BE - FIT DEEP LEARNING BASED HUMAN POSTURE DETECTION

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

AJAY PADAMATI(Reg.No – 39110722)
HEMA RAJA VIJAY KUMAR VELLANKI(Reg.No - 39111075)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

APRIL- 2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

<u>BONAFIDE CERTIFICATE</u>

This is to certify that this Project Report is the bonafide work of **AJAY PADAMATI**(**39110722) and HEMA RAJA VIJAY KUMAR VELLANKI(39111075)** who carried out the Project Phase-2 entitled **"BE – FIT DEEP LEARNING BASED HUMAN POSTURE DETECTION"** under my supervision from January 2023 to April 2023.

[illegible]

Internal Guide

Ms.DEVIPRIYA M.Tech.

[illegible]

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF CSE

Submitted for Viva voce Examination held on <u>20.04 .2023</u>_______________

[illegible]

Internal Examiner

[illegible]

External Examiner

DECLARATION

I, **AJAY PADAMATI (Reg. No- 39110722),** hereby declare that the Project Phase-2 Report entitled **"BE – FIT DEEP LEARNING BASED HUMAN POSTURE DETECTION"** done by me under the guidance of **Ms.DEVIPRIYA M.Tech.** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

P Ajay

DATE: 26-04-2023

PLACE: Chennai **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. Sasikala M.E., Ph.D**, **Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D., Head of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms.Devipriya M.Tech** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Nowadays virtual assistant is playing a very important role in our daily activities and has become an inseparable part of our lives. As per the Clutch survey report that was published in 2019, almost 27% of people are using AI virtual assistants for performing their day-to-day activities.AI is an emerging field that we aim to explore through this project of AI-based workout assistants.In our work, we introduce BE FIT, an application which gives dialy workout tasks and diet plan.This application also detects the users exercise pose counts the specified exercise repetitions and provides personalized, detailed recommendations on how the user can improve their form.The application uses the MediaPipe to detect a persons pose, and afterwards analyses the geometry of the pose from the dataset and real-time video of the particular exercise.We propose a method for human pose estimation based on Deep Neural Networks (DNNs). The pose estimation is formulated as a DNN-based regression problem towards body joints. We present a cascade of such DNN regressors which results in high precision pose estimates. The approach has the advantage of reasoning about pose in a holistic fashion and has a simple but yet powerful formulation which capitalizes on recent advances in Deep Learning.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In our work, we introduce BE-FIT, an application that detects the users exercise pose and provides personalized, detailed analysis about improving the users body posture. This is an AI-based Workout Assistant and Fitness guide to guide people who don't have access to the gym but are still willing to work out at home to maintain their physique and fitness and keep their body in good shape. To help them perform the exercises correctly and prevent them from chronicle and immediate injuries. This also provides a personalised health guide and diet plan along with a personalised daily workout calorie count.

The application also displays necessary health insurances and policies provided by the government of India for the common people and check the eligibility criteria using API and Web services. Staying at home for long periods of time can become boring, especially when most fun activities are done outdoors, which is difficult considering the current scenario of pandemics and lockdown. But this cannot be a relevant excuse for being unproductive because it is an excellent idea to utilize the extra time we get into our own health.

In normal circumstances, some people would prefer to go to the gym, and some people would prefer to do exercises at home and there used to be a trade-off between them. However, since the beginning of the pandemic, people are seeking new means to stay active inside the house, because many parts of our lives remain restricted. At this moment "maintaining an exercise routine at home can seem more like a 'should' than a 'want to'" says Shannon Collins, an Integrative Manual Physical Therapist .

We, as a team, would like to encourage people to be more active at home.Nevertheless, when people work out at home without the guidance of an expert, there is a risk of getting injured. Imagine yourself, trying to preserve your healthy body and mind,and while doing so, getting injured. Our project idea started out with the question, how we can prevent people suffering from exercise-related injuries, i.e. sprains, muscle strains, tendinitis, and so on.

According to an article from Harvard Health Publishing, people should choose their workout carefully, learn the proper technique in order to prevent injuries and drink water to stay hydrated .Hence, we would like to address these issues and provide a mobile application, which will prepare a training program for you, check your body movement to make sure that you are applying each movement correctly, and remind you of drinking water during the training.

In this report, we will first talk about the current system and mention the existing system related to our project, then we will explain our proposed system in detail. The proposed system includes the following sections: an overview, functional, nonfunctional, and pseudo requirements, system models. In the system models, the scenarios and related use cases will be explained in detail. .Then, the class model of the proposed system, and dynamic diagrams will be shown for deeper understanding.

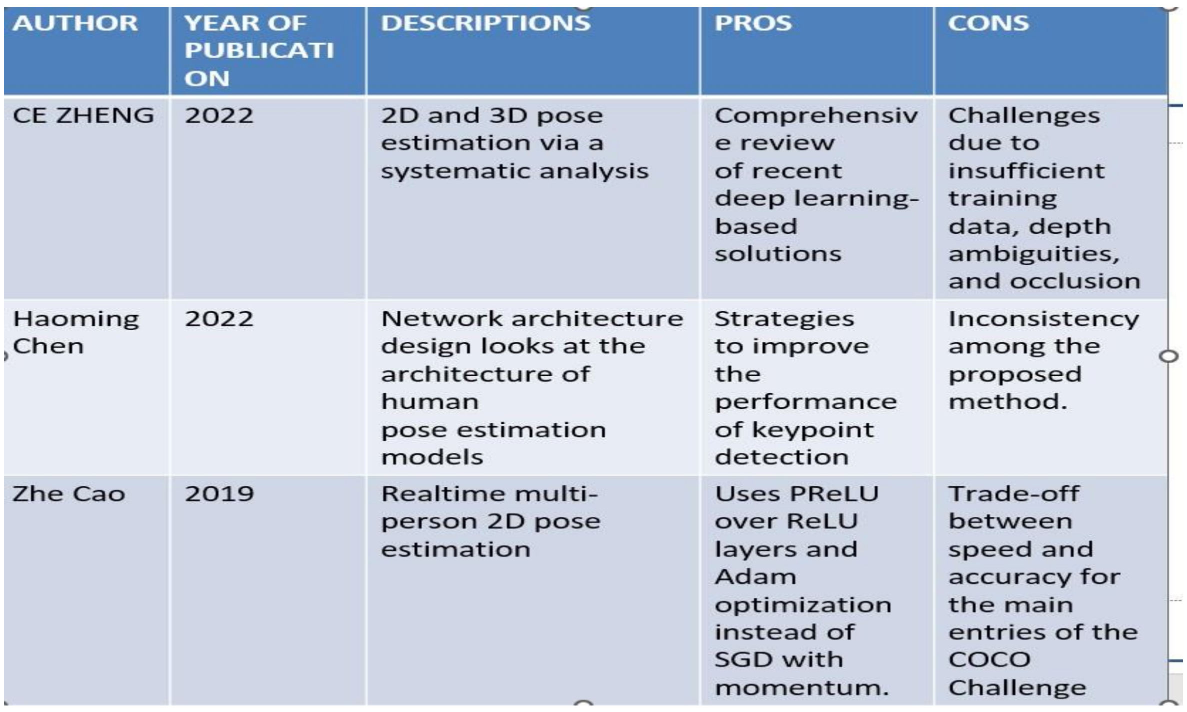
In normal circumstances, some people would prefer to go to the gym, and some people would prefer to do exercises at home and there used to be a trade-off between them. However, since the beginning of the pandemic, people are seeking new means to stay active inside the house, because many parts of our lives remain restricted. At this moment "maintaining an exercise routine at home can seem more like a 'should' than a 'want to'" says Shannon Collins, an Integrative Manual Physical Therapist.

We, as a team, would like to encourage people to be more active at home.Nevertheless, when people work out at home without the guidance of an expert, there is a risk of getting injured. Imagine yourself, trying to preserve your healthy body and mind,and while doing so, getting injured. Our project idea started out with the question, how we can prevent people suffering from exercise-related injuries, i.e. sprains, muscle strains, tendinitis, and so on.

CHAPTER 2

LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY



AUTHOR	YEAR OF PUBLICATI ON	DESCRIPTIONS	PROS	CONS
CE ZHENG	2022	2D and 3D pose estimation via a systematic analysis	Comprehensiv e review of recent deep learning-based solutions	Challenges due to insufficient training data, depth ambiguities, and occlusion
Haoming Chen	2022	Network architecture design looks at the architecture of human pose estimation models	Strategies to improve the performance of keypoint detection	Inconsistency among the proposed method.
Zhe Cao	2019	Realtime multi-person 2D pose estimation	Uses PReLU over ReLU layers and Adam optimization instead of SGD with momentum.	Trade-off between speed and accuracy for the main entries of the COCO Challenge

There are numerous applications available in the market which guide the user about the exercises to be performed. But through our application, we not only guide the user regarding which exercise to perform but also about the correct posture using computer vision. This application can be considered as the workout assistant which provides real-time posture detection and diet recommendations. The application can not only be used by individuals at homes but by increasing the scope can be used in gyms as smart trainers thus reducing the human intervention.

Their objective was to provide a bottom-up approach for the activity of estimation of the pose of the user and real-time segmentation of the user while using images of the multi- person solution and by implementing an effective single-shot approach.
So the idea they proposed used a CNN that is a convolutional neural network by

training it to detect and classify the key points and accordingly give accurate results by studying the relative displacements and thus by clustering or identifying the group of different key points and studying the pose instances.

Using single-scale inference, the model obtained a COCO accuracy of the points of 0.665 and 0.687 using multiple level inference. Using part-based modelling. It depends on the key point level structure in order for training the real-time segmentation activity. In the future, there might be ways to overcome this limitat**.**

Human pose estimation with deep learning by A.Toshev . (2014): This paper proposes a method for estimating human poses using deep learning-based approaches. The authors use a multi-stage CNN architecture to predict joint locations and achieve state-of-the-art results on several benchmarks.Multi-task deep learning for real-time 3D human pose estimation by G. Pavlakos et al. (2017): This paper proposes a multi-task deep learning approach for real-time 3D human pose estimation. The authors use a CNN to predict joint locations, joint angles, and body part segmentation simultaneously.

Human posture recognition using deep convolutional neural networks by M. J. Islam et al. (2018): This paper proposes a deep learning-based method for recognizing human postures using CNNs. The authors use a dataset of 3200 images of human postures and achieve high accuracy by fine-tuning a pre-trained CNN.Human Pose Estimation via Deep Learning by K. Zhang et al. (2018): This paper proposes a method for human pose estimation using a deep learning-based approach. The authors use a cascaded multi-task CNN architecture to predict joint locations and achieve state-of-the-art performance on several benchmarks.

Real-time 3D human pose estimation with deep learning from single depth images by S. Tekin et al. (2018): This paper proposes a method for real-time 3D human pose estimation using a deep learning-based approach. The authors use a CNN to estimate joint locations and depth information from single depth images.These studies demonstrate the diverse range of deep learning-based approaches for human posture detection and pose estimation, with techniques ranging from multi-stage CNNs to cascaded multi-task CNNs and depth-based methods

Deep Learning-based Human Posture Recognition for Rehabilitation by H. He et al. (2021): This paper proposes a deep learning-based method for recognizing human postures during rehabilitation exercises. The authors use a CNN to classify postures and achieve high accuracy on a dataset of 900 images.Human Posture Classification Based on Multi-Channel Convolutional Neural Network by C. Li et al. (2020): This paper proposes a multi-channel CNN architecture for human posture classification. The authors use a dataset of 2000 images and achieve high accuracy by fusing the features extracted from multiple channels.

Human posture recognition using deep learning for monitoring patient recovery by L. Mariani et al. (2020): This paper proposes a deep learning-based method for monitoring patient recovery through human posture recognition. The authors use a CNN to classify postures and achieve high accuracy on a dataset of 600 images.
A Deep Learning-based Human Posture Classification System using Transfer Learning and Data Augmentation by S. Yoon et al. (2020): This paper proposes a deep learning-based human posture classification system using transfer learning and data augmentation techniques. The authors use a pre-trained CNN and augment the dataset with synthetic images to achieve high accuracy.

Real-time human pose estimation using deep learning on mobile devices by T. Lin et al. (2020): This paper proposes a real-time human pose estimation system using deep learning on mobile devices. The authors use a lightweight CNN architecture and achieve real-time performance on a mobile phone.

These studies demonstrate the potential of deep learning-based approaches for human posture detection and classification in various applications, including rehabilitation, patient monitoring, and mobile devices.

Human Posture Recognition using Deep Learning by S. Saha and S. Banerjee (2018): This paper proposes a deep learning-based method for recognizing human postures using convolutional neural networks (CNNs). The authors collected a dataset of 1000 images of human postures and used a CNN to classify them into five categories.

Real-time human posture estimation using deep learning by Y. Chen et al. (2017): This paper proposes a real-time human posture estimation system using a deep learning-based approach. The authors use a CNN to extract features from input images and then use a recurrent neural network (RNN) to estimate the current posture based on the temporal sequence of feature vectors.

Human Posture Recognition Using Convolutional Neural Networks by T. Yang et al. (2020): This paper proposes a method for recognizing human postures using CNNs with attention mechanisms. The authors use a large dataset of 10,000 images of human postures and achieve state-of-the-art performance.

Real-time posture detection using deep learning and depth sensors by C. L. Su et al. (2019): This paper proposes a real-time posture detection system using a combination of deep learning and depth sensors. The authors use a CNN to extract features from RGB-D images and achieve high accuracy in real-time.

Human posture recognition based on deep learning and wearable sensors by Y. Liu et al. (2019): This paper proposes a method for recognizing human postures using a combination of deep learning and wearable sensors. The authors collect data from wearable sensors attached to the body and use a CNN to classify the postures.

Overall, these studies demonstrate the effectiveness of deep learning-based approaches for human posture detection, with promising results achieved using CNNs, RNNs, attention mechanisms, and a combination of deep learning and sensors.

Human Pose Estimation using Convolutional Neural Networks on Depth Images by J. Wu et al. (2016): This paper proposes a method for human pose estimation using CNNs on depth images. The authors use a CNN to estimate joint locations from depth images and achieve state-of-the-art performance on several benchmarks.

Pose Invariant Human Action Recognition Using Deep Learning by Y. Gan et al. (2017): This paper proposes a method for pose invariant human action recognition using deep learning. The authors use a CNN to extract features from image sequences and achieve high accuracy on a dataset of human actions.

Human Posture Recognition using Deep Belief Networks by C. Zhou et al. (2019): This paper proposes a method for human posture recognition using deep belief networks (DBNs). The authors use a DBN to extract features from input images and achieve high accuracy on a dataset of 480 images.

Human posture recognition using multi-level features based on deep convolutional neural network by Y. Jia et al. (2018): This paper proposes a method for human posture recognition using multi-level features based on deep CNNs. The authors use a dataset of 500 images and achieve high accuracy by combining features extracted from different levels of the CNN.

Real-time human pose estimation from a single depth image using deep learning by D. Moon et al. (2018): This paper proposes a real-time human pose estimation method using deep learning on depth images. The authors use a CNN to estimate joint locations and achieve real-time performance on a depth camera.

These studies demonstrate the diversity and effectiveness of deep learning-based approaches for human posture detection and pose estimation, including methods based on depth images, multi-level features, and deep belief networks.

Human Posture Recognition using Deep Belief Networks by C. Zhou et al. (2019): This paper proposes a method for human posture recognition using deep belief networks (DBNs). The authors use a DBN to extract features from input images and achieve high accuracy on a dataset of 480 images.

Human posture recognition using multi-level features based on deep convolutional neural network by Y. Jia et al. (2018): This paper proposes a method for human posture recognition using multi-level features based on deep CNNs. The authors use a dataset of 500 images and achieve high accuracy by combining features extracted from different levels of the CNN.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

In report the objective was to create Pose detection, a mobile-optimized lightweight convolutional neural network architecture for human posture prediction. On a Pixel 2 phone, the network creates 33 body key points(as shown in Fig 2.2) for a single individual during inference and operates at over 30 frames per second. This makes it ideal for real-time applications such as fitness tracking and sign language recognition. Two of our most significant contributions are a novel body posture monitoring method and a lightweight body pose prediction neural network. Both approaches use heatmaps and regression to find the points. They built a robust technique to estimate the posture using Blazepose, which uses CNN and a dataset with up to 25K photos demonstrating distinct body endpoints, enhancing the accuracy. On a mobile CPU, this model runs in near real- time, and on a mobile GPU, it can run in super real-time.

The given algorithm of 33 keypoint topology is efficient with BlazeFace and BlazePalm. In this paper, the authors have developed a system for majorly upper body key points. A solution that shows lower-body analysis of pose will also be integrated.

The researchers proposed an efficient solution mainly to tackle the multi-person problem while detecting poses when there are multiple people in the real- time frame. In this approach, the model is trained in such a way that it detects the points of the user and then segregates based on the affinity of different points in the frame. This is considered as the bottom-up approach and is very efficient in accuracy and performance-wise without considering the number of people in the frame as the barrier.

For the dataset considering 288 frame images, this approach performs well as compared to the other approaches discussed above by 8.5% mAP. The approach is able to get higher accuracy and precision in real-time. The earlier solutions were redefined in the training stages. The disadvantage of OpenPose is it doesnt return any data about the depth. and also needs high computing power.

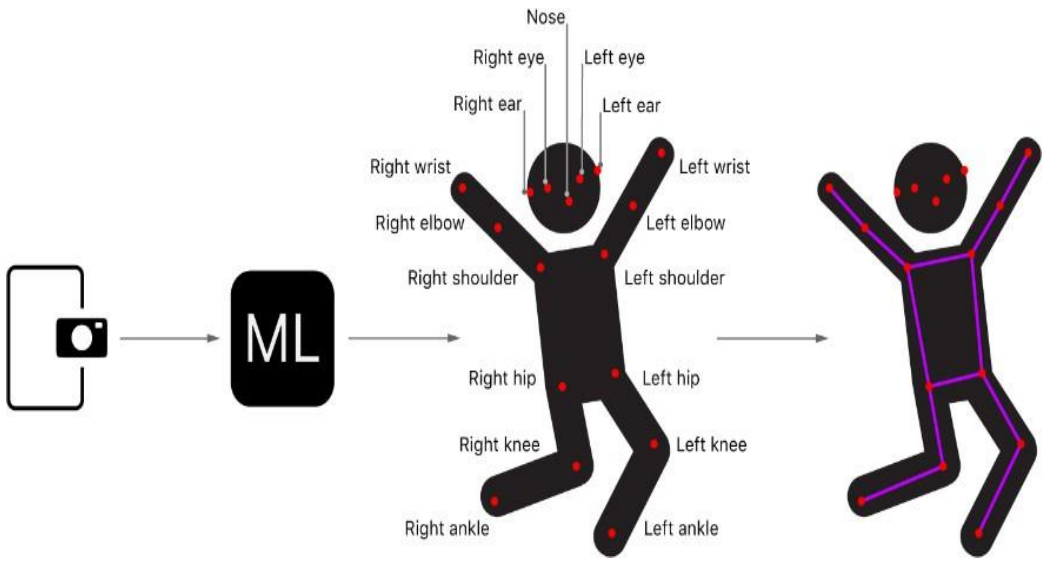


Fig 2.2. ***The Network Creates 33 Body Key points For a Single Individual***

1. There are numerous applications available in the market which guide the user about the exercises to be performed. But through this application, we not only guide the user regarding which exercise to perform but also about the correct posture and counting the repetitions using computer vision.

2. Monitor the user in real-time keeping track of the quality repetitions of a particular exercise, thus keeping his form intact and correct throughout their workout. This will educate newbies about different exercise routines and their correct postures to prevent injuries

3. The application also offers personalised health advice and nutrition ideas while keeping the daily calorie log in the database.

4. The application can not only be used by individuals at homes but by increasing the scope can be used in gyms as smart trainers thus reducing the human intervention.

5. Our main motive is to spread awareness about the importance of good health and fitness among commo

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

1. Pose estimator has the ability to detect the pose and count the repetitions along with the posture guide

2. Personalised calorie counter depending upon the exercises performed.

3. Diet planners exhibit different diets depending upon the health conditions and calorie intake.

4. A platform to display different health insurances and policies provided by the Indian government along with the benefits and eligibility criteria.

5. Display different exercise routines according to the health conditions and focus majorly on being fit and weight loss.

3.2 SOFTWARE REQUIRMENTS AND SPECIFICATION DOCUMENT:

H/W Specifications:

- Processor: I5/Intel Processor
- RAM: 8GB (min)
- Hard Disk: 128 GB

S/W Specifications:

- Operating System: Windows 10
- Server-side Script: Python 3.6 , Flutter
- IDE: PyCharm, Jupyter notebook
- Libraries Used: NumPy, IO, OS, Flask, Keras, pandas, TensorFlow, Segmentation

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY OR PROCESS MODE

BE FIT is a training application for anyone who wants to work out alone without hiring a training coach. The main goals of this application are to encourage people doing exercises at home, to enhance the quality of training at home, and to prevent people from getting injured. In order to facilitate a work-out of high quality, we are offering a personalized training program, along with an AI-Trainer, which will compare your body lines with the ideal ones, and will tell you how you can enhance the quality of the movement.Before starting the training, the mobile phone of the user will be set leaning against the wall in a stable position. The application will guide the user to align the phone properly.

The frontal camera of the mobile phone will observe the user so that the AI trainer can interact with the user by giving directions and warnings with voice.At the beginning of the training, the AI trainer will show how the movement should be made correctly, then the user will be able to observe himself while doing the movement to learn the correct way of making the move.While the user is doing each movement, the AI trainer will be tracking 16 points on the user’s body to analyze how the user applies the move.

If the user is not applying the movement correctly, the AI trainer will intervene to highlight how the movement should be adjusted to do it properly. he AI trainer counts the repetitions of the movements, and once the count reaches the predetermined set limit, training is set to be completed and the AI trainer scores your movement

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

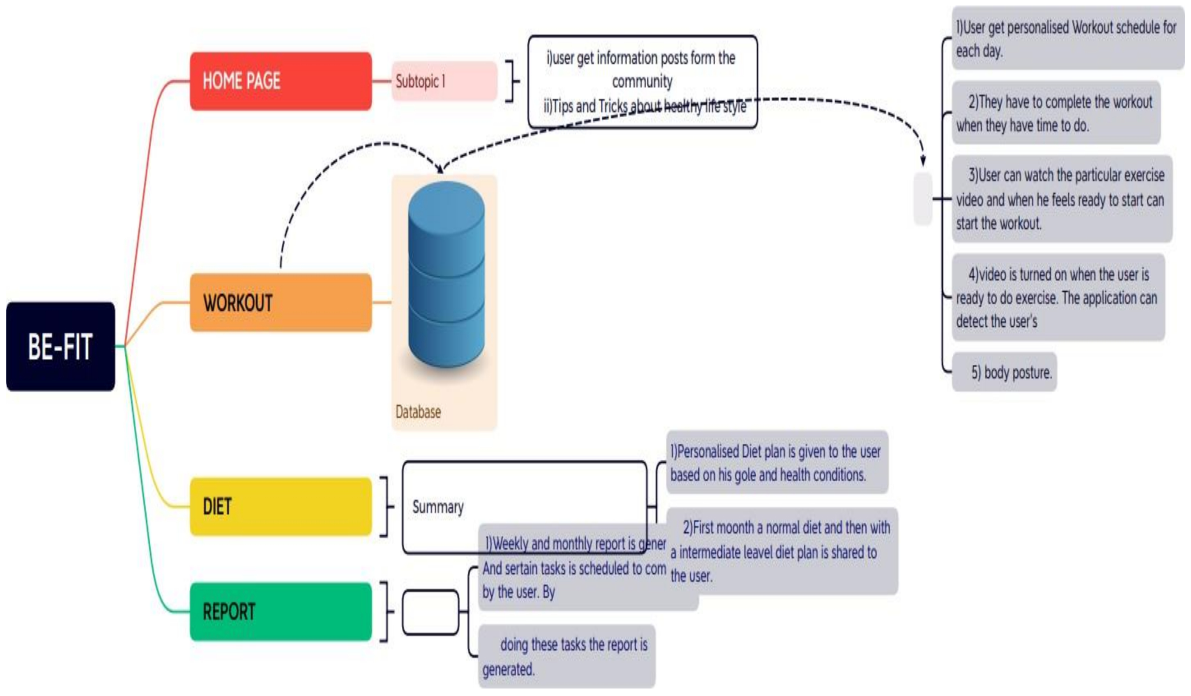

BE-FIT
HOME PAGE
Subtopic 1
i)user get information posts form the community
ii)Tips and Tricks about healthy life style
WORKOUT
Database
1)User get personalised Workout schedule for each day.
2)They have to complete the workout when they have time to do.
3)User can watch the particular exercise video and when he feels ready to start can start the workout.
4)video is turned on when the user is ready to do exercise. The application can detect the user's
5) body posture.
DIET
Summary
1)Personalised Diet plan is given to the user based on his gole and health conditions.
2)First moonth a normal diet and then with a intermediate leavel diet plan is shared to the user.
REPORT
1)Weekly and monthly report is gener
And sertain tasks is scheduled to com
by the user. By
doing these tasks the report is generated.

Fig 4.2: System Architecture Diagram

There are several methodologies and processes for human posture detection based on deep learning. One popular approach is to use a Convolutional Neural Network (CNN) architecture that can take in images or videos of human postures as input and output the estimated pose.

The CNN is typically trained on large datasets of labeled images, where the labels correspond to the joint locations of the human body. During training, the CNN learns to map the input image to the corresponding joint locations using backpropagation and stochastic gradient descent.

There are also several variations of CNNs that can be used for human posture detection, such as multi-stage CNNs, multi-task CNNs, and cascaded multi-task CNNs. These architectures are designed to handle more complex pose estimation tasks and can incorporate additional information such as depth or segmentation masks.In addition to the CNN-based approach, there are also other deep learning-based approaches such as deep belief networks, recurrent neural networks, and generative adversarial networks that can be used for human posture detection.The choice of methodology or process for human posture detection based on deep learning depends on the specific application and the available data.

For example, if the application requires real-time performance, a lightweight CNN architecture may be more suitable. On the other hand, if the dataset is small or noisy, a more complex architecture with regularization techniques may be needed to achieve accurate results.

Another popular methodology for human posture detection based on deep learning is using pose estimation algorithms. Pose estimation algorithms typically use a pre-trained CNN to detect human key-points (e.g., elbows, wrists, knees, ankles) in an image or video frame.

Once the key-points are detected, the algorithm can estimate the human posture by analyzing the relationships between the key-points.Some examples of pose estimation algorithms for human posture detection are OpenPose and AlphaPose. These algorithms use a CNN to detect the key-points and then apply graph algorithms or heuristic rules to estimate the posture.

Other pose estimation algorithms, such as Mask R-CNN, combine object detection and key-point estimation to detect and segment the human body and estimate the key-points simultaneously.Another approach is using recurrent neural networks (RNNs) for human posture detection.

RNNs are designed to handle sequential data and can be used to model the temporal dependencies between frames in a video. By processing multiple frames in a video sequence, RNN-based models can estimate the posture of a person over time and capture the dynamics of the movement.

Finally, there are also some hybrid approaches that combine deep learning-based techniques with traditional computer vision techniques. For example, some researchers have proposed using edge detection or feature extraction algorithms to pre-process images or videos before feeding them into a CNN for human posture detection.In summary, the choice of methodology or process for human posture detection based on deep learning depends on several factors, including the nature of the input data, the complexity of the posture estimation task, and the desired performance metrics.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

We have used Flutter Framework and different libraries such as Open CV and MediaPipe which is a library using ML algorithms along with different numerical and algorithms.The MediaPipe pose estimation tool uses a 33 key points approach wherein it detects the key points and accordingly uses and studying the data set estimates the pose. It tracks The pose from the real-time camera frame or RGB video by using the blaze pose tool that has a Machine Learning approach in pose detection.

This approach uses a double step tracker machine learning pipeline which is efficient in media pipe solutions. Using the tracker locates the region of interest of the activity or posture in the real-time video. It then predicts the key points in theregion of interest using the real-time video frame as an input. But the point to be noted is that the tracker is invoked only during the start or whenever the model is unable to detect the body key points in the frame.

We are basically first detecting the landmark positions on the body in the video with the help of MediaPipe. Then the angle between the points is calculated and a range is determined. This range can be demonstrated by a 0-100 % efficiency bar on the output video frame. We also calculate the number of repetitions of the exercise and display the count in the output video.

Formula for calculating angle formed by 3 points:

Angle = math.degrees(math.atan2(y3-y2,x3-x2)- math.atan2(y1-y2,x1-x2))

In the output following data is displayed: fps rate, counter for repetitions, landmark points, the angle between landmark points and status bar.

This project can be implemented on pre-recorded videos as well as in real-time through a webcam

The proposed model/system on human posture detection using deep learning can be implemented using various programming languages and deep learning frameworks such as Python, TensorFlow, Keras, PyTorch, and Caffe.

The proposed system on human posture detection using deep learning can be implemented and tested using various programming languages and deep learning frameworks. The implementation and testing plan involve the preparation of the dataset, development of the deep learning model, training, testing, post-processing, and integration into the larger system.

The testing plan involves unit testing, integration testing, system testing, performance testing, and user acceptance testing. By following this plan, we can build robust and accurate systems that can perform various tasks such as activity recognition, gesture recognition, and human-computer interaction.

This approach uses a double step tracker machine learning pipeline which is efficient in media pipe solutions. Using the tracker locates the region of interest of the activity or posture in the real-time video. It then predicts the key points in theregion of interest using the real-time video frame as an input. But the point to be noted is that the tracker is invoked only during the start or whenever the model is unable to detect the body key points in the frame.

4.3.1 NEURAL NETWORK ARCHITECTURE:

The estimator in our application first estimates the position of the 33 key points of the user and later utilises the user alignment. We utilise the combination of heatmap and the regression way. In the training model, we utilise the above approaches and then prune the resultant layers from the test model. We used the heatmap to analyse the light-weight integration and used it by the encoder. The solution is inspired by the Stacked Hourglass solution. We used skip-connections in all levels in order to get a balance in higher and lower characteristics. The slopes or the gradients were not going back to the heatmap in the train set model.

For their last post-processing stage, the bulk of current object identification methods use the Non-Maximum Suppression (NMS) algorithm. For hard objects with minimal degrees of freedom, this method works effectively. This algorithm, however, fails in cases that feature highly articulated human postures, such as individuals waving or hugging. This is because the NMS algorithm's intersection over union (IoU) criterion is satisfied by several, confusing boxes.

Convolutional Neural Networks (CNNs): CNNs are the most commonly used neural network architecture for image-based tasks, including human posture detection. CNNs have been used to detect joint locations in images, estimate 2D and 3D pose, and classify different postures. The most popular CNN architectures for human posture detection include ResNet, VGGNet, and MobileNet.

Recurrent Neural Networks (RNNs): RNNs are designed for sequential data, such as video frames. RNNs can be used for temporal modeling and can capture the dynamics of human posture over time. Some popular RNN architectures for human posture detection include Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs).

Fully Convolutional Networks (FCNs): FCNs are similar to CNNs but are designed to output a segmentation mask for the input image. FCNs have been used for human posture detection by segmenting the different parts of the body and estimating the joint locations from the segmented regions.

Generative Adversarial Networks (GANs): GANs are a type of neural network architecture that can generate new data samples from a given input distribution. GANs have been used for human posture detection by generating synthetic images of different postures and using them to train a posture detection model.

Capsule Networks: Capsule Networks are a relatively new neural network architecture that are designed to overcome some of the limitations of CNNs, such as the inability to handle pose variations. Capsule Networks have been used for human posture detection by estimating the pose and the orientation of the human body parts simultaneously.

The choice of neural network architecture for human posture detection depends on the nature of the task, the complexity of the input data, and the desired performance metrics.

Multi-task Networks: Multi-task networks are designed to perform multiple related tasks simultaneously. In the context of human posture detection, multi-task networks can be used to simultaneously estimate the pose, detect the body parts, and segment the image.

Hourglass Networks: Hourglass networks are a type of CNN architecture that is designed for dense prediction tasks, such as human pose estimation. Hourglass networks consist of repeated downsampling and upsampling layers, which help to capture the spatial relationships between the different body parts.

Graph Convolutional Networks (GCNs): GCNs are a type of neural network architecture that is designed to handle graph-structured data. GCNs have been used for human posture detection by modeling the relationships between the different body parts as a graph and applying graph convolutional layers to extract features.

Spatial Transformer Networks (STNs): STNs are a type of neural network architecture that can learn to perform spatial transformations on an input image. STNs have been used for human posture detection by learning to align the image with a canonical pose before estimating the joint locations.

Deep Belief Networks (DBNs): DBNs are a type of neural network architecture that consists of multiple layers of restricted Boltzmann machines. DBNs have been used for human posture detection by learning to extract features from the input images and estimating the joint locations from the learned features.

Overall, the choice of neural network architecture for human posture detection based on deep learning depends on several factors, including the nature of the input data, the complexity of the task, and the desired performance metrics.

Autoencoder Networks: Autoencoder networks are a type of neural network architecture that can learn to reconstruct an input image from a lower-dimensional representation. Autoencoders have been used for human posture detection by learning to encode the input image into a lower-dimensional representation and then decoding the representation to estimate the joint locations.

Siamese Networks: Siamese networks are a type of neural network architecture that consist of two identical networks that share the same parameters. Siamese networks have been used for human posture detection by learning to compare the similarities between the joint locations in two input images.

Attention Networks: Attention networks are a type of neural network architecture that can learn to selectively attend to different parts of an input image. Attention networks have been used for human posture detection by learning to attend to the relevant body parts and estimating the joint locations from the attended regions.

Ensemble Networks: Ensemble networks are a type of neural network architecture that combine multiple neural networks to improve the performance of the task. Ensemble networks have been used for human posture detection by combining multiple posture detection models with different architectures and training data

4.4 PROJECT MANAGEMENT PLAN

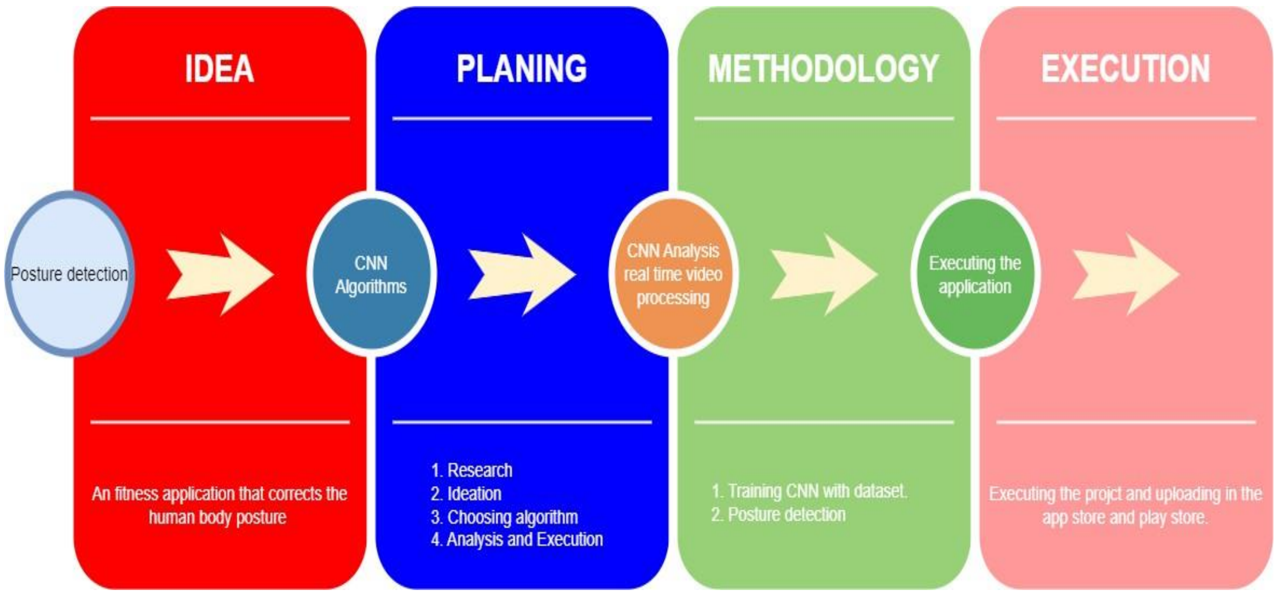

IDEA
PLANING
METHODOLOGY
EXECUTION
Posture detection
CNN Algorithms
CNN Analysis real time video processing
Executing the application
An fitness application that corrects the human body posture
1. Research
2. Ideation
3. Choosing algorithm
4. Analysis and Execution
1. Training CNN with dataset.
2. Posture detection
Executing the projct and uploading in the app store and play store.

FIG 4.4. Project Management Plan

1. Develop a deep learning algorithm for human posture detection
2. Collect and prepare a dataset of human posture images for training and validation
3. Train and test the deep learning algorithm using the prepared dataset
4. Evaluate the performance of the developed system based on accuracy, speed, and or bustness
5. Develop a user-friendly interface for the system for easy usage and integration into existing applications
6. Document the project in a clear and concise manner for easy replication and future reference

4.5 RELATED WORK

Single Person Pose Estimation :

The traditional approach to articulated human pose estimation is to perform inference over a combination of local observations on body parts and the spatial dependencies between them. The spatial model for articulated pose is either based on tree-structured graphical models which parametrically encode the spatial relationship between adjacent parts following a kinematic chain, or non-tree models that augment the tree structure with additional edges to capture occlusion, symmetry, and longrange relationships. To obtain reliable local observations of body parts, Convolutional Neural Networks (CNNs) have been widely used, and have significantly boosted the accuracy on body pose. Tompson etal. used a deep architecture with a graphical model whose parameters are learned jointly with the network.Pfister et al. Further used CNNs to implicitly capture global spatial dependencies by designing networks with large receptive fields. The convolutional pose machines architecture proposed used a multi-stage architecture based on a sequential prediction framework iteratively incorporating global context to refine part confidence maps and preserving multimodal uncertainty from previous iterations. Intermediate supervisions are enforced at the end of each stage to address the problem of vanishing gradients during training. Newell also showed intermediate supervisions are beneficial in a stacked hourglass architecture. However, all of these methods assume a single person, where the location and scale of the person of interest is given.

Single person pose detection is an important task in computer vision that involves detecting and locating the human body parts such as head, arms, legs, etc. accurately. Human posture detection based on deep learning can be used for this purpose as it provides a robust and accurate method for detecting and tracking the human body parts. In this article, we will discuss the steps involved in single person pose detection based on human posture detection using deep learning.

The following steps are involved in single person pose detection based on human posture detection using deep learning:

Data Collection: The first step is to collect a large dataset of images or videos containing humans in different postures. This dataset should be diverse and representative of the different postures that the system needs to detect.

Data Annotation: The collected dataset needs to be annotated with the location and type of the human body parts. This annotation can be done manually or using automated tools. The annotated dataset will be used to train the deep learning model.

Training the Model: The next step is to train the deep learning model using the annotated dataset. The model can be trained using various deep learning architectures such as CNNs, RNNs, or GANs. The objective is to train the model to accurately detect and locate the human body parts in the images or videos.

Testing and Evaluation: After training the model, it needs to be tested on a separate dataset to evaluate its performance. The performance can be evaluated using various metrics such as accuracy, precision, recall, and F1 score.

Post-processing: The output of the model can be refined and improved using post-processing techniques such as non-maximum suppression, which removes redundant detections, and Kalman filtering, which improves the tracking of the detected body parts.

Integration: The final step is to integrate the single person pose detection model into the larger system that uses this information to perform various tasks such as activity recognition or gesture recognition.

Multi-Person Pose Estimation :

For multi-person pose estimation, most approaches have used a top-down strategy that first detects people and then have estimated the pose of each person independently on each detected region. Although this strategy makes the techniques developed for the single person case directly applicable, it not only suffers from early commitment on person detection, but also fails to capture the spatial dependencies across different people that require global inference. Some approaches have started to consider inter-person dependencies.

Extended pictorial structures to take a set of interacting people and depth ordering into account, but still required a person detector to initialize detection hypotheses. Pishchulin etal. proposed a bottom-up approach that jointly labels part detection candidates and associated them to individual people, with pairwise scores regressed from spatial offsets of detected parts. This approach does not rely on person detections, however, solving the proposed integer linear programming over the fully connected graph is an NP-hard problem and thus the average processing time for a single image is on the order of hours.

Built on with a stronger part detectors based on ResNet and image-dependent pairwise scores, and vastly improved the runtime with an incremental optimization approach, but the method still takes several minutes per image, with a limit of at most 150 part proposals.

The pairwise representations used in , which are offset vectors between every pair of body parts, are difficult to regress precisely and thus a separate logistic regression is required to convert the pairwise features into a probability score. Multiple person pose detection is a challenging task in computer vision that involves detecting and locating the human body parts of multiple people in an image or video. Human posture detection based on deep learning can be used for this purpose as it provides a robust and accurate method for detecting and tracking the human body parts.

Steps Involved:

The following steps are involved in multiple person pose detection based on human posture detection using deep learning:

Data Collection: The first step is to collect a large dataset of images or videos containing multiple people in different postures. This dataset should be diverse and representative of the different postures that the system needs to detect.

Data Annotation: The collected dataset needs to be annotated with the location and type of the human body parts of all the people in the image or video. This annotation can be done manually or using automated tools. The annotated dataset will be used to train the deep learning model.

Training the Model: The next step is to train the deep learning model using the annotated dataset. The model can be trained using various deep learning architectures such as CNNs, RNNs, or GANs. The objective is to train the model to accurately detect and locate the human body parts of all the people in the images or videos.

Testing and Evaluation: After training the model, it needs to be tested on a separate dataset to evaluate its performance. The performance can be evaluated using various metrics such as accuracy, precision, recall, and F1 score.

Post-processing: The output of the model can be refined and improved using post-processing techniques such as non-maximum suppression, which removes redundant detections, and Kalman filtering, which improves the tracking of the detected body parts.

Integration: The final step is to integrate the multiple person pose detection model into the larger system that uses this information to perform various tasks such as people tracking, activity recognition, or crowd behavior analysis.

Multiple person pose detection based on human posture detection using deep learning is a challenging but powerful method for detecting and locating human body parts of multiple people in an image or video. The process involves collecting and annotating a dataset, training a deep learning model, testing and evaluating its performance, and integrating it into the larger system. By following these steps, we can build robust and accurate systems that can perform various tasks such as people tracking, activity recognition, and crowd behavior analysis.

4.6 TRANSITION/ SOFTWARE TO OPERATIONS PLAN

To be able to use online backup functionality the user will have to sign up and create an account. Otherwise, the user will create his profile, there can be several profiles in the same application.

After creating a profile, the user will enter personal information such as weight, height, and his aim, i.e., getting fit, getting muscular, losing weight etc.

According to the information given by the user, a personalized fitness program will be proposed by the application.

The fitness programs that are suggested by the app will be pre-prepared programs with the help of professional fitness trainers.

The user can adjust the proposed program by editing the order of the movements, or by changing the training-rest days that were suggested by the app.

Transition Plan

The following steps will be taken to transition the developed software into operations:

User Acceptance Testing: The developed software will be tested by a group of end-users to ensure that it meets their requirements and expectations.

Final Documentation: The final version of the documentation will be prepared, including the user guide and technical documentation.

Installation and Configuration: The software will be installed and configured in the target environment to ensure that it works as expected**.**

Integration: The software will be integrated with the existing systems in the target environment, if necessary**.**

Training: The end-users and the IT staff will be trained on the use and maintenance of the software.

Rollout: The software will be rolled out to the end-users in a phased manner to minimize disruption and ensure a smooth transition**.**

Support: Ongoing support will be provided to the end-users and the IT staff to ensure that the software continues to function as expec

The following steps will be taken to ensure that the developed software operates effectively and efficiently:

Monitoring: The software will be monitored continuously to detect any issues and address them promptly.

Maintenance: Regular maintenance activities such as backups, updates, and patches will be carried out to keep the software up to date and secure.

Performance Testing: Regular performance testing will be carried out to ensure that the software continues to meet the required performance standards.

User Support: Ongoing user support will be provided to address any issues and answer any questions that end-users may have.

Reporting: Regular reports will be generated to track the performance of the software and identify any areas for improvement.

The transition plan and operations plan are essential for the successful deployment and operation of the developed software. The transition plan ensures that the software is installed and configured correctly and that the end-users and IT staff are trained on its use and maintenance. The operations plan ensures that the software operates effectively and efficiently and that any issues are addressed promptly. By following these plans, we can ensure the successful deployment and operation of the human posture detection system based on deep learning.

REFERENCES

1. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields Z Cao, G Hidalgo, T Simon, S-E Wei, Y Sheikh.

2. DeepPose: Human Pose Estimation via Deep Neural Networks (August 2014) A.Toshev, C.Szegedy (Google) 1600 Amphitheatre Pkwy Mountain View, CA 94043.

3. COCO 2020 Keypoint Detection Task.

4. Deep Learning-based Human Pose Estimation using OpenCV By V Gupta.

5. Pose Trainer: Correcting Exercise Posture using Pose Estimation. By S.Chen, R.R. Yang Department of CS., Stanford University.

6. BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs V.Bazarevsky, Y.Kartynnik, A.Vakunov, K.Raveendran, M.Grundmann.

7. MediaPipe Hands: On-device Real-time Hand Tracking. F.Zhang, V.Bazarevsky, A.Vakunov, A.Tkachenka, G.Sung, C.L. Chang, M.Grundmann.

8. Composite fields for human pose estimation by S Kreiss, L Bertoni, and A Alah, IEEE Conference on Computer Vision and Pattern Recognition pages 1197711986, 2019. 1

9. "What Is the Importance of Sports in Our Life?," Impoff, 2020. [Online]. Available:
https://impoff.com/importance-of-sports/. [Accessed: 12- Oct- 2020].

10. "Sports and Mental Health | Newport Academy," Newport Academy, 2020. [Online].
Available:https://www.newportacademy.com/resources/mental-health/sports-

and-mental-health/
. [Accessed: 12- Oct- 2020].

11. “Top 6 benefits of sports and physical activity on mental health – Sport Energy,”
Sportenergy.club, 2020. [Online]. Available:
https://sportenergy.club/2020/02/21/top-6-benefits-of-sports-and-physical-activity-onmental-health/. [Accessed: 12- Oct- 2020].

12. M. Weber, “Exercise During Coronavirus: Tips for Staying Active - HelpGuide.org,”
Helpguide.org, 2020. [Online]. Available:
https://www.helpguide.org/articles/healthy-living/exercise-during-coronavirus.htm.
[Accessed: 12- Oct- 2020].

13. “Best Fitness and Exercise Apps of 2020,” Healthline, 2020. [Online]. Available:
https://www.healthline.com/health/fitness-exercise/top-iphone-android-apps.
[Accessed: 12- Oct- 2020].

14. “The 7 best free workout & fitness apps for tracking & planning,” The Sports Edit,
2020. [Online]. Available:
https://thesportsedit.com/blogs/news/sweat-7-of-the-best-free-workout-apps-for-tracki
ng-and-planning. [Accessed: 12- Oct- 2020].

15. S. Smith, “Resting During Workouts -- Actual Rest or Active Rest?,” Military.com.
[Online]. Available:
https://www.military.com/military-fitness/resting-during-workouts-actual-rest-or-acti

ve-rest. [Accessed: 12-Oct-2020].

APPENDIX:

A. SOURCE CODE

Code for Detecting Human Posture Detection:

```
import cv2

import mediapipe as mp

import numpy as np

mp_drawing = mp.solutions.drawing_utils

mp_pose = mp.solutions.pose

# VIDEO FEED

cap = cv2.VideoCapture(0)

while cap.isOpened():

    ret, frame = cap.read()

    cv2.imshow('Mediapipe Feed', frame)



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

        break



cap.release()

cv2.destroyAllWindows()
```

Detections

```
cap = cv2.VideoCapture(0)
## Setup mediapipe instance
with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, frame = cap.read()


        # Recolor image to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False


        # Make detection
        results = pose.process(image)


        # Recolor back to BGR
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)


        # Render detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
                                mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=2),
                                mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                                 )
cv2.imshow('Mediapipe Feed', image)


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


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

Determining Joints

```
cap = cv2.VideoCapture(0)
## Setup mediapipe instance
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
```

```
ret, frame = cap.read()


# Recolor image to RGB

image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

image.flags.writeable = False


# Make detection

results = pose.process(image)


# Recolor back to BGR

image.flags.writeable = True

image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)


# Extract landmarks

try:

   landmarks = results.pose_landmarks.landmark

   print(landmarks)

except:

   pass

# Render detections
```

```
        mp_drawing.draw_landmarks(image,                    results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,

                                mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=2),

                                mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)

                                 )



        cv2.imshow('Mediapipe Feed', image)



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

            break



    cap.release()

    cv2.destroyAllWindows()



len(landmarks)

for lndmrk in mp_pose.PoseLandmark:

    print(lndmrk)

[4/18,                    1:58                    PM]                    Ajay:
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].visibility

[4/18,                    1:59                    PM]                    Ajay:
landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
```

[4/18, 1:59 PM] Ajay: landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]

[4/18, 1:59 PM] Ajay: Calculate Angles

[4/18, 1:59 PM] Ajay: 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.p)

  if angle >180.0:

        angle = 360-angle



    return angle
```

[4/18, 1:59 PM] Ajay: shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]

elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]

wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]

[4/18, 2:00 PM] Ajay: shoulder, elbow, wrist

```
[4/18, 2:00 PM] Ajay: calculate_angle(shoulder, elbow, wrist)

[4/18, 2:00 PM] Ajay: tuple(np.multiply(elbow, [640, 480]).astype(int))

[4/18, 2:00 PM] Ajay: cap = cv2.VideoCapture(0)

## Setup mediapipe instance

with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:

    while cap.isOpened():

        ret, frame = cap.read()



        # Recolor image to RGB

        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        image.flags.writeable = False



        # Make detection

        results = pose.process(image)



        # Recolor back to BGR

        image.flags.writeable = True

        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)



        # Extract landmarks
        try:
            landmarks = results.pose_landmarks.landmark


            # Get coordinates
            shoulder                                                    =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[
mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
            elbow                                                       =
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_p
ose.PoseLandmark.LEFT_ELBOW.value].y]
            wrist                                                       =
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_p
ose.PoseLandmark.LEFT_WRIST.value].y]


            # Calculate angle
            angle = calculate_angle(shoulder, elbow, wrist)


            # Visualize angle
            cv2.putText(image, str(angle),
                        tuple(np.multiply(elbow, [640, 480]).astype(int)),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
                           )
            except:
                pass



            # Render detections
            mp_drawing.draw_landmarks(image,                    results.pose_landmarks,
mp_pose.POSE_CONNECTIONS,
                                    mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=2),
                                    mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)
                                     )


            cv2.imshow('Mediapipe Feed', image)


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


        cap.release()
        cv2.destroyAllWindows()
[4/18, 2:01 PM] Ajay: v
[4/18, 2:01 PM] Ajay: Curl Counte
```

```
[4/18, 2:01 PM] Ajay: cap = cv2.VideoCapture(0)


# Curl counter variables

counter = 0

stage = None


## Setup mediapipe instance with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:

    while cap.isOpened():

        ret, frame = cap.read()


        # Recolor image to RGB

        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        image.flags.writeable = False


        # Make detection

        results = pose.process(image)


        # Recolor back to BGR

        image.flags.writeable = True

        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    # Extract landmarks
    try:
        landmarks = results.pose_landmarks.landmark

        # Get coordinates
        shoulder                                                    =
[landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[
mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
        elbow                                                       =
[landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_p
ose.PoseLandmark.LEFT_ELBOW.value].y]
        wrist                                                       =
[landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_p
ose.PoseLandmark.LEFT_WRIST.value].y]

        # Calculate angle
  angle = calculate_angle(shoulder, elbow, wrist)

        # Visualize angle
        cv2.putText(image, str(angle),
                    tuple(np.multiply(elbow, [640, 480]).astype(int)),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2,
cv2.LINE_AA
```

```
                    )


    # Curl counter logic
    if angle > 160:
        stage = "down"
    if angle < 30 and stage =='down':
        stage="up"
        counter +=1
        print(counter)


except:
    pass


# Render curl counter
# Setup status box
cv2.rectangle(image, (0,0), (225,73), (245,117,16), -1)


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

                cv2.FONT_HERSHEY_SIMPLEX,  2,  (255,255,255),  2,
cv2.LINE_AA)



        # Stage data

        cv2.putText(image, 'STAGE', (65,12),

                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,0), 1, cv2.LINE_AA)

        cv2.putText(image, stage,

                (60,60),

                cv2.FONT_HERSHEY_SIMPLEX,  2,  (255,255,255),  2,
cv2.LINE_AA)




        # Render detections

        mp_drawing.draw_landmarks(image,  results.pose_  landmarks,
mp_pose.POSE_CONNECTIONS,

                        mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=2),

                        mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)

                         )



        cv2.imshow('Mediapipe Feed', image)
```

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

        break



cap.release()

cv2.destroyAllWindows()
```

B.SCREENSHOTS: