2D keypoints for all people in the image.

Confidence maps: Each confidence map is a 2D representation of the belief that a particular body part can be located in any given pixel. Ideally, if a single person appears in the image, a single peak should exist in each confidence map if the corresponding part is visible; if multiple people are in the image, there should be a peak corresponding to each visible part j for each person k. So, the number of maps is the same as the total number of the body parts. We take the maximum of the confidence maps instead of the average so that the precision of nearby peaks remains distinct, as illustrated in the right figure. At test time, we predict confidence maps, and obtain body part candidates by performing non-maximum suppression.

$$S = (S_1, S_2.., S_J) \ , \ S_J \epsilon R^{w*h}, j \epsilon 1...j, \tag{2}$$

where j is the total number of body parts

Part Affinity Fields (PAFs): PAFs are a series of 2D vector fields that encode the position and orientation of limbs across the image domain. For each pair of body part detections, a confidence measure of the relationship, i.e., that they belong to the same individual, is needed. Detecting an additional midpoint between each pair of parts on a limb and checking for its occurrence between candidate component detections is one way to quantify the association. When people congregate, as they often do, these midpoints are more likely to accept false associations.

$$L = (L_1, L_2.., L_C) \ , \ L_C \epsilon R^{w*h*2}, c \epsilon 1...c, \tag{3}$$

where C is the total number of limbs

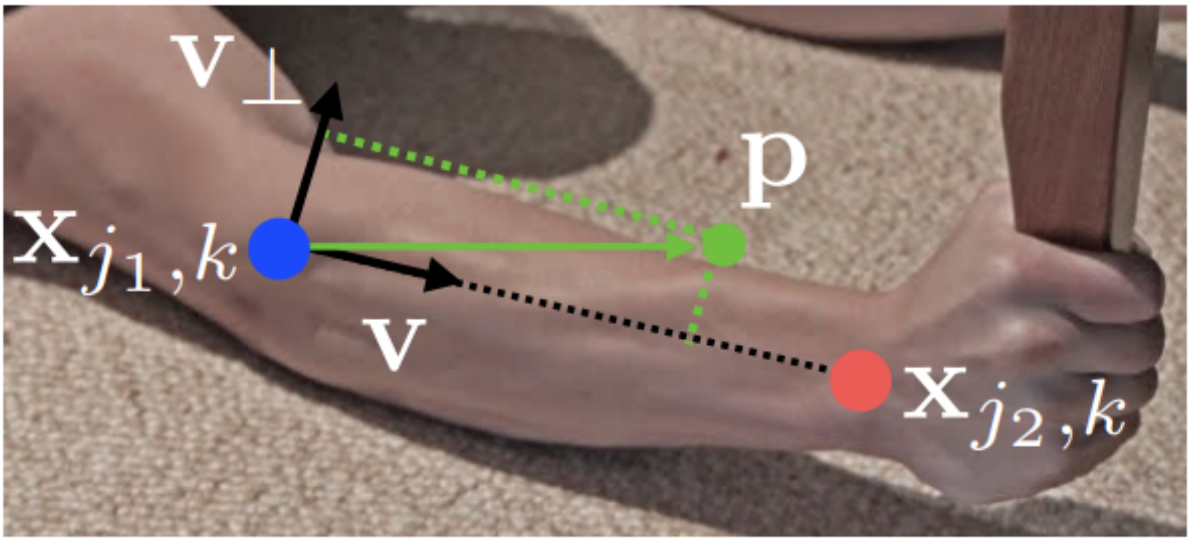


Figure 3.12: Part affinity field Visualization

Bipartite Matching: Determining Z is a K-dimensional matching problem when it comes to finding the full body pose of multiple people. This problem is NP-Hard, and there are numerous relaxations. In this paper, we add two domain-specific relaxations to the optimization. Relaxation 1: To make a spanning tree skeleton, choose the smallest number of edges possible. Relaxation 2: Break the matching problem down even more

into a set of bipartite matching subproblems. Determine the matching in adjacent tree nodes independently.

Results: For comparison on the MPII dataset, the authors have used the toolkit to measure mean Average Precision (mAP) of all body parts following the "PCKh" metric from. For the 288 images subset, this method outperforms previous state-of-the-art bottom-up methods by 8.5% mAP. Remarkably, the inference time is 6 orders of magnitude less. For the entire MPII testing set, this method without scale search already outperforms previous stateof-the-art methods by a large margin, i.e., 13% absolute increase on mAP. Using a 3 scale search (×0.7, ×1 and ×1.3) further increases the performance to 75.6% mAP. The mAP comparison with previous bottom-up approaches indicate the effectiveness of the novel feature representation, PAFs, to associate body parts. Based on the tree structure, the greedy parsing method achieves better accuracy than a graphcut optimization formula based on a fully connected graph structure

Method	Hea	Sho	Elb	Wri	Hip	Kne	Ank	**mAP**	s/image
Subset of 288 images as in [1]									
Deepcut [1]	73.4	71.8	57.9	39.9	56.7	44.0	32.0	54.1	57995
Iqbal et al. [41]	70.0	65.2	56.4	46.1	52.7	47.9	44.5	54.7	10
DeeperCut [2]	87.9	84.0	71.9	63.9	68.8	63.8	58.1	71.2	230
Newell et al. [48]	91.5	87.2	75.9	65.4	72.2	67.0	62.1	74.5	-
ArtTrack [47]	92.2	**91.3**	80.8	71.4	**79.1**	72.6	67.8	**79.3**	**0.005**
Fang et al. [6]	89.3	88.1	80.7	75.5	73.7	**76.7**	**70.0**	79.1	-
Ours	**92.9**	**91.3**	**82.3**	72.6	76.0	70.9	66.8	79.0	**0.005**
Full testing set									
DeeperCut [2]	78.4	72.5	60.2	51.0	57.2	52.0	45.4	59.5	485
Iqbal et al. [41]	58.4	53.9	44.5	35.0	42.2	36.7	31.1	43.1	10
Levinko et al. [71]	89.8	85.2	71.8	59.6	71.1	63.0	53.5	70.6	-
ArtTrack [47]	88.8	87.0	75.9	64.9	74.2	68.8	60.5	74.3	**0.005**
Fang et al. [6]	88.4	86.5	78.6	**70.4**	74.4	**73.0**	**65.8**	76.7	-
Newell et al. [48]	**92.1**	89.3	78.9	69.8	76.2	71.6	64.7	77.5	-
Fieraru et al. [72]	91.8	**89.5**	**80.4**	69.6	**77.3**	71.7	65.5	**78.0**	-
Ours (one scale)	89.0	84.9	74.9	64.2	71.0	65.6	58.1	72.5	0.005
Ours	91.2	87.6	77.7	66.8	75.4	68.9	61.7	75.6	**0.005**

Figure 3.13: Results of openpose

3.7.2. Blazepose

The CV4ARVR workshop at CVPR 2020 featured Blazepose, one of the most recent approaches to human body pose perception. They used a tool that is slightly different from traditional approaches and yields faster results. Human pose tracking is performed by employing machine learning (ML) to infer 33, 2D landmarks of a body from a

single frame. In contrast to current pose models based on the standard COCO topology, BlazePose is uniquely suited for health applications since it precisely localizes more keypoints. Blazepose provides real-time output on mobile phones with CPU inference, while current state-of-the-art approaches require more computation. For more precise modeling, it can also use GPU inference.

It is designed to be a lightweight convolutional neural network architecture for human pose estimation that is suited for real-time inference on mobile devices. The network produces 33 body keypoints for a single person and runs at over 30 frames per second on a Pixel 2 phone. This makes it particularly suited to real-time use cases like fitness tracking and sign language recognition. It is a superset of COCO.

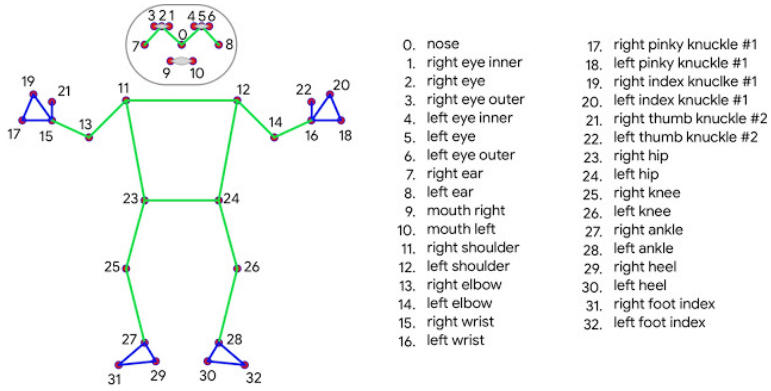
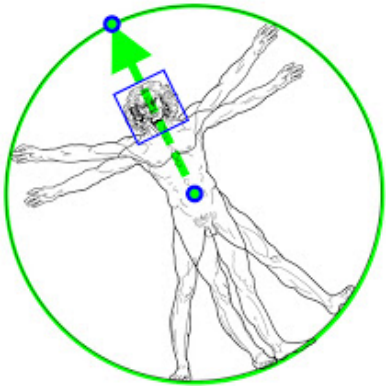


Figure 3.14: Keypoints given by blazePose

Architecture and pipeline: For pose estimation, it uses the two step detector-tracker method. Using a detector, this pipeline first locates the pose region-of-interest (ROI) inside a frame. ROI is the approximation of where the joints could be located in a frame. Then secondly, the tracker subsequently predicts all 33 pose keypoints from the detected ROI. For videos, ROI is derived from the first frame, so a detector is not required for every frame.

Since the pose detection and tracking is aimed to be very fast, the detection should be performed in milliseconds time for a frame. To accomplish this, they have approached with the assumption that the strongest signal to the neural network about the position of the torso is the person's face (due to its high-contrast features and comparably small variations in appearance). Therefore, pose detector goes with the assumption that the head should be visible for single-person use case.

Detection of person is done by using a proxy. For detection, it uses a face detector as the proxy (Blaze face). Blazeface only detects the location of face in a very fast time. This can be used to draw two points (that describe the human body center, rotation and scale as a circle) as shown in the figure and then draw out a region of interest using it as a reference.



The following items are predicted:

1. the midpoint of a person's hips
2. the radius of a circle circumscribing the whole person
3. and the incline angle of the line connecting the shoulder and hip midpoints.

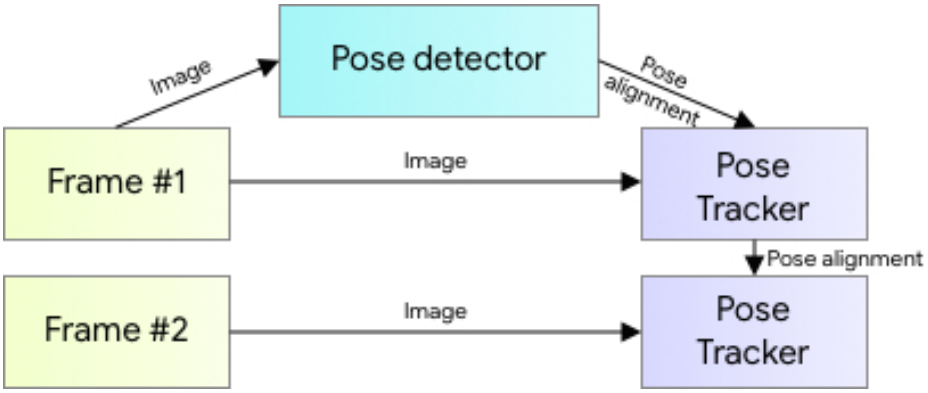


Figure 3.15: Diagram of pose tracker

It does not use compute-intensive heatmap prediction, instead it uses a regression approach that is supervised by a combined heat map/offset prediction of all keypoints.

Combined heatmap, offset, and regression approach is used as shown in the figure. This approach is partially inspired by Stacked Hourglass approach,but the difference is that there is stacking of a tiny encoder-decoder heatmap-based network and a subsequent regression encoder network. Skip-connections is utilized between all the stages of the network to achieve a balance between high and low-level features. However, the gradients from the regression encoder are not propagated back to the heatmap trained features. This not only improves the heatmap predictions, but also substantially increases the coordinate regression accuracy.

Dataset in this model: Compared to the majority of existing pose estimation solutions that detect keypoints using heatmaps, they have instead restricted the dataset to those cases where either the whole person is visible, or where hips and shoulders keypoints can be confidently annotated. To ensure the model supports heavy occlusions that are not present in the dataset, occlusion-simulating augmentation is used. Training dataset consists of 60K images with a single or few people in the scene in common poses and 25K images with a single person in the scene performing fitness exercises. All of the images were annotated by humans.

The pose estimation component of predicts the location of all 33 person keypoints with three degrees of freedom for each joint:

1. x coordinate
2. y coordinate
3. visibility
4. Two virtual alignment keypoints described above

Accuracy of this method:

	Mean 2D Euclidean error, normalized by torso size	*PCK@0.2*
Heatmaps	16.2	83.6
Regression without Heatmaps loss	15.9	79.9
Regression with heatmap regularization	**14.4**	**84.1**

Figure 3.16: Accuracy of blaze pose

Metric: Percent of Correct Points with 20% tolerance (PCK@0.2) : The point is detected correctly if the 2D Euclidean error is smaller than 20% of the corresponding person's torso size. In comparison to human performance, annotators were asked to annotate several samples redundantly and obtained an average PCK@0.2 of 97.2.

3.8. Evaluation metrics in pose estimation

Since this is a regression problem, the most commonly used metric is the loss function - Mean Squared Error, MSE(Least Squares Loss). The model will attempt to regress to the correct coordinates, i.e. incrementally move to the ground truth coordinates. A Mean Squared Error loss function is used to train the model to output continuous coordinates. Some major evaluation metrics specific to pose estimation are:

3.8.1. Percentage of Correct Parts - PCP

If the distance between the two predicted joint locations and the true limb joint locations is at most half of the limb length (PCP at 0.5), a limb is considered detected and a correct part. The better the model, the higher the PCP. The disadvantage is that it can penalize those with shorter limbs.
Calculation: For a specific part, PCP = (No. of correct parts for entire dataset) / (No. of total parts for entire dataset) Take a dataset with 10 images and 1 pose per image. Each pose has 8 parts - (upper arm, lower arm, upper leg, lower leg) x2 No of upper arms = 10 * 2 = 20 No of lower arms = 20 No of lower legs = No of upper legs = 20 If upper arm is detected correct for 17 out of the 20 upper arms i.e 17 (10 right arms and 7 left) $\rightarrow$ PCP = 17/20 = 85

3.8.2. Percentage of Correct Key-points - PCK

A detected joint is considered correct if the distance between the predicted and the true joint is within a certain threshold. The threshold can either be:
PCKh@0.5 is when the threshold = 50% of the head bone link
PCK@0.2 == Distance between predicted and true joint < 0.2 * torso diameter Sometimes 150 mm is taken as the threshold. Alleviates the shorter limb problem since shorter limbs have smaller torsos and head bone links. PCK is used for 2D and 3D (PCK3D). Again, the higher the better.

3.8.3. Percentage of Detected Joints - PDJ

A detected joint is considered correct if the distance between the predicted and the true joint is within a certain fraction of the torso diameter. PDJ@0.2 = distance between predicted and true joint < 0.2 * torso diameter.

3.8.4. Object Keypoint Similarity (OKS) based map

It is commonly used in the COCO keypoints challenge.

$$\frac{\Sigma_i exp(-d_i^2/2s^2k_i^2)\delta(v_i > 0)}{\Sigma_i \delta(v_i > 0)} \tag{4}$$

where,

d_i *is the Euclidean distance between the detected keypoint and the corresponding ground truth*

v_i *is the visibility flag of the ground truth,*

s *is the object scale, and*

k *is a per − keypoint constant that controls falloff.*

It is calculated from the distance between predicted points and ground truth points normalized by the scale of the person.

3.9. Techstack

3.9.1. Tensorflow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.Tensorflow provides stable Python, C and Javascript API across all platform.TensorFlow computations are expressed as stateful dataflow graphs.Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

3.9.2. Numpy

Numpy is a library for Python programming language that supports large multidimensional arrays and matrices along with various mathematical operations that can be performed on them. It is an open source software. They provide the primitives to represent all the pose frame data in the project. Numpy is extensively used to perform various trigonometric and vector operations on the pose data in the project. Numpy makes use of parallelism through Single Instruction Multiple Data (SIMD) which helps to speed up addition, comparison and slicing of document matrices.

3.9.3. Matplotlib

Matplotlib is a python library which is used for data visualization. It has been one of the main tools for visualizing the data in the project. Matplotlib provides various plots like scatter plot, histogram, bar graph, pie chart etc to visualize various types of data. It is open source and can be used freely.

3.9.4. OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being a BSD-licensed product,OpenCV makes it easy for businesses to utilize and modify the code.The library has more than 2500 optimized algorithms, which includes a compre-hensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects,classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images to-gether to produce a high resolution image of an entire scene, find similar images from animage database, remove red eyes from images taken using flash, follow eye movements,recognize scenery and establish markers to overlay it with augmented reality, etc.

3.9.5. Pandas

Pandas is a python library for data manipulation and analysis.Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features. Pandas was mainly used in the project for data reprocessing.

3.9.6. Flask

Flask is a lightweight WSGI web application framework written in python.t is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.It provides services like session based authentication, basic http authentication, role based access, password hashing, token authentication, login tracking etc. It also provides basic click commands for managing users and roles. All these features are made possible by integrating various flask libraries and extensions like Flask-Login, Flask-Mail, passlib etc.

3.9.7. CSS and Bootstrap

CSS is used to style HTML elements. It provides an intuitive and easy way to style different HTML elements. It uses simple syntax and number of English keywords to specify the number of various style properties. It allows separation of presentation and content which improves content accessibility, provides more flexibility and control in specification of presentation characteristics. Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

4. METHODOLOGY

4.1. System Block Diagram

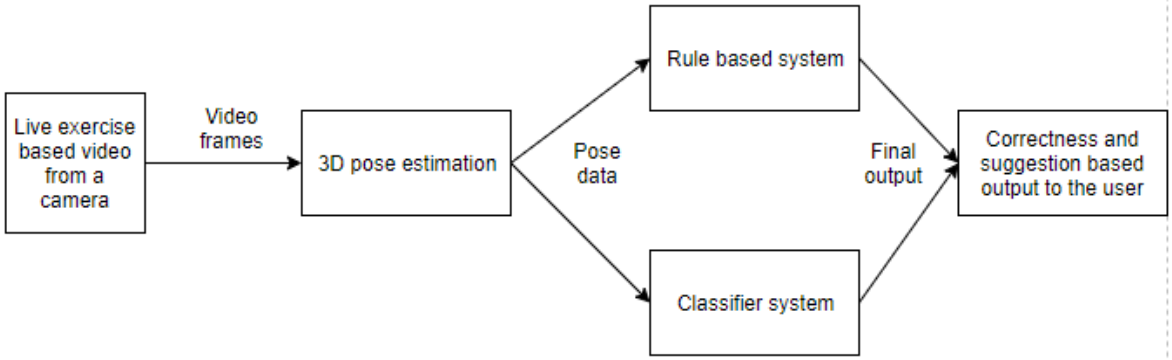


Figure 4.1: Block diagram of system

Our system consists of a web application which opens a camera attached to the computer, i.e. an external webcam from a laptop computer. The camera will be used to capture a normal RGB format video whose frames are fed into the 3D pose estimation model. The video will be such that the person performing an exercise routine is clearly visible. Pose estimation is performed to get the detailed pose data and this data is then fed to a rule based system as well as a classification and comparison based system to analyze the exercise routine. Four exercise types are supported: bicep curl (side view camera orientation), bicep curl (front view camera orientation), push-ups and shoulder press. If the exercise is performed correctly, the repetition count of the exercise is automatically recorded and displayed in the user interface of our application. If the person does not perform the exercises according to the rules, then feedback is generated to notify the user and display the appropriate steps for correction. All the correct and incorrect repetitions of the exercise are tracked along with the feedback for incorrect sequences. The progress of the individual exercise is also stored in a database for a complete log of history. The

replays of incorrect exercises can also be viewed from the user interface.
Our project consists of two major components - pose estimation (real time) in a video and analysis and feedback generation of gym/exercise posture.

4.2. Pose estimation steps

The pose estimation part can be described as: Given a video of a person who is performing a given exercise, a 3D pose estimation model is built with the task of producing a 3D pose that matches the spatial position of the depicted person. The following steps are followed:

4.2.1. Available datasets for pose estimation

Available datasets for Pose Estimation are
MPII: The MPII human pose dataset is a multi-person 2D Pose Estimation dataset comprising of nearly 500 different human activities, collected from Youtube videos. It is a state of the art benchmark for evaluation of articulated human pose estimation. The images were systematically collected using a collection of everyday human activities.
COCO : The COCO keypoints dataset is a multi-person 2D Pose Estimation dataset with images collected from Flickr. COCO is the largest 2D Pose Estimation dataset, to date, and is considering a benchmark for testing 2D Pose Estimation algorithms.
HumanEva : HumanEva is a single-person 3D Pose Estimation dataset, containing video sequences recorded using multiple RGB and grayscale cameras. Ground truth 3D poses are captured using marker-based motion capture (mocap) cameras. HumanEva was the first 3D Pose Estimation dataset of substantial size.
Human3.6M : Human3.6M is a single-person 2D/3D Pose Estimation dataset, containing video sequences in which 11 actors are performing 15 different possible activities were recorded using RGB and time-of-flight (depth) cameras. 3D poses are obtained using 10 mocap cameras. Human3.6M is the biggest real 3D Pose Estimation dataset, to date.

4.2.2. Analysis of MPII Dataset

Annotations of this dataset are provided in a matlab structure which has to be parsed. The main joints which are represented in the dataset are given by joint ids:

Joint id - Joint

0 - r ankle,

1 - r knee,

2 - r hip,

3 - l hip,

4 - l knee,

5 - l ankle,

6 - pelvis,

7 - thorax,

8 - upper neck,

9 - head top,

10 - r wrist,

11 - r elbow,

12 - r shoulder,

13 - l shoulder,

14 - l elbow,

15 - l wrist

4.2.3. Analysis of COCO Dataset

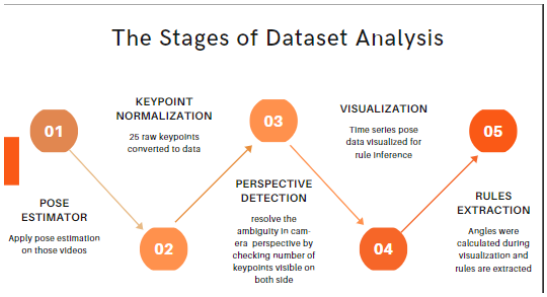


Figure 4.2: Dataset analysis steps

COCO dataset is one of the popular available datasets which is used in the field of pose estimation. It consists of 250,000 people instances which has been divided into 80 unique categories. For pose estimation task of a particular type, all the data may not be relevant as it is more desirable to get data which is exactly applicable for a field.

However, this is a more generalized dataset which may consist of random or unwanted type of images. The dataset is composed of image files and annotation files. The annotation file is a JSON containing all metadata about a single person (or some other categories). Here we can parse the data to find the position and size of bounding boxes, area, polygons for segmentation, keypoints, file names of source images, etc.

Firstly, the COCO object was loaded which acts a wrapper for json annotation data. Then the metadata for each image was loaded. A dictionary containing general information like image width, height, name, licensing, etc and annotations were also found and then loaded. In the obtained list, each element (dictionary) represented information about a single person.

Converting COCO metadata to Pandas dataframe: To convert the COCO metadata into a pandas dataframe we used a number of tools for data science, like matplotlib, sklearn-kit, and pandas api. Filtering, visualization, and manipulation of data becomes much easier. These libraries are commonly used in python and mainly for data visualization. In the next step, we merged both tables (left join operation) and glue the training set to the validation set. Also, we adedd a new column source with the value 0 for the training set and 1 for the validation set. Such information is necessary because we need to know in which folder we should search for an image. Finally, we obtain a pandas dataframe. This dataframe represents the entire dataset.

Number of people in images: Multiple people as well as single person are present in the dataset. The distribution of single person and multi person was also calculated for reference.

Figure 4.3: COCO dataset test images

From observing the test images, most of the COCO images contained a single person. But, there was a significant number of images with 13 people.

There is even an image with 19 annotations marked as non-crowd.

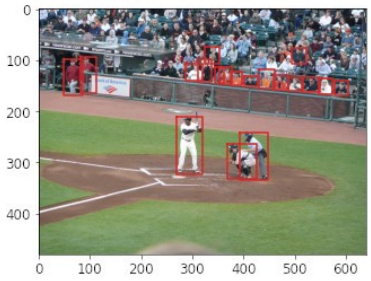


Figure 4.4: COCO dataset multiple people

Adding Extra Columns: Once we have converted our COCO into pandas dataframes, we can very easily add extra columns, calculated from the existing ones. Extracting all keypoint coordinates into different columns was done. We also provided a column with scale factors. Knowledge about the scales of a bounding box for an individual, in particular, can be extremely useful. For example, we may want to discard all too-small occurrences or perform up-scaling. We used sklearn for this purpose. In general, sklearn transformers are powerful tools used for cleaning, reducing, expanding, and generating feature representations in data science models.

Photos of different widths and heights can be found in the COCO dataset. We normalized the x,y coordinates of a nose in each picture so that we could draw dots in the output map to reflect noses. We calculated the average width and height of all pictures. We can use any value since the scale factor will be determined solely by it. We can then examine some videos. For example, we'd like to examine a few images with head positions that are incredibly close to the image's bottom edge. To do so, we use the column normalized nose (y) to filter the dataframe.

Number of keypoints: Additional useful knowledge is the number of bounding boxes with a particular number of keypoints. The iscrowd flag in bounding boxes determines if the content should be viewed as a crowd (no keypoints) or as an individual (should have keypoints). In general, iscrowd is used to describe a bounding box containing several small instances of people, such as a tennis audience.

Scales: This is another important metric as training pose estimation deep neural network models are sensitive to scale variation of people in samples.

Stratified sampling of COCO dataset: We want to make sure that each subset includes an equal proportion of unique data groups as we split the entire dataset into training sets, validation sets and test sets. As a result, stratified sampling is important.

Background removal: It is used to separate the background from the image and suppress any noise. The aim of foreground detection is to detect changes in image sequences, which is one of the most important tasks in the field of computer vision and image processing. Any technique that allows the foreground of an image to be removed for further processing is known as background subtraction (object recognition etc.).

Bounding box creation: It is used to isolate the image from the background and reduce noise. Background subtraction is a procedure that enables the foreground of an image to be removed for further processing (object recognition etc.).

Camera calibration and image registration: If multiple cameras' inputs are used, image registration is required. Camera calibration assists in the translation of recorded ground truth into standard world coordinates in 3D Human Pose Estimation.

4.2.4. Pose estimation models

We used pretrained models as well as trained some models ourselves. The main pose estimation model that we have used is the open-source Openpose. They have released these models and an API implementation in the form of Python code, C++ implementation and Unity Plugin. These resources can be downloaded from OpenPose repository.

We also trained our own model for pushup keypoint detection and normal body part pose estimation using blazepose.This was not used in the main part of our application as openpose offered better and more accurate realtime implementation in the context of normal computers, i.e. blazepose was more suited to mobile devices.

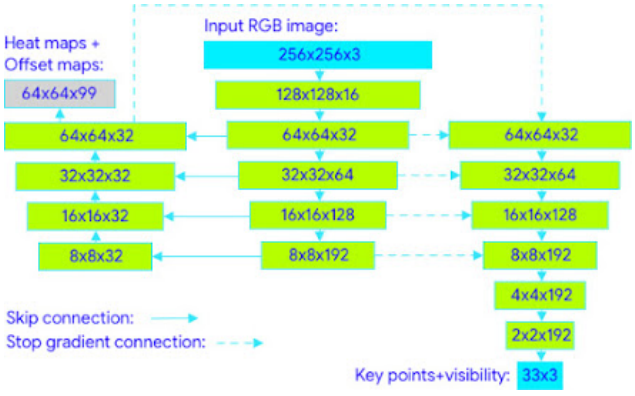


Figure 4.5: Blazepose model

4.3. Video Capture and Rules based Classification

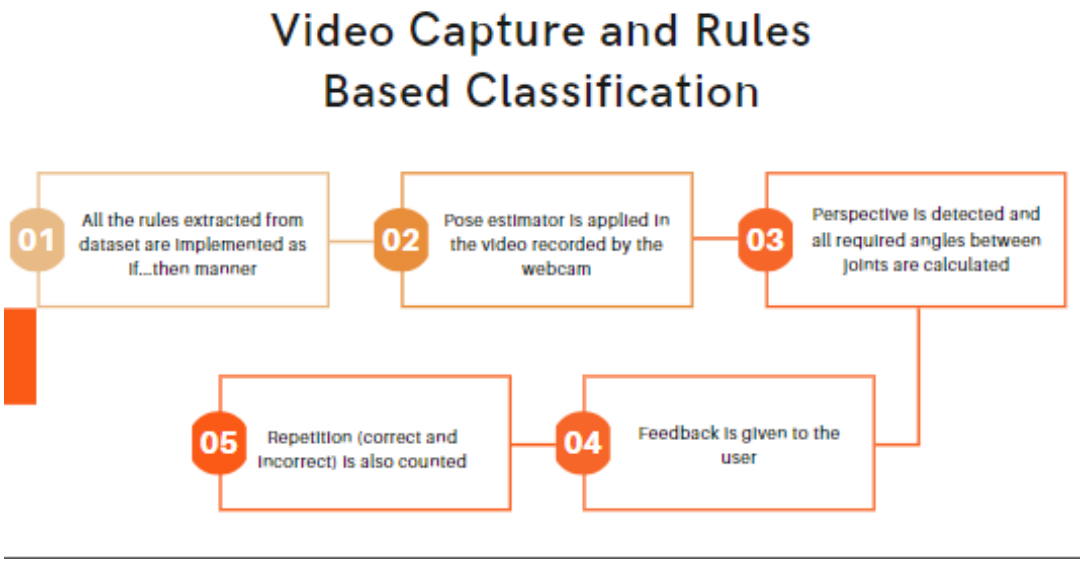


Figure 4.6: Classification diagram

The frames of a live video are sent for pose estimation followed by our previous analysis of the required angles and joint positions. After one repetition of an exercise is performed, the frames which have elapsed so far, are sent to our feedback generator to output correctness and feedback.

4.3.1. Video Recording

There are two options in the system, either the user records the video themselves performing exercise or they can also choose to use the system while performing exercise to get live feedback. In first option, the user records themselves performing a selected exercise. The video is recorded from a particular perspective, either facing the camera or side to the camera that allows the exercises to be seen. However, there are no requirements to the position and distance for the video. For second option, while performing the real time exercise, the user opens the webcam and the webcam starts to record the video frame by frame and sends it to the pose estimator which is running in the backend.

Since our system supports all common video formats, any software can be used to record video.

4.3.2. Pose Estimator

For Pose estimation, we use deep convolutional neural networks (CNNs) to label RGB images. We first used the pose estimation model which is recently proposed by google called Blazepose model. Blazepose model gave good results on pushup dataset but the accuracy of it isn't that good for real time for other exercises and also isn't compatible for other libraries, we choose to use pre trained model, OpenPose, for pose detection. OpenPose introduces a novel approach to pose estimation using part affinity fields, which are vectors that encode the position and orientation of limbs. OpenPose is both accurate and efficient, while also scalable to multiple people without scaling up the run-time.
OpenPose output consists of lists containing the coordinate predictions of all keypoint locations, and their corresponding prediction confidence. We consider the predictions of 18 key points of the pose, which include the nose, neck, shoulders, elbows, wrists, hips, knees, and ankles.

4.3.3. Parsing and Mapping

The obtained keypoints have to be parsed and stored in readable format for future tasks. The keypoints are obtained as:

0. "Nose",
1. "Neck",
2. "RShoulder",
3. "RElbow",
4. "RWrist",
5. "LShoulder",
6. "LElbow",
7. "LWrist",
8. "MidHip",
9. "RHip",
10. "RKnee",
11. "RAnkle",
12. "LHip",
13. "LKnee",
14. "LAnkle",
15. "REye",
16. "LEye",
17. "REar",
18. "LEar",
19. "LBigToe",
20. "LSmallToe",
21. "LHeel",
22. "RBigToe",
23. "RSmallToe",
24. "RHeel",
25. "Background"

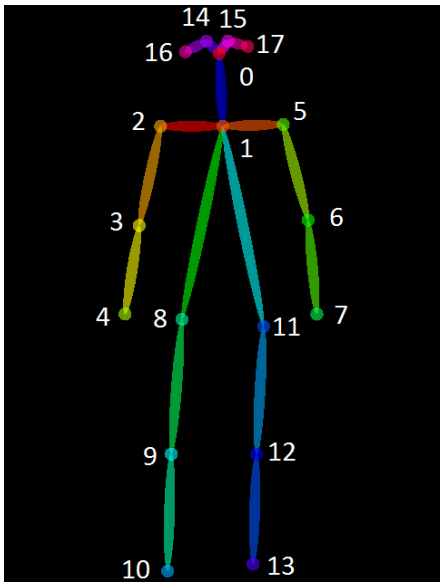


Figure 4.7: Pose output format (BODY25)

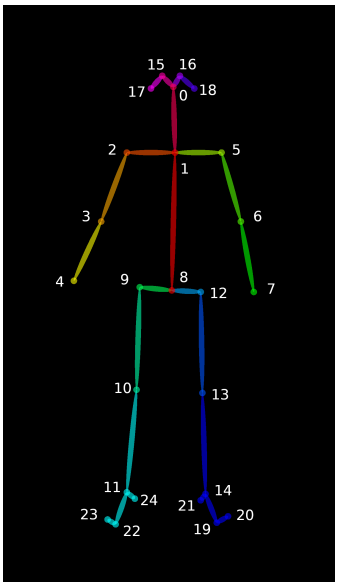


Figure 4.8: Pose output format (COCO)

They are stored in our custom data structure:

class PoseData:

nose: Joint

neck: Joint

rshoulder: Joint

relbow: Joint

rwrist: Joint

lshoulder: Joint

lelbow: Joint

lwrist: Joint

rhip: Joint

rknee: Joint

rankle: Joint

lhip: Joint

lknee: Joint

lankle: Joint

reye: Joint

leye: Joint

rear: Joint

lear: Joint

mhip: Joint

Each joint is of the following structure:

class Joint:

x: float

y: float

confidence: float

A part is made up of two joints:

class Part:

joint1: Joint

joint2: Joint

4.3.4. Keypoint Normalization

Estimated pose output from pose estimator may vary with many relative factors such as different body length measurement, distance from the camera, as well as other relative factors. In order to system be able to be compatible for any size and length, we normalize the pose based on the torso's length of the body in pixels as we observed that the length

of torso remain almost constant throughout the entire frame of the video. The torso length is basically the average of the distance from the neck keypoint to the right and left hip keypoint. So the distances after normalization are thus represented as ratios of torso length.

$$NormalizedArmVectorLength = \frac{OriginalArmVectorLength}{TorsoLength} \tag{5}$$

4.3.5. Perspective Detection

For some exercises, the video is recorded at a certain angle to be able to capture all the required keypoints into the frame. For example, double arm bicep curl is recorded with user facing directly to the camera whereas the single arm bicep curl is recorded from the side of the body. Therefore, it is crucial to detect the perspective of the body for all the exercises which are recorded from the side of the camera in order to evaluate the exercise properly. Perspective detection is performed by measuring the visible keypoints of all part throughout all the frames of the exercise. Whichever side that has the most part visible in the video is selected as the required perspective. This method almost accurately predict the perspective.

4.3.6. Geometric Evaluation

After perspective detection, all the required vectors are created from keypoints of interest. Required angles for evaluation are calculated by using vector dot product.

$$\begin{aligned} \vec{BA} &= \vec{A} - \vec{B} \\ \vec{BC} &= \vec{C} - \vec{B} \\ \Theta &= \arccos \frac{\vec{BA} \cdot \vec{BC}}{|\vec{BC}||\vec{BA}|} \end{aligned} \tag{6}$$

After calculating all the angles, we use personal training guidelines and our own recorded videos to design geometric heuristics, evaluating on the body vectors.

4.3.7. Rules Evaluation

First the rules were collected from the various sources. Then all the datasets videos collected from various sites were visualized and the rules were extracted from the video in angle form. Then those rules were implemented in if then fashion. Required angles in each frame were evaluated with implemented rules to label them as proper and improper. We obtained data from various locations like youtube and gym consulting websites. We also recorded some exercise at home performing by ourselves. Data then was refined and analysed.

Datasets given by gym instructor:

1. Stand with feet about shoulder-width apart. Keep the back straight and feet planted flat on the floor. Your arms holding weights should hang down. 2. Hold the dumbbells across the thighs horizontally, palms facing back toward the thighs. Ensure that you have a firm grip.

3. Brace the abdominable muscles

4. Lift the weights upward, inhaling with arms out infront and palms facing down. Keep a slight bend in the elbows to reduce stress on joints. Pause when arms are approximately horizontal to floor.

5. Return dumbbells to starting position

Datasets our Perspective:

1. Torso vector and shoulder elbow vector should be parallel

2. Torso vector should not move

3. Angle between wrist and shoulder should always be 180 degrees.

4. Torso vector and shoulder vector angle will increase until it becomes perpendicular and decreases again at initial state marking one set being completed.

4.3.8. Machine Learning based evaluation

Another approach to evaluate the exercise posture can be through various machine learning methods.

Dynamic Time Warping (DTW):

Duration of exercise can vary among the recorded videos which results in a different keypoint vector length for each example. Different feature vector lengths present a

challenge to compare two different exercise sequence so to overcome this we have used a method called dynamic time warping. In DTW, we can dynamically identify the keypoint in the second sequence that corresponds to a given point in first video sequence. DTW distance were calculated for a recorded video with all the correct and incorrect dataset and nearest neighbor classification was performed to produce the output of correct or incorrect for a given excercise sequence.

4.3.9. Text and Voice Feedback

Feedback is provided to the user as per the rules which angles calculated from their recorded video or live video from webcam hit. Feedbacks are displayed in the text, and also for convenience we've used text-to-speech API to convert displayed text into voice. That way, user need not look at screen to read feedback and can get it through voice of the computer.

4.3.10. Repetition Counter

We've used a repetition counter to count the number of reps in exercise.The diagram below depicts two cycles (repetitions): A-C (1st cycle) and E-G (2nd cycle). Simply put, if the angle crosses the range of motion's middle line (point B), it counts as one repetition. This counts as a correct repetition if the following angles (trend) pass the upper/lower limits of the range of motion (point C) and then continue to the opposite side of the range of motion (point E). Otherwise, it will be considered a incorrect repetition (point G). As a result, the first cycle will be considered correct, while the second will be incorrect.

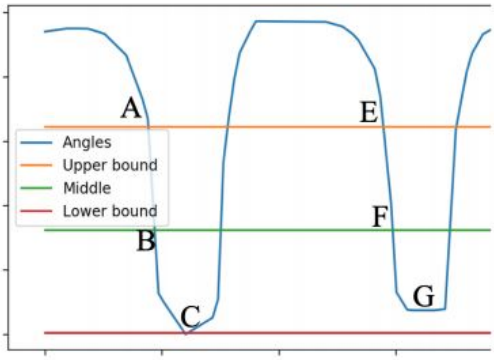


Figure 4.9: Sample of two cycles (repetitions)

4.3.11. View History

System also saves the progress of the user's exercise. All the sets and repetitions of the users is saved in MongoDB database. After user finishes the excercise, the Webapp will request for backend to save the record of the user.
User gives the detail of the date and exercise of which they want to see the history to the webapp and it will request for that particular to the backend. Backend respond with data and webapp displays it on the screen.

4.4. Web Application Development

A web application to display all the required information was developed. The different parts of the web application are described below.

4.4.1. Frontend

User interface is at the heart of our application, because the ultimate goal of this project is to give service to the users as easy as they feel. The features of the application's front end are described below:

Features :
Video is captured using the webcam of the client. Then it is sent to the backend for processing and the feedback+pose data are communicated back to the client in real time. At the moment, we have used a python library airtc which uses webRTC protocol. Real Time video is taken from camera and sent to the server for pose estimation of that video. Then implemented rules are used to analysis if the exercise is performed correctly or not. First video was captured from Jinja application using webcam. Then screenshot of that video is taken specifying the time interval. Then according to the specified time interval the screenshot is converted into base64 string and sent to flask server. We have used socket-io for real-time communication between client and server. Client is in Jinja and server is in flask. When server gets the base64 encoding then it is decoded and pose estimation algorithm is processed. First perspective is detected so as to which side we take reference for pose estimation and all required angles between the joints are calculated. Repetition occurred in the video is counted and correct and incorrect reps is

counted and feedback is given to user which is sent back to client side.

Technologies Used :

Flask, HTML/CSS and Javascript are used to develop the front end of the application. CSS and Bootstrap, are used to make the app smooth, intuitive and aesthetically pleasing.

4.4.2. Backend and Database:

The backend of the application is developed in Python using Flask. MongoDB Database is used to store and manage the application's data.

4.4.3. Rest API:

Flask Framework is used to develop Rest API to communicate with the Javascript frontend. The developed API returns the data saved in the database as specified in JSON format. Postman and Talend are used to test and document the APIs for easy communication between front end and back end developers.

4.4.4. Testing and Documentation

Test-driven deployment Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: requirements are turned into very specific test cases, then the software is improved so that the tests pass. This is opposed to software development that allows software to be added that is not proven to meet requirements.

Agile software development Agile software development comprises various approaches to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers. It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change.

5. SYSTEM DIAGRAMS

5.1. Sequence Diagrams

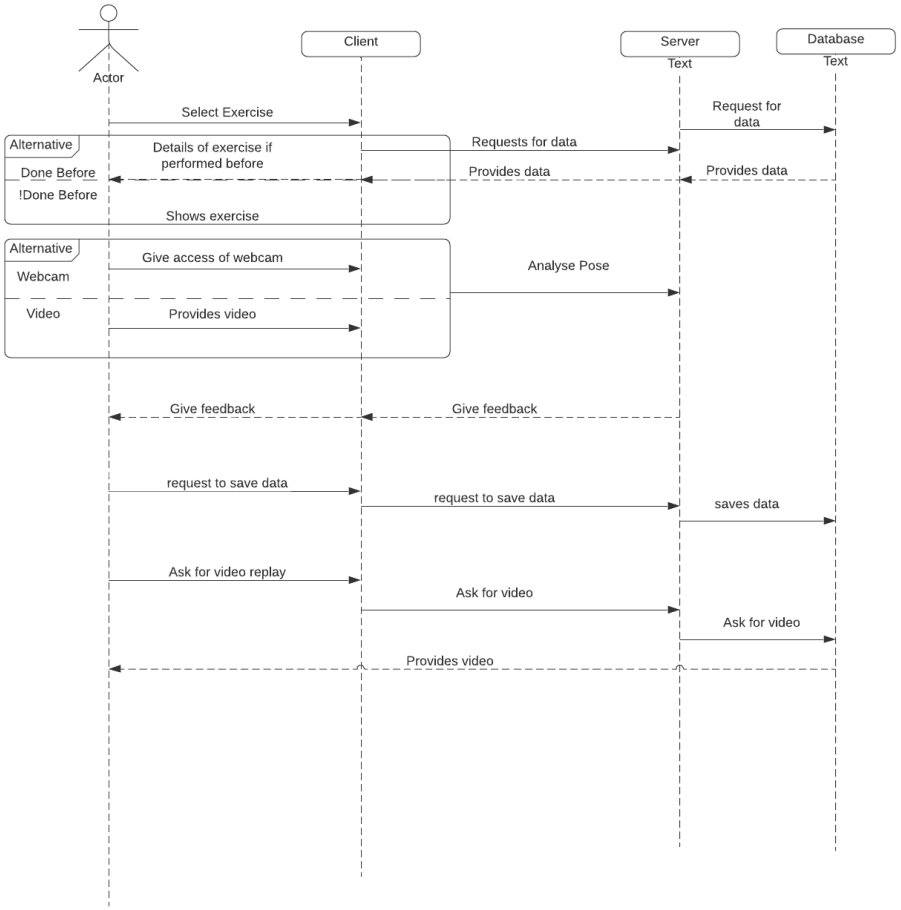


Figure 5.1: Sequence diagram

The above diagram shows the sequence in which the user operates the application. The user gives details to the application of what exercise is to be performed. Application then estimates the pose and see whether the exercise performed is correct or not. If incorrect gives the feedback that it is wrong and if right gives the feedback that it is right. Then the data is saved in the database and is shown in the UI.

5.2. State Chart Diagram

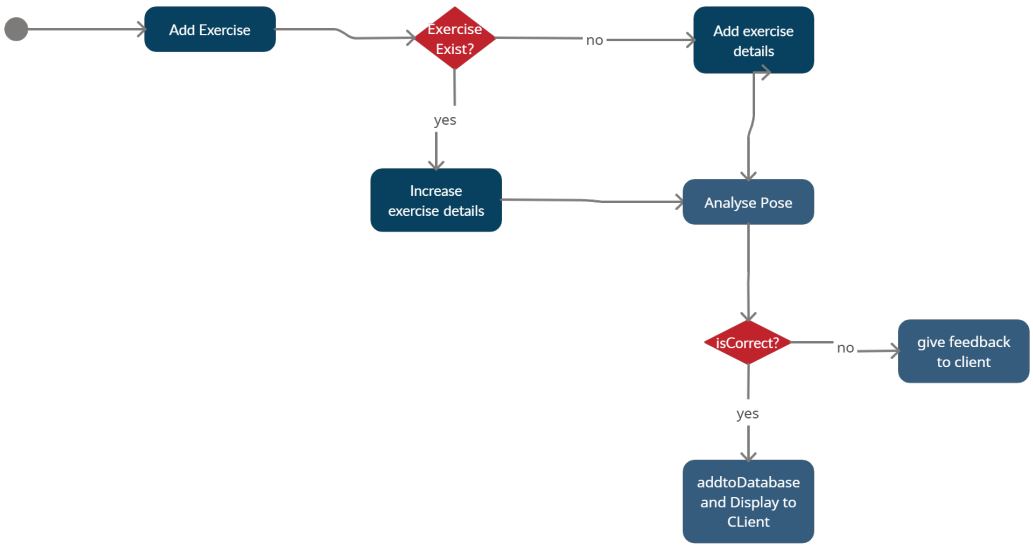


Figure 5.2: State chart diagram

The above diagram shows the various states in which the user operates the application. The user gives details to the application of what exercise is to be performed. Application then estimates the pose and see whether the exercise performed is correct or not. If incorrect gives the feedback that it is wrong and if right gives the feedback that it is right. Then the data is saved in the database and is shown in the UI.

5.3. Activity Diagram

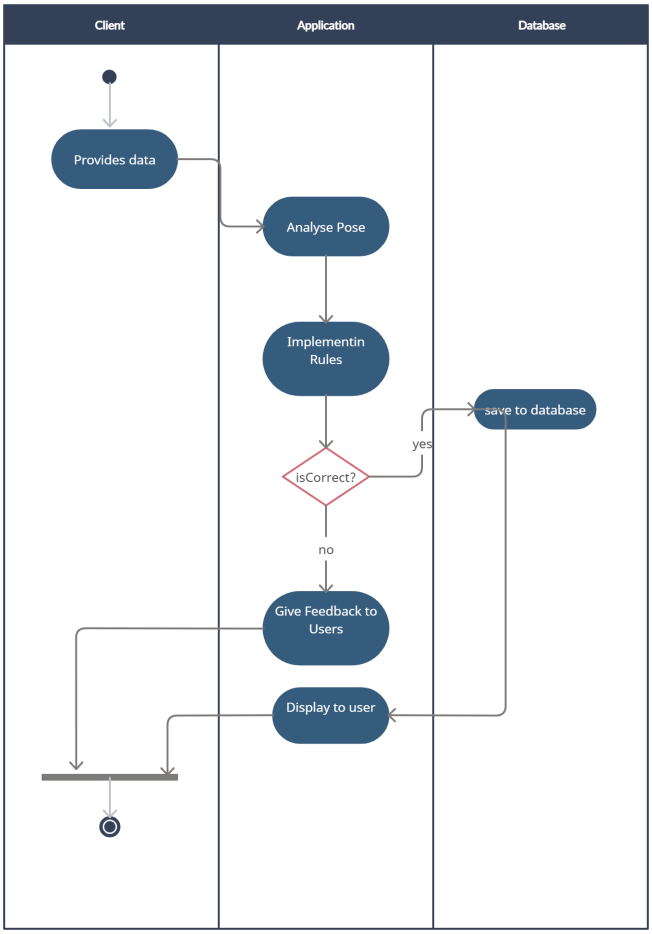


Figure 5.3: Activity diagram

The above diagram is an activity diagram depicting activities involved with three major components: user, database and system/application. Application can be initiated by user. Then application finds whether exercise is done right or wrong and give feedbacks to user

5.4. Use case Diagram

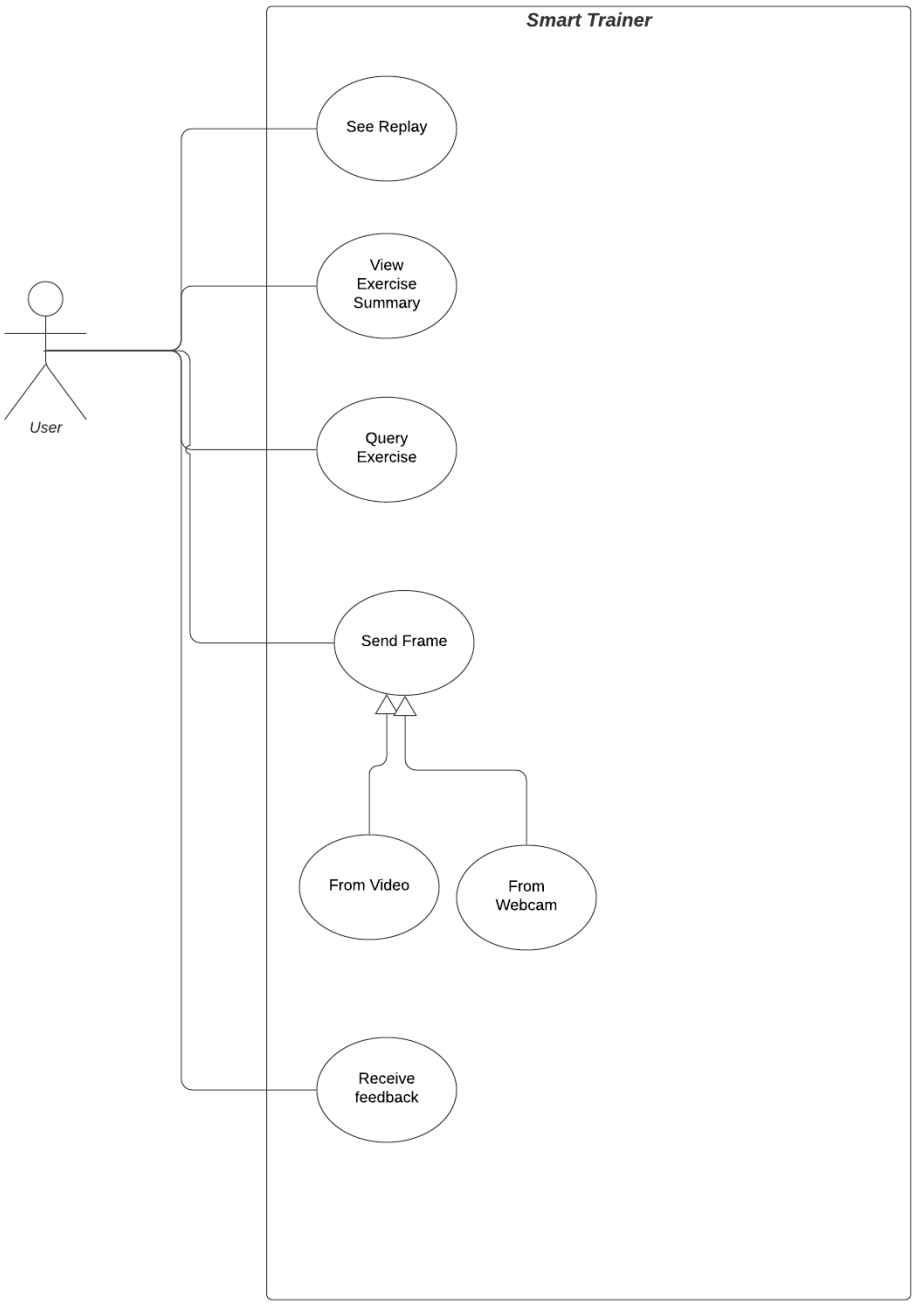


Figure 5.4: Use case diagram

6. RESULTS

6.1. Pose Estimation

We tried different libraries so that we could get the result as we expect. We tried various pose estimation library like posenet and tf-pose- estimation, but we found out openpose suits our application the best so, we used openpose. We send the video to openpose which to get the required pose of all frames and use the pose estimated points to analyse the output of how user is performing the exercise.

6.1.1. Openpose

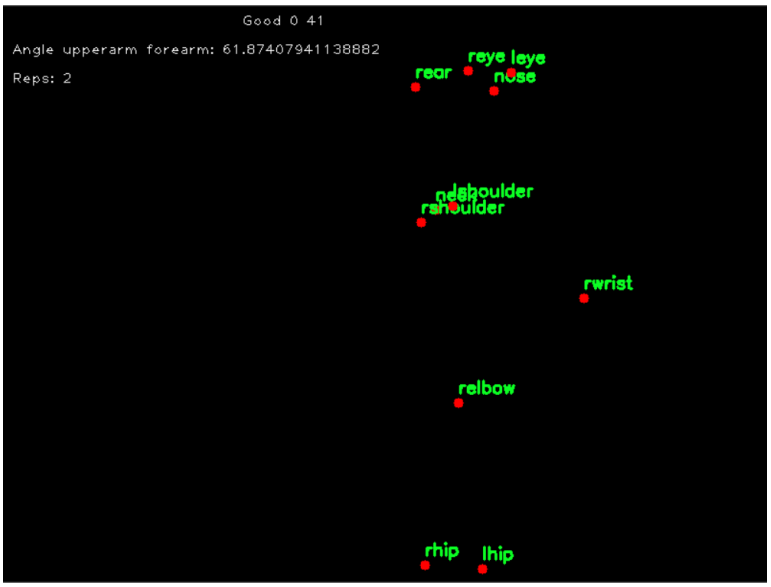


Figure 6.1: Pose estimation plot

In above figure, we have plotted all the joints obtained from pose estimation and line was drawn between the moving joints and angle was visualized.

6.1.2. Blazepose for pushup evaluation

Figure 6.2: Blazepose pushup output

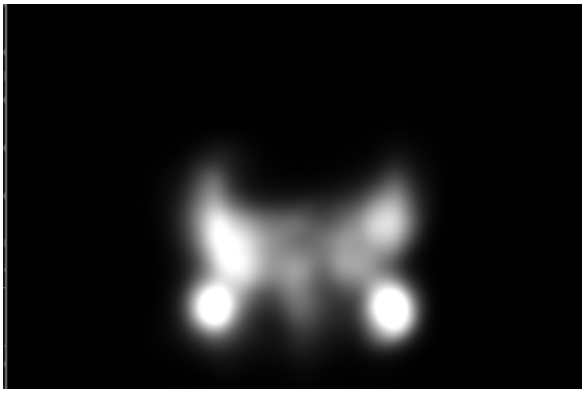


Figure 6.3: Blazepose heatmap pushup output

Figure 6.4: Blazepose heatmap pushup up position output

Evaluation metrics are as follows:

BlazePose - Heatmap branch:

PCKs: 0.760

MAE: 0.032

Number of parameters: 885,559

Training for full body pose estimation for 29 epochs:

joints loss: 0.0325

heatmap loss: 0.6929

joints pck2: 0.1141

joints mae2: 0.1896

heatmap pck1: 0.0459

heatmap mae1: 0.3187

val loss: 0.0297

val joints loss: 0.0297

val heatmap loss: 0.6930

val joints pck2: 0.1096

val joints mae2: 0.1909

val heatmap pck1: 0.0510

val heatmap mae1: 0.3218

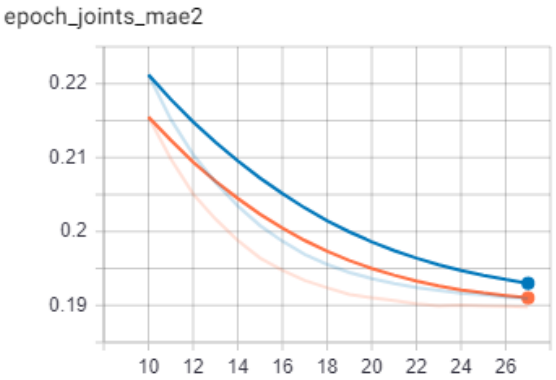


Figure 6.5: Tensorboard joints mae2

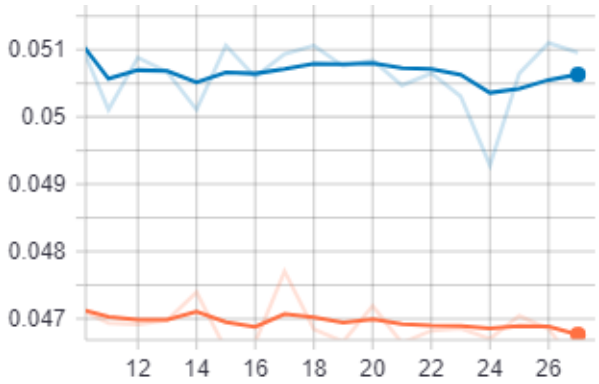


Figure 6.6: Tensorboard heatmap pck1

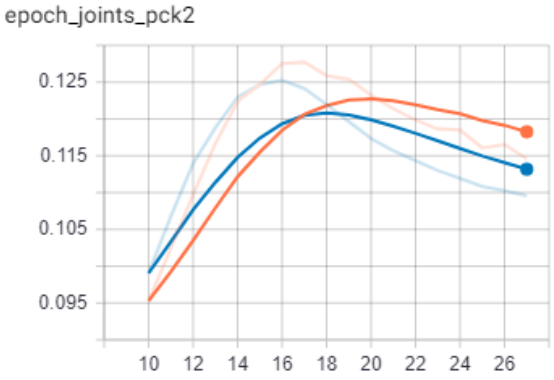


Figure 6.7: Tensorboard joints pck2

6.2. Exercise Result

6.2.1. Bicep Curl

The range of angle between the upper arm and the torso (measuring if the user rotates the shoulder when lifting) and the minimum angle between the upper arm and the forearm are both taken into account in our geometric algorithm (measuring how high the user lifts). If the angle between the upper arm and the torso is greater than 35 degrees, we consider it excessive shoulder rotation. This is flagged as not curling the weight all the way up if the minimum angle between upper arm and forearm is greater than 70 degrees. All good form videos are correctly classified by our geometric algorithms. The majority of incorrect form exercises are labeled as such, while the rest are labeled as correct form. These mislabeled examples appear to be on the border between proper form and clearly improper form upon visual inspection: tweaking our algorithms will, as expected, adjust

the decision boundary, and thus our rigor when evaluating user exercise posture.

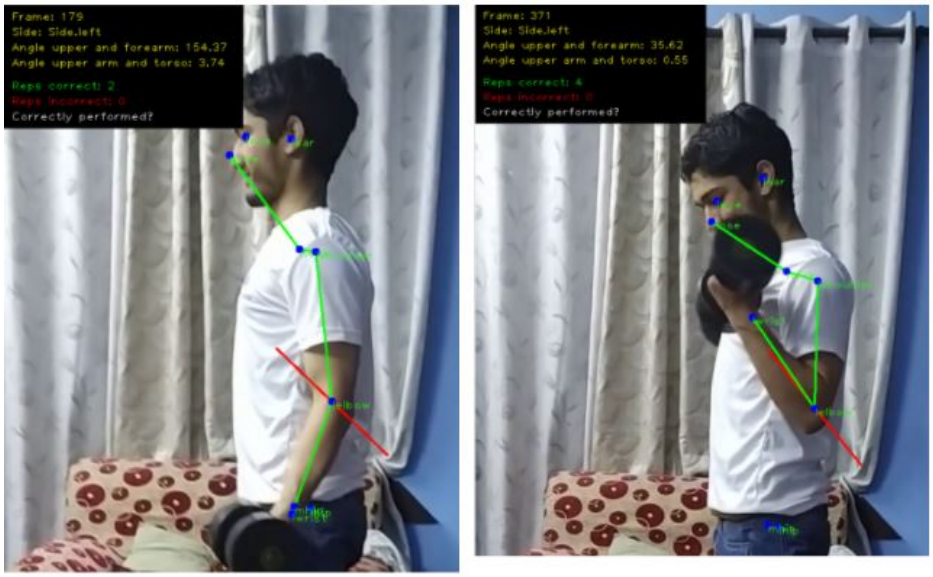


Figure 6.8: Pose estimator on bicep curl exercise

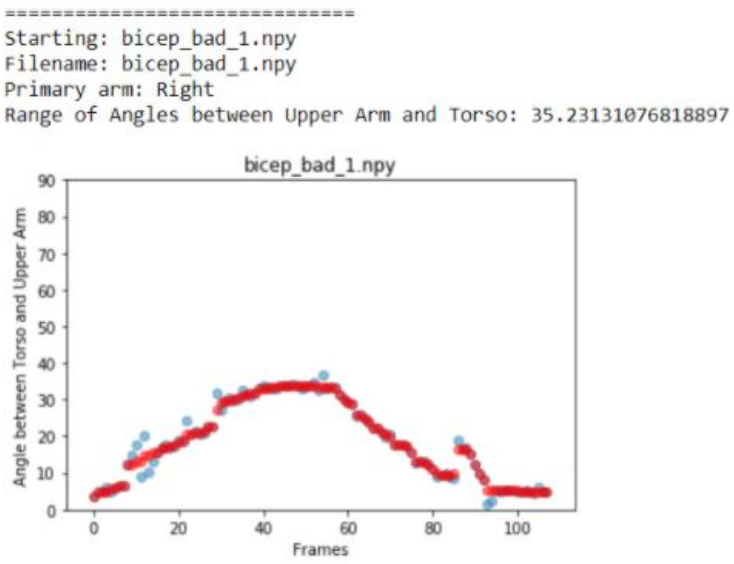


Figure 6.9: Angle between upper arm and torso for improper bicep curl form

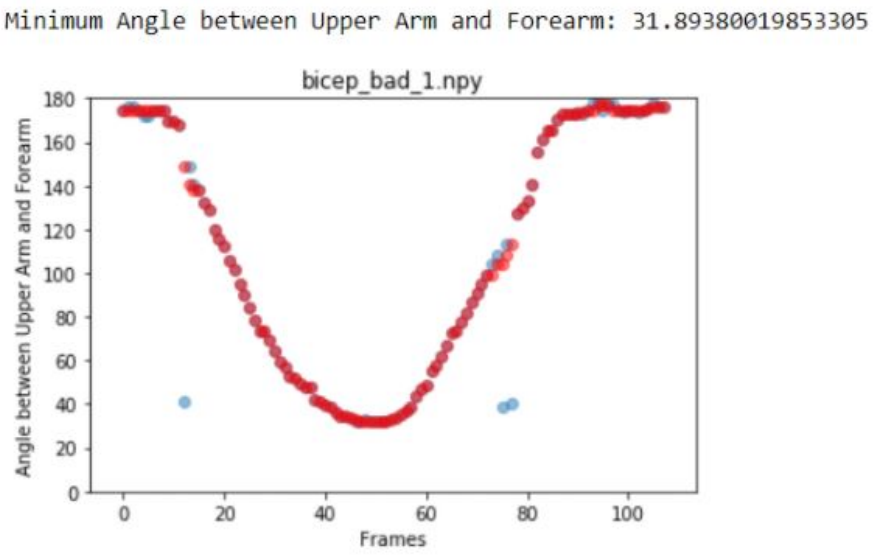


Figure 6.10: Angle between upper arm and forearm for improper bicep curl form

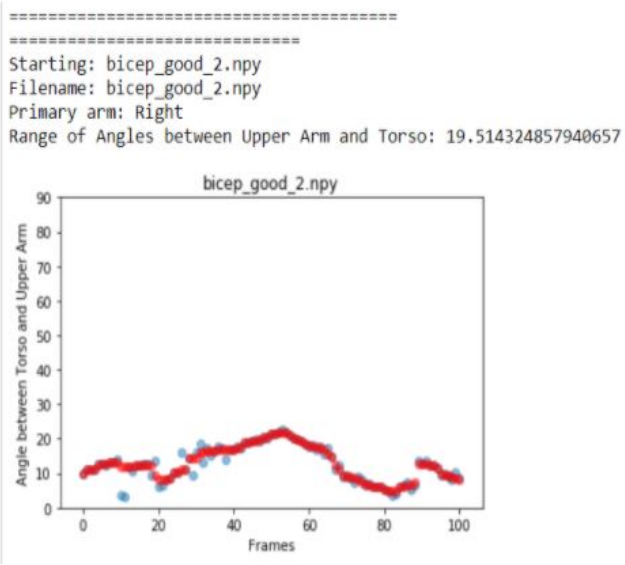


Figure 6.11: Angle between upper arm and torso for proper bicep curl form

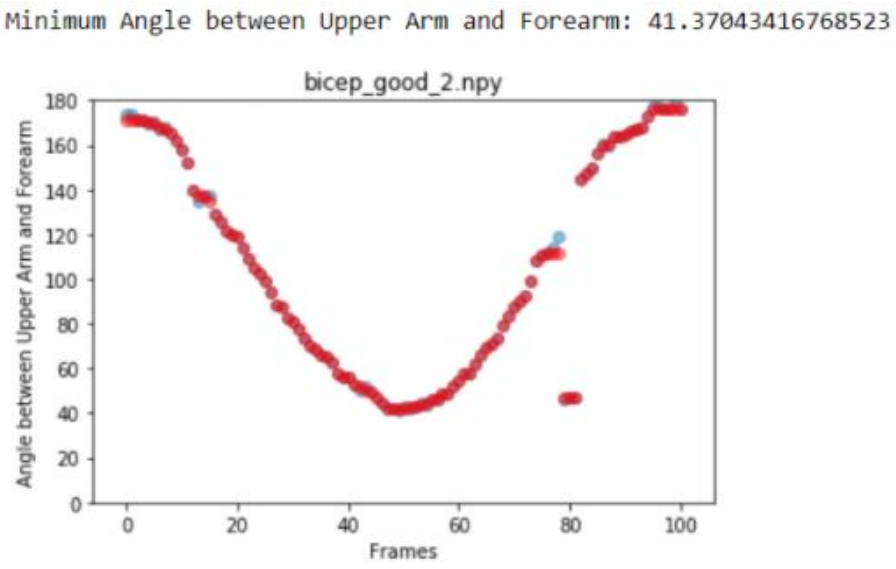


Figure 6.12: Angle between upper arm and forearm for proper bicep curl form

DTW classifier is also used for Bicep Curl exercise of 7 video sequence on which 4 were correct and 3 were incorrect and the data is split into training data and test data. The result of DTW classifier of Biceps Curl exercise is shown in figure below.

	precision	recall	f1-score	support
correct	0.80	1.00	0.89	4
incorrect	1.00	0.67	0.80	3
avg / total	0.89	0.86	0.85	7

Figure 6.13: Result of DTW classfier on bicep curl

6.2.2. Front Raise

In our geometric algorithm, we take into account the torso's range of motion (as well as the angle between the upper arm and the torso if the user rotates the body when lifting) and the minimum angle between the upper arm and the torso (measuring how high the user lifts). DTW classifier was used for 12 video sequence on which 6 were correct and 6 were incorrect.

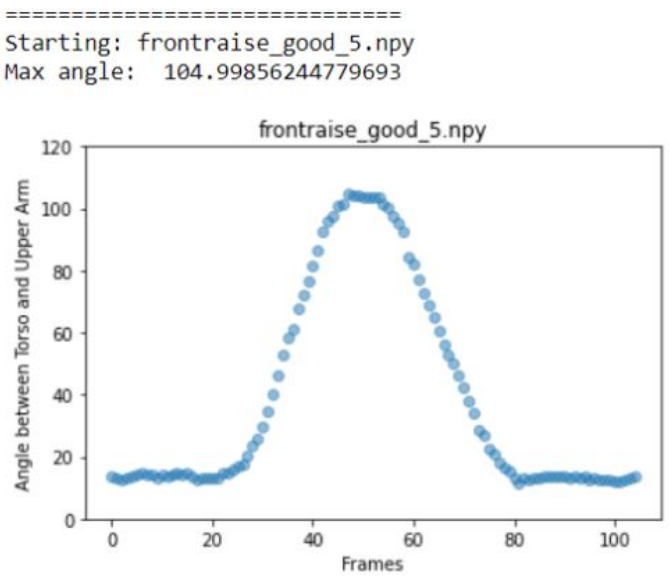


Figure 6.14: Angle between arm and torso in proper front raise form

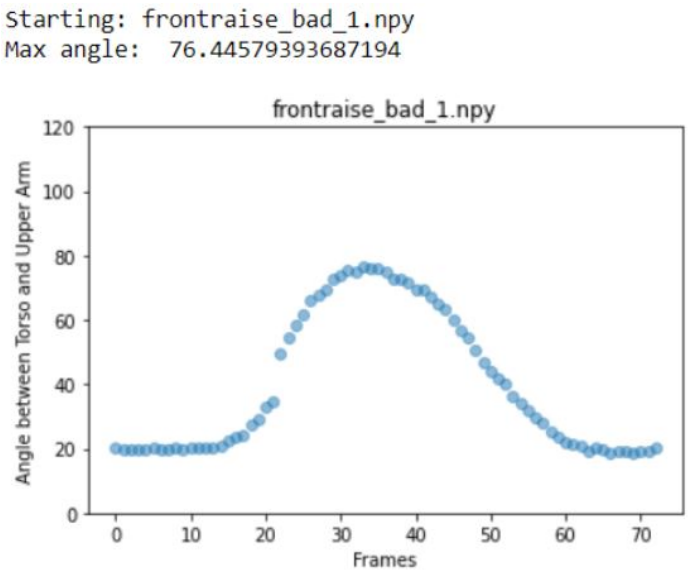


Figure 6.15: Angle between arm and torso in improper front raise form

	precision	recall	f1-score	support
correct	1.00	1.00	1.00	6
incorrect	1.00	1.00	1.00	6
avg / total	1.00	1.00	1.00	12

Figure 6.16: DTW classifier on front raise

6.2.3. Double Arm Bicep Curl

We discovered that the maximum angle between the forearm and upper arm is between 145 and 160 degrees, and that any movement less than 140 degrees is labeled as "Curling not performed all the way to the top." The difference in angle is clearly seen in the diagram below.DTW classifier was used for 15 video sequence on which 8 were correct and 7 were incorrect.

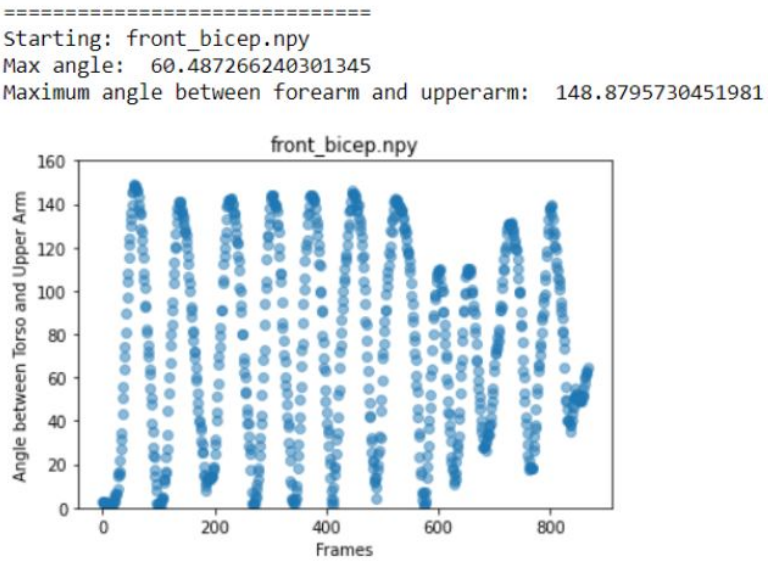


Figure 6.17: Angle between upper arm and forearm in double arm bicep curl

	precision	recall	f1-score	support
correct	1.00	0.75	0.86	8
incorrect	0.71	1.00	0.83	5
avg / total	0.89	0.85	0.85	13

Figure 6.18: DTW classifier on double arm bicep curl

6.2.4. Shoulder Press

Our shoulder press geometric algorithm calculates the back range of motion using the neck and hip keypoints, the arm range of motion using the elbow and neck keypoints, and the maximum angle achieved between the upper arm and forearm vectors. If the user's back range of motion is too wide, we advise them to keep their back straight. We

advise the user not to roll their shoulders during the lift if their elbow is behind their neck. We advise the user to lift the weight all the way up if the angle between the upper arm and forearm is too small.DTW classifier was used for 15 video sequence on which 7 were correct and 8 were incorrect.

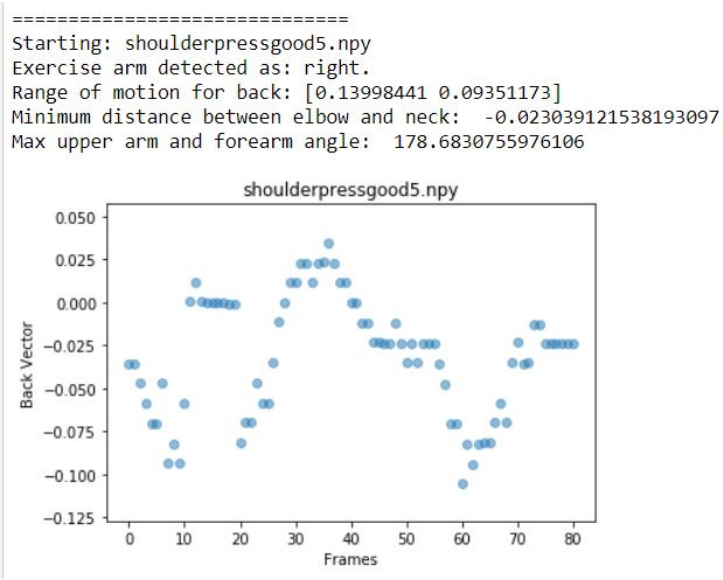


Figure 6.19: Back vector for proper shoulder press form

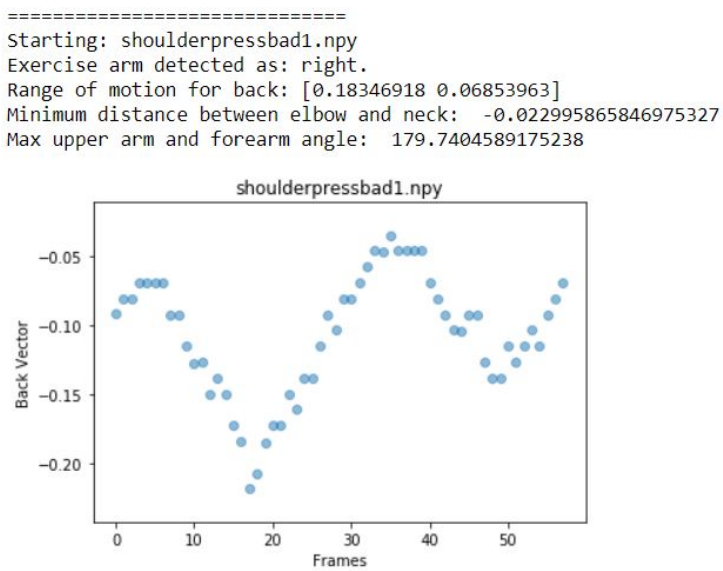


Figure 6.20: Back vector for improper shoulder press form

	precision	recall	f1-score	support
correct	0.67	0.86	0.75	7
incorrect	0.83	0.62	0.71	8
avg / total	0.76	0.73	0.73	15

Figure 6.21: DTW classifier on shoulder press

6.3. Data Visualization

Multiple videos of good exercise and bad exercises were sent to openpose and output joints was saved as numpy arrays and was plotted in opencv to visualize between data. It is then used to make rules of exercises, what exercise is good and what is bad.

6.3.1. MPII Data Visualization

Examples of MPII dataset visualization:

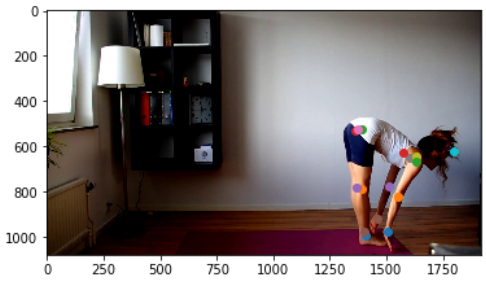


Figure 6.22: MPII pose estimation plot

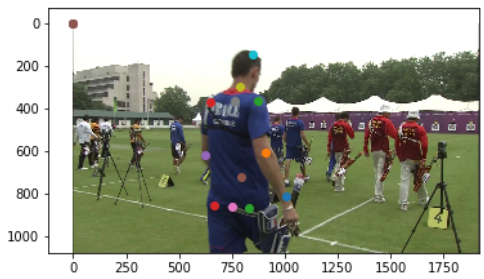


Figure 6.23: MPII pose estimation plot II

NAME	r ankle_X	r ankle_Y	r knee_X	r knee_Y	r hip_X	r hip_Y	l hip_X	l hip_Y	...
015601864.jpg	620	394	616	269	573	185	647	188	...
015599452.jpg	-1	-1	-1	-1	806	543	720	593	...
005808361.jpg	804	711	816	510	908	438	1040	454	...
086617615.jpg	301	461	305	375	201	340	294	342	...
060111501.jpg	980	322	896	318	865	248	943	226	...

Figure 6.24: MPII data format

6.3.2. COCO Data Visualization

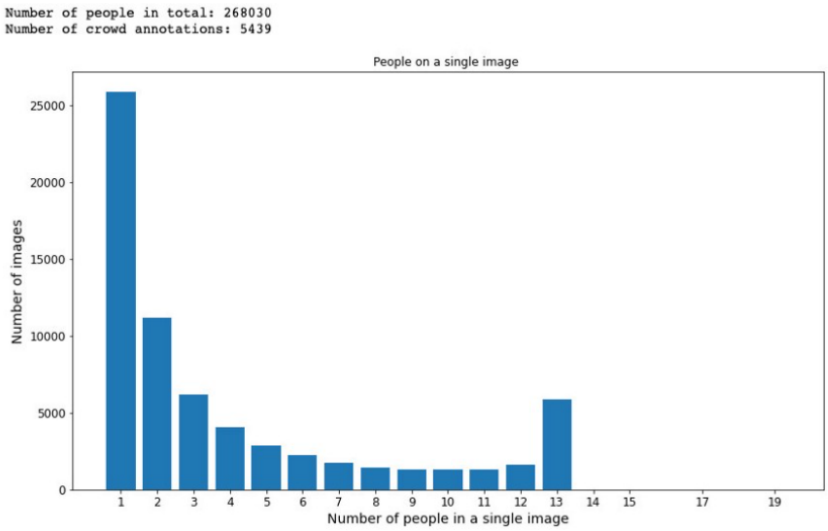


Figure 6.25: Number of people plot

Figure 6.26: COCO dataset example of unwanted image

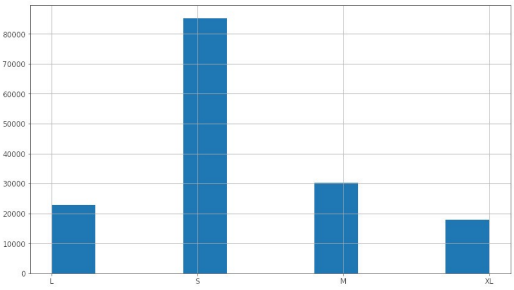


Figure 6.27: Size of persons

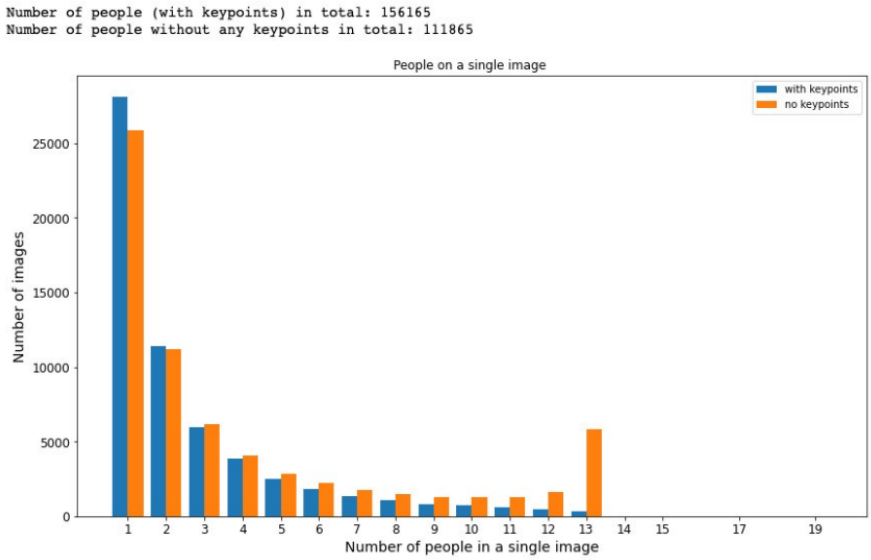


Figure 6.28: Number of people with and without keypoints

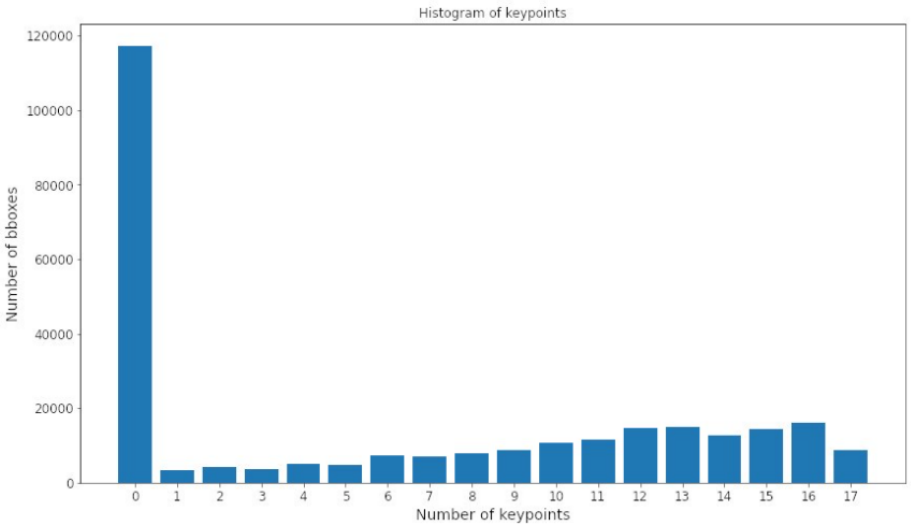


Figure 6.29: Number of keypoints

	Num keypoints in train set %	Num keypoints in val set %	Diff 100%
0	0.429208	0.422755	0.645234
1	0.012459	0.012632	0.017297
2	0.015354	0.015722	0.036713
3	0.013491	0.015176	0.168498
4	0.018128	0.020538	0.240985
5	0.016962	0.019720	0.275784
6	0.026579	0.028808	0.222892
7	0.025775	0.027354	0.157882
8	0.028869	0.029171	0.030260
9	0.032313	0.034079	0.176564
10	0.039346	0.041894	0.254766
11	0.042554	0.040712	0.184178
12	0.053321	0.058342	0.502103
13	0.054765	0.054889	0.012373
14	0.046166	0.042439	0.372704
15	0.052799	0.053072	0.027219
16	0.059619	0.052163	0.745653
17	0.032290	0.030534	0.175567

Figure 6.30: Number of keypoints in train and test in percent

6.3.3. Pose Data Visualization

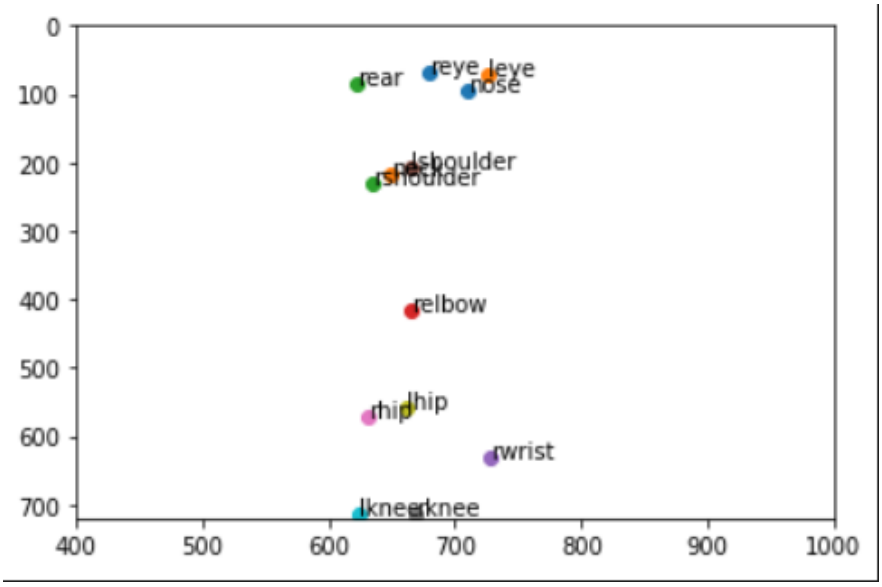


Figure 6.31: Plot of joints and x y coordinate

```
Data shape:  (114, 18, 3)
Mean torso:  360.11485467535044
114
nose <1.9708212276881114, 0.26258344739832407, 0.835979>
neck <1.8021833633746602, 0.6053263206721059, 0.584495>
rshoulder <1.763887248063247, 0.6434558224733541, 0.613341>
relbow <1.845452891964497, 1.1546371792267567, 0.54861>
rwrist <2.0198889064316887, 1.7531906607106573, 0.78617>
lshoulder <1.845719473580761, 0.5780572428417762, 0.333104>
lelbow <0.0, 0.0, 0.0>
lwrist <0.0, 0.0, 0.0>
rhip <1.7532267603045266, 1.5844972585604844, 0.261292>
rknee <1.8565076983639413, 1.987218773980178, 0.137317>
rankle <0.0, 0.0, 0.0>
lhip <1.8347313125853828, 1.551885440837532, 0.124535>
lknee <1.731453151417804, 1.9871632361434566, 0.101476>
lankle <0.0, 0.0, 0.0>
reye <1.8891583925725985, 0.19182796572575952, 0.935494>
leye <2.0141213020881454, 0.19704741162086012, 0.914553>
rear <1.7261881644965915, 0.235362132107575, 0.886749>
lear <0.0, 0.0, 0.0>
```

Figure 6.32: Data structure for joints

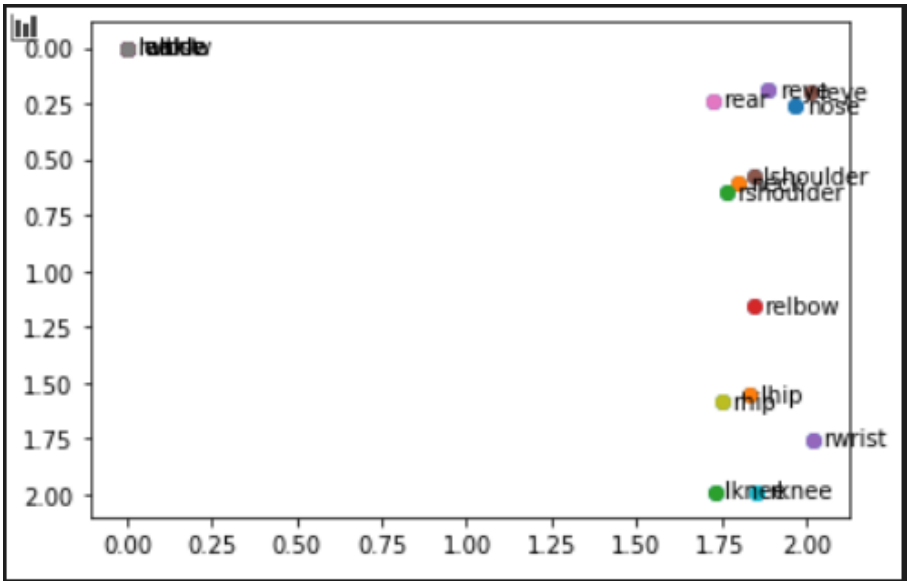


Figure 6.33: Normalized joints xy plot

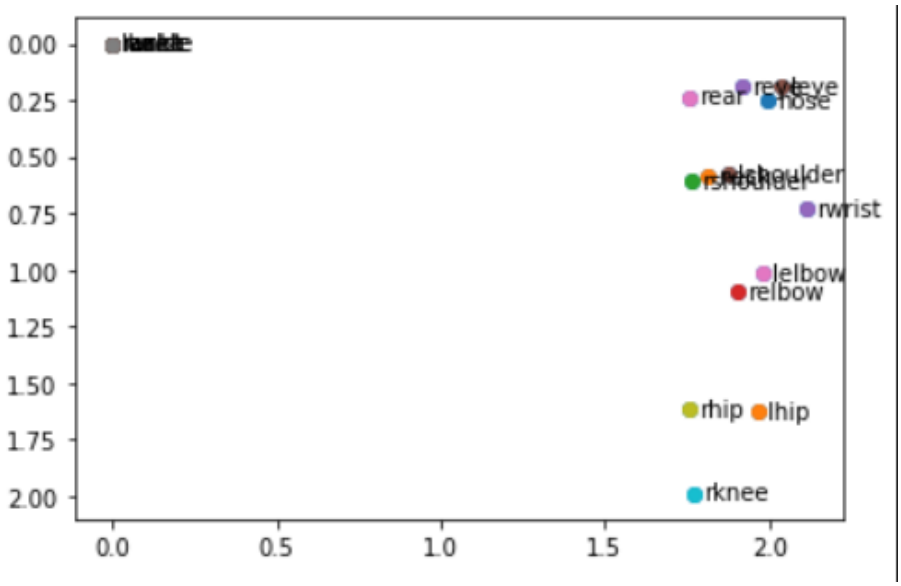


Figure 6.34: Normalized joints xy plot 2

After this we also checked if pose estimation was giving output correctly or not. For Pose Estimation we followed everything as said in Methodology section. The output given by pose Estimation:

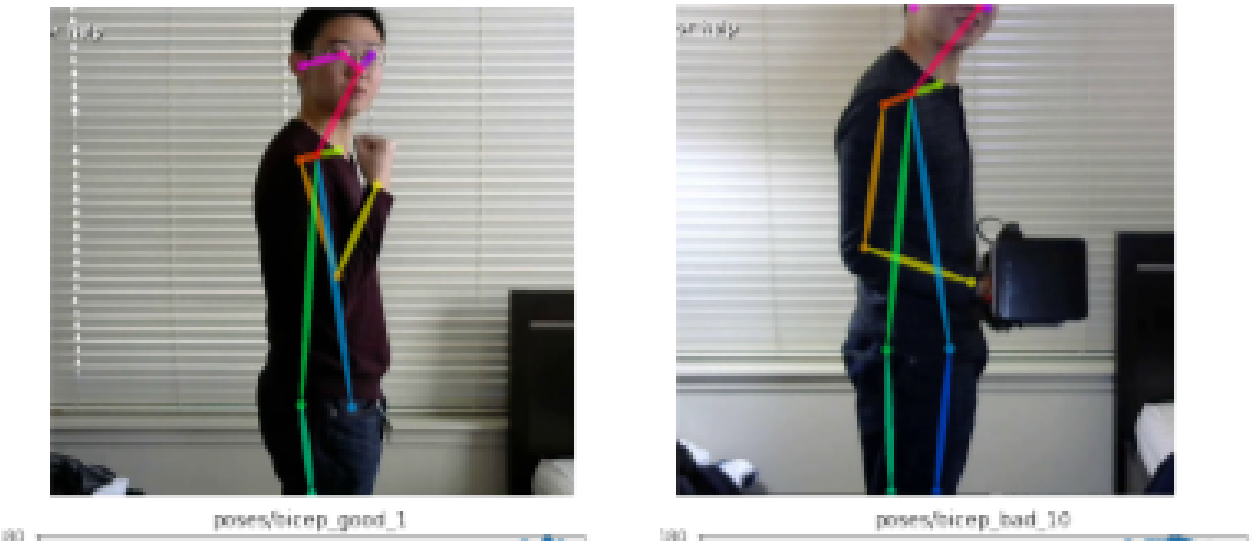


Figure 6.35: Bicep curl correct/incorrect from dataset

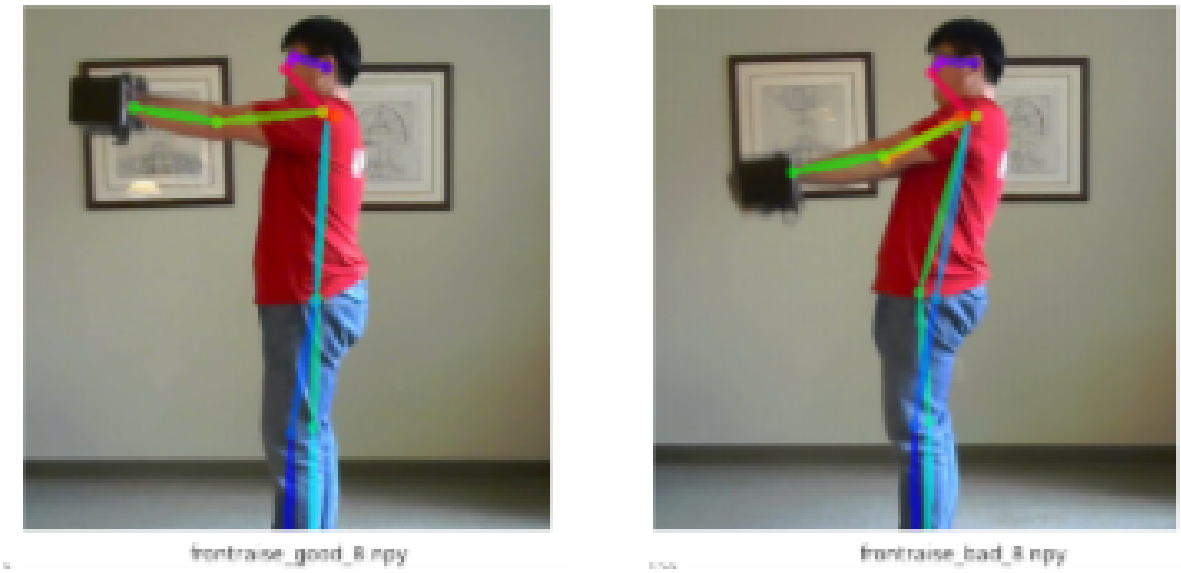


Figure 6.36: Front raise correct/incorrect from dataset

6.4. Real Time Feedback

We have added the functionality of performing the pose estimation as well as analysis in real time. The frames of a live video are sent for pose estimation followed by our previous analysis of the required angles and joint positions. After one repetition of an exercise is performed, the frames which have elapsed so far, are sent to our feedback generator to output correctness and feedback.

We have built a web application to display all the required information which includes pose visualization, feedback,etc. Video is captured using the webcam of the client. Then it is sent to the backend for processing and the feedback+pose data are communicated back to the client in real time.

Real Time video is taken from camera and sent to the server for pose estimation of that video. Then implemented rules are used for analyzing if the exercise is performed correctly or not. Firstly, video was captured from application using webcam. Then screenshot of that video is taken specifying the time interval. Then according to the specified time interval the screenshot is converted into base64 string and sent to flask server. We have used socket-io for real-time communication between client and server. When server gets the base64 encoding then it is decoded and pose estimation algorithm is processed and the output is again sent to react application using socket-io for display.

6.5. Interface

Interface is created using flask. In flask, Jinja is used to render the information in html. Users gives information of what exercise is to be performed. After the data is fetched about the exercise which has been done before by the users. User also can search for the exercise and see their improvement over time. For database, MongoDB is used. To send the data realtime between the server and the client flask-socketio is used. In application, the video related how the exercise is to be done is specified and feedback if the performed exercise is wrong is shown. Also the user can later see the parts of the video which was performed badly so that correction can be made and posture can be analysed at free time. This helps user to benefit more about how the exercise should be performed and how user is performing the exercise.

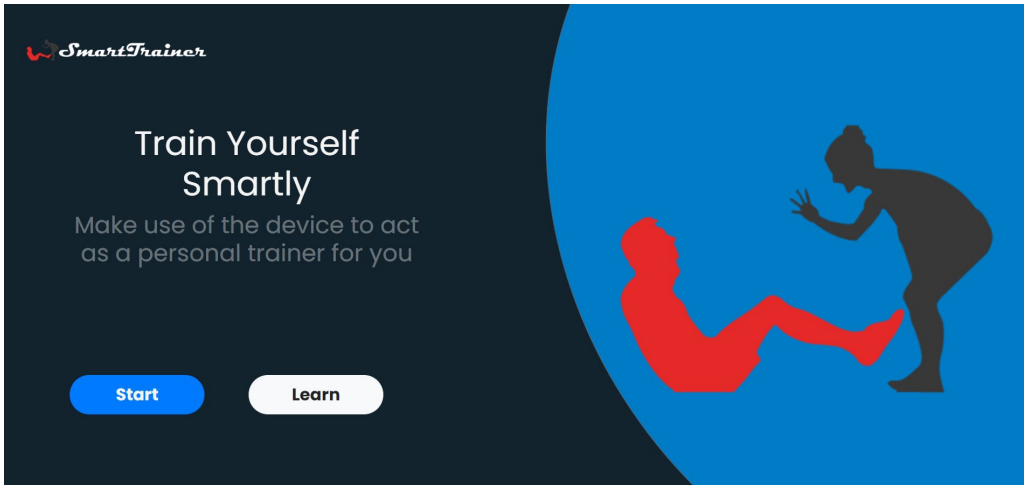


Figure 6.37: Landing page

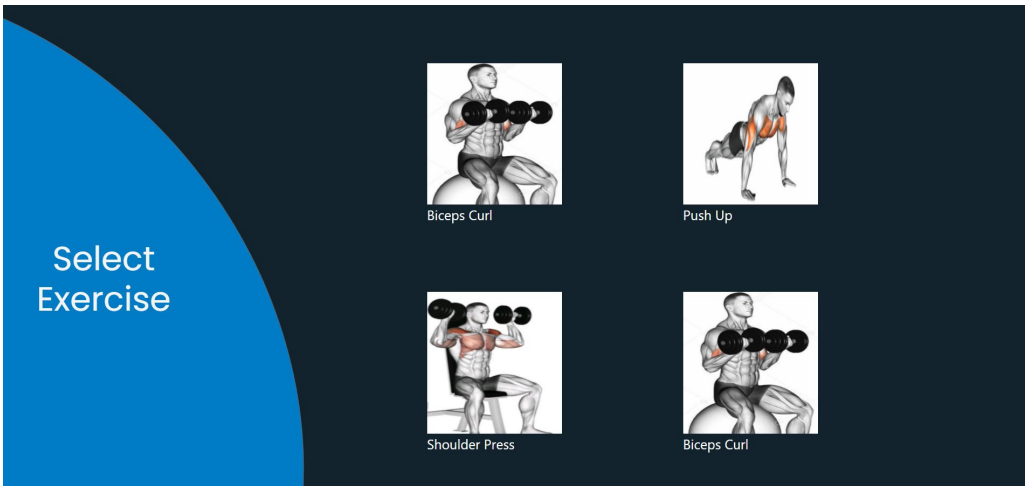


Figure 6.38: Exercise selection page

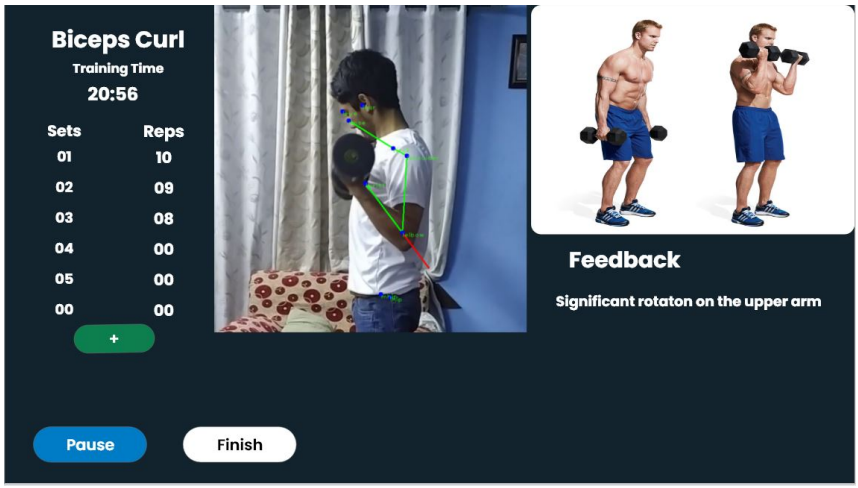


Figure 6.39: Exercise page

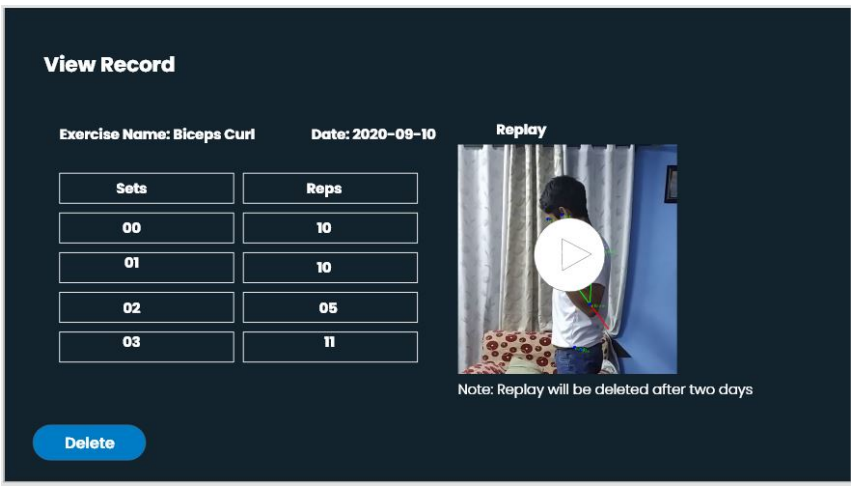


Figure 6.40: Exercise record page

7. PROBLEM FACED

7.1. Data Collection

Data which we collected was very raw. Ten minutes of trainer's youtube video had only 1 or 2 min of useful information also in different parts. So it was difficult to refine the data we had got. While starting the project at the beginning we were planning to collect the data by visiting the gym ourselves. But due to regulations issued by the government to minimize the corona virus, all the gym were closed for significant amount of the time , so collection of data by visiting the gym was very difficult so project was not on much progression as planned.

7.2. Computational Problem

To perform real time, we need very high specifications hardware, which is to expensive so there is problem taking pose estimation in real time. While performing pose estimation in real time, it gives fps of 3 to 5. To improve it better hardware should be used.

8. PROJECT APPLICATION

Pose Estimation has high range of applications. It can be used for analysing the humans movements and create animation for virtual world. It can be used to analyse different movements of people but we have used the application of pose estimation to find whether the exercise done by user is right or wrong. Exercise performed can be injury prone if not done properly and correctly. There are various casualty which even a good fit people have faced by performing the exercise incorrectly. So, this application implements a set of rules whether the exercise is performed well or not and give feedback to the user. This helps every user to perform the exercise correctly and see back whether the exercise they were doing helped in improvement of their exercise or not. It simply uses a web camera and when given a correct angle performs the pose estimation sees the joints of body and implements certain rules and see whether the performed exercised is correct or not.
Along with the feedback, user also can collect the data from the specified dates and then keep the records of their progress to see it back in the future and analyse about it and see their progress. They can also see the parts of the video where they have performed it the wrong way and analyse about it later on.

9. CONCLUSION AND FURTHER WORKS

9.1. Conclusion

In this report, we introduced an end to end application that uses pose estimation and machine learning to provide personalized feedback on fitness exercise form. Pose estimation is applied to evaluate the exercises video to generate the key points. The keypoints are then processed by a rule based system to check the correctness of exercise pose. Exercise pose is classified correct and incorrect by the evaluation system and correct feedback is selected provided to the user in case of incorrect exercise pose.

9.2. Further Work

Several extensions can be added for future work in this project. This project can be easily extended to the smartphones with an application that record the video and get feedback at any place or time. The system can be further improved to automatically detect the type of exercise from the video without having to manually select it. Similarly, we have considered only the most common exercises like bicep curl, front raise. This proof of concept application can be developed into a full fledged product by introducing the other exercises along with their rules, feedback and exercise tracking system. Another direction can be improvement in the granularity of the feedback received from the system. Instead of specifying what is wrong in the exercise form, we can also add feedback on where the user's pose needs improvement (e.g. back, shoulders) and suggest necessary action to improve it.

REFERENCES

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. Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2d pose estimation using part affinity fields,” in CVPR, 2017.

4. G. Hidalgo, Z. Cao, T. Simon, S.-E. Wei, H. Joo, and Y. Sheikh, “OpenPose library,” https://github.com/ CMU-Perceptual-Computing-Lab/openpose.

5. K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask r-cnn,” in ´ ICCV, 2017.

6. H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, “RMPE: Regional multiperson pose estimation,” in ICCV, 2017.

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. M Andriluka, “Pictorial structures revisited: People detection and articulated pose estimation,” in CVPR, 2009.

11. V. Bazarevsky and I. Grishchenko and K. Raveendran and T. Zhu and F. Zhang and Matthias G. "BlazePose: On-device real-time body pose tracking", 2020

12. S. Chen and R. Yang, "Pose trainer: correcting exercise posture using pose estimation", 2020

13. D. Berndt and J. Clifford. "Using dynamic time warping to find patterns in time series," In AAAIWS, 1994.

14. F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. Black. "Keep it smpl: Automatic estimation of 3d human pose and shape from a single image," in ECCV, 2016

15. A. Newell, K. Yang, and J. Deng. "Stacked hourglass networks for human pose estimation," on ECCV, 2016.

16. G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy. "Towards accurate multi-person pose estimation in the wild" In CVPR, 2017.

17. J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. "Real-time human pose recognition in parts from single depth images," In CVPR, 2011.

18. A. Toshev and C. Szegedy. "Deeppose: Human pose estimation via deep neural networks," In CVPR, 2014.

19. S. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. "Convolutional pose machines," In CVPR, 2016.

20. P. Zell, B. Wandt, and B. Rosenhahn. "Joint 3d human motion capture and physical analysis from monocular videos," In CVPR Workshops, 2017

21. Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann. "Real-time facial surface geometry from monocular video on mobile gpus," CoRR, abs/1907.06724, 2019.

22. S. Kreiss, L. Bertoni, and A. Alahi. "Pifpaf: Composite fields for human pose estimation," In proceedings of the IEEE conference on computer vision and pattern recognition

23. T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. Lawrence ´ Zitnick. "Microsoft coco: Common objects in context," In European conference on computer vision

24. A. Newell, K. Yang, and J. Deng. "Stacked hourglass networks for human pose estimation," In European conference on computer vision

25. K. Sun, B. Xiao, D. Liu, and J. Wang. "Deep high-resolution representation learning for human pose estimation," In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,

26. T. Alatiah and C. Chen. "Recognizing Exercises and Counting Repetitions in Real Time," May 2020.