**Proposed System**

The proposed system for this work uses a webcam to capture the user's movements, and a trained MediaPipe model to estimate their body pose in real-time. The system can then provide feedback to the user on how to correct their form to achieve a better workout. Figure (figure number) shows the workflow of the project starting from the acquisition of the InfiniteRep dataset, which is a large dataset of weightlifting exercises captured from multiple camera angles and stored in an MPED-V AVC (.mp4) format. The model is first trained on the data and fine-tuned based on its performance in order to be able to accurately apply it to a real-time webcam video feed.

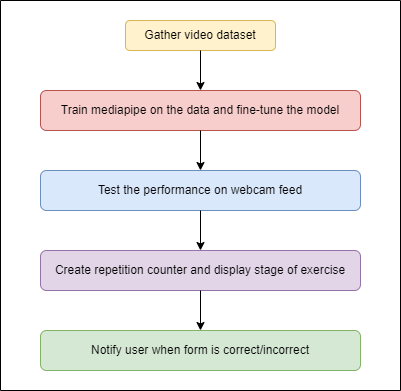


Figure: Project workflow

**3.2 InfiniteRep dataset**

The dataset used for training is InfiniteRep, which is a synthetic, open-source dataset for fitness and physical therapy (PT) applications. The InfiniteRep dataset contains 1,000 videos of diverse avatars doing multiple reps of 10 common exercises. It includes massive variation in the environment, lighting conditions, avatar demographics, and movement trajectories. From cadence to kinematic trajectory, each rep is done slightly differently, just like real humans.

It features:

* 1,000 videos (5–30 seconds each)
* 10 exercises
* Lifelike rep behavior (no two reps are done in the same way)
* 7 indoor home or gym scenes
* Diverse lighting conditions
* Varied demographics, including body shape, skin tones, and clothing
* 18 label and annotation types

The dataset allows engineers to build rep counting models, pose estimation models, form-correction models, segmentation models, activity classification models, and much more.

**3.2 MediaPipe**

MediaPipe is a powerful cross-platform framework for building multimodal machine learning pipelines, including but not limited to computer vision, audio processing, and natural language processing. It was developed by Google and is open source, allowing developers to use it to create their own applications. One of the most popular features of MediaPipe is its ability to perform pose estimation.

Pose estimation is the process of determining the position and orientation of a person's body parts in space. It has many applications, such as motion capture for animation, sports analysis, and physical therapy. MediaPipe makes it easy to perform pose estimation using a variety of different models and algorithms.

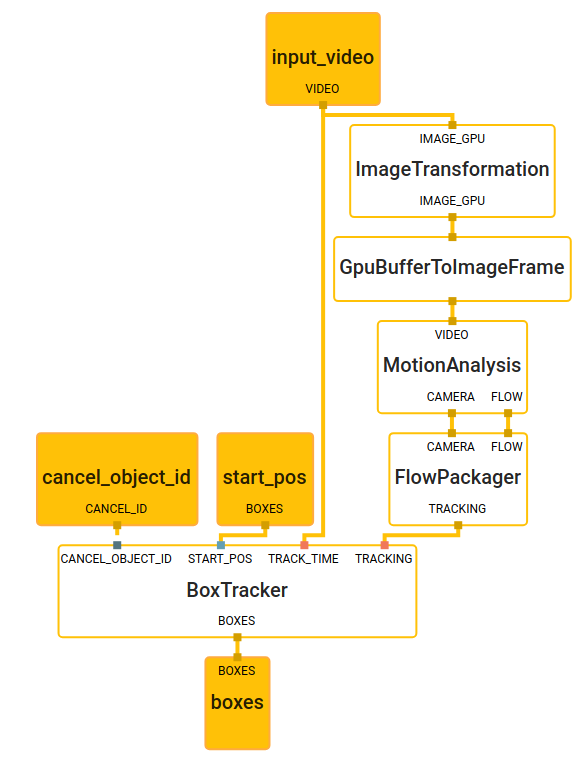


Figure: MediaPipe architecture

MediaPipe offers several pre-trained pose estimation models that can be used out of the box. These models are designed to work with a variety of input sources, including images and video. The models are trained on large datasets of labeled images and use complex neural networks to infer the pose of a person in real-time.

One of the most popular pose estimation models provided by MediaPipe is the "Holistic" model. This model is designed to estimate the pose of a person's entire body, including their face, hands, and body. It uses a multi-stage approach that first detects landmarks on the face, then the hands, and finally the body. The model outputs a set of 3D coordinates for each landmark, which can be used to reconstruct the person's pose.



Figure: MediaPipe body landmarks

To use the MediaPipe pose estimation models in an application, code that captures input from a camera or video file and feeds it into the model, needs to be written. MediaPipe provides a set of Python APIs that make it easy to work with the pre-trained models. Models can be customized by training them on custom datasets, which requires some knowledge of machine learning and computer vision.

MediaPipe is a powerful tool for pose estimation and other machine learning tasks. It provides a range of pre-trained models that can be used out of the box, as well as the ability to customize models to fit your specific needs. With its easy-to-use APIs and comprehensive documentation, MediaPipe is a great choice for developers looking to add pose estimation to their applications.

**3.3 Form detection and correction**

Once the joints and body landmarks are detected, they are then checked for correctness by the use of functions which calculate the angles between them based on the specific exercise being performed. This is done by calculating the element-wise arc tangent of two co-ordinates, x and y in the form of  x/y, by choosing the quadrant correctly.

Arctan (or) tan-1 is the inverse function of the tangent function. This means if -

*y = tan-1(x)*

Then,

*tan (y) = x.*

Also, it is known that if f and f-1 are inverse functions of each other then-

*f(f-1(x)) = f-1(f(x)) = x*

From this, tan(arctan x) = arctan(tan x) = x in the respective domains.

The quadrant, or in other words, the branch is chosen such that the arc tangent of (x, y) is the signed angle in radians between the ray ending at the origin and passing through the point (1, 0), and the ray ending at the origin and passing through the point (y, x).

The value obtained is in radians and is then converted to degrees and altered such that any angle greater than 180 is reset to 0. This is done to calculate the variation of the posture from the accurate form, instead of just obtaining the numeric value without being able to use it to derive insights.

For the purpose of this work, we have considered four different exercises, which are push-ups, leg extensions, lunges and bicep curls. If the user's form is incorrect, the system provides real-time feedback on where the user is making a mistake, and what can be done to correct it. This feedback is in the form of visual cues such as highlighting the correct and incorrect body positions and displaying a text message on how to correct it. Audio notes are also triggered when the user maintains incorrect posture for longer than 3 seconds, so that they can be alerted of the lack of proper form, without having to look at the screen.

In order to effectively locate the source of the incorrect posture for each exercise, we only consider the landmarks and connections between them that are relevant to that particular exercise. For example, for a bicep curl, the knees, feet and facial landmarks and connections are irrelevant and hence, don’t need to be displayed. Similarly, for other exercises, only those landmarks and connections that could potentially be the source of incorrect form are displayed. When the user’s form is deemed correct, the connections are displayed in green colour along with a message reiterating that it is correct. On the other hand, when the user’s form is incorrect, the connections are displayed in red with a message reiterating the same.

Along with information about the exercise form, the number of repetitions performed by the user along with their current stage of exercise are also displayed. When the angle between the relevant joints for a given exercise cross a certain threshold, the repetition counter is incremented and the stage of the exercise is updated. For example, while doing a leg extension, the stages can be “up” or “down” indicating that the user’s legs are either up or down. Similarly, in the case of lunges, the stages can be “front” or “back” respectively.

(Insert sub-sections about storing user’s progress in a database and creation of milestones to motivate them)

**Experimental setup**

Some of the videos in the InfiniteRep dataset are shot from distant angles. These can be used to increase the robustness of the model to detect poses even under undesirable conditions. The data not only contains variability in the lighting conditions and camera angles, but also in the form of the exercises being performed.

Figure 3 shows the demonstrator performing a push-up with correct form with the angle between the knee, elbow and hip being within a certain threshold level.



Figure: Correct exercise form

However, Figure 4 shows the demonstrator performing a push-up with an incorrect posture with the angle exceeding the threshold value which can lead to potential harm to a real user if remained undetected.



Figure: Incorrect exercise form

**4.1 Model parameters**

Mediapipe provides a variety of pre-trained models that can be used for different tasks, including object detection, face detection, hand tracking, pose estimation, and more. Each model comes with a set of parameters that can be customized to adjust the performance and accuracy of the model.

Here are some of the common parameters used in a Mediapipe model:

1. Input Size: This parameter specifies the size of the input image or video frame that the model expects. The input size can affect the model's performance and accuracy, as well as the processing time required for inference.

2. Confidence Threshold: This parameter specifies the confidence level required for the model to detect or recognize a particular object or feature. For example, in pose estimation, the confidence threshold can be used to filter out detections with low confidence scores, which may indicate that the pose is incorrect.

3. Model Complexity: This parameter specifies the complexity of the model, which can affect its accuracy and performance. More complex models can achieve higher accuracy but may require more processing power and time to run.

4. Frame Rate: This parameter specifies the rate at which the model processes video frames. A higher frame rate can result in more accurate and smooth tracking but may also require more processing power and time.

5. Non-Maximum Suppression (NMS) Threshold: This parameter is used to remove duplicate or overlapping detections that can occur when multiple detections are made on the same object or feature. The NMS threshold determines the level of overlap required for the model to discard a duplicate detection.

6. Landmark Count: This parameter specifies the number of landmarks that the model should detect for a particular object or feature. For example, in hand tracking, the model may be configured to detect 21 landmarks per hand, which can be used to track the hand's position and movement.

7. Smoothing: This parameter is used to smooth out the model's output to reduce jitter or noise in the tracking results. Smoothing can be achieved using various techniques, such as filtering or interpolation.

8. Detection confidence: Detection confidence is a score between 0 and 1 that represents the probability that the model has detected the presence of a particular landmark or feature in an image or video frame. For example, in pose estimation, detection confidence measures the likelihood that a specific body joint has been identified in an image. This parameter is used to filter out low-confidence detections, which can help improve the accuracy of the model.

9. Tracking confidence: Tracking confidence is a score between 0 and 1 that represents the probability that the model has correctly tracked a particular landmark or feature across multiple frames of a video. In other words, tracking confidence measures the reliability of the model's estimates over time. For example, in a gesture recognition application, tracking confidence can be used to determine if a hand has remained in the same position for a sufficient amount of time to constitute a recognizable gesture.

These are some of the common parameters used in a Mediapipe model. Depending on the specific task and model, there may be additional parameters that can be customized to achieve the desired performance and accuracy.

Both detection confidence and tracking confidence are used to determine the overall confidence of the model's predictions. A high detection confidence score and a high tracking confidence score together indicate that the model is highly confident that it has accurately identified and tracked a particular landmark or feature. Conversely, a low detection confidence score or a low tracking confidence score can indicate that the model's predictions may be less reliable. These parameters can be adjusted in many Mediapipe models to fine-tune the tradeoff between accuracy and speed of the model.

Setting the detection and tracking confidence parameters to a very high value such as 0.9, can result in the model only detecting and tracking the user’s pose only under perfect lighting conditions, camera quality and angle of video capture. On the other hand, a low value such as 0.2 would make the model inaccurately detect landmarks where they don’t exist, which could lead to inconsistent form detection. The model is capable of detecting key joints and body parts such as the shoulders, elbows, wrists, hips, knees, and ankles, which are essential for proper exercise form.

While applying the model to real-time webcam video feed, we process the feed every 10 frames to save the computation of every single frame. By default, the input feed received from OpenCV is in the Blue-Green-Red (BGR) format. However, MediaPipe can only process Red-Green-Blue format as shown in Figure 5. To solve this problem, we convert the feed from BGR to RGB using OpenCV’s inbuilt functions.

