HOME TRAINER: PROVIDING FEEDBACK FOR EXERCISES

A PROJECT REPORT

Submitted By

VIGNESH R. C. 312217104186

UJJWEL BALWAL 312217104179

SRINIVASA ARUN YERAGUDIPATI 312217104166

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

SSN COLLEGE OF ENGINEERING

KALAVAKKAM 603110

ANNA UNIVERSITY :: CHENNAI - 600025

April 2021

ANNA UNIVERSITY : CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report titled **"HOME TRAINER: PROVIDING FEEDBACK FOR EXERCISES"** is the *bonafide* work of "**VIGNESH R. C. (312217104186)**, **UJJWEL BALWAL (312217104179)**, and **SRINIVASA ARUN YERAGUDIPATI (312217104166)**" who carried out the project work under my supervision.

Dr. CHITRA BABU
HEAD OF THE DEPARTMENT
Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Dr. T. T. MIRNALINEE
SUPERVISOR
Professor,
Department of CSE,
SSN College of Engineering,
Kalavakkam - 603 110

Place:
Date:

Submitted for the examination held on. . . . . .

Internal Examiner **External Examiner**

ACKNOWLEDGEMENTS

We thank the Almighty GOD for giving us the strength and knowledge to do this project.

We would like to thank and express our deep sense of gratitude to our guide **Dr. T. T. MIRNALINEE**, Professor, Department of Computer Science and Engineering, for her valuable advice and suggestions as well as her continued guidance, patience and support that helped us to shape and refine our work.

Our sincere thanks to **Dr. CHITRA BABU**, Professor and Head of the Department of Computer Science and Engineering, for her words of advice and encouragement. Also, we would like to thank our project Coordinator **Dr. B. BHARATHI**, Associate Professor, Department of Computer Science and Engineering for her valuable suggestions throughout this project.

We express our deep respect to the founder **Dr. SHIV NADAR**, Chairman, SSN Institutions. We also express our appreciation to our Principal, **Dr. V. E. ANNAMALAI**, for all the help he has rendered to us during this course of study.

We would like to extend our sincere thanks to all the teaching and non-teaching staff of our department who have contributed directly our indirectly during the course of our project work. Finally, we would like to thank our parents and friends for their patience, cooperation and moral support throughout our lives.

VIGNESH R. C. UJJWEL BALWAL SRINIVASA ARUN YERAGUDIPATI

ABSTRACT

Increasing sedentary lifestyles have become a major problem and the need for everyone to adopt a fit and healthy lifestyle has increased a lot over time. However, not everyone has access to expensive gyms and fitness studios and professional trainers to guide them. We create a user friendly chat bot application that provides feedback for a predefined exercise, informing the user whether the exercise was performed as intended, or if it needs improvement, with specific points of feedback as to how the user can improve their exercise posture. The user provides a pre-recorded video of them performing the exercise, or directly from the webcam. The chat bot acts as frontend of a prediction system. The prediction system provides feedback via two alternative approaches: Geometric approach and ML-DTW Approach. In geometric approach, specific angle ranges and values made by specific limbs are measured and compared with standard predefined values, and feedback such as ”You are not curling your hands enough” and ”Exercise performed correctly” is displayed. In the ML-DTW approach, the angles measured are compared with example videos in a dataset using a modified version of the K-Nearest Neighbours algorithm with Dynamic Time Warping as the distance metric, and similar feedback is provided. We obtain an F1-Score of **1.0** and **0.72** for Squat and Bicep Curl respectively.

TABLE OF CONTENTS

ABSTRACT **iii**

LIST OF TABLES **vi**

LIST OF FIGURES **vii**

1 INTRODUCTION **1**

2 LITERATURE SURVEY **2**

3 PROPOSED METHODOLOGY **4**

3.1 Video Input 5

3.2 Pose Estimator 6

3.3 Processing Coordinates 8

3.4 Feedback Generator 9

3.4.1 Geometric Approach 10

3.4.2 ML-DTW Approach 14

3.4.3 Dynamic Time Warping algorithm 14

3.4.4 Modified KNN Algorithm for Classification 15

4 EXPERIMENTAL SETUP **18**

4.1 Collecting the Dataset 18

4.2 Installing OpenPose And Processing Video 19

4.3 Application Interface 22

5 PERFORMANCE ANALYSIS **24**

5.1 Performance metrics used 24

5.2 Results obtained 26

5.2.1 Geometric Approach 26

5.2.2 ML-DTW Approach (Using KNN) 27

6 CONCLUSION AND FUTURE WORK **30**

LIST OF TABLES

5.1 Squat Confusion Matrix 27
5.2 Bicep Curl Confusion Matrix 28

LIST OF FIGURES

3.1 Proposed System Flow 4
3.2 Proposed Architecture Diagram 5
3.3 Architecture of the multi-stage CNN used in OpenPose. 7
3.4 Illustrates the overall pipeline of OpenPose. 7
3.5 Possible keypoint mappings 9
3.6 Squat Analysis 12
3.7 Bicep Curl Analysis 13
3.8 Dynamic Time Warping Matching [16] 14

4.1 Bicep Curl and Squat Samples 19
4.2 Sample of videos processed by Openpose 22
4.3 Sample UI Windows 23

5.1 Precision and Recall 25

CHAPTER 1

INTRODUCTION

The 2019-2020 coronavirus pandemic has made the people confined to their homes. This situation has made it difficult for people to go to a gym for workout or for exercise. Also, it is not possible for gym trainers to help out their clients either due to gyms being closed or due to social distancing rules. Lockdown imposed in most parts of the country also increase a sedentary lifestyle due to the lack of any physical activity. Exercise and physical activity are vital for blood glucose management and overall health in individuals with diabetes and prediabetes [1]. Physical activity is associated with numerous health benefits. Even a very small amount of physical activity can help high-risk youngsters [2]. Hence, there is a need for a method of getting people to do physical activity, within their own homes, despite the absence of a trainer. We try to leverage computer vision to tackle the problem: using the technology to predict and give feedback regarding the accuracy of exercises one performs.

We develop a user friendly application to provide feedback for the exercises done by people without a professional trainer.

CHAPTER 2

LITERATURE SURVEY

Today, a lot of data is generated in the sports and fitness industry by tracking human body movements. This data can be analysed using machine learning and deep learning methods to improve the training quality of athletes and improve their performance. Arnold [3] shows how data from sensors can be used to automatically give advice on how to continue exercising or to adjust the sports equipment during the physical activity based on Neural Networks, Hidden Markov models and Support Vector Machines. Mustafa Acikkar, Mehmet Fatih Akay, Kerem Tuncay Ozgunen, Kadir Aydin and Sanli Sadi Kurdak [4] present a new approach based on support vector machines to predict whether an athlete is aerobically fit or not.

IMUs (Inertial Measurement Units) has been used for human motion tracking, which is used for the real-time evaluation of repetitive physical exercise [5]. Three-axis accelerometer has been incorporated into a workout glove to track hand movements and put another accelerometer on a user's waist to track body posture. This helps to recognise the exercise that is begin performed along with the number of repetitions of each exercise [6]. Naïve Bayes Classifier and Hidden Markov Models are used to recognise the type of exercise. Adaptive counting approach have been used to capture weightlifting exercise repetitions despite fluctuations in the speed of performing. The counting is based on mean acceleration information and a sensor-based peak detection method [7]. A Machine Learning technique called Linear Discriminant Analysis has been used

for analyzing data coming from wearable inertial measurement units, (IMUs) to classify or count exercises [8]. RecoFit [9], a system for automatically tracking repetitive exercises, provides real-time and post-workout feedback via an arm-worn inertial sensor, with no user-specific training and no intervention during a workout. A Kinect based algorithm has also been developed for real-time monitoring of physical rehabilitation exercises [10]. A recognition system for home based physiotherapy exercises has been built using Generative Bayesian Network by combining the information from the three main components of a physiotherapy exercise (the motion patterns, the stance knowledge, and the exercise object) [11]. Finally, a novel post processing step is employed to estimate the exercise repetitions counts. GymCam [12], a camera-based system automatically detects, recognizes and tracks multiple people and exercises simultaneously in unconstrained environments without any user intervention.

These techniques are more focused on professional athletes, and also require additional sensors to correct exercise posture. We provide feedback for exercises done by anyone without attaching any sensors to the user in the comfort of their home.

CHAPTER 3

PROPOSED METHODOLOGY

Consider the example of squats. The method of doing squats is to bend your knees, go into an almost crouching position, until your upper legs are parallel to the ground and return to standing position. If all the conditions are met, the feedback will be ”Exercise performed correctly”. If not, the feedback will suggest what is performed wrong and mention changes to be made. For example, ”Upper leg is not parallel to ground, please ensure that the upper leg is parallel to the ground” or ”Angle between Upper leg and Lower leg is too large. Try going down further”. Figure 3.1 shows the proposed system flow for same such generic case.

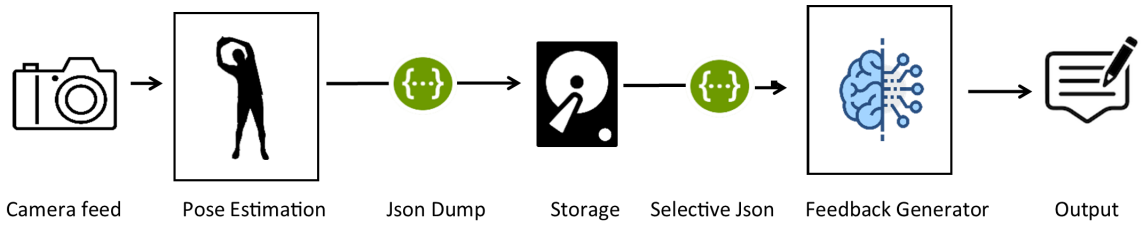

FIGURE 3.1: Proposed System Flow

The system consists of two major modules - the Pose Estimator module and the Feedback generator module. Both the modules are discussed in the upcoming section.

The figure 3.2 depicts the architecture diagram of the proposed application that has the following modules.

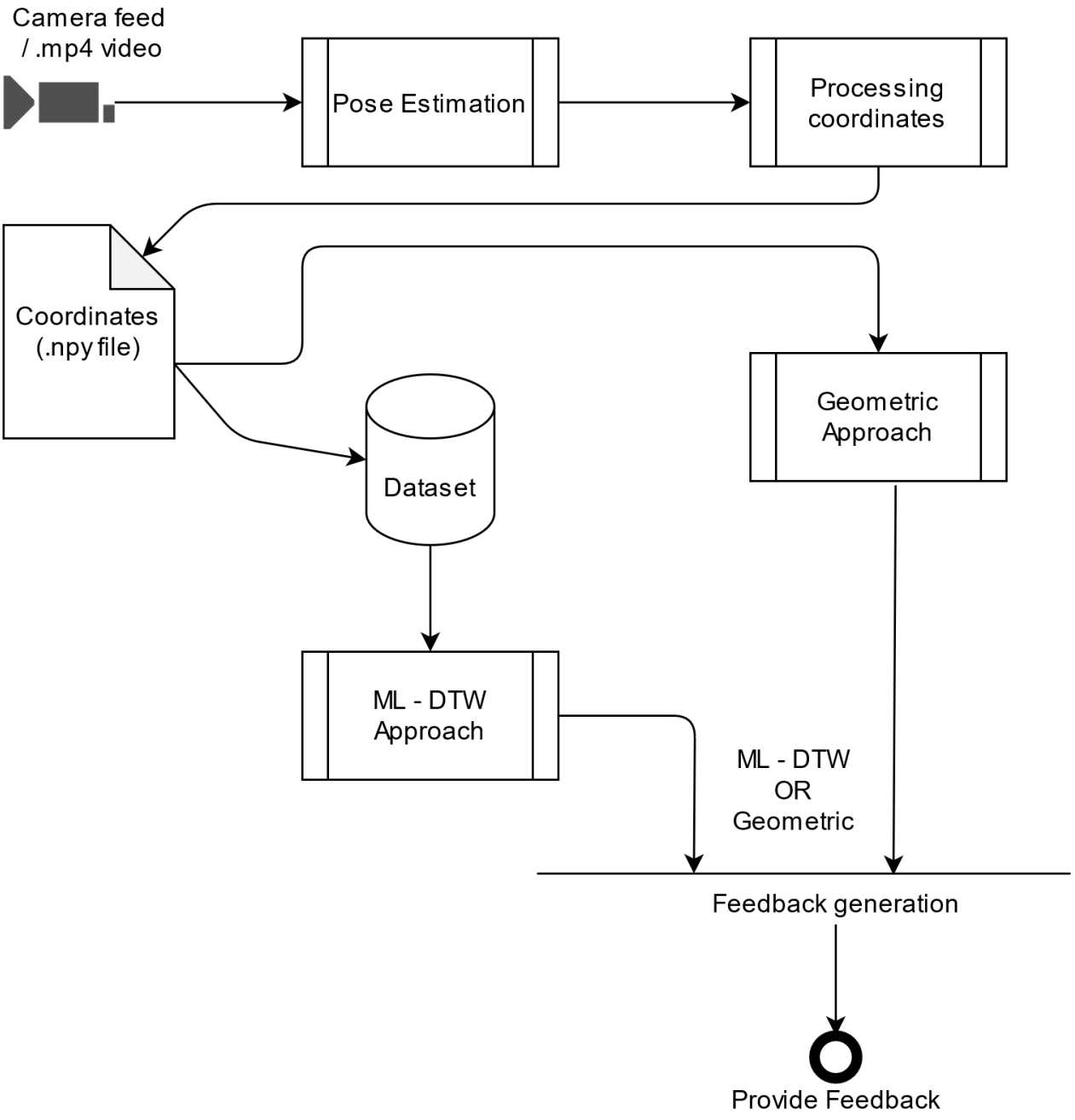

FIGURE 3.2: Proposed Architecture Diagram

3.1 Video Input

Getting the source video is an interesting problem because we need to get just enough of the video stream which is necessary to classify the exercise accordingly,

but does not include unnecessary movement which will derail our algorithms into thinking the user performed the exercise wrong, where in case they might just be adjusting to the initial position.

We have two methods to achieve this; we can have a video prerecorded and trimmed easily using ready to use apps that specialize in the same. We can then use this video feed to make the prediction.

Alternatively, we can have a timed phase in the camera recorder feed itself: The software says a timer (let's say 3..2..1) and the video recording starts, then after a predefined timer (let's say 5 seconds), the recording stops and this whole recorded stream is sent for processing. The user is expected to be in starting position and start the exercise as soon as the initial timer ends and the video recording starts.

3.2 Pose Estimator

The Pose Estimator module is the first major module which makes up the application. The input to this module is a video of the user performing a particular exercise. A pose estimation algorithm is used to get 25 key-points representing every prominent joint of the body. These coordinates are the output of the module, which is then stored as a file on the host system.

In our application, the Pose Estimator plays its role in the dataset preparation and the prediction of the input. The training example videos are given as input to the pose estimation algorithm and the coordinates that are generated by the algorithm are stored on the system disk as a numpy file with the extension ".npy". These

numpy files are important as they help us encode, store and analyze the data as running the pose estimation algorithm is computationally expensive.

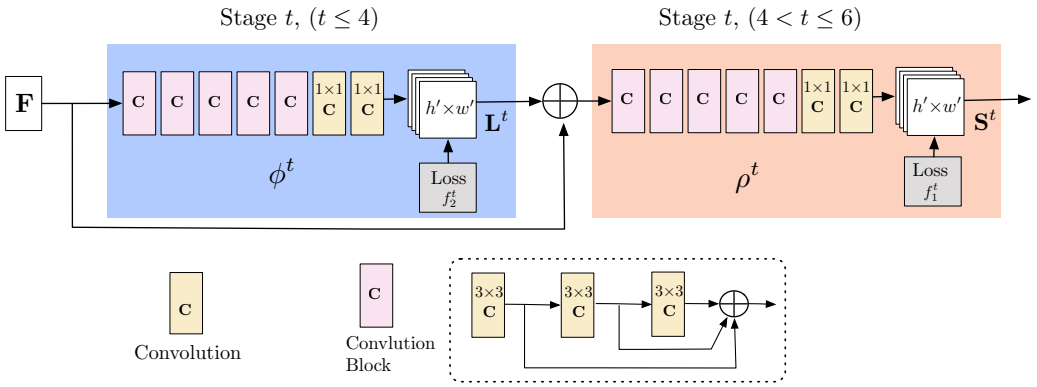

FIGURE 3.3: Architecture of the multi-stage CNN used in OpenPose.

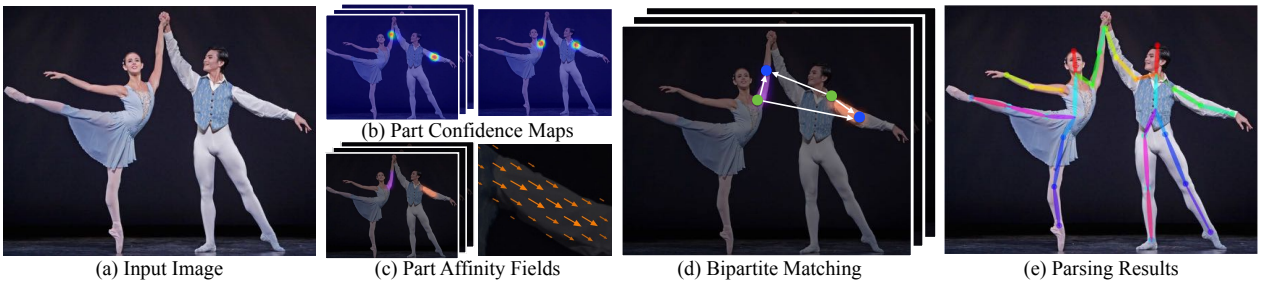

FIGURE 3.4: Illustrates the overall pipeline of OpenPose.

OpenPose[13] is one of the key component of our pose estimation module. OpenPose is a real-time multiperson detection library developed at CMU's Perceptual Computing Lab. It is the state-of-the-art pose estimation algorithm capable of jointly detecting a person's body, face and foot keypoints.

The architecture of the multi-stage CNN used in OpenPose is shown in figure 3.3. The first set of stages predicts Part Affinity Field (PAFs) $\mathbf{L}^t$, while the last set predicts confidence maps $\mathbf{S}^t$.

The system takes, as input, a color image of size $w \times h$ (Fig. 3.4a) and produces the 2D locations of anatomical keypoints for each person in the image (Fig. 3.4e).

First, a feedforward network predicts a set of 2D confidence maps $\mathbf{S}$ of body part locations (Fig. 3.4b) and a set of 2D vector fields $\mathbf{L}$ of part affinity fields (PAFs), which encode the degree of association between parts (Fig. 3.4c). The set $\mathbf{S} = (\mathbf{S}_1, \mathbf{S}_2, ..., \mathbf{S}_J)$ has J confidence maps, one per part, where $\mathbf{S}_j \in R^{w \times h}$, $j \in \{1 \ldots J\}$. The set $\mathbf{L} = (\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_C)$ has C vector fields, one per limb, where $\mathbf{L}_c \in R^{w \times h \times 2}$, $c \in \{1 \ldots C\}$. We refer to part pairs as limbs for clarity, but some pairs are not human limbs (e.g., the face). Each image location in $\mathbf{L}_c$ encodes a 2D vector. Finally, the confidence maps and the PAFs are parsed by greedy inference (Fig. 3.4d) to output the 2D coordinates of the keypoints (body parts) for all the people in the image.

3.3 Processing Coordinates

The implemented algorithm does not work directly onto raw video streams, instead, it expects the video in a numpy array stream format, where each element of the array signifies the position of one of the limbs of human body.

We are using OpenPose, a pose estimation algorithm which performs real time multiperson 2D pose estimation using part affinity fields (a set of 2D vector fields that encode the location and orientation of limbs over the image domain) to estimate the pose and get the key points of the joints involved in the exercise. Figure 3.5 shows the various keypoint movements that we can capture using OpenPose. We use some of these keypoints to find out angles between required body parts, or calculate some other required metrics. We create create a numpy array consisting of all these keypoints for each frame of the video stream. This

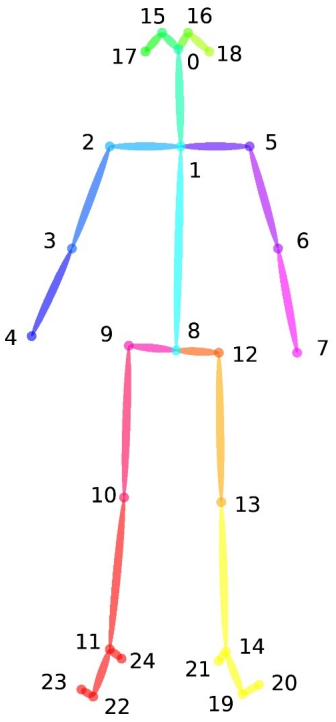

FIGURE 3.5: Possible keypoint mappings

newly generated numpy array is now ready to be passed onto the next phases of our implementation pipeline.

3.4 Feedback Generator

The next major module of the system is the Feedback Generator module. This module takes as input the coordinates file stored on the system by the Pose Estimator module as input. The module analyzes the coordinates and calculates the required values to judge the correctness of an exercise. The feedback is then output to the user.

The Feedback Generator outputs the feedback as a string. First, it indicates the user whether he has performed the exercise correctly. If it is incorrect, a short description is provided about the location of the error and the corrective measure to improve the body posture in that location. For Example If the squat exercise was performed correctly, the feedback would be, ”Exercise Performed Correctly”. If the exercise was performed incorrectly, the feedback would be, ”Exercise performed Incorrectly. You are not bending your back too much, make sure your back is approximately perpendicular to your thighs”.

Using the created numpy stream of data, we have developed two approaches to predict the correctness of an exercise:

3.4.1 Geometric Approach

The Geometric Approach is used as a tool to explore data and get the differentiating features of exercise posture. We require to find out the key features for an exercise before we can continue coding them in the geometric approach. We consult online resources to find out what are the key movements to perform the exercise in focus. Then we narrow down the body parts for which we need to calculate the movements for.
For squats we have identified two key features:

- The upper leg lower leg angle (the angle formed by thigh and lower leg below knees)
- The upper leg torso angle (the angle between thigh and spine)

For Bicep curl we have identified the two key features:

- The upper arm torso angle (the angle formed by elbow to shoulder and the spine)
- The upper arm forearm angle (the angle formed by shoulder to elbow and elbow to palm)

We plot critical angles involved in performing an exercise, analyse those angles to gain some insights about the optimal range of movement in a correct and incorrect exercise. These optimal range of angles is packaged into a set of rules, similar to a decision tree, with the appropriate feedback mapped to each condition. These values are measured by plotting and analyzing various examples of the training data and determining the correct range of the angles. The critical features are also used in the ML-DTW approach to automatically classify exercise posture without any manual analysis of the same.

We do empirical analysis to approximate a range of angles for using it as a threshold for the features. For the squat exercise, the Figure 3.6 shows the distribution of measured angles for each frame of an example video. Here, it can be observed that if the range of the angle between the upper leg and torso is less than 110 degrees, the exercise can be classified as good. If it is greater than 110 degrees, the exercise is classified as bad, and feedback provided would be, ”You are bending forward too much”. Similarly, if the minimum angle between the upper leg and lower leg is greater than 37 degrees, the exercise is classified as bad, with the feedback being, ”You are bending your knees too much”. If both the values are within the limits, the exercise is classified as good, with the feedback being, ”Exercise performed correctly”.

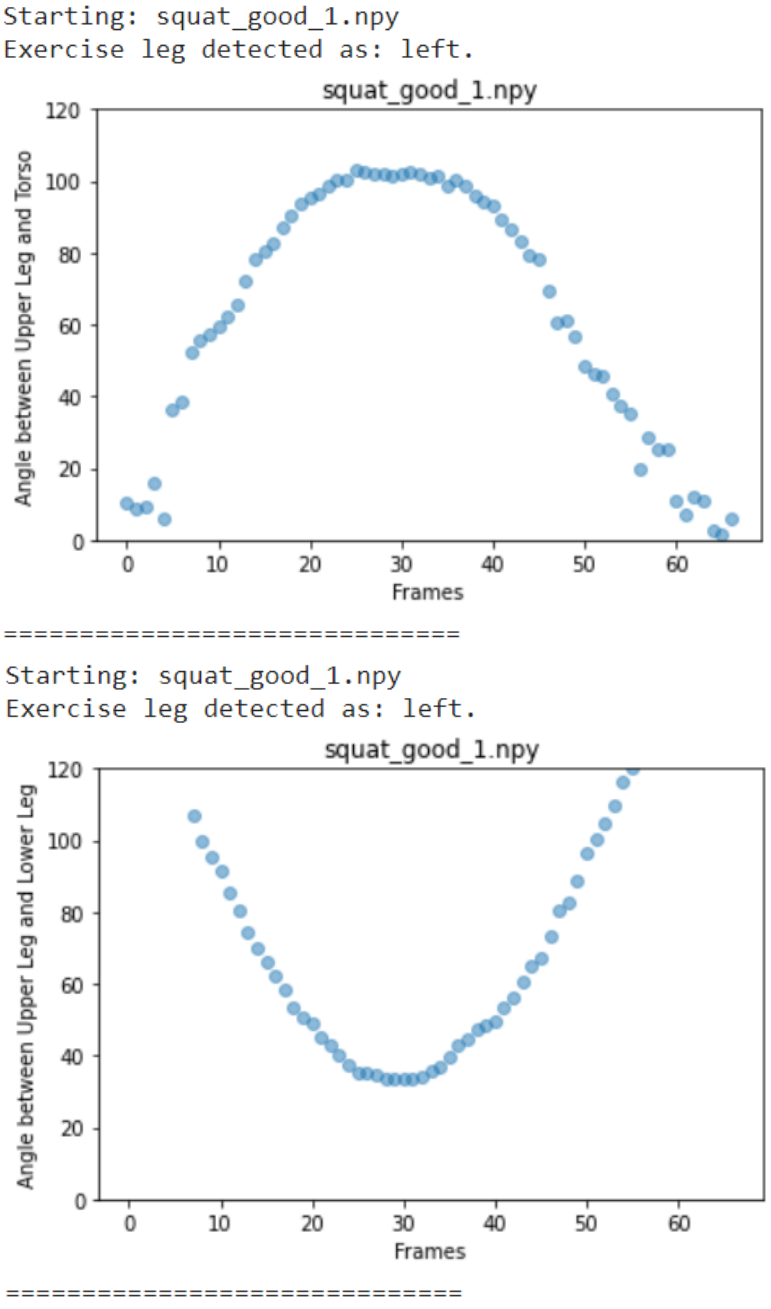

FIGURE 3.6: Squat Analysis

For the bicep-curl (Figure 3.7) exercise, a similar analysis is performed. If the range of the angle between the upper arm and torso is more than 12 degrees, the exercise is classified as bad, with the feedback being, ”You are moving your upper arm forward too much”. If the minimum angle between the upper arm and

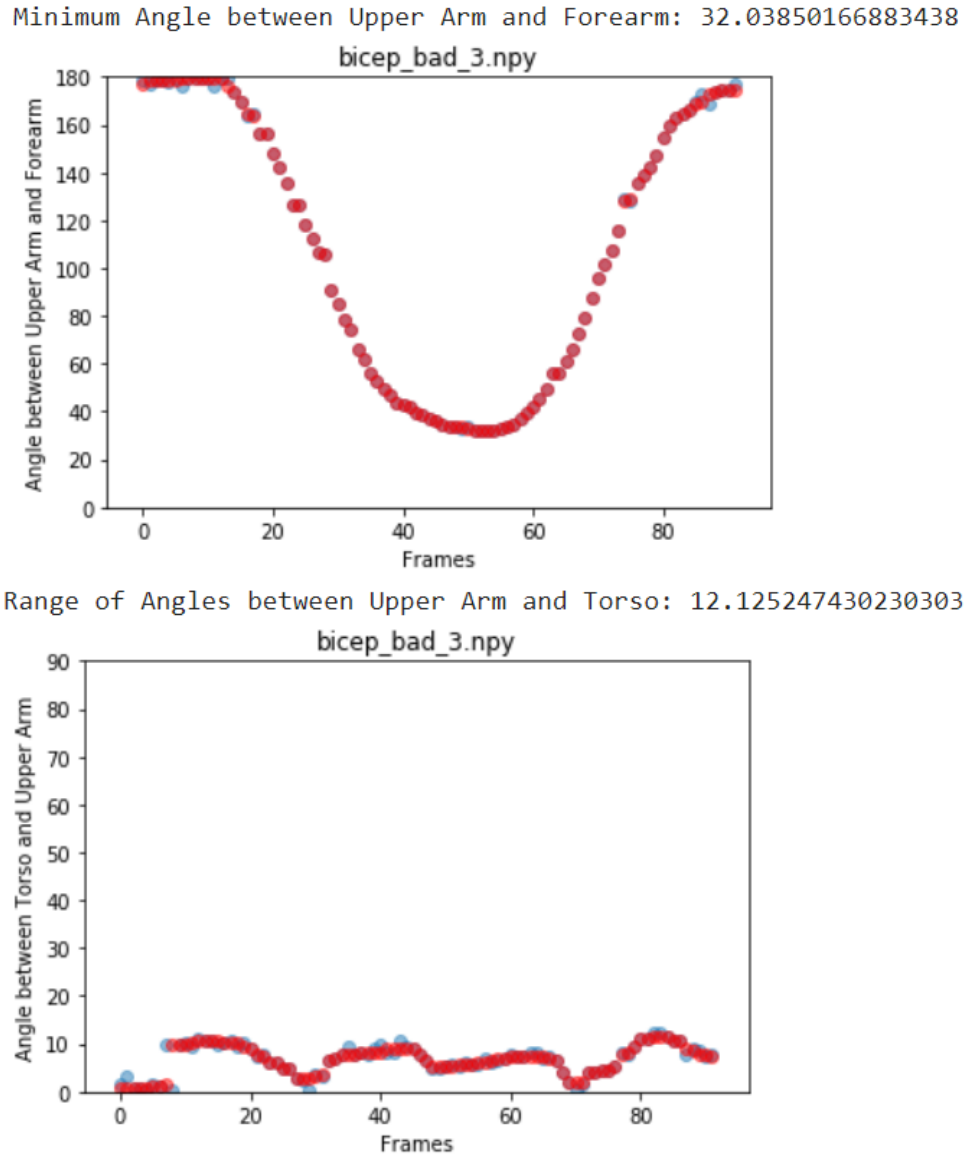

FIGURE 3.7: Bicep Curl Analysis

lower arm is less than 32 degrees, the exercise is classified as bad, with the feedback being, ”You are bending your arms too much”. If both the angles are within limits, the exercise is performed correctly, and the feedback is ”Exercise performed correctly”.

3.4.2 ML-DTW Approach

We create a dataset of correct and incorrect exercises and compare a newly coming stream to these baselines, we calculate the deviation in performing the exercise using Dynamic Time Warping Algorithm and classify it as 'Bad' if the incoming sample is closer to the wrong examples, 'Good' if the sample is closer to the good examples. If classified as 'Bad', then we find the distance of each feature (for example - upper leg lower leg angle in case of squat) from good and bad examples. If a feature is closer to the bad examples we give feedback to the user to improve that feature of the exercise. This process is done for all features for an exercise. The feedback for all features are accumulated and reported to the user.

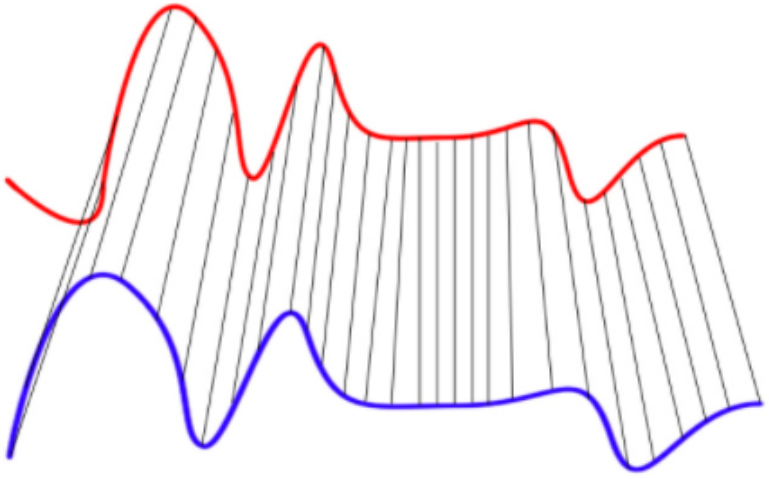

FIGURE 3.8: Dynamic Time Warping Matching [16]

3.4.3 Dynamic Time Warping algorithm

Dynamic time warping [14] is a pattern detection algorithm used mainly in the fields of speech recognition. We take the stream of continuously monitored limb

angles and related positions as a time series data; and using dynamic time warping we match it with one of the standard template to detect what exercise is being performed.

The DTW is specifically used here as we are matching an incoming video stream to an existing standard template videos. The pace in which each user performs the exercise may vary. This difference of pace should not impact the feedback provided. Using OpenPose, we are able to get a stream of keypoint coordinates for an exercise video. The objective is to match this stream of data, which will resemble a specific pattern to the standard template pattern. DTW provides an optimal match for faster and slower variations of the same exercise, if one visualizes the process, it is sort of warping a pattern over a standard pattern by either stretching or compressing parts of the keypoint stream patterns, while also checking if the patterns are actually compatible with each other, as two different patterns should not be warped together.

The complexity of computing DTW is $O(m*n)$ where m and n represent the length of each sequence.

3.4.4 Modified KNN Algorithm for Classification

The k-nearest-neighbour[15] (KNN) Algorithm is a supervised algorithm that falls in to the category of lazy learner, i.e. the algorithm does not have a distinct training phase. It doesn't learn the values of certain parameters of a learning function from the training data but "memorizes" the training dataset instead.

Algorithm 1 DTW Algorithm

```
1: procedure DTWDISTANCE(s : array[1..n], b : array[1..m])
2:     DTW ← array[0..n, 0..m]
3:     for i ← 0 to n do
4:         for j ← 0 to m do
5:             DTW[i, j] ← infinity
6:         end for
7:     end for
8:     DTW[0,0] ← 0
9:     for i ← 1 to n do
10:        for j ← 1 to m do
11:            cost ← d(s[i],t[j])    ▷ d is the distance metric, euclidean distance
12:            DTW[i, j] ← cost + min(DTW[i, j − 1],DTW[i − 1, j − 1],DTW[i − 1, j])
13:        end for
14:    end for
15:    return DTW[n,m]
16: end procedure
```

The algorithm's basic workflow is as such. It gets an input which it has to classify, and it compares it with the elements of the training data, to calculate the similarity of the input with the elements of the training data. This comparison is done with a function called as a distance metric, which takes in two values as input and outputs a number, the similarity (or distance) between the two. Based on these similarity numbers, the k 'most nearest'elements are chosen, and the class in which the maximum number of elements belong to, and is assigned to the given input. For our problem, a very specific version of this KNN algorithm is adopted owing to the specific nature of the problem. We use the Modified KNN algorithm in our application, which is the KNN algorithm with certain changes. The modifications performed are as follows.

- The distance metric used in the modified algorithm is the Dynamic Time

Warping distance. The standard KNN algorithm in the sklearn library cannot be directly used as the training data we use does not have same dimensions.

- The value of k is set to the size of the training data. This is essential because all of the training data should be considered for making a decision. Also, the size of the training data is relatively smaller, hence comparison with all the elements of the training data is computationally not strenuous.

Algorithm 2 Modified KNN Algorithm

```
1:  procedure KNNPREDICT(Xtrain : array[1..n, 1..m], Ytrain : array[1..n], y :
    array[1..m])
2:      good ← array[]                          ▷ Stores distances to good examples
3:      bad ← array[]                           ▷ Stores distances to bad examples
4:      for i ← 1 to n do
5:          dist ← DTWDistance(Xtrain[i], y)
6:          if Ytrain[i] = 1 then
7:              append dist to good
8:          else
9:              append dist to bad
10:         end if
11:     end for
12:     goodScore ← mean(good)
13:     badScore ← mean(bad)
14:     prediction ← 0
15:     if goodScore ≤ badScore then
16:         prediction ← 1
17:     end if
18:     return prediction                       ▷ 1: Correct, 0: Incorect
19: end procedure
```

CHAPTER 4

EXPERIMENTAL SETUP

4.1 Collecting the Dataset

In order to evaluate the exercises performed by the user and provide relevant feedback, we collect a set of video examples from which the required features would be extracted. One approach we tried was using our personal camcorders and tripod stands. The camcorder could be easily adjusted using the tripod stand. The tripod stand would give us a proper angle to record our videos and would not cause unwanted movements in the video. However, this approach came with certain disadvantages. First, the camcorders we had did not store the video in the proper format. This meant converting the format into the required .mp4 format which is acceptable to the pose estimation algorithm. Second, the camcorders we had were not at par with the cameras present in the average smartphone or webcam, which we believe will be the device mainly used by the user to record his exercise video. Hence, we decided to use our personal smartphones to record the videos for our dataset.

The videos were recorded by us keeping in mind the devices and surroundings of the user. We used our personal smartphones placed on a small table supported by items like books. We used different backgrounds to provide variety to the dataset. Also, the dataset does not comprise of videos of only a single person, as there might be different people with different characteristics using this application.

A custom dataset (Figure 4.1) consisting of 100 good and bad videos for each exercise is used. The video samples are varied in terms of lighting conditions, background, the perspective of the exercise performed. Each video consists of multiple iterations of an exercise. The videos are taken on a generic smartphone so that it will closely resemble a standard laptop webcam or smartphone camera when put to use in the future. The dataset currently follows this structure for two exercises, namely squats and bicep curl.

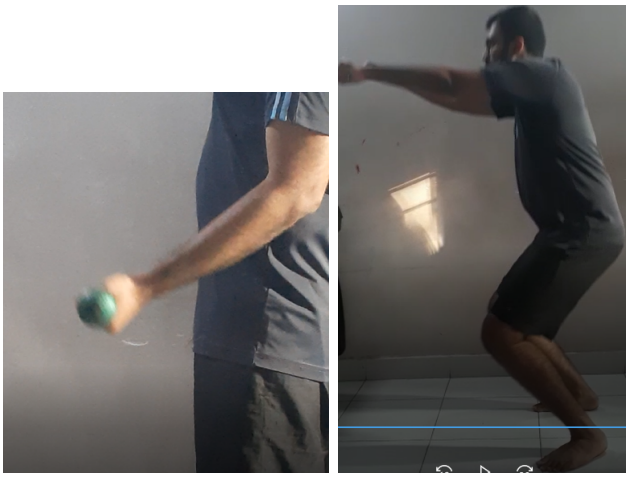

FIGURE 4.1: Bicep Curl and Squat Samples

4.2 Installing OpenPose And Processing Video

OpenPose is the pose estimation algorithm which is used in our application to extract the coordinates of the joints from the videos present in the dataset and also the user's input. OpenPose is to be executed as a binary file (.bin) on Linux systems or as an executable file (.exe) on Windows systems. In order to generate these files, the OpenPose source code, which is available on GitHub, is to be built

using cmake. Building the OpenPose executable file is time consuming, and hence cannot be repeated often. Hence, it must be built once and run many times.

We reduced our processing times significantly by using NVIDIA GPUs for running OpenPose. OpenPose predicts the keypoints of the joints of the body and outputs the coordinates. A visual representation of the coordinates, shown in figure 4.2 is also obtained as output, but is not of much help as it is the coordinates that we require. We leveraged Google Colab's free tier GPUs for generating numpy array (.npy) files of each exercise. These numpy array files are used to analyse posture and give feedback.

The output of OpenPose is a set of coordinates of all the joints of the body for every frame. These coordinates are output as a collection of .json files, one file for each frame. A 3-second video would contain around 180 .json files, each having the coordinates of 25 keypoints of the human body. The content of a single .json file is displayed below

Label 1 is a sample JSON file for one frame. The decimal numbers after the pose_keypoints_2d are basically triplets, the x-coordinate, y-coordinate and the confidence respectively. The confidence metric is the certainty with which OpenPose can determine that the predicted coordinate is true. These JSON files are parsed and the required coordinates of the joints are extracted. The confidence value is also used in determining whether the coordinate exists or not.

The parsed coordinates are stored in a numpy array. This numpy array is stored on disk by using the numpy.save( ) function, which is stored on disk as a binary .npy file. The .npy file can be loaded back using the numpy.load( ) command. A binary file cannot be read using any text editor.

Listing 1 JSON output for one squat video frame


```
{
  "version": 1.3,
  "people": [
    {
      "person_id": [-1],
      "pose_keypoints_2d":[
        285.99, 542.126, 0.716152, 415.491, 424.362,
        0.362589, 515.589, 477.31, 0.273697, 533.205,
        642.211, 0.903962, 491.984, 806.975, 0.848426,
        377.269, 430.237, 0.447373, 377.234, 698.011,
        0.775014, 359.484, 951.187, 0.687087, 839.299,
        430.238, 0.160332, 833.444, 430.221, 0.0968932,
        1068.92, 441.977, 0.154369, 1186.66, 456.733,
        0.122549, 845.238, 427.233, 0.155467, 1074.78,
        468.506, 0.11419, 1195.51, 456.634, 0.211489,
        0, 0, 0, 268.3, 524.447, 0.855203, 0, 0, 0,
        274.293, 456.634, 0.886998, 0, 0, 0, 0, 0, 0,
        1201.31, 459.617, 0.135234, 0, 0, 0, 0, 0, 0,
        0, 0, 0
      ],
      "face_keypoints_2d": [],
      "hand_left_keypoints_2d": [],
      "hand_right_keypoints_2d": [],
      "pose_keypoints_3d": [],
      "face_keypoints_3d": [],
      "hand_left_keypoints_3d": [],
      "hand_right_keypoints_3d": []
    }
  ]
}
```

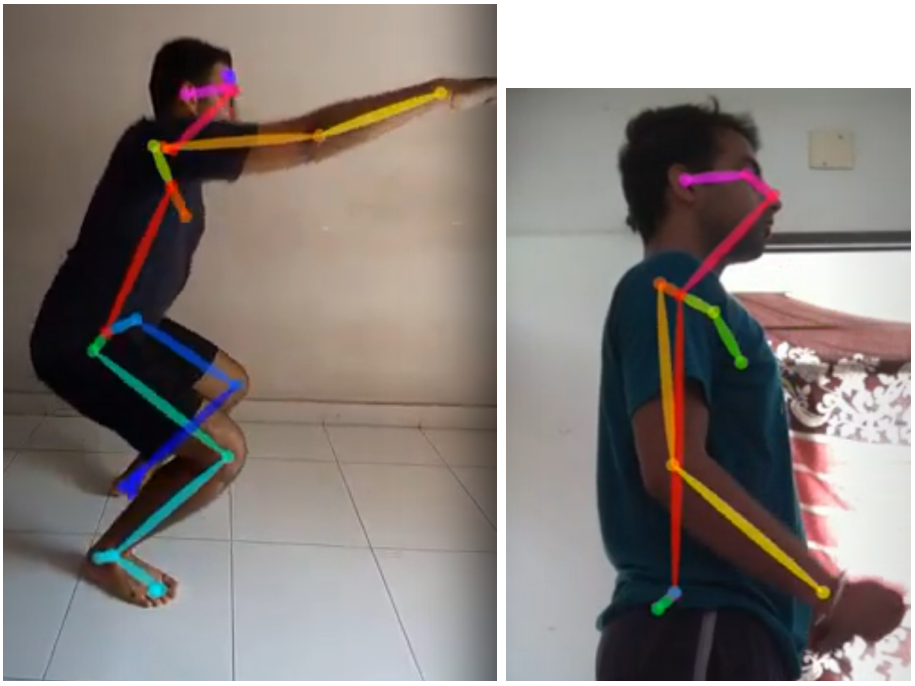

FIGURE 4.2: Sample of videos processed by Openpose

4.3 Application Interface

We create a chat bot frontend which interacts with the user and interfaces with the pose estimation algorithm backend. The user interface is designed and coded in tKinter, which is a Python binding to the Tk GUI toolkit.

The user is expected to choose one of the supported exercises, after which the bot prompts the user to submit one video record a live video for preprocessing; or submit a preprocessed video file for prediction.

After a successful candidate video is submitted from the user, the algorithm executes the prediction pipeline and resulting feedback is projected to the user via the chat-bot itself.

The UI screen shown in the Figure 4.3 depicts the basic chat-bot look and feel of the application. This type of interface has been chosen as it is very interactive and

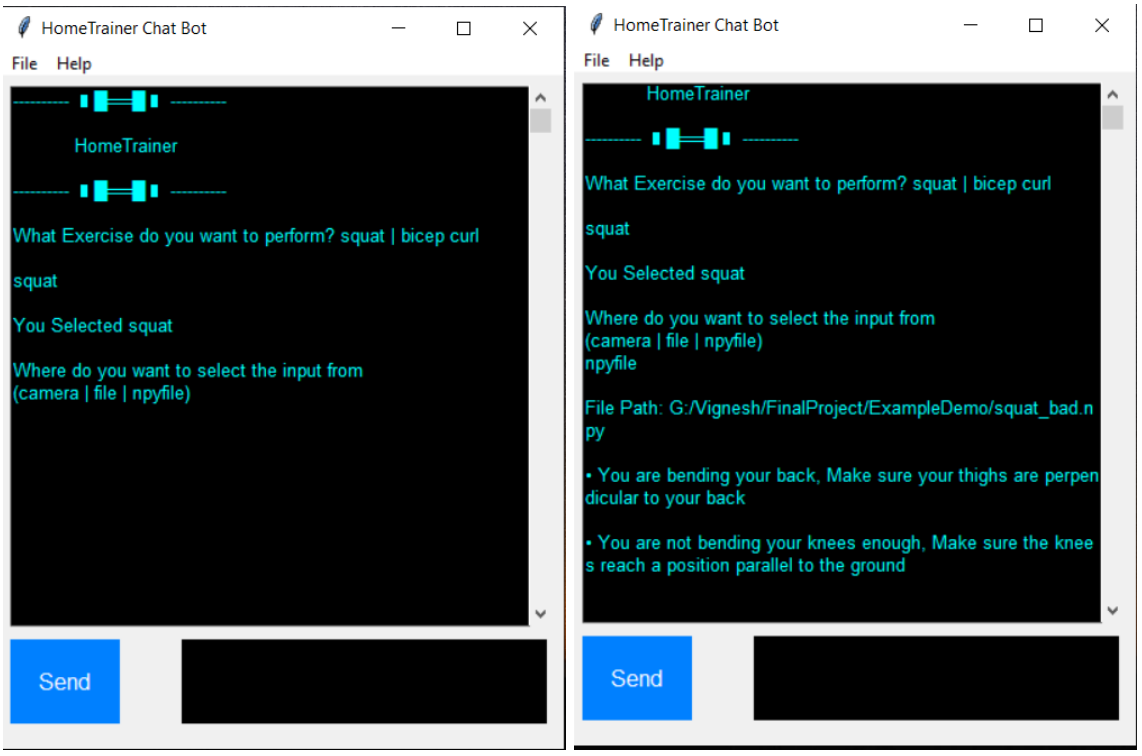

FIGURE 4.3: Sample UI Windows

abstracts the inner workings of the application from the user.

The user has to type the name of the exercise, then select the options for uploading either the .npy file, video file, or record a new video directly from the camera present in the system.

The feedback is displayed in the same chat window and the user can restart their session to try again, or can choose to input another file for the selected exercise. The UI is simple and focused so that it helps the user to concentrate on the task of improving his exercise posture, and reduces the number of packages and dependencies to install on a desktop system.

CHAPTER 5

PERFORMANCE ANALYSIS

5.1 Performance metrics used

We used two metrics namely precision and recall to measure our implementation. The formulae 5.1 for precision and recall are shown below.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Where, TP = True Positives,

FP = False Positives,

TN = True Negatives,

FN = False Negatives.

- A True Positive is an outcome where the positive class is predicted correctly.

- A True Negative is an outcome where the negative class is predicted correctly.
- A False Positive is an outcome where the positive class is predicted incorrectly.
- A False Negative is an outcome where the negative class is predicted incorrectly.

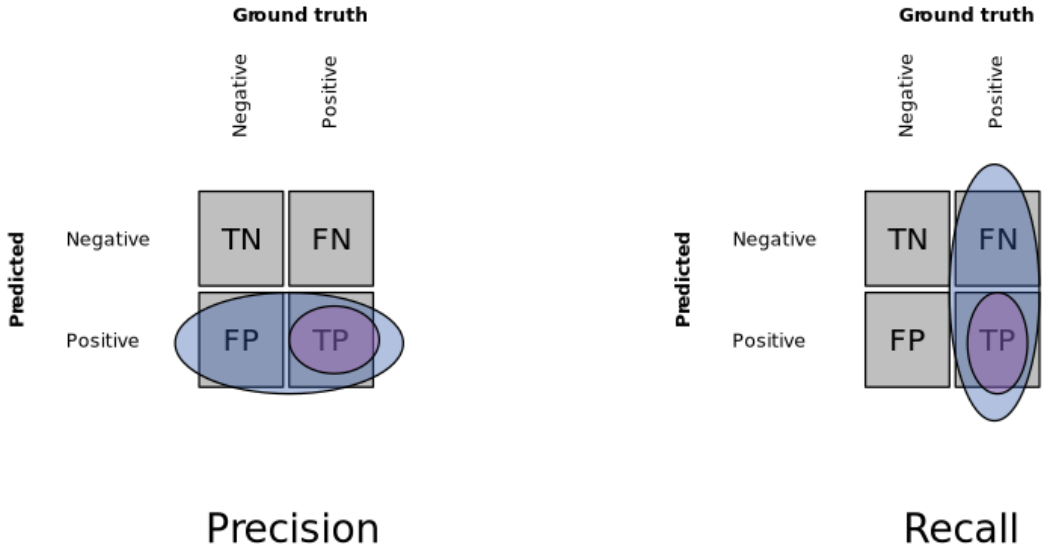

FIGURE 5.1: Precision and Recall

5.2 Results obtained

5.2.1 Geometric Approach

Geometric approach is the baseline for classifying exercises into correct or incorrect posture by observing critical angles involved in the exercise.This helps us label the dataset for the Machine Learning Approach. We provide two cases each for squat and bicep curl, one with correct posture and other with incorrect posture.

- **Case 1** : The subject maneuvered their upper and lower legs in a range of angles from 47 to 109 degrees while performing Squat. The tolerance range of a prefect squat is from 35 to 120 degrees which is 85 degrees. The user is moving lesser than the requirement for a successful squat, we conclude the subject performed the exercise incorrectly. Moreover, we provide specific feedback as to how to improve the exercise. In this case ”You are not bending knees enough, make sure you bend enough so that your thighs are parallel to the ground” is returned to the user.

- **Case 2** : The subject maneuvered their upper and lower legs in a range of angles 30 to 110 degrees while performing Squat. Since these angle ranges are very close to the tolerance range i.e. 35 to 120 degrees, we conclude the subject performed the exercise correctly. Hence, ”Exercise Performed Correctly” is returned to the user.

- **Case 3** : The minimum angle between upper arm and forearm of the subject is 35 degrees. The minimum angle between a upper arm and forearm for a

perfect Bicep Curl is approximately 40 degrees. Since the minimum angle deviation for this feature is close to the minimum angle range of perfect Bicep Curl, we conclude the exercise is performed correctly. Hence, ”Exercise Performed Correctly” is returned to the user.

- **Case 4** : The minimum angle between upper arm and forearm of the subject is 98 degrees. Since the minimum angle deviation for this feature is very high compared to the minimum angle range of perfect Bicep Curl, we conclude the exercise is performed incorrectly. In this case ”You are not curling enough, you should start parallel to the legs and curl till your fists reach close to your shoulder” is returned as feedback.

5.2.2 ML-DTW Approach (Using KNN)

Geometric approach provides us with accurate results but with one caveat: we need to feed the rules explicitly using the optimal range of angles. We try to remove this requirement in this approach by building a KNN classifier for detecting whether the user has performed an exercise correctly or incorrectly.
The classifier predicts positive, if the exercise is performed with correct posture, negative if it is performed with incorrect posture. The Confusion Matrix for Squat exercise is shown in Table 5.1

		Predicted		
		Positive	Negative	Total
Actual	Positive	21	0	21
	Negative	0	21	21
	Total	21	21	42

TABLE 5.1: Squat Confusion Matrix

- The Precision is **1.0**
- The Recall is **1.0**
- The F1-Score is **1.0**

The Confusion Matrix for Bicep Curl exercise is shown in Table 5.2

		Predicted		
		Positive	Negative	Total
Actual	Positive	10	2	12
	Negative	6	14	20
	Total	16	16	32

TABLE 5.2: Bicep Curl Confusion Matrix

- The Precision is **0.63**
- The Recall is **0.83**
- The F1-Score is **0.72**

From the results obtained we infer that, the squat has a perfect F1-Score of **1.0**. This is due to the following reasons:

- The squat focuses only on the lower part of the body.

- There are not a lot of variations in which a squat can be performed in a wrong way.
- The features that we have identified are enough to perfectly identify the exercises with correct and incorrect posture.

For the bicep curl exercise, the F1-Score is **0.72**.
The score is lower compared to the squat due to the following reasons:

- There are multiple variations of performing the bicep curl exercise.
- We have considered two key features of upper arm forearm angle, upper arm torso angle. There are more features that could affect the correctness of this exercise.

From the results obtained, it can be inferred that the application produces satisfactory predictions for a given input and provides constructive feedback to the user to correct his/her exercise. We treat the results obtained from Geometric Approach as absolute because they are based on deterministic rules that have been hard-coded for the specific exercises, it acts as a feature engineering tool which we use to label the exercise videos for generating the confusion matrix while determining the accuracy of ML-DTW Approach.

These labelled videos are used in the ML-DTW approach to automatically classify exercise posture and provide feedback.

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this project, we present Home Trainer, a web application that provides feedback for the exercise performed by the user by utilizing machine learning and geometric analysis. A state-of-the-art pose estimation algorithm is used to identify the required coordinates of the body parts, which are used in the evaluation of the exercise performed. Two approaches - a geometric approach and a machine learning approach have been proposed by us. We formulate these two approaches for two exercises namely squats and bicep curls.

There are many ways in which future work is possible from this current project work. One such area is real time pose estimation and feedback. Providing feedback to the user at real time will increase the interactivity as the user can instantly correct his actions while he is performing the exercises. Another area of development is to port the application into the mobile field by modifying OpenPose, which can only run in a Windows or Linux environment. Finally, The application can also be made into an online application with OpenPose running in a cloud environment.

REFERENCES

1. Sheri R Colberg, Ronald J Sigal, Jane E Yardley, Michael C Riddell, David W Dunstan, Paddy C Dempsey, Edward S Horton, Kristin Castorino, and Deborah F Tate. 2016. 'Physical activity/exercise and diabetes: a position statement of the American Diabetes Association', Diabetes Care 39, 11 (2016), 2065–2079

2. Ian Janssen and Allana G LeBlanc. 2010. 'Systematic review of the health benefits of physical activity and fitness in school-aged children and youth', International journal of behavioral nutrition and physical activity 7, 1 (2010), 40.

3. Arnold Baca, 'Methods for Recognition and Classification of Human Motion Patterns – A Prerequisite for Intelligent Devices Assisting in Sports Activities', University of Vienna, In Proceedings of 7th Vienna International Conference on Mathematical Modelling, 21 April 2016.

4. Mustafa Acikkar, Mehmet Fatih Akay, Kerem Tuncay Ozgunen, Kadir Aydin, Sanli Sadi Kurdak, 'Support vector machines for aerobic fitness prediction of athletes', In Proceedings of Expert Systems with Applications: An International Journal, March 2009.

5. Goran Šeketa, Dominik Džaja, Sara Žulj, Luka Celić, Igor Lacković, and Ratko Magjarević. 2015. 'Real-Time Evaluation of Repetitive Physical Exercise Using Orientation Estimation from Inertial and Magnetic Sensors', In First European Biomedical Engineering Conference for Young Investigators. Springer, 11–15.

6. Keng-Hao Chang, Mike Y Chen, and John Canny. 2007. ‘Tracking free-weight exercises’, In International Conference on Ubiquitous Computing. Springer, 19–37.

7. Christian Seeger, Alejandro Buchmann, and Kristof Van Laerhoven. 2011. ‘Adaptive gym exercise counting for myHealthAssistant’, In Proceedings of the 6th International Conference on Body Area Networks. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 126–127.

8. C Crema, A Depari, A Flammini, E Sisinni, T Haslwanter, and S Salzmann. 2017. ‘IMU-based solution for automatic detection and classification of exercises in the fitness scenario’, In Sensors Applications Symposium (SAS), 2017 IEEE. IEEE, 1–6.

9. Dan Morris, T Scott Saponas, Andrew Guillory, and Ilya Kelner. 2014. ‘RecoFit: using a wearable sensor to find, recognize, and count repetitive exercises’, In Proceedings of the 32nd annual ACM conference on Human factors in computing systems. ACM, 3225–3234.

10. D Antón, A Goñi, A Illarramendi, et al. 2015. ‘Exercise recognition for Kinect-based telerehabilitation’, Methods Inf Med 54, 2 (2015).

11. Ilktan Ar and Yusuf Sinan Akgul. 2014. ‘A computerized recognition system for the home-based physiotherapy exercises using an RGBD camera.’, IEEE Transactions on Neural Systems and Rehabilitation Engineering 22, 6 (2014), 1160–1171.

12. Rushil Khurana, Karan Ahuja, Zac Yu, Jennifer Mankoff, Chris Harrison, and Mayank Goel. 2019. ‘GymCam: Detecting, Recognizing and Tracking Simultaneous Exercises in Unconstrained Scenes. ’, Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 2, 4, Article 185 (December 2018), 17 pages.

13. Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh, 'OpenPose: Real time Multi-Person 2D Pose Estimation using Part Affinity Fields', IEEE Transactions on Pattern Analysis and Machine Intelligence, 2019

14. Donald J. Berndt, James Clifford, 'Using Dynamic Time Warping to Find Patterns in Time Series', AAAI Technical Report WS-94-03, 1994

15. Zhang, Zhongheng. 'Introduction to machine learning: k-nearest neighbors', Annals of translational medicine vol. 4,11 (2016): 218.

16. Modified the Euclidean vs DTW image to include only the DTW image, **Source :** `https://commons.wikimedia.org/wiki/File:Euclidean_vs_DTW.jpg`.