Pose Estimation And Gym Tracker Using Movenet Algorithm

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Saif – 19BCE1225

Raghavendra – 19BCE1178

Raghava – 19BCE1716

Submitted to

Dr. Sajidha Syed

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

DECLARATION

This is to declare that this report has been written by us as part of our coursework. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found plagiarized, we shall take full responsibility for it.

19BCE1225 Saif Shaik

19BCE1716 P Raghava Ratna

19BCE1178 K.V Sai Raghavendra

Place: Vellore Institute of Technology, Chennai

Date: 28th April 2022

CERTIFICATE

Certified that this project report entitled "Pose Estimation Using Machine Learning" is a bonafide work of **K V Sai Raghavendra (19BCE1178), P Raghava Ratna (19BCE1716), Saif (19BCE1225)** and they carried out the Project work under my supervision and guidance for CSE1901 - Technical Answers to Real World Problems (TARP).

19BCE1225 Saif Shaik

19BCE1716 P Raghava Ratna

19BCE1178 K.V Sai Raghavendra

ACKNOWLEDGMENT

We would like to express our special thanks to our TARP Professor **Dr. Sajidha syed** who gave us this opportunity to work on this project and expand our knowledge in the field of AI & ML as well as for guiding us on the same.

We are also sincerely grateful to the Vellore Institute of Technology, Chennai for giving us the occasion tolearn in depth the concepts of Machine learning and AI in Computer Science and technical problems solving to enhance our basic fundamentals and grow interest in the area.

A special mention to our friends and families who helped and encouraged us throughout the journey despite the distance barrier and communication discomfort to ideate and move ahead with the project within the time frame.

<u>**TABLE OF CONTENTS**</u>

1.INTRODUCTION

1.1 MOTIVATION

Now a days fitness is the most important part of the life to survive and this pandemic teached us how much important is nature and our body exercise.

In our day to day lives we spend more than 7 hours on our smart gadgets, they have become a big part of our daily lives. This reality is what is changing the way we use screens and we are getting attracted towards the screen and are not physically fit

The existing system of fitness app is it just shows the exercise and plan for it but we can modify it by or modifying it personal AI trainer where it detects our body posture and count the number of push-ups , jumping …etc you have done and keep a track and observe it.

.

1.2 WHY DID WE CHOOSE THIS PROJECT?

This system is mainly developed to properly know which is very handily and also detects the imaginary data and also various objects. And

- First we will gather data on the user’s movements while they perform the exercise.
- Next, determine how correct or incorrect the user’s movements were.

Finally, show the user via the interface what mistakes they may have made and keep a track on it so that they can be fit both mentally and physically

2.ABSTRACT

Human Pose Estimation is a technique to identify the poses of the body parts and joints in humans. It is a way to capture the set of coordinates by defining the human body joint points like eyes, ears, arms, et c. Based on these key points we can compare various movements and draw insights. Human pose estimation is important because It can be used in real-time scenarios like social distancing detectors where we can combine the human poses of each individual person and the distance between each individual. Another significant area where pose estimation plays a vital role is autonomous driving, with the help of human pose detection, the computer can sense humans and avoid accidents.

In this project, we are going to estimate the human pose estimation with MoveNet Model which takes the video as input and detects the poses, key points, and key point confidence sore of each part of the human body. It detects the 17 key points of a Human body. Then we will build a gym tracker which will detect the number of push-ups done, jumping and raising the Dumble, etc.

3.LITERATURE SURVEY

Author name	Dataset	Metrics and model	Accuracy
Grégory Rogez · Jonathan Rihan · Carlos Orrite-Uruñuela · Philip H.S. Torr Fast Human Pose Detection Using Randomized Hierarchical Cascades of Rejectors	MOBO walking dataset the database contains 25 individuals walking on a treadmill in the CMU 3D room. The subjects perform four different walk patterns: slow walk, fast walk, incline walk and walking with a ball.	Bayesian formulation to compute the log-likelihood ratios which can be used to determine the importance of different regions in the image	HOG feature boxes are rescaled in proportion to the new classifier window scale. The sampling coordinates for the HOG boxes can be rescaled at no extra cost in computation time and the proper pose is estimated
Baole Ai, Yu Zhou, Yao Yu, Sidan Du Nanjing University Human Pose Estimation using Deep Structure Guided Learning	The Frames Labeled In Cinema (FLIC) dataset and the Leeds Sports Dataset (LSP). FLIC dataset contains 5003 images obtained from popular Hollywood movies. 3987 images are used for training dataset and 1016 images	predict the joints heat-maps, we use a 2D Gaussian distribution Next its carried out by Network architecture Then limb loss (Amputation is the loss or removal of a body part such as a finger,toe,hand,foot,arm or leg) and evaluation is done	convolutional neural network learning for human pose estimation. CNN architecture is efficient and has holistic view on the input image. It captures the contextual information of the whole image and it shows more accuracy

	for test dataset		
Gregory Rogez ´ 1, Jonathan Rihan2, Srikumar Ramalingam2, Carlos Orrite1 and Philip H.S. Torr2 Randomized trees for human pose detection	HumanEva-I dataset contains 7 calibrated video sequences that are synchronized with 3D body poses obtained from a motion capture system. The database contains 4 subjects performing a 6 common actions (e.g. walking, jogging, gesturing, etc.).	We combine ideas from hierarchical clustering then we build hierarchical decision tress then we learn a series of regressors to estimate the 3D pose	human pose detection and recognition using randomized trees. Unlike most previous works, our pose detection algorithm is applicable to more challenging scenarios involving extensive viewpoint changes and moving camera. Moreover, our random forest classifier allows to model distribution over pose
Qi Dang; Jianqin Yin; Bin Wang; Wenqing Zheng. 2D human pose using deep learning	The data is extracted almost from all the sources such as youtube, images from internet, FLICKER and Hollywood	Uses deep learning method. DCNN Percentage of Correctly estimated body Parts (PCP) which evaluates stick predictions. Heatmap based model for 2D estimation (heatmap).	2D is very accurate with heatmap model and direct regression. Cannot be applied to 3D model since generative

	movies.		and discriminative models are not very good in this case.
M. Dimitrijevic Human body pose detection using Bayesian spatio-temporal templates	Live data capturing using 6 motion cameras.	During a training phase, we use statistical learning techniques to estimate and store the relevance of the different silhouette parts to the recognition task. Chamfer distance for silhouette containing n points. Penalty term for removing clutter. And a formula to compare chamfer distances.	Approach tested on a specific human pose, is generic and could be applied for any other actions performed in roughly similar ways but with substantial individual variations. This method, with its accurate 3-D pose detections, is a key step towards robust full 3-D body pose tracking algorithms.
Malte Hoffmann , Esra Abaci Turk , Borjan Gagoski , Leah Morgan , Paul Wighton · Rapid head-pose	Live data is used with pregnant woman consent by MRI (Data specific to this work are	rating randomly selected clinical HASTE stacks from our picture. MSER is detected using local search, estimation of slices is done using spatial	Brain-localization accuracy is assessed in terms of the Euclidean distance. The algorithm

detection for automated slice prescription of fetal-brain MRI	acquired using an IRB-approved protocol). A total of *n* = 41 full-uterus stacks of T2*-weighted slices from 18 different fetuses in the third trimester (26–37 weeks GA, mean ± SD 31.0 ± 2.7) with varying oblique and double-oblique FOV relative to the brain are used for evaluation.	regularization. Finally finding the clusters that represent the eyes.	detects brain locations and orientations in agreement with the “ground truth” for 95.1% of EPI stacks from healthy fetuses. For younger fetuses it yields only 18.2%.
A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks,".	Frames Labelled Dataset set and Leeds Pose Dataset	Uses Deep Neural Network (DNN) with Alexnet as base model which is Based on regression approach. Applied cascade regressor approach for better accuracy.	Achieved 69 % accuracy beating the previous state of art models.
Z. Cao, T. Simon, S. Wei and Y. Sheikh, "Realtime Multi-person 2D Pose	MPII Human Pose dataset	Image is passed through a network to extract feature maps and here they used the first 10 layers of the VGG-19 model and then the feature maps are	Achieved 79.7 % mAP beating previous state of art models.

Estimation Using Part Affinity Fields,".		processed through multiple stages of CNN to generate a set of 2D confidence maps and a set of part affinity fields and then uses a bipartite graph matching algorithm for identifying poses.	
A. Singh, S. Agarwal, P. Nagrath, A. Saxena and N. Thakur, "Human Pose Estimation Using Convolutional Neural Networks,".	MPII Human Pose dataset	They aim to identify x-y pixel coordinates for 15 body joint points and aim to label the images using CNN and use the regression approach for human posture estimation. used Backpropagation to optimize weights and perform mini-batch gradient descent on batch of 128.	Cannot able to beat the previous state of art models.

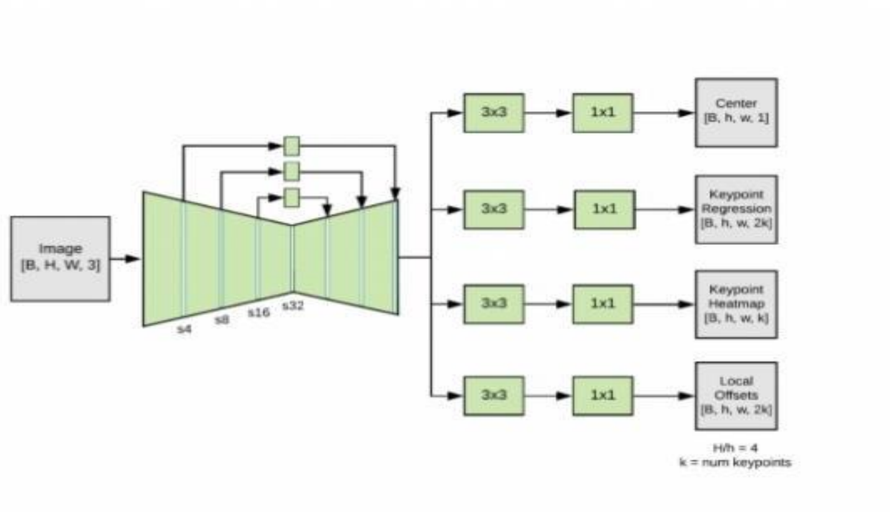
4. Proposed Work

We are going to detect the human pose with movenet algorithm. It is going to detect the 17 key points of the human body. The data is live data which is captured through the camera. The 17 points include the nose, left eye, right eye, left ear, right ear, left shoulder, right shoulder, left elbow, right elbow, left wrist, right wrist, left hip, right hip, left knee, right knee, left ankle, right ankle. All those important parts which can be considered as joints and can be detected when there is motion. The data is live data which is captured through the camera. From the video which is being rendered the video is converted to frames which consist individual images where the movenet algorithm runs and the points are detected. These frame are stored into arrays using interpretar. The Movenet model returns the coordinates of each key point and the confidence score of each key point in the body. According to this score they join the connections between two body parts if their confidence value is greater than the threshold value. First pixel like points is being kept and then those points are joined. A stick like figure will be obtained with the selected color. Then for measuring the number of pushups and raising the dumble we need to extract the angle between the elbow, shoulder and wrist to count the number of push ups and so on.

Movenet Algorithm Architecture: It was trained on COCO dataset.

The architecture consists of two components: a feature extractor and a set of prediction heads. The feature extractor in movenet is MobileNetV2 with an attached feature pyramid network which gives rich feature map output. The prediction head consists of four parts:

Person Center heatmap: predicts the geometric center of person instances



Keypoint regression field: predicts full set of keypoints for a person used for grouping keypoints into instances.

Person keyheatmap: predicts the location of all keypoints independent of person instances.

Offset field: predicts local offsets from each output feature map pixel to the precise sub pixel location of each keypoint.

5.IMPLEMENTATION

In this project, Various data-set including the real time data to estimate and check the accuracy and time constraint of detecting the pose.

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

```
model = tf.lite.Interpreter(model_path='lite-model_movenet_singlepose_lightning_3.tf
model.allocate_tensors()
```

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    cv2.imshow('Video', frame)

    if cv2.waitKey(10) & 0xFF==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

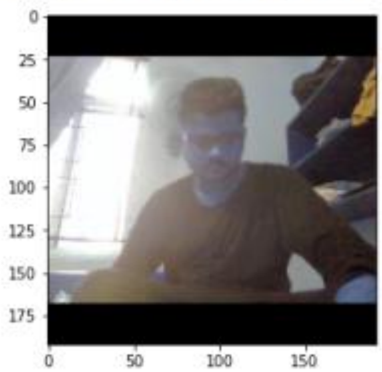
```
counter=0
stage=''
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    image = frame.copy()
    image = tf.image.resize_with_pad(np.expand_dims(image, axis=0), 192,192)
    in_image = tf.cast(image, dtype=tf.float32)

    in_details = model.get_input_details()
    out_details = model.get_output_details()

    model.set_tensor(in_details[0]['index'], np.array(in_image))
    model.invoke()
    keypoints = model.get_tensor(out_details[0]['index'])
    #print(keypoints)

    rightshoulder=keypoints[0][0][6]
    rightshoulder_c=np.array(rightshoulder[:2]*[480,640]).astype(int)
    rightelbow=keypoints[0][0][8]
    rightelbow_c=np.array(rightelbow[:2]*[480,640]).astype(int)
    rightwrist=keypoints[0][0][10]
    rightwrist_c=np.array(rightwrist[:2]*[480,640]).astype(int)


   # if(rightshoulder[2]<0.4 or rightelbow[2]<0.4 or rightwrist[2]<0.4):
    #    print("Your hand is not visible clearly")
    #    continue

    angle=calculate_angle(rightshoulder_c,rightelbow_c,rightwrist_c)
 #   print(rightshoulder_c)
  #  print(rightelbow_c)
  #  print(rightwrist_c)
   # print(angle)


    cv2.putText(frame, str(angle),
```

```
                                tuple(rightelbow_c),
                                cv2.FONT_ITALIC, 2, (125, 242, 245), 1,cv2.LINE_AA)

                # Curl counter logicq
        if angle > 160:
            stage = "down"
        if angle < 30 and stage =='down':
            stage="up"
            counter +=1
            print("No.of curls are",counter)



        cv2.rectangle(frame, (0,0), (225,73), (245,117,16), -1)

            # Rep data
        cv2.putText(frame, 'REPS', (15,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
        cv2.putText(frame, str(counter),
                        (10,60),
                        cv2.FONT_ITALIC, 2, (0,255,255), 2, cv2.LINE_AA)

            # Stage data
        cv2.putText(frame, 'Stage', (65,12),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
        cv2.putText(frame, stage,
                        (60,60),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

        draw_connections(frame, keypoints, EDGES, 0.2)
        draw_keypoints(frame, keypoints, 0.2)

        cv2.imshow('Video', frame)

        if cv2.waitKey(1) & 0xFF==ord('q'):
            break

cap.release()
cv2.destroyAllWindows()
```

```
No.of curls are 1
No.of curls are 2
```

```
plt.imshow(tf.cast(np.squeeze(image), dtype=tf.int32))
```

```
<matplotlib.image.AxesImage at 0x1a5cb5500a0>
```



```python
img = frame.copy()
```

```python
img.shape
```

```
(480, 640, 3)
```

```python
model.get_input_details()
```

```
[{'name': 'serving_default_input:0',
  'index': 0,
  'shape': array([  1, 192, 192,   3]),
  'shape_signature': array([  1, 192, 192,   3]),
  'dtype': numpy.float32,
  'quantization': (0.0, 0),
  'quantization_parameters': {'scales': array([], dtype=float32),
   'zero_points': array([], dtype=int32),
   'quantized_dimension': 0},
  'sparsity_parameters': {}}]
```

```python
model.get_output_details()
```

```
[{'name': 'StatefulPartitionedCall:0',
  'index': 312,
  'shape': array([ 1,  1, 17,  3]),
  'shape_signature': array([ 1,  1, 17,  3]),
  'dtype': numpy.float32,
  'quantization': (0.0, 0),
  'quantization_parameters': {'scales': array([], dtype=float32),
   'zero_points': array([], dtype=int32),
   'quantized_dimension': 0},
  'sparsity_parameters': {}}]
```

```python
model.get_tensor(model.get_output_details()[0]['index'])
```

```
array([[[[0.33563852, 0.4918412 , 0.6013999 ],
         [0.27699625, 0.53435105, 0.29148144],
         [0.28251183, 0.44304937, 0.4752282 ],
         [0.29927588, 0.5901612 , 0.591939  ],
         [0.29590386, 0.38014507, 0.6551827 ],
         [0.48948607, 0.7017464 , 0.68989825],
         [0.5230725 , 0.27884543, 0.60958445],
         [0.7827147 , 0.8914031 , 0.4876485 ],
         [0.8648485 , 0.24485788, 0.32038832],
         [0.8178784 , 0.8858621 , 0.06766882],
         [0.87350154, 0.28361183, 0.05250168],
         [0.93995637, 0.6879045 , 0.04860159],
         [0.9658277 , 0.40127382, 0.0224943 ],
         [0.84900236, 0.92112637, 0.06929153],
         [0.8356112 , 0.26262954, 0.02063032],
         [0.87454265, 0.9394624 , 0.02284241],
         [0.84174293, 0.02700099, 0.02180329]]]], dtype=float32)
```

```python
keypoints[0][0]
```

```
array([[0.33563852, 0.4918412 , 0.6013999 ],
       [0.27699625, 0.53435105, 0.29148144],
       [0.28251183, 0.44304937, 0.4752282 ],
       [0.29927588, 0.5901612 , 0.591939  ],
       [0.29590386, 0.38014507, 0.6551827 ],
       [0.48948607, 0.7017464 , 0.68989825],
       [0.5230725 , 0.27884543, 0.60958445],
```

```
       [0.7827147 , 0.8914031 , 0.4876485 ],
       [0.8648485 , 0.24485788, 0.32038832],
       [0.8178784 , 0.8858621 , 0.06766882],
       [0.87350154, 0.28361183, 0.05250168],
       [0.93995637, 0.6879045 , 0.04860159],
       [0.9658277 , 0.40127382, 0.0224943 ],
       [0.84900236, 0.92112637, 0.06929153],
       [0.8356112 , 0.26262954, 0.02063032],
       [0.87454265, 0.9394624 , 0.02284241],
       [0.84174293, 0.02700099, 0.02180329]], dtype=float32)
```

```
keypoints.shape
```

```
(1, 1, 17, 3)
```

```
righteye = keypoints[0][0][2]
righteye_c=np.array(righteye[:2]*[480,640]).astype(int)
leftelbow = keypoints[0][0][7]
print(righteye)
print('right eye',righteye_c)
```

```
[0.28251183 0.44304937 0.4752282 ]
right eye [135 283]
```

```
shaped = np.squeeze(np.multiply(model.get_tensor(model.get_output_details()[0]['inde
shaped
```

```
array([[1.61106491e+02, 3.14778366e+02, 6.01399899e-01],
       [1.32958202e+02, 3.41984673e+02, 2.91481435e-01],
       [1.35605679e+02, 2.83551598e+02, 4.75228190e-01],
       [1.43652420e+02, 3.77703171e+02, 5.91938972e-01],
       [1.42033854e+02, 2.43292847e+02, 6.55182719e-01],
       [2.34953313e+02, 4.49117699e+02, 6.89898252e-01],
       [2.51074791e+02, 1.78461075e+02, 6.09584451e-01],
       [3.75703068e+02, 5.70497971e+02, 4.87648487e-01],
       [4.15127277e+02, 1.56709042e+02, 3.20388317e-01],
       [3.92581644e+02, 5.66951752e+02, 6.76688179e-02],
       [4.19280739e+02, 1.81511574e+02, 5.25016822e-02],
       [4.51179056e+02, 4.40258865e+02, 4.86015901e-02],
       [4.63597298e+02, 2.56815243e+02, 2.24942993e-02],
       [4.07521133e+02, 5.89520874e+02, 6.92915320e-02],
       [4.01093388e+02, 1.68082905e+02, 2.06303187e-02],
       [4.19780474e+02, 6.01255951e+02, 2.28424110e-02],
       [4.04036608e+02, 1.72806346e+01, 2.18032897e-02]])
```

```
for i in shaped:
    y, x, conf = i
    print(int(y), int(x), conf)
```

```
161 314 0.6013998985290527
132 341 0.2914814352989197
135 283 0.4752281904220581
143 377 0.5919389724731445
142 243 0.6551827192306519
234 449 0.6898982524871826
251 178 0.6095844507217407
375 570 0.48764848709106445
415 156 0.3203883171081543
392 566 0.06766881793737411
419 181 0.05250168219208717
451 440 0.04860159009695053
463 256 0.022494299337267876
407 589 0.06929153203964233
401 168 0.02063031867146492
419 601 0.0228424109518528
404 17 0.021803289651870728
```

```python
def draw_keypoints(frame, keypoints, threshold):
    y, x, c = frame.shape
    shaped = np.squeeze(np.multiply(keypoints, [y,x,1]))

    for i in shaped:
        y, x, conf = i
        if conf > threshold:
            cv2.circle(frame, (int(x), int(y)), 4, (0,255,255), -1)
```

```python
EDGES = {
    (0, 1): 'm',
    (0, 2): 'c',
    (1, 3): 'm',
    (2, 4): 'c',
    (0, 5): 'm',
    (0, 6): 'c',
    (5, 7): 'm',
    (7, 9): 'm',
    (6, 8): 'c',
    (8, 10): 'c',
    (5, 6): 'y',
    (5, 11): 'm',
    (6, 12): 'c',
    (11, 12): 'y',
    (11, 13): 'm',
    (13, 15): 'm',
    (12, 14): 'c',
    (14, 16): 'c'
}
```

```python
shaped[0], shaped[1]
```

```
(array([294.90077019, 286.57598495,   0.77177668]),
 array([275.12403488, 307.74776459,   0.46545959]))
```

```python
for edge, color in EDGES.items():
    p1, p2 = edge
    y1, x1, c1 = shaped[p1]
    y2, x2, c2 = shaped[p2]
    print((int(x2), int(y2)))
```

```
(341, 132)
(283, 135)
(377, 143)
(243, 142)
(449, 234)
(178, 251)
(570, 375)
(566, 392)
(156, 415)
(181, 419)
(178, 251)
(440, 451)
(256, 463)
(256, 463)
(589, 407)
(601, 419)

(168, 401)
(17, 404)
```

```
def draw_connections(frame, keypoints, edges, threshold):
    y, x, c = frame.shape
    shaped = np.squeeze(np.multiply(keypoints, [y,x,1]))

    for edge, color in edges.items():
        p1, p2 = edge
        y1, x1, c1 = shaped[p1]
        y2, x2, c2 = shaped[p2]

        if (c1 > threshold) & (c2 > threshold):
            cv2.line(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0,255,0), 2)
```

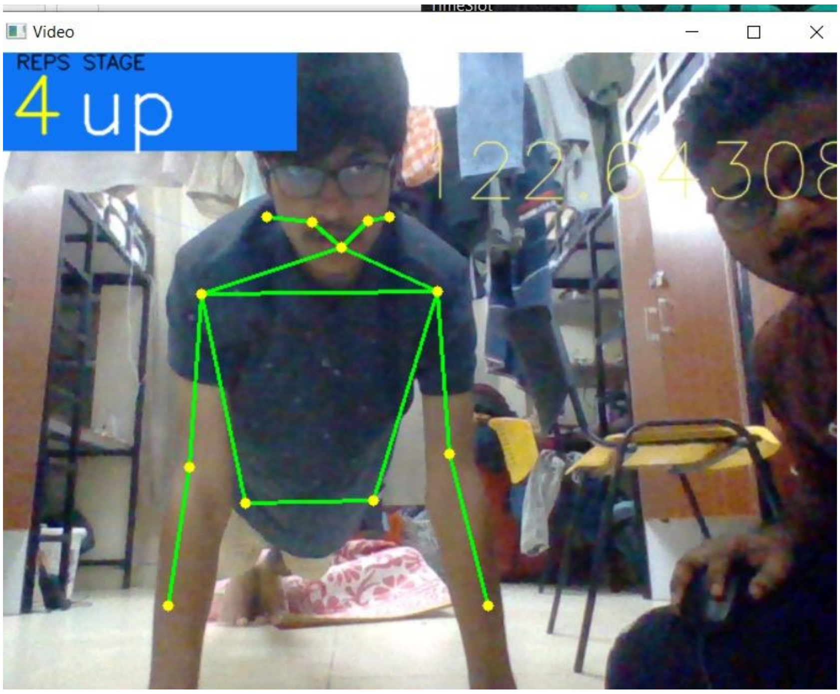
```
def calculate_angle(a,b,c):
    a = np.array(a) # First
    b = np.array(b) # Mid
    c = np.array(c) # End

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians*180.0/np.pi)

    if angle >180.0:
        angle = 360-angle

    return angle
```

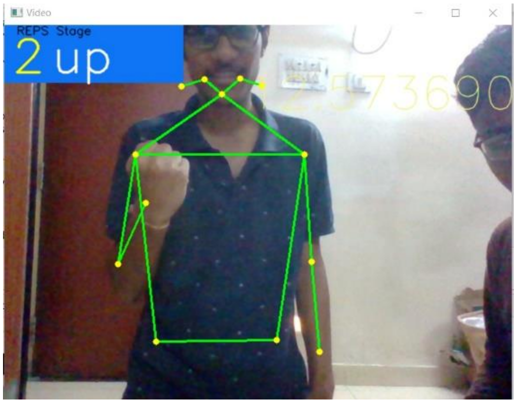
```
counter=0
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    image = frame.copy()
    image = tf.image.resize_with_pad(np.expand_dims(image, axis=0), 192,192)
    in_image = tf.cast(image, dtype=tf.float32)

    in_details = model.get_input_details()
    out_details = model.get_output_details()

    model.set_tensor(in_details[0]['index'], np.array(in_image))
    model.invoke()
    keypoints = model.get_tensor(out_details[0]['index'])
    #print(keypoints)

    rightshoulder=keypoints[0][0][6]
    rightshoulder_c=np.array(rightshoulder[:2]*[480,480]).astype(int)
    rightelbow=keypoints[0][0][8]
    rightelbow_c=np.array(rightelbow[:2]*[480,480]).astype(int)

    leftshoulder=keypoints[0][0][5]
    leftshoulder_c=np.array(leftshoulder[:2]*[480,480]).astype(int)
    leftelbow=keypoints[0][0][7]
    leftelbow_c=np.array(leftelbow[:2]*[480,480]).astype(int)

    if(rightshoulder_c[1]>rightelbow_c[1] and leftshoulder_c[1]>leftelbow_c[1]):
        stage='down'
    if(rightshoulder_c[1]<rightelbow_c[1] and leftshoulder_c[1]<leftelbow_c[1]) and
        stage='up'
        counter+=1
        print('Number of Pushups',counter)
```

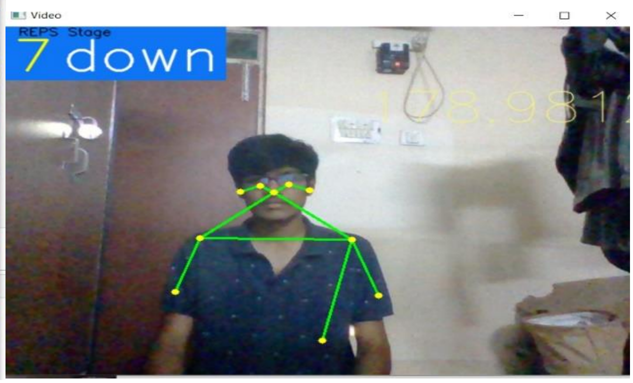
```
            # pushups


    cv2.rectangle(frame, (0,0), (225,73), (245,117,16), -1)

        # Rep data
    cv2.putText(frame, 'REPS', (15,12),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
    cv2.putText(frame, str(counter),
                (10,60),
                cv2.FONT_ITALIC, 2, (0,255,255), 2, cv2.LINE_AA)

        # Stage data
    cv2.putText(frame, 'STAGE', (65,12),
                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)
    cv2.putText(frame, stage,
                (60,60),
                cv2.FONT_HERSHEY_SIMPLEX, 2, (255,255,255), 2, cv2.LINE_AA)

    draw_connections(frame, keypoints, EDGES, 0.2)
    draw_keypoints(frame, keypoints, 0.2)

    cv2.imshow('Video', frame)

    if cv2.waitKey(1) & 0xFF==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

6.Predictive Analytics

Predictive analytics encompasses a variety of statistical techniques from data mining, predictive modelling, and machine learning, that analyze current and historical facts to make predictions







These are some predictive analysis where counted the number of sets done by the person of one particular exercise.

7.CONCLUSION

We trained through defining various parameters (POSE) to understand if the person body joints. And identify the pose and keep the track of the count if any person Is doing any kind of fitness activity like push-up, Dumble lifting and etc.

Therefore, with the advancements in machine learning and we have successfully implemented the pose estimation using move net model and counted the number of curls and pushups done by the person accurately. This project can be extended by implementing in the android app where the gym users can use it

.

8.FUTURE WORK

Our project can be used in future to predict and do training for various other poses. But as now we have trained on our real time video dataset as well. Our project can be used in the current situation of COVID-19. Various attention is given to treat covid-19 patients, but post covid-19 treatment is not efficient and often ignored.

Various reports have claimed certain deaths post covid-19 due to ‘No proper immune’ and ‘fitness’ in the body.

Therefore, our further dataset with proper specifications of ketoacidosis level can help the health authorities to know if the patient has severe diabetes or not, if the patient will be vulnerable to COVID-19 and if the patient will develop post-covid symptoms. This data will enable the health authorities to pay attention to those patients with proper facilities and treatment.

9.REFERENCES

REFERENCE JOURNALS:

[1] Q. Dang, J. Yin, B. Wang and W. Zheng, "Deep learning based 2D human pose estimation: A survey," in *Tsinghua Science and Technology*, vol. 24, no. 6, pp. 663-676, Dec. 2019, doi: 10.26599/TST.2018.9010100.

[2] human body pose detection using bayesian spatio-temporal templates. dimitrijevic, miodrag, vincent lepetit, and pascal fua. "human body pose detection using bayesian spatio-temporal templates." computer vision and image understanding 104.2-3 (2006): 127-139.

[3] Hoffmann M, Turk EA, Gagoski B, Morgan L, Wighton P, Tisdall MD, Reuter M, Adalsteinsson E, Grant PE, Wald LL, van der Kouwe AJW. Rapid head-pose detection for automated slice prescription of fetal-brain MRI. Int J Imaging Syst Technol. 2021 Sep;31(3):1136-1154. doi: 10.1002/ima.22563. Epub 2021 Mar 1. PMID: 34421216; PMCID: PMC8372849.

[4] A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks,".

[5] Z. Cao, T. Simon, S. Wei and Y. Sheikh, "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields,".

[6] A. Singh, S. Agarwal, P. Nagrath, A. Saxena and N. Thakur, "Human Pose Estimation Using Convolutional Neural Networks,"