|  | **MINISTRY OF EDUCATION AND TRAINING** |
| --- | --- |

| **FPT UNIVERSITY** |
| --- |
| Capstone Project Document |
| AI Personal Trainer for Exercise: Pose Detection and Repetition Counting with Deep Learning |

| **<GSU25AI05>** | |
| --- | --- |
| **Group Members** | <Nguyễn Xuân Vỹ> <SE173571>  <Thân Phú Sĩ> <SE172056>  <Ngô Đình Anh Quốc> <SE171534>  <Huỳnh Vĩnh Phúc> <SE172818> |
| **Supervisor** | Nguyễn Quốc Trung |
| **Ext Supervisor** |  |
| **Capstone Project code** | SU25AI33 |

- Ho Chi Minh, <08>/<2025> -

# Table of Contents

[Acknowledgement 4](#_3dcxe8qd89ws)

[Definition and Acronyms 4](#_g0q4upax1p6b)

[List of tables 4](#_i940uv7apg4c)

[List of figures](#_89yo46b3yl85) 5

[I. Project Introduction 5](#_7vkwa0r996dn)

[1. Overview 5](#_40kbb6npfoue)

[2. Project Background](#_bbzm5vnqs0xa) 6

[3. Project Objective](#_i1ti44dvzu8w) 6

[4. Problem Statement](#_4cafu8tbt2yu) 6

[5. Significance of the Project](#_cvzie5fpej8r) 7

[6. Project Scope & Limitations](#_m64lagg4ornm) 7

[II. Project Management Plan](#_oc0asjn3zmv2) 8

[1. Team Work](#_wa6xj97oyih2) 8

[2. Project Management Approach](#_1mfbrfkxgegy) 8

[III. Existing Systems/State of the Art](#_a23ihnlbews6) 10

[1. Overview of the Field](#_wvm4196mt13v) 10

[2. Historical Context](#_2zwtm58k30pg) 10

[3. Key Studies and Theories](#_92hkubz7wwjx) 10

[4. Technological Advancements](#_6mjn8b9poow9) 11

[5. Comparison of Existing Systems](#_unsqzgxgjej9) 11

[6. Gaps in the Literature/Technology](#_kifbh6lqzw83) 11

[7. Justification for the Project](#_3ojp9tb5xkj1) 11

[IV. Methodology](#_ytvq05kdyqi) 12

[IV. Methodology (LSTM)](#_kxp7y2e9lxf1) 12

[1. Research Questions and Objectives](#_f4p75he1fg1w) 12

[2. Data Collection and Preprocessing](#_yaoist40hlwe) 12

[3. Feature Selection and Engineering](#_78l9yt6sznsp) 14

[4. Model Training and Validation](#_yqsh342sp8r) 15

[5. Evaluation Metrics](#_7mv12dao2l30) 16

[6. Implementation Plan](#_w8vkcddls5d9) 17

[7. Ethical Considerations 18](#_gveiadwqyn9h)

[IV. Methodology (YOLOv8)](#_tkw5y5f9ep9) 18

1. [Research Questions and Objectives](#_4esq6giogogz) 18

2. [Data Collection and Preprocessing](#_x42vp61t85h6) 19

[3.](#_78l9yt6sznsp) [Feature Selection and Engineering](#_or5und5lxudv) 20

[4.](#_yqsh342sp8r) [Model Training and Validation](#_tt8vthphziwd) 20

5. [Evaluation Metrics](#_posvbrpcevxm) 21

[6.](#_w8vkcddls5d9) [Implementation Plan](#_4ethuseuihqo) 22

[7.](#_gveiadwqyn9h) [Ethical Considerations](#_tb730z3w5agm) 22

IV. [Methodology (Exercise error detection/Repetition Counting)](#_wa8z65j73cku) 23

[1. Research Questions and Objectives](#_f4p75he1fg1w) 23

[2. Data Collection and Preprocessing](#_yaoist40hlwe) 23

[3. Feature Selection and Engineering](#_78l9yt6sznsp) 24

[4. Model Training and Validation](#_yqsh342sp8r) 24

5[. Implementation Plan](#_w8vkcddls5d9) 25

6[. Ethical Considerations](#_gveiadwqyn9h) 25

[V. System Design and Implementation](#_i56tzc59xtth) 26

1[. Data Flow and Processing](#_oetty7nqe7hc) 26

2[. Deployment Strategy](#_4hd5ynt0x7j7) 27

3[. Scalability and Maintenance](#_d5j63n2i2fjv) 28

[VI. Results and Discussion](#_tlo4ocjadjha) 29

[1. Results and Analysis](#_dupiph5qiiy0) 29

[2. Discussion](#_ydbeeigfi2ju) 36

[3. Recommendations](#_c9iuea6h7ehf) 37

[VII. Conclusion](#_vwiu0gw6t5v1) 38

[1. Summary of Findings](#_d427fxlzmeg4) 38

[2. Contributions and Reflections on the Project](#_nr07l2b6oo2l) 38

[3. Limitations and Future Work](#_ovbm6gieeli8) 38

[VIII. References](#_29fh2zyxzc7v) 39

# Acknowledgement

We cannot express enough thanks to our instructors/teachers from FPT University for their continued support and encouragement. Our sincere thanks to our teacher Mr. Nguyen Quoc Trung for his advice and guidance throughout the development of this project.

We could not have achieved the completion of this project without the support of our family and relatives. Their support has always been a motivation for us throughout the process.

# Definition and Acronyms

| **Acronym** | **Definition** |
| --- | --- |
| AI | Artificial Intelligent |
| DL | Deep Learning |
| ML | Machine Learning |
| PM | Project Manager |
| PMP | Project Management Plan |
| WBS | Work Breakdown Structure |
| LSTM | Long Short-Term Memory |
| Bi-LSTM | Bidirectional Long Short-Term Memory |
| YOLOv8 | You Only Look Once version 8 |
| BGR | Blue-Green-Red color model |
| RGB | Red-Green-Blue color model |
| FPS | Frame(s) per second |

# List of tables

Table 1: Numbers of videos for each exercise taken 13

Table 2: Numbers of videos corresponding to video file type 13

Table 3: Numbers of video corresponding to FPS 14

Table 4: Number of video corresponding to resolution 14

Table 5: Confusion matrix for the Bicep Curl class 17

Table 6: Confusion matrix for the Lunge class 17

Table 7: Confusion matrix for the Plank class 17

Table 8: Confusion matrix for the Sit-up class 17

Table 9: Confusion matrix for the Squat class 17

Table 10: Macro average and weighted average for precision, recall, f1-score 17

Table 11: Model training with Bi-LSTM on 30 frames sequences 29

Table 12: Model training with Bi-LSTM on 60 frames sequences 30

Table 13: Model training with Bi-LSTM+Attention on 30 frames sequences 31

Table 14: Model training with Bi-LSTM+Attention on 60 frames sequences 32

# List of figures

Figure 1: Work Breakdown Structure (WBS) of the project 9

Figure 2: Image of a person doing bicep curls 12

Figure 3: LSTM Data processing 15

Figure 4: System Architecture Diagram 19

Figure 5: Dateset after bounding boxes 20

Figure 6: Confusion matrix for Bi-LSTM on 30 frames sequence 29

Figure 7: Confusion matrix for Bi-LSTM on 60 frames sequence 30

Figure 8: Confusion matrix for Bi-LSTM+Attention on 30 frames sequence 31

Figure 9: Confusion matrix for Bi-LSTM+Attention on 60 frames sequence 32

Figure 10: Confusion matrix of YOLOv8 33

Figure 11: Evaluation metrics of YOLOv8 33

Figure 12: Matrix error Bicep Curl 34

Figure 13: Matrix error Lunge 35

Figure 14: Matrix error Plank 35

Figure 15: Matrix stage Squat 36

# I. Project Introduction

## 1. Overview

### 1.1 Project Information

Project name: AI Personal Trainer for Exercise: Pose Detection and Repetition Counting with Deep Learning

Project code: SU25AI33

Group name: GSU25AI05

Team members:

* Nguyen Xuan Vy: Team leader
* Than Phu Si: Member
* Ngo Dinh Anh Quoc: Member
* Huynh Vinh Phu: Member

Supervisor: Mr. Nguyen Quoc Trung

### 1.2 Project Overview

The system will utilize state-of-the-art human pose estimation techniques to extract skeleton keypoints from video data using MediaPipe framework, followed by deep learning models with one being Bi-LSTM with attention block and the second being Yolov8 to classify different exercises and then check the status of their form ("Correct" or "Incorrect" and at which body parts of the users are they doing it wrong) and detect the transition between exercise states for accurate repetition counting.

This project aims to deliver a prototype that can monitor exercise performance in real-time, providing users with immediate feedback, and tracking progress, helping users improve their posture and efficiency in training sessions, especially in home workout scenarios.

## 2. Project Background

Exercises are a means of improving human health, improving their weights or at the very least strengthening your stamina as well as your immune system. Their posture plays a crucial role in the effectiveness and safety of physical exercises. Such can be helped with if they train themselves in a gym where feedback from the trainers can be used to fix their form. However, for many people, especially beginners or people who do their exercises mainly by themselves as they have no time to go to such places, they tend to perform these exercises incorrectly due to lack of knowledge and instruction. Incorrect posture not only reduces the benefit of a workout but also increases the risk of injuries. This project proposes an AI-based personal trainer system capable of recognizing human exercise postures and accurately counting the number of valid repetitions.

## 3. Project Objective

The goal of this project is to develop machine learning models for 5 common home exercises, where the models can detect any form of incorrect movement while a person is performing a multiple of the 5 exercises. In addition, a web application is developed that uploads the model to the internet to either analyze exercise videos and give feedback or run real time through the camera and send feedback via voice.

## 4. Problem Statement

There are 3 problems that this project intends to solve. First is to have the model to understand what exercise it is currently seeing from the human posture, different exercises have different set of movement and key point to identify what that exercise is, having a model to recognize what action a person is doing between 5 exercises is the core of the project as this will heavily affect the later 2 problems with its most common problem in this field of work is the model’s inability to greatly differentiate different actions when the amount of exercises it needs to detect increases as some of these actions sometimes overlap one another making it hard to know which exercise is which let alone when it is trying to do it to multiple person at once. The second problem is to have the model to be able to recognize incorrect postures or errors when the exercise is being performed, this is also one of the part this project aim to achieve, this part is done right after the exercise pose is detected in the first problem, there are some simple errors of which this project can do is through simply calculating the angles, but there are some that are rather more complicated so it needs to be solved through machine learning of that specific error. The last problem of this project is the model’s ability to count repetition of an exercise, repetition can only be approved after the model detect what that exercise is and if that exercise is done correctly in one cycle, which itself brings multiple challenges to solve from knowing what a full cycle of an exercise is, having the repetition still counts while the model gets confused of what the exercise is being done for a brief moment.

## 5. Significance of the Project

Aside from this project's aim is to make 2 models which later will choose the best one to run the project results, those 2 models also incorporate two different machine learning processes which themselves can use the same dataset but with different purposes that can still run smoothly without overcomplicating the maths behind it. The project also researches about other different ways than the usual libraries that are used to recognize human posture such as Mediapipe or method such as Bi-LSTM that are commonly used for such works about detecting actions to see if we can further simplify the solution for such problems with other ways that is rather uncommon in this field such as Yolov8 which is more known for detecting objects.

## 6. Project Scope & Limitations

This project researches deep learning knowledge such as computer vision, neural network and MediaPipe/Ultralytics YOLOv8 frameworks. This project mainly uses Python programming language, Open CV for image processing, Sci-kit learn, Roboflow for building machine learning model, Streamlit for the web application that uses the trained model for feedback on incorrect form to exercise videos or realtime.

# II. Project Management Plan

## 1. Team Work

### 1.1 Team Structure and Roles

The team consist of 4 members:

* Nguyen Xuan Vy: Team leader, work on #1 approach for the project, exercise recognition part in general, data collecting, communication between members on the project plans as well as planning out future actions, project paper writer and main communication with the supervisor.
* Huynh Vinh Phuc: Member, work on #2 approach for the project, exercise recognition part in general., data collecting
* Than Phu Si: Member, work on #1 approach for the project, exercise errors recognition part in general, data collecting
* Ngo Dinh Anh Quoc: Member, work on #2 approach for the project,work on #1 approach for the project,exercise recognition part in general, build website application, data collecting.

### 1.2 Communication Plan

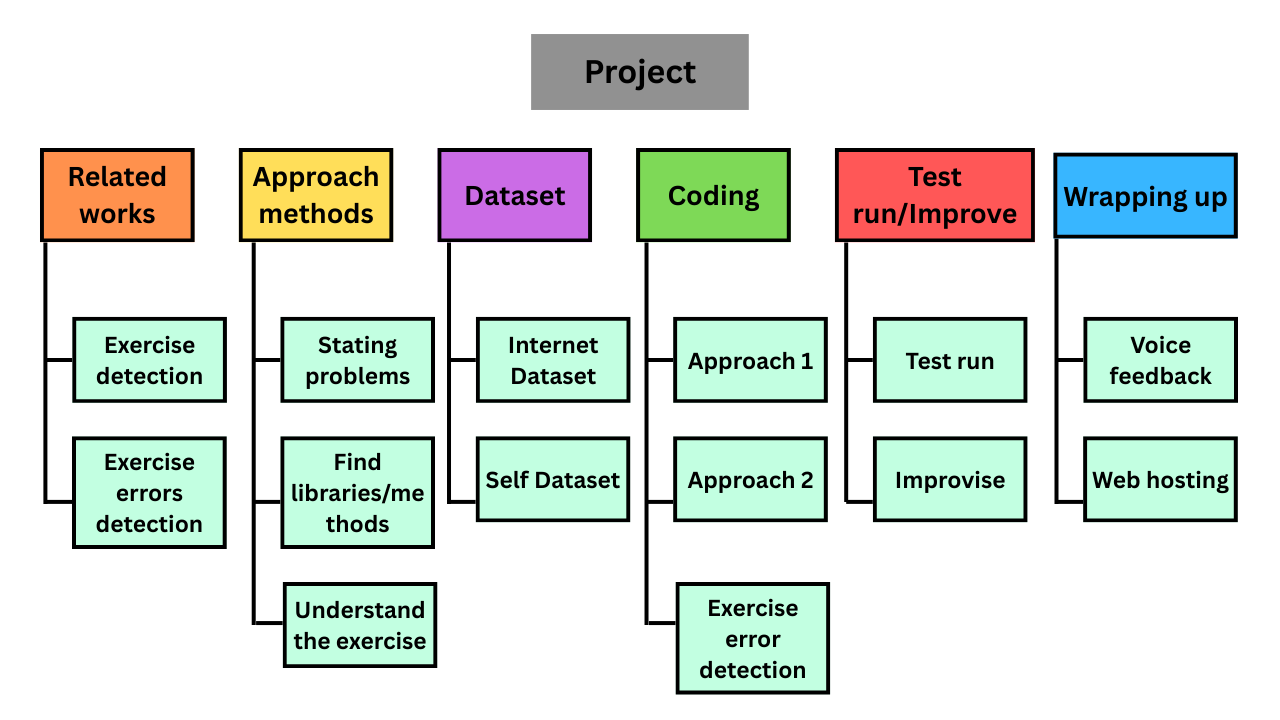
The team communicates mainly through Zalo which is a chatting app. Once per week the team will report their progress to the supervisor teacher and twice per week the team will hold its own meeting through either google meet or offline meeting at a designated location.

## 2. Project Management Approach

### 2.1 WBS

The project will have phases including:

* Related work research: This phase has 2 processes. With the first one being searching about works that have done about detecting actions to be more specifically detecting exercises and the second one being researching about works that have done about detecting errors of an action on a human which in this case is about mistakes or errors that can be made when doing an exercise. The end result for this phase is running these existing codes of these works to see which of them could align with our aims and interests so we can further learn and understand what they do and what their aims, strengths and weaknesses are so we can make our own works that inherit their strengths and can cover their weaknesses.
* Approach methods planning : This comes after the first phase which in this phase has 3 processes with the first one being stating what problems could appear when doing this project and the second process is to find popular methods libraries/methods that were used in this field of work and the third one being to understand the nature of the exercise we are researching from how to do the exercise the correct way, what defines a repetition for that exercise and so on.
* Dataset: This phase has 2 processes. The first one is searching for existing datasets including available dataset which have come with the existing projects that we have researched about as well as looking up for dataset on the internet. The second is to self record ourselves for more data to be used for the model with research on what is the correct way to do certain exercises to do it correctly as well as recording ourselves doing them incorrectly. The result of this process will create a folder which holds the data that will be used for our models.
* Coding: This has 3 processes which the first 2 processes symbolise for our 2 different approaches for solving the problem which include coding the model in order to train it with our dataset and then run it for the results as well as evaluation values for us to understand how well the code is doing. The third process is coding a system that can detect errors and count repetitions of an exercise which is trained from the dataset which the first 2 processes have but has more information about incorrect exercises data which will be used for training to run it for the results as well as evaluating and build logic to count the repetition for the exercise depending on if that action is correct or not. The 3rd process will go with both the 1st and the 2nd process to create our desired model.
* Test running/Improvising: As the name implies, this phase only has 2 processes which go in a loop and that is to run our completed models, from them we will note down how well our code runs. The second process is based on how well the test run goes, this phase tries to improvise its next text run with several different ways from changing the learning properties , changing values in the logic of how the code runs.
* Wrapping up: After reaching our desired output we finished the whole project up by implementing these 2 processes. The first one is to implement voice response to the output of the code and the second is uploading the code to a web hosting site. From that we have completed our project.



*Figure 1. Work Breakdown Structure (WBS) of the project*

### 2.2 Risk Management

The team usually plans ahead its progress and what the team will do as well as making a deadline a month before the actual deadline to prevent the risk of lateness.  
As for the biggest risk, which is models producing bad or unusable results, the team planned to combat it by making 4 different models at once in case such a thing could happen.  
For work loss risk, we save all our finished work on multiple platforms from our local device, on cloud such as Google Drive and on GitHub to make sure if our work goes missing from accidents or we made any bad decisions to our work , we can safely revert it back.

### 2.3 Quality Management

This project already has an advantage of having 2 different approaches that were inspired from past works which can be used to compare between. It also came with the benefit of them being a recently made work that is popular so we can also compare our outcomes to their works too. And for testing and validating the models of ours we use a variety of different values to judge how good a model is such as precision, recall, f1-score, loss values, accuracy values, etc and lastly for safe measurement we also list out many possibilities of the users using the model such as how tall they are , how fast they are doing the exercise and at what angles are they doing the exercise which we also record a person that is completely not in our dataset to do these kind of testings to see if our model can run well on strange unknown data.

# III. Existing Systems/State of the Art

## 1. Overview of the Field

The application of computer vision and artificial intelligence for exercise recognition and posture analysis has developed rapidly in recent years. The general goal of these studies is to help users practice with correct techniques, prevent injuries, and improve training effectiveness. Such systems are gradually becoming important tools not only in the field of sports but also in medical rehabilitation, thanks to their ability to monitor and provide real-time feedback.

## 2. Historical Context

In the past, most exercise monitoring systems relied on wearable sensors, such as accelerometers or gyroscopes. While this approach provided accurate data, it was inconvenient and difficult to deploy on a large scale. The advancement of deep learning models in computer vision opened a new direction, especially through object detection and pose estimation techniques using regular cameras. This laid the foundation for many modern exercise monitoring systems, among which the application of MediaPipe and YOLO marked significant turning points.

## 3. Key Studies and Theories

Google’s MediaPipe has become a widely used tool for extracting 33 human body keypoints, creating essential input data for detailed motion analysis. Alongside this, YOLOv8, part of the YOLO family of models, has demonstrated outstanding effectiveness in object detection with fast processing speeds that meet real-time requirements. Several experimental studies have highlighted the feasibility of combining these tools. In our project,we research that the Exercise-Correction[[1]](#79wh54q86jsj) project employs MediaPipe pose estimation to calculate joint angles and provide users with visual feedback on their form. While this system effectively identifies incorrect postures, its functionality is limited to angle-based detection and lacks advanced features such as customizable model selection or multi-input options. Similarly, the Exercise\_Recognition\_A [[2]](#ikqtp6hfa4e0) project utilizes MediaPipe pose landmarks in combination with machine learning models to classify basic exercises like squats or push-ups.Its scalability is restricted due to the small dataset and limited exercise categories.

3D-Pose-Based-Feedback-For-Physical-Exercises[[3]](#gffrg4qv563k), explores the use of 3D pose estimation to enhance the accuracy of motion tracking and error detection. This approach provides richer spatial information than 2D pose estimation but requires higher computational power and is less feasible for lightweight, real-time applications on consumer-grade hardware.

From these studies, two key insights emerge: first, pose estimation frameworks like MediaPipe are reliable for extracting motion features in real time, but alone they are insufficient for robust exercise classification; second, advanced models such as YOLO can improve exercise recognition by identifying the context of movement, yet they must be integrated with pose analysis for detailed feedback. These insights directly influenced the design of our project, which combines YOLOv8 for exercise detection with MediaPipe for posture analysis, while extending functionality through repetition counting, audio alerts, and overlay visualization.

## 4. Technological Advancements

Recent advancements in computer vision technologies allow the development of exercise monitoring systems using only common hardware such as personal computers and webcams. MediaPipe provides fast and lightweight processing capabilities, while YOLOv8 improves the speed and accuracy of exercise detection. The use of GPUs for inference acceleration has made these systems increasingly feasible for real-world applications.

## 5. Comparison of Existing Systems

Different approaches present their own strengths and limitations. The experimental system using BiLSTM + Attention combined with Mediapipe pose data was capable of classifying action sequences but faced challenges in detecting specific exercises. To address this, the proposed system leverages the power of YOLOv8 for exercise detection and Mediapipe for posture analysis, while adding customized logic for exercises such as squats, planks, lunges, sit-ups, and bicep curls, resulting in higher effectiveness. This solution not only counts repetitions and detects errors but also provides direct feedback through overlays displayed on the video. Compared to other research works, the group’s system extends functionalities, supports multiple input modes, and delivers a more user-friendly experience for general users.

## 6. Gaps in the Literature/Technology

## Despite significant progress, most existing systems still support only a limited number of exercises and struggle with scenarios involving multiple people in the same frame. Moreover, the ability to provide personalized feedback for individual users has not yet been fully addressed, although this is a crucial factor for improving training effectiveness.

## 7. Justification for the Project

Arising from these gaps, this project was developed to leverage the strengths of YOLOv8 in exercise detection combined with Mediapipe in posture analysis, while adding advanced functions such as repetition counting, audio alerts, and intuitive overlay visualization. The system also allows users to choose between webcam and video file inputs, thereby enhancing flexibility and broader applicability. With these improvements, the project not only overcomes several limitations of existing systems but also demonstrates the feasibility of applying artificial intelligence to support personal exercise training. This contributes to improving public health by enabling more people to exercise effectively at home without needing to travel to fitness centers.

# IV. Methodology

# IV. Methodology (LSTM)

## 1. Research Questions and Objectives

First and foremost, the initial step of this thesis is to decide which exercise to use to train the machine learning model for incorrect posture correction. Since the team wanted to try out their own ability, we decided to go for 5 exercises. Each exercises have to meet the following criteria:

* An exercise is popular among people who exercise at home or who is inexperience to
* An exercise must contain at least 1 common mistakes that affect the course of action

Based on the criteria, we decided on 4 exercises: bicep curls, basic plank, basic squat, lunge and basic sit up.

After that, the next step is to identify at least 1 or 2 common errors for each exercise and then develop a strategy for detecting them. Here are the identified common errors:

* Bicep curl: Loose upper arm, weak peak contraction and lean-back standing posture.
* Basic plank: Lower back and high back.
* Basic squat: Foot placement too tight/wide as well as for knees
* Lunge: Knee’s angles and knee over toe while going down.
* Sit up: Arm’s movement shift upwards



*Figure 2. Image of a person doing bicep curls*

*Source: https://www.menshealth.com/uk/how-tos/a748583/dumbbell-bicep-curls/*

## 2. Data Collection and Preprocessing

### 2.1 Self collect data

The data contains 2 types of videos or records of people performing exercises that are collected through the team's self-recorded videos of us (there are a total of 4 contributors for this) which each of us must do these:

* Record 15 videos of each exercise, in these 15 videos are made from 3 different camera angle from low/middle/high and from the 3 angles, 5 points of view is recorded from the camera which are facing left, facing right, facing front, facing front left and facing front right if the participant performs the exercise naturally.
* Record 6 videos of each exercise, in these 6 videos are made from 3 different camera angles from low/middle/high and from the 3 angles, 3 points of view are recorded from the camera which are facing left, facing right, facing front if the participant performs the exercise incorrectly.

For the videos provided by the contributor, they must meet the following criteria:

* For exercises that involve being still, such as Plank, the maximum duration of the exercise is 13 seconds. For exercises that require movements, such as bicep curls, participants must perform the exercise around 10-13 repetitions per video.
* The video is shot in an environment with good lighting so the person’s body can be seen.
* The participant must be in frame throughout the video.

### 2.2. Public Dataset from Internet

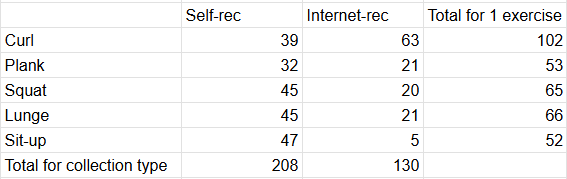
With an exercise such as Plank, there is not much movement during the exercise, while we also record videos of us doing plank in the self-record dataset, we find that taking images of planks exercise online, extending it to multiple frames which the frames are the exact same of the image and make it as a synthesis videos.

The public dataset is from 2 sources with the first being Kaggle which is an online community platform for models or custom datasets of multiple fields of work, the found dataset contains 2 folders of 2 exercises (which are bicep curl, squat), each folder contains videos of people correctly doing the corresponding exercise on multiple angles. The second source is Youtube or similar platforms which are platforms that are used for people to upload many kinds of videos and exercise is one of them. We download and crop the videos of 5 exercises and place them into the corresponding folder to join it with the general folder.

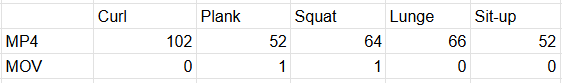
### 2.3. Data preprocessing/details

After the videos are finished with its collecting phase, they are then passed through video editing tools that are used mostly for cropping videos, removing unnecessary parts of them which are scenes that the person is not doing the intended exercise. These cropped videos will then be exported back with the same type of video file as well as their resolution and size.

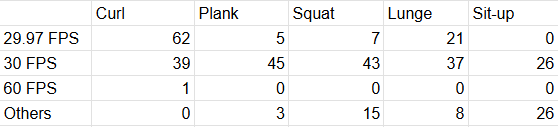
The videos we collected can be seen below:



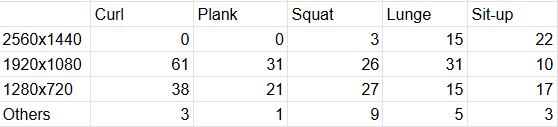
*Table 1. Numbers of videos for each exercise taken*



*Table 2. Numbers of videos corresponding to video file type*



*Table 3. Numbers of video corresponding to FPS*



*Table 4. Number of video corresponding to resolution*

## 3. Feature Selection and Engineering

### 3.1. Feature Selection & Extraction

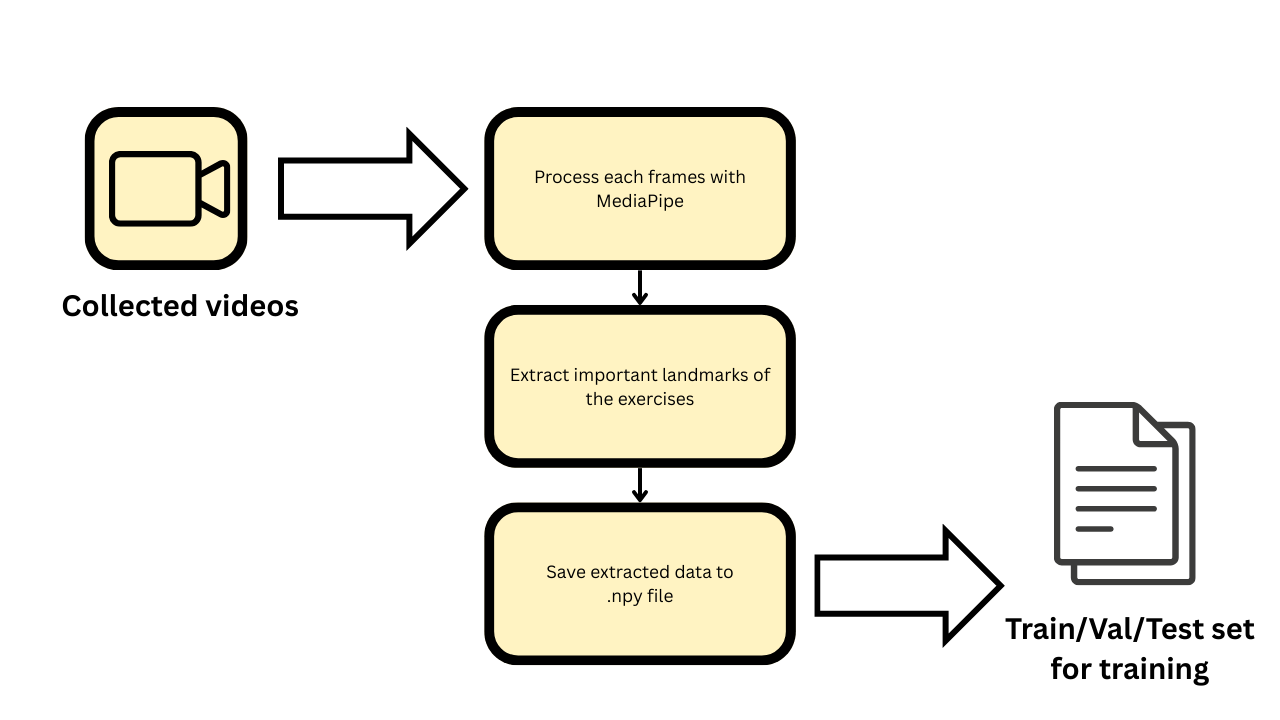
Out of all the libraries that are specialized in detecting human poses and after considerations of what our project is aiming for, our top priority is to find a library that is just sufficient enough for detecting exercises which alone does not need it to be too complex. We decided to go for MediaPipe, which its strong side is its own processing speed and a more resource-friendly library and a plus which is its advantage to do real time work. While we acknowledge that there are libraries like OpenPose or MMpose that offer these kind of more complex detections to improve our overall results, the trade off is they are too time consuming to use both from the input video for training and the output video for result as well as being resource-intensive for a result that is not much different than what MediaPipe can offer.

Now that the library is chosen, we move to the next step is to convert video frames from BGR to RGB and process them through the MediaPipe model which holds the normalized 2D coordinates of the detected pose landmarks which at core is a list of “Landmark” objects and in MediaPipe, there are 33 “Landmark” objects which corresponds to the 33 key points of the human body (e.g., nose, left shoulder, right hip, left ankle, etc.) that MediaPipe Pose detects that can be shown with the figure below:

And in each of these “Landmark” object has four numerical attributes:

* X: The horizontal coordinate of the landmark, normalized to [0.0, 1.0] by the image width.
* Y: The horizontal coordinate of the landmark, normalized to [0.0, 1.0] by the image height.
* Z: The depth coordinate of the landmark, normalized to [0.0, 1.0]. The depth is relative to the hip center, with the origin at the center of the hips. Positive values mean the landmark is closer to the camera, and negative values mean it's further away.
* Visibility: A float value in [0.0, 1.0] indicating the likelihood of the landmark being visible (and not occluded) in the image. A higher value means higher visibility confidence.

With the 4 values for each of these 33 “landmark” objects are then flattened into a single 132-element 1D NumPy array (33 \* 4 = 132 features per frame). If no pose is detected, an array of zeros is used, all of this can be portrayed with the figure below:



*Figure 3. LSTM Data processing*

### 3.2. Feature Engineering

An exercise is a dynamic set of actions, if taken only a frame of that video, it would be impossible to tell if the person is at the top, middle, or bottom of the movement. Exercise involves a sequence of movements over time, so it is reasonable to use sequences of frames instead of individual frames to understand the temporal dynamics of an exercise. This also allows the model to differentiate between different exercises based on their motion patterns.

Videos can be of different durations, so for our network model, it is crucial that every input needs to have the exact same number of frames which we later then decided to pick 30 frames for each sequence and this comes with a problem which our video dataset has many long videos as well as short videos and to achieve this we use two ways:

* Sampling: Which is used for longer videos. The general problem is if a video is longer than 30 frames, simply taking the first 30 frames might miss out important parts of the exercise that happen later but taking 60 frames would make the input too long for our fixed-size model. The way we solve this is to evenly sample frames from the entire video which means calculating intervals to pick frames that are spread out across the video’s duration. Like for a 60-frame video, we might pick every second frame, for every 90-frame video, every third frame and so on.
* Padding: Which is for shorter video. Opposite to the longer videos, videos that are shorter than 30 frames won't have enough data to fill the 30-frame sequence. So we padded the sequence. What this means is we take all the existing frames and then for the remaining slots, we repeat the keypoints from the last available frame.

And after extracting the sequences, we save them into individual files on disk (as .npy file) to avoid redundant computation as well for faster training.

## 4. Model Training and Validation

This section will talk about 2 models that were used for this method, with the first one being Bi-LSTM and the second one being Bi-LSTM with an Attention block.

Bi-LSTM are made for sequential data like time-series keypoints, as they can learn long-term dependencies and patterns over time. Bi-LSTM processes the sequence in both forward and backward directions or in terms of this project being both the past and future frames for understanding the complete motion of an exercise.

Attention block allows the model to weigh the importance of different frames within a sequence. This means that the model can focus on the important moments of an exercise (e.g., the peak of a squat or the lowest point of a push up) rather than treating all frames equally. This was made for the reason of enhancing the model’s ability to discern subtle differences between exercises.

In the models come with these specifications below:

* Input: Takes sequences of 30 frames with each frame represented by 132 keypoint features (33 body landmarks, with x,y,z coordinates and visibility value).
* Bi-LSTM Layer: Process these sequences using 256 hidden units both forward and backward, capturing temporal patterns and dependencies.
* Attention Layer: A custom mechanism that learns to assign varying importance to different frames within the 30-frames sequence, enabling the model to focus on most critical moments of an exercise
* Dense Layers: After the attention layer, a fully connected layer with 512 neurons processes the weighted features, followed by a dropout layer to prevent overfitting
* Output Layer: A final dense layer outputs the probability distribution over all exercise classes using a softmax activation which means showing the likelihood of what that exercise is.

The model is trained with an Adam optimizer with an initial learning rate of 0.01 with a loss function that is categorical cross-entropy which is common to use for multi-class classification with one-hot encoded labels. With a batch size of 32 and max epoch of maximum 500 as the training can stop early if the results are already met.

## 5. Evaluation Metrics

Evaluation metrics are used to measure the quality of trained machine learning models. This part is a must for any project and there are many types of evaluation metrics to check how good the model is. These are the following metrics that were used to evaluate the trained models in this paper:

* Confusion matrix: provides a detailed overview of the classification. For a better performing model, true positive and false negative must be high while false negative and false positive should be low.
  + True positive (TP): Number of correctly predicted positive samples.
  + False positive (FP): Number of negative samples incorrectly predicted as positive.
  + True Negative (TN): Number of correctly predicted negative samples.
  + False Negative (FN): Number of positive samples incorrectly labelled as negative.
* Precision: tells the ratio of the true negative to the total of true negative and false positive.
* Recall: tells the proportion that the model is accurately classifying the true negatives which can also be called sensitivity.
* F1 score: defined as the mean of precision and recall. The better the precision and recall, the better the F1-score.
* Support: The number of actual occurrences of that exercise class in the test dataset.
* Accuracy: The average of the total number of the true negatives of all classes divided by the number of support..
* Macro avg: metric to check the unweighted overall performance of precision, recall, f1-score with class imbalance by taking the values of the precision from the 5 exercise classes and divide it by the number of exercise classes which is 5, same can be said for recall and f1-score.
* Weighted avg: the average of the metrics, weighted by the support for each class by multiplying the precision of each class with its numbers of support, then all these values are added up and divided by the total number of support of all exercise classes. Same can be said with recall and f1-score.

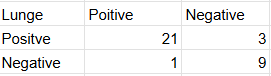
We took a total of 34 support that are the same for the test on each model below to ensure the evaluation is as accurate as possible. The evaluations can be calculated with these tables below, these values are rounded into 2 decimal digits:

* Bicep Curl: Precision = 0.75, recall = 0.75, F1 Score = 0.75



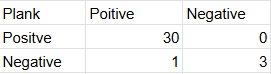
*Table 5. Confusion matrix for the Bicep Curl class*

* Lunge: Precision = 0.75, recall = 0.9, F1 Score = 0.82



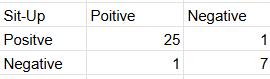
*Table 6. Confusion matrix for the Lunge class*

* Plank: Precision = 1.00, recall = 0.75, F1 Score = 0.86



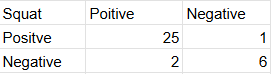
*Table 7. Confusion matrix for the Plank class*

* Sit-up: Precision = 0.88, recall = 0.88, F1 Score = 0.88



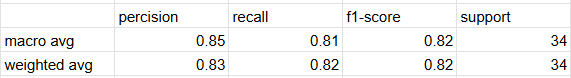
*Table 8. Confusion matrix for the Sit-up class*

* Squat: Precision = 0.86, recall = 0.75, F1 Score = 0.80



*Table 9. Confusion matrix for the Squat class*

From all the value above can the model give out the accuracy = 0.82 with all the value of macro average and weighted average below:



*Table 10. Macro average and weighted average for precision, recall, f1-score*

## 6. Implementation Plan

The implementation of this model is purely on Python through the usage such as Mediapipe for most of the work from annotating the videos for the dataset to using it for detecting body key points of a person when running the model for the results.

As for the dataset, after we finished extracting the sequence of each video, they are saved in individual files in a dedicated folder. This allows us to continue extracting key points or shorten the training phase time as this step takes the longest and most computationally expensive process in our model training phase.

To combat the issue of that process taking a lot of computing resources and not damaging our devices we decided to move our training process entirely to Google Colab, which is a coding site provides free resources as well as offering multiple utilities such as accessing folders in other applications in its own line of application for this long session of data extracting although the downside of that being the time it takes for this process to finish quintuples comparing to doing it on a local hard drive but the trade of is acceptable as it does not have a large impact in our model total development time.

We also use Streamlit which is a web platform which helps us upload our model to the internet which will be used for making a web application for our model so that users can access.

## 7. Ethical Considerations

### 7.1. Dataset ethics

For the data collection phase, the participants, which is the team ourselves and our relatives, have given their consent to use their image as well as video which include them doing the exercises as well as the environment of where they did the exercises.

### 7.2. Input/Output ethics

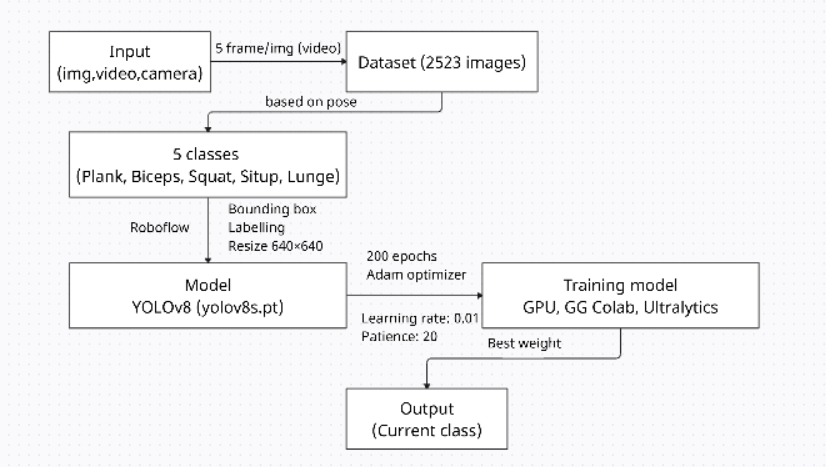
By using our web application, the users have agreed to give us the rights to use the content of that video for our model to run its work. Aside from that, we do not use these videos for other means such as data collecting for our own model or for any third parties.

# IV. Methodology ( YOLOv8)

## 1. Research Questions and Objectives

Besides LSTM, what other methods can be used to recognize exercise movements such as plank, bicep curl, squat, lunge, and sit-up from video-extracted images?

The objective is to explore whether YOLOv8 (You Only Look Once version 8)[[4]](#3grajag3y4c), a spatial object detection model, can effectively classify these five types of exercises based solely on visual features from still frames, without relying on sequence-based models like LSTM. The aim is to build, train, and evaluate a YOLOv8-based recognition system using both public and self-collected data, and to analyze its performance in terms of detection accuracy, precision, and general applicability to real-world use. [[5]](#8ykmaie1rjud)



*Figure 4. System Architecture Diagram*

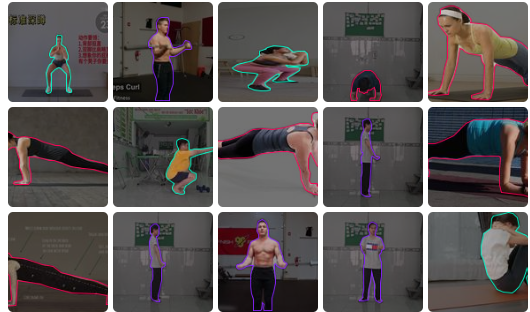
## 2. Data Collection and Preprocessing

The dataset used in this project consists of 2,523 annotated images, collected from two main sources to ensure both diversity and real-world relevance:

* Public data: A subset of the images was obtained from an open-access dataset on Kaggle and video on Youtube. [[12]](#q7x92cx2g474)
* Self-recorded data: The remaining images were extracted from videos that I recorded personally, performing the five target exercises (plank, bicep curl, squat, lunge, and sit-up). To enhance dataset variability, these recordings were captured under different camera angles, lighting conditions, and positions.

For frame extraction, each video was processed using OpenCV, sampling at a rate of 1 image every 5 frames. This strategy balanced motion diversity and redundancy, ensuring that the dataset captured distinct postures and transitional phases of movement.

The annotation process was carried out using Roboflow [[6]](#1zoaphwms0mu), where bounding boxes were manually drawn around the subject and labeled into one of the five classes. Roboflow also provided dataset management and direct export in YOLO format [[7]](#u5s0uppc951u), fully compatible with the Ultralytics YOLOv8 training pipeline.



*Figure 5. Dateset after bounding boxes*

Preprocessing included the following steps:

* Resizing: All images were resized to 640×640 pixels, meeting YOLOv8 input requirements.
* Data cleaning: Images with unclear postures, excessive motion blur, poor lighting, or incorrect annotations were excluded. Only frames with clearly visible and distinguishable movements were retained.

## 3. Feature Selection and Engineering

For this project, I used YOLOv8, a deep learning-based object detection model that does not require manual feature selection. Instead of traditional handcrafted features (e.g., edge detectors, keypoint descriptors), YOLOv8 automatically learns hierarchical visual features directly from the input images during training using convolutional neural networks (CNNs).

The primary inputs to the model are RGB images that have been resized to 640×640 pixels. Each image contains a single individual performing one of five predefined exercise movements (plank, bicep curl, squat, lunge, or sit-up). These images are annotated with bounding boxes and class labels, which indicate the location and category of the movement. These annotations serve as ground truth targets, while the image pixel values act as raw input features that the model processes through convolutional layers.

YOLOv8 is a deep learning model designed for object detection, it automatically learns features during the training process without the need for manually crafted descriptors. Through its convolutional architecture, YOLOv8 extracts multi-level representations—ranging from edges and textures in early layers to complex shapes and semantic information in deeper layers.

## 4. Model Training and Validation

For this project, I chose to use YOLOv8s, a lightweight yet powerful object detection model developed by Ultralytics. YOLOv8 is well-suited for real-time applications due to its speed, accuracy, and ease of deployment. Among the YOLOv8 variants, the “s” (small) version was selected to balance performance with faster training time and lower computational cost[[8]](#lom28jomkjt4), which is ideal for prototyping and running on limited hardware such as Google Colab[[9]](#2wt8y2it90gm).

The model was trained for 200 epochs on Google Colab using GPU acceleration. Training configurations included a batch size of 16, a learning rate of 0.01, and input image size fixed at 640×640 pixels. The dataset was split into 70% training, 20% testing, and 10% validation. No cross-validation was applied, but performance was continuously monitored across epochs through metrics like precision, recall, and mean average precision (mAP).

The choice of YOLOv8 over other models such as LSTM, CNN+LSTM hybrids, or pose-estimation-based methods is due to its end-to-end capability of detecting and classifying objects (exercise poses) directly from raw images, without requiring sequential temporal data or skeleton tracking. This makes it more robust for frame-based analysis from static images.

## 5. Evaluation Metrics

To assess the performance of the YOLOv8 model in exercise movement recognition, I use standard object detection evaluation metrics, including Precision, Recall, mAP@0.5, and mAP@0.5:0.95. These metrics evaluate both the classification accuracy and the localization ability (i.e., how well the bounding box fits the object).

Precision (0.963)

Precision measures the proportion of true positive predictions among all positive predictions made by the model. A high precision indicates that the model makes very few false positive predictions — in other words, when the model predicts a movement, it is usually correct.

Recall (0.967)

Recall quantifies the model’s ability to identify all relevant instances in the data. A high recall means that the model rarely misses a movement that appears in the image.

mAP@0.5 (0.982)

Mean Average Precision at IoU threshold = 0.5. This metric evaluates the average precision across all classes, considering a prediction correct if the Intersection over Union (IoU) with the ground truth is greater than or equal to 0.5. It reflects how accurately the model detects and classifies objects.

A

mAP@0.5:0.95 (0.775)

This is the average of mAP calculated across multiple IoU thresholds from 0.5 to 0.95 (in steps of 0.05). It is a more rigorous metric that assesses the model’s ability to precisely localize the object, not just roughly detect it.

mA

## 6. Implementation Plan

## The implementation of the model is carried out using the Ultralytics YOLOv8 framework in a Python environment. The training and deployment pipeline is designed to support real-time action recognition, from data preparation to inference.

## To annotate the dataset, I used Roboflow, which allows for manual labeling of images and automatic export in YOLO format. The annotated dataset is then used to train the model in Google Colab, which provides access to free GPUs, making it suitable for running long training sessions efficiently without relying on local hardware. The model is trained and evaluated within this cloud-based environment using PyTorch [[10]](#4y1o3s319za4).

## OpenCV is used extensively for preprocessing and post-processing tasks, such as extracting frames from videos, resizing images, and visualizing detection results during inference[[11]](#cfvp43q2hfsv). Once training is complete, the model is saved in .pt format and integrated into a Python script capable of performing live inference using webcam input or pre-recorded videos. This script leverages YOLOv8’s inference functions to detect and classify exercise movements in real time.

## 7. Ethical Considerations

## This project fully adheres to ethical principles related to data privacy, fairness, and responsible AI usage. All self-recorded videos used in the dataset were created by us, without involving other individuals. This ensures that no personal or biometric data from third parties is collected or processed, thereby avoiding any infringement on privacy rights. Additionally, the publicly sourced data from Kaggle is used in accordance with its licensing terms, ensuring proper and fair use of open datasets [[12]](#q7x92cx2g474).

## To promote fairness and reduce bias, the dataset includes diverse video samples captured under different lighting conditions, camera angles, and body positions. Although no data augmentation techniques were applied, this natural variation contributes to more generalizable model performance across different real-world scenarios. During data cleaning, any unclear or ambiguous postures were removed to maintain label consistency and reduce noise in the training data.

## Importantly, the model is intended solely for educational, research, and exercise-support purposes. It is not designed for surveillance, behavioral monitoring, or commercial exploitation. By keeping the application scope clear and focused, the project minimizes the risk of misuse and aligns with ethical standards in machine learning deployment [[13]](#n3ywj1y7czeg) [[14]](#prf5rl6yv22k)

# IV. Methodology (Exercise error detection/Repetition counting)

## 1. Research Questions and Objectives

The project's objective is to design, implement, and evaluate a multi-stage system for detailed exercise form analysis. The aim is to move beyond simple exercise classification and address the following goals:

1. To extract and utilize human skeletal data (pose keypoints) as the primary feature set for form analysis.
2. To develop a hybrid approach, combining rule-based biomechanical logic with specialized machine learning models, to detect common posture errors.
3. To create a real-time feedback loop that can count repetitions, track duration, and display specific error messages to the user.

## 2. Data Collection and Preprocessing

The dataset for the form correction system was not based on raw images, but on structured keypoint data derived from a public video collection available on GitHub. This dataset provided a robust foundation for training the various analysis models.

Data Source and Scale: The data was sourced from the "Exercise-Correction" repository by NgoQuocBao1010 [[1]](#79wh54q86jsj), supplemented with a few additional self-recorded videos. This repository contains labeled keypoint data extracted from videos showing both correct and incorrect exercise forms. The scale and class distribution for each exercise are detailed below:

* Bicep Curl:  
  Total Dataset: 15,976 frames.  
  Train Set: Comprised 15,372 frames (8,238 "Correct", 7,134 "Lean-back-error").  
  Test Set: Comprised 604 frames (339 "Correct", 265 "Lean-back-error").
* Lunge (Stage Classification):  
  Total Dataset: 25,449 frames.  
  Train Set: Comprised 24,244 frames (8,232 "Down", 7,864 "Init", 8,148 "Mid").  
  Test Set: Comprised 1,205 frames (416 "Down", 402 "Init", 387 "Mid").
* Lunge (Error Classification):  
  Total Dataset: 19,014 frames.  
  Train Set: Comprised 17,907 frames (9,114 "Incorrect", 8,793 "Correct").  
  Test Set: Comprised 1,107 frames (561 "Incorrect", 546 "Correct").
* Plank:  
  Total Dataset: 29,230 frames.  
  Train Set: Comprised 28,520 frames (9,904 "Correct", 9,546 "Low back", 9,070 "High back").  
  Test Set: Comprised 710 frames (241 "High back", 235 "Low back", 234 "Correct").
* Squat:  
  Total Dataset: 5,013 frames.  
  Train Set: Comprised 4,160 frames (2,127 "down", 2,033 "up").  
  Test Set: Comprised 853 frames (430 "down", 423 "up").

Data Extraction and Labeling: The source project had already performed the keypoint extraction using MediaPipe Pose [[15]](#u3ddks42ix9a). Each row in the datasets represents the flattened keypoint coordinates of a single video frame, accompanied by a label defining the posture's correctness or stage.

Data Splitting and Preprocessing: For training the machine learning models, the datasets were split into training and testing sets using an 80/20 ratio. To ensure model performance was not biased by the scale of coordinate values, a StandardScaler from scikit-learn [[16]](#nisaqoyint29) was fit on the training data and used to normalize the feature vectors for both training and testing sets.

## 3. Feature Selection and Engineering

This project employs an engineered-feature approach, where domain knowledge of biomechanics is used to create meaningful inputs from the raw keypoint data.

Base Features: The initial features are the coordinates of the 33 keypoints provided by MediaPipe [[15]](#u3ddks42ix9a). For specific models, a subset of the most relevant landmarks was selected to reduce dimensionality and focus on the most informative joints.

Engineered Features for Rule-Based Systems: For the Squat and Sit-up analyses, which are primarily rule-based, the features were meticulously engineered. The thresholds for these rules were not arbitrary but were derived from statistical analysis of the dataset:

* Squat Analysis: By analyzing the mean and distribution of the ratios between shoulder-width, feet-width, and knee-width across 853 sample data points, specific thresholds were established. For instance, the conclusion that a feet/shoulder width ratio outside the range of [1.2, 2.8] indicates incorrect foot placement was based on the observation that the mean ratio in the dataset was 1.8. Similar data-driven analysis was performed to set the thresholds for knee placement in different stages of the squat.
* Sit-up Analysis: The angle thresholds (up\_threshold=90, down\_threshold=120) and the leg\_movement\_threshold=0.05 were determined through empirical testing and iterative refinement to find values that best separated the exercise stages and reliably detected instability across various test videos.

Features for ML Models: For the Bicep Curl, Lunge, and Plank models, the feature vectors consist of the flattened, normalized coordinates of the pre-selected important landmarks. The models learn the complex non-linear relationships between these keypoint positions to make predictions.

## 4. Model Training and Validation

A systematic approach was taken to select the best machine learning model for each specific task.

Model Selection Process: For each exercise (Bicep Curl, Lunge, Plank, Squat), a suite of seven common classification algorithms was evaluated: Logistic Regression (LR), Support Vector Machine (SVC), K-Nearest Neighbors (KNN), Decision Tree (DTC), SGD Classifier (SGDC), Gaussian Naive Bayes (NB), and Random Forest (RF). All models were trained on the 80% training split of the respective dataset using their default hyperparameters from scikit-learn [[16]](#nisaqoyint29). Their performance was then compared on the 20% test set. The model demonstrating the highest overall performance, particularly in terms of F1-score and Accuracy, was selected for the final implementation.

Final Model Choices:

* Bicep Curl (Error): Although Random Forest showed slightly higher metrics, K-Nearest Neighbors (KNN) was chosen for its excellent balance of high accuracy (99.8%) and computational efficiency, making it ideal for real-time applications.
* Lunge (Stage): Support Vector Machine (SVC) was selected. While KNN had a marginally higher accuracy, SVC provided highly robust and stable predictions for distinguishing the three distinct stages of the lunge, achieving an accuracy of 99.3%.
* Lunge (Error): Logistic Regression (LR) was chosen for its simplicity and near-perfect performance in this binary classification task, achieving 98.9% accuracy with a good balance of precision and recall.
* Plank (Posture): Logistic Regression (LR) was the clear top performer, achieving an outstanding accuracy of 99.6% in this multi-class classification problem.
* Squat (Stage): Logistic Regression (LR) was selected, achieving 99.8% accuracy and demonstrating robust performance for this binary stage classification task.

This methodical selection process ensures that each component of the form correction system utilizes a model that is not only highly accurate but also well-suited for the specific complexity of its task.

## 5. Implementation Plan

The system is implemented as a modular, real-time pipeline in Python, orchestrated by a central ExerciseDetector class.

Core Technologies: OpenCV is used for video capture and rendering. MediaPipe Pose [[15]](#u3ddks42ix9a) serves as the foundational layer for extracting skeletal keypoints. Scikit-learn [[16]](#nisaqoyint29) and Pickle are used for training, saving, and loading the machine learning models and data scalers.

System Error/Counting Architecture:

1. An initial exercise type is selected.
2. The main loop captures a frame using OpenCV.
3. The frame is passed to MediaPipe Pose to get the landmark coordinates.
4. The ExerciseDetector class routes these landmarks to the appropriate analysis module (e.g., PlankPoseAnalysis, SquatPoseAnalysis).
5. Each analysis class contains the specific logic for that exercise. It either applies the data-driven rule-based engine (for Squat, Sit-up) or uses its pre-trained ML model (loaded from .pkl files for Bicep, Lunge, Plank) to analyze the pose.
6. The module returns a dictionary containing results (rep count, stage, error flags, feedback messages).
7. Finally, OpenCV is used to draw the skeletal overlay, status boxes, and all feedback text onto the original frame, which is then displayed to the user.

## 6. Ethical Considerations

## The development and intended use of this form correction system adhere to key ethical principles.

## Data Provenance and Privacy: The project utilizes a publicly available, anonymized dataset of keypoints, respecting the privacy of the individuals in the original videos. The real-time system processes video locally and does not store or transmit user video data, ensuring a high standard of privacy.

* Fairness and Bias: The system is based on MediaPipe Pose, which is trained on a diverse dataset, making it relatively robust. The form correction models were also trained on a substantial dataset containing thousands of examples per class. However, it is acknowledged that performance may vary for individuals with body types or performing exercise variations not represented in the training data.
* Accuracy and Responsibility: The high accuracy scores (over 98% for all selected models) demonstrate the system's technical reliability. However, it is designed as a fitness aid, not a medical or diagnostic tool. The feedback is an algorithmic interpretation and should be used as guidance. The application's purpose is strictly for educational and motivational support in a personal fitness context and is not intended for surveillance or mandated performance evaluation.

# V. System Design and Implementation

## 1. Data Flow and Processing

**1.1 LSTM**

In the first model, the raw data before involving in any processing or alteration is taken as either a MP4 or a MOV video file which comes with multiple resolution types, these raw videos are then taken into a variety of video editing tools which are mostly used to crop the unneeded part of the videos so that the videos can only have the scene where the person is actually training without other activities. This will give out the corresponding file type videos same as the raw videos with the same resolution.

After that the cropped videos files are then processed into folder which holds exactly 30 or 60 numpy files that represent for the the number of the frame sequence which is a 132-element 1D NumPy array (33 \* 4 = 132 features per frame), if no pose is detected, an array of zeros is used.

A folder containing all of the processed folder is then put into the model for training by splitting it into 3 sets: training set, validation set and testing set which take respectively 75%/15%/10% of the total which after finished making callbacks and fixing them to use a proper file path will we be getting a new .keras file for training, after training we will get our final weight file under the form of a .h5 file.

As for the part when the model is being used, the input data will be the user input video which is treated as pixel data then it gets transformed into keypoint data which every frames of it gets turned into a single one-dimensional array of numerical values of the 4 values x,y,z and visibility of MediaPipe. When the sequence reaches its required length it will then pass to the pre-trained model which is the .h5 file which translates the model's predictions and counts and then finally return it back to visual data for the user as a MP4 file.

**1.2. YOLOv8**  
The data flow of our system begins with data collection from multiple sources. We use publicly available datasets from Kaggle and YouTube, including both videos and live camera streams. In addition, we recorded our own videos from multiple angles to increase dataset diversity and improve model robustness.

After collection, the videos are processed by extracting frames at a ratio of 5 frames per image. This sampling rate is chosen to ensure that all key movements are captured without producing redundant frames. The extracted frames are then resized to 640×640 pixels to match the input requirements of the YOLOv8 model.

Next, the images are annotated and categorized into five classes corresponding to different human exercise poses (plank, bicep curl, squat, lunge, sit-up). The labeled dataset is then used to train the YOLOv8 model, which learns to detect and classify exercise movements.

Once the YOLOv8 inference is performed, the system outputs the predicted class label (one of the five exercises). However, this result is only an intermediate step. The detected class is then passed to a post-processing model responsible for Evaluating correctness of the pose (providing immediate feedback if the movement is incorrect), Counting repetitions (reps) for each exercise and Tracking progress and ensuring users maintain proper form.

Finally, the integrated system provides a complete product, where the user can see not only the recognized exercise type, but also receive real-time reminders, correctness evaluation, and rep counting through the application interface.

## 2. Deployment Strategy

### 2.1 Deployment Environment

## The system is deployed and executed directly on the local machine environment with the following components:

## Interface framework using Streamlit, displayed through the local web browser (localhost).

## Processing libraries are OpenCV to process image and video,Mediapipe to extract body keypoints,PyTorch + Ultralytics YOLO work as exercise recognition,and Scikit-learn gives detailed movement analysis (counting reps, detecting errors).

## About models,we use YOLO (best.pt) for exercise recognition.Adding detailed classification models (.pkl) for analyzing squat, plank, lunge, and bicep curl exercises. Dependencies are fully listed in the requirements.txt file.

### 2.2 Deployment Procedures

## To deploy and operate the system, the first step is to install the required libraries. It is recommended to create a virtual environment to manage the packages efficiently, and all dependencies can then be installed using the command pip install -r requirements.txt. After installation, the system follows a defined directory structure in which app11.py serves as the main file to launch the Streamlit application, b4.py functions as the backend module responsible for processing logic and overlay visualization, the model/ directory contains the YOLO.pt(best.bt) and .pkl models, the result/ directory stores the processed output videos, and the requirements.txt file specifies the required dependencies. Once the structure is set, the application can be executed by navigating to the project directory and running the command streamlit run app11.py in the terminal, after which a localhost link will be provided for access.

## During operation, users can choose between two input modes: using a webcam or uploading a video file. For webcam input, several configuration options are available, including width, height, and target FPS. If the target FPS is set higher than 30, the model will still operate at a maximum of 30 FPS because the current model only achieves that level of performance. The system then recognizes the type of exercise, analyzes posture, displays overlays containing information such as repetitions, exercise stage, and errors, and saves the output video in the result/ folder. Additional functionalities include the ability to select a YOLO.pt(we only have best.pt now) model stored in the model/ directory for exercise recognition, which allows flexible replacement or experimentation with different versions of YOLO. Furthermore, the application supports customization of key YOLO parameters such as the confidence threshold and IoU threshold, although altering these is not recommended since the default values are optimized for best performance; improper adjustments could slow down the system or lead to unexpected results. This will be improved in future updates to ensure stable performance across parameter ranges. Users may also enable or disable an audio alert (Beep), which emits a sound whenever an incorrect posture is detected, thereby providing immediate feedback. In addition, output display options are available, enabling users to either view the processed video directly within the application interface or save it in the result/ folder for later review. These functions provide a high degree of flexibility, allowing users to tailor the application to their specific needs.

## Finally, the system supports updates and maintenance by allowing the replacement or updating of YOLO.pt and .pkl models to improve accuracy. Whenever new libraries are added or existing ones are changed, the requirements.txt file must be updated accordingly to ensure smooth operation.

## 3. Scalability and Maintenance

All the libraries in the code are on the versions that can be run together as some libraries tend to require older versions of other libraries as well as the kernel itself, which is one of the tackles we have to deal with when we started this project. These library versions are still considered up to date by the newer version standards so there should be no problems in missing values or functions if the code is run under a new version of mediapipe as you only need to know what version of other libraries are. This also was made to help with further training and maintaining the system as it could be trained by a new set of data on top of the existing data it had learned for better accuracy, this also applies to when training a new exercise.

# VI. Results and Discussion

## 1. Results and Analysis

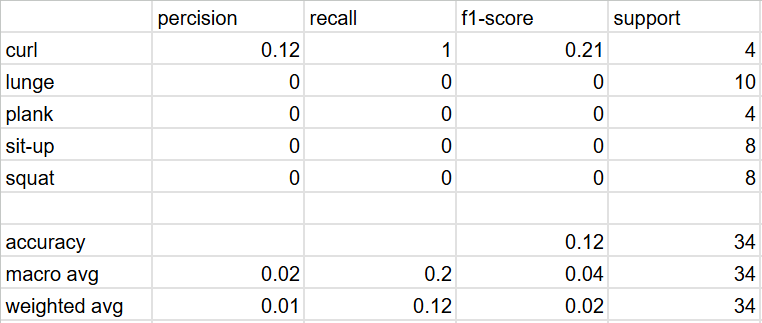
This part will be talking about the result we gain as well as comparing different models as well as different values when using to train our models including models from model 1, model 2 and the error detection, repetition counting model.

### 1.1. Model 1

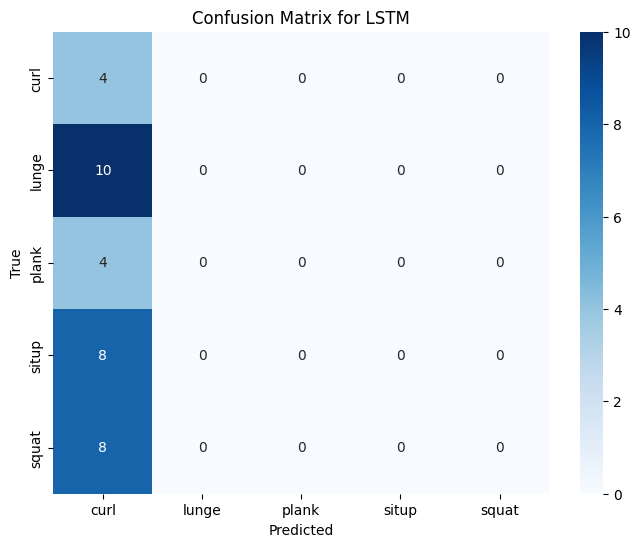
#### 1.1.1. Bi-LSTM only

##### 1.1.1.1. 30 frames sequence

The result gained from this model is not good for any practical use as it can only predict a small portion of the curl class with the other class being unable to be recognized, with its classification accuracy being 11.765% which is rounded up to be 12% in the table:



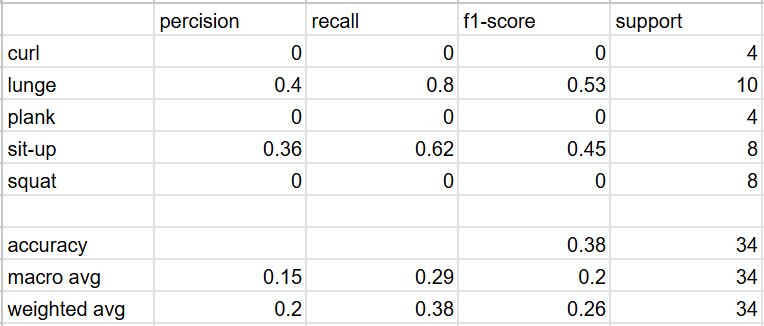
*Table 11. Model training with Bi-LSTM on 30 frames sequences*



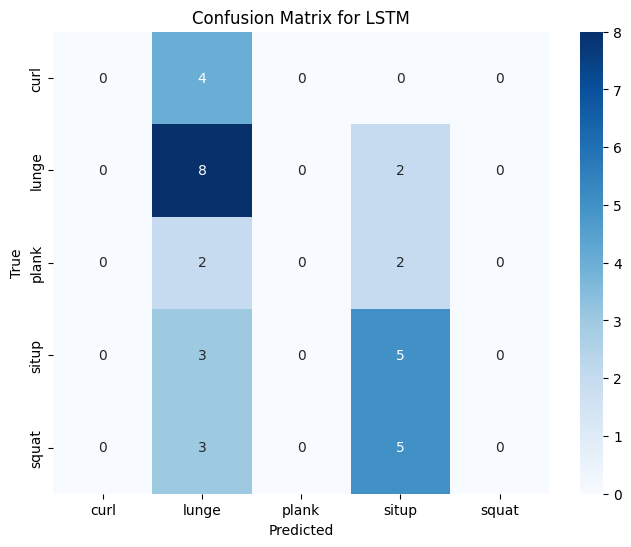
*Figure 6. Confusion matrix for Bi-LSTM on 30 frames sequence*

##### 1.1.1.2. 60 frames sequence

The result from training 60 frames sequence is partially better than training using 30 frames sequence but it can only detect actions like lunge and situp while the other 3 class being unable to be recognized, with its classification accuracy being 38.235% which is rounded down to be 38% in the table:



*Table 12. Model training with Bi-LSTM on 60 frames sequences*



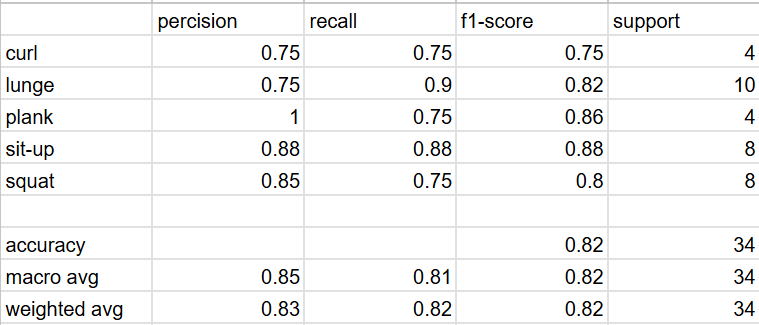
*Figure 7. Confusion matrix for Bi-LSTM on 60 frames sequence*

Through the results, we could conclude the 60 frames sequence model has a better performance between the two of them. The problem being why the 30 frames sequence model being able to detect bicep curl action but the 60 frames sequence model not able to, we concluded that for the performance of the model to improve, it solely boils down to the nature of the exercises themselves if they are a long exercise or a short one, a simple repetitive one or a more complex set of action ones and to know what is the perfect amount of frames to capture both the key movements of all the exercises. In short, increasing the frame sequence might not help improving the model performance.

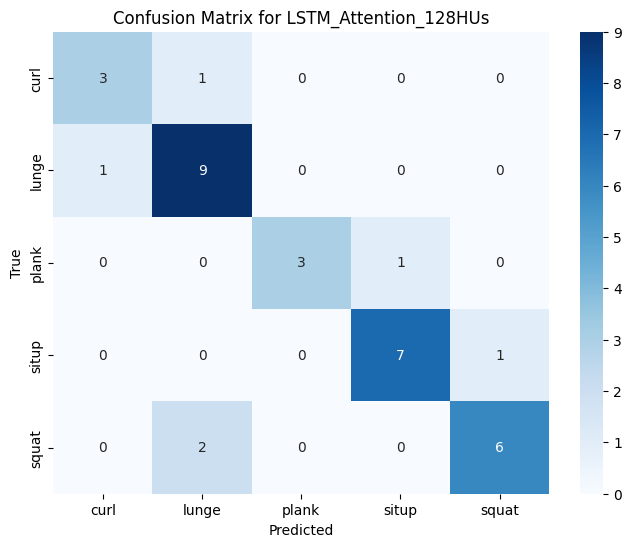
#### 1.1.2. Bi-LSTM with Attention

##### 1.1.2.1. 30 frames sequence

The result of training 30 frame sequence but now with the effect of the Attention block greatly improves its performance than the results of the model without the Attention block with it being able to detect all classes with the classification accuracy of 82.353%:



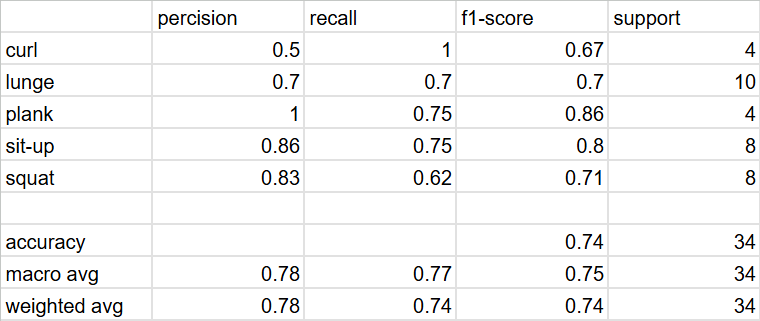
*Table 13. Model training with Bi-LSTM+Attention on 30 frames sequences*



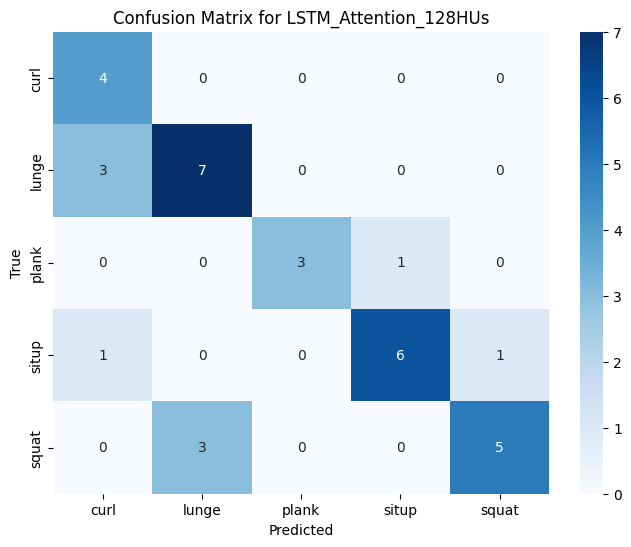
*Figure 8. Confusion matrix for Bi-LSTM+Attention on 30 frames sequence*

##### 1.1.2.2. 60 frames sequence

The result for the model is still better than both of the models that has no attention block with it being able to detect all 5 classes with the classification accuracy of 73.529% rounded up to 74% in the table:



*Table 14. Model training with Bi-LSTM+Attention on 60 frames sequences*

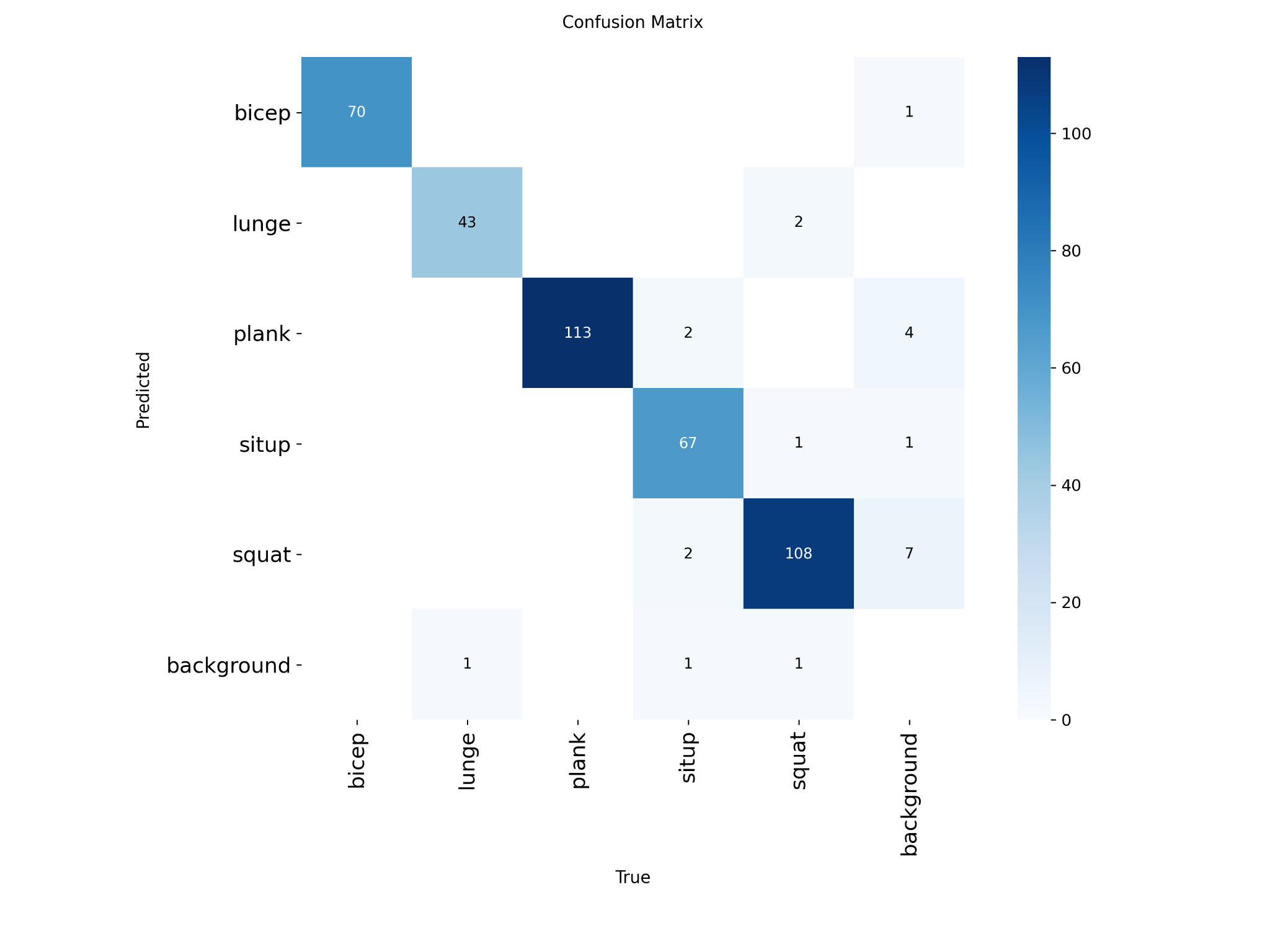


*Figure 9. Confusion matrix for Bi-LSTM+Attention on 60 frames sequence*

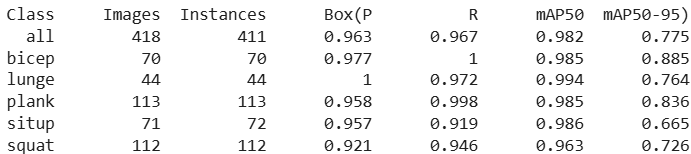
Through the results of the models that are with the Attention block, its performance is greatly increased and it also confirmed our assumption that the model does best when a correct amount of frames is being trained with the 30 frames sequence model accuracy being higher than the 60 frames sequence model.

### 1.2. Model YOLOv8

This evaluation is based on the best model tested on the validation set.



*Figure 10. Confusion matrix of YOLOv8*



*Figure 11.Evaluation metrics of YOLOv8*

The confusion matrix shows that the model performs very well across all five exercise classes, with the majority of predictions concentrated on the correct diagonal. For instance, the plank and squat classes achieve particularly high accuracy, with very few misclassifications into other categories. The sit-up and bicep curl classes are also classified correctly in most cases, though there are occasional mislabels into nearby movements. The lunge class demonstrates strong performance as well, but with slightly fewer samples compared to other classes, its robustness may be more sensitive to dataset size.

Overall, the results indicate that the model generalizes effectively to the validation data, achieving high precision and recall for all classes. Minor confusion occurs mainly between movements with visually similar poses, but the error rates remain low. This confirms that the chosen best model is reliable for exercise recognition on the given dataset.

### 1.3. Model error and analyse

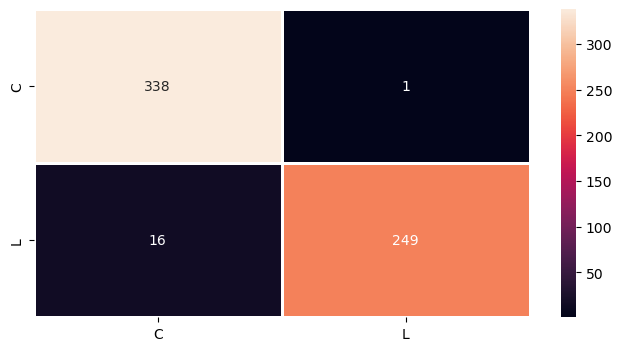
The performance of the machine learning components of the system was rigorously evaluated on the held-out test set using a range of standard classification metrics.

Primary Metrics:

* Accuracy: The main indicator of overall model performance.
* Precision, Recall, and F1-Score: Used to provide a more nuanced understanding of performance, especially in multi-class problems or where class imbalance might be a concern. These metrics were calculated on a per-class basis.
* Confusion Matrix: Generated for each model to visualize specific error patterns and understand which classes were most often confused with one another.

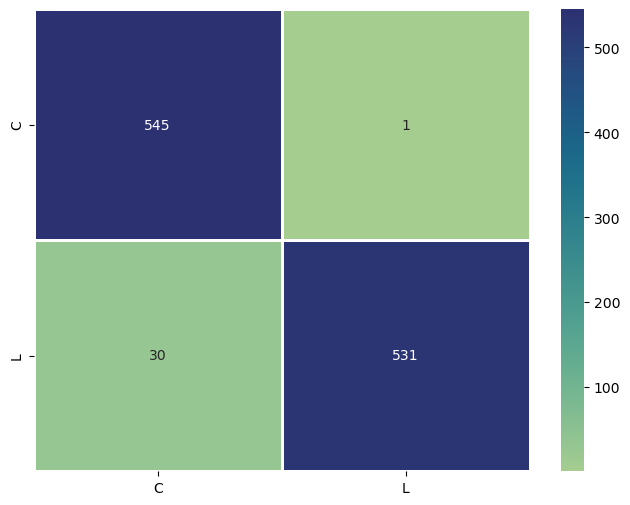
Quantitative Results: The selected models demonstrated exceptional performance on their respective test sets:

* Bicep Curl (KNN): Accuracy: 97.2%. F1-Score: 97.1%.



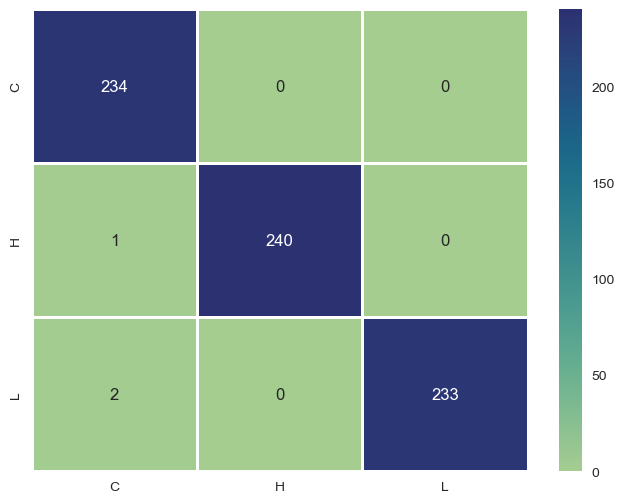
*Figure12. Matrix error Bicep Curl*

* Lunge Stage (SVC): Accuracy: 95.2%. F1-Score: [94.9%, 91.9%, 98.2%].
* Lunge Error (LR): Accuracy: 97.2%. F1-Score: 97.2%.



*Figure 13. Matrix error Lunge*

* Plank Posture (LR): Accuracy: 99.6%. F1-Score: 99.6%.



*Figure 14. Matrix error Plank*

* Squat Stage (LR): Accuracy: 99.4%. F1-Score: 99.4%.

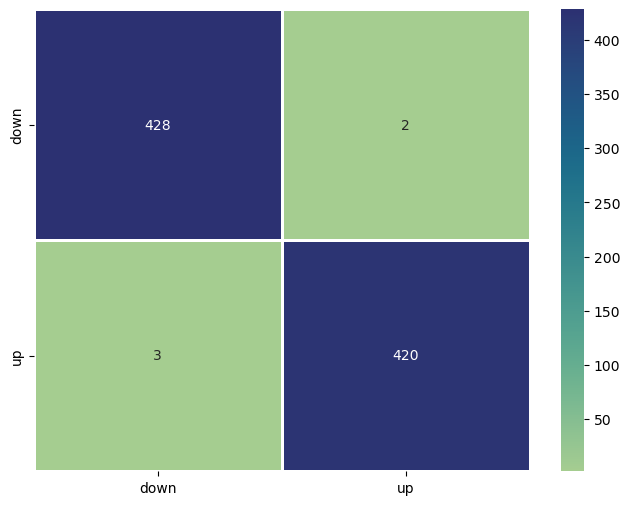


Figure 15. Matrix stage Squat

These high metric scores validate the effectiveness of the chosen features and models, providing a strong quantitative foundation for the system's reliability in real-time analysis.

## 2. Discussion

## The results of the system indicate that real-time exercise recognition and posture analysis can be effectively implemented by combining transfer models for exercise detection with machine learning models for detailed posture classification. The system successfully identified five types of exercises—squat, plank, lunge, sit-up, and bicep curl—while also providing additional feedback such as repetition counts, exercise stage detection, and posture error alerts. These results directly address the research question of whether computer vision and deep learning models can support home-based training through automated monitoring and feedback. When comparing two types of action recognition models, our group determined that YOLO slightly outperforms Bi-LSTM + Attention in detecting exercises within action sequences.

## Compared with related studies, the system shows similarities in applying deep learning techniques to the fields of sports and rehabilitation, particularly through the use of pose estimation (Mediapipe) and object detection (YOLO) for exercise monitoring. However, the system extends beyond these approaches by integrating additional features such as overlay visualization, audio alerts, support for both webcam and video file input, and the inclusion of more exercise types, thereby improving usability for general users.

## Nevertheless, several limitations remain. First, the system’s processing performance is capped at 30 FPS, which may pose restrictions in scenarios requiring higher frame rates. Second, accuracy can be influenced by environmental factors such as lighting conditions, camera quality, user distance from the camera, and complex backgrounds, which may occasionally result in false detections. Third, the current models are trained on specific exercises and therefore lack the ability to generalize to more complex or compound movements without additional training data. Furthermore, because the training dataset was designed for individual users, the system cannot detect multiple people within the same frame.

## In a broader context, these results demonstrate the potential of computer vision–based systems to support personal fitness training, especially for individuals without access to professional instructors. If the current limitations can be addressed, the system could evolve into a powerful fitness assistant. However, it should still be considered a supplementary tool rather than a full replacement for expert guidance, particularly for advanced or injury-prone exercises. Future directions may include expanding the dataset, optimizing model performance, and incorporating adaptive learning mechanisms to deliver personalized feedback. At a more advanced level, the system could be improved to detect specific posture errors and provide detailed corrective instructions to users.

## 3. Recommendations

## Based on the results achieved and the limitations identified, the group proposes several recommendations to improve the system in the future. First, the training dataset should be expanded in both the number of participants and the diversity of exercises. Adding data from different individuals under varied environmental conditions (lighting, distance, and camera angles) will allow the model to generalize better and reduce errors in real-world scenarios.

## Next, the performance of the model should be optimized to increase the processing speed beyond the current limit of 30 FPS.

## In addition, the system should be upgraded to support multi-person detection within the same frame, thereby extending its applicability to fitness classes or group training sessions. At the same time, it is necessary to integrate mechanisms for detecting specific posture errors and provide detailed corrective instructions, enabling users to improve their technique more effectively and safely.

## Finally, a promising direction is to develop a personalized feedback system based on each user’s training history. This would allow the system to provide recommendations tailored to the progress and abilities of each individual, transforming the application into an intelligent training assistant rather than merely a basic repetition counter.

# VII. Conclusion

## 1. Summary of Findings

While working on this project, the team was able to further strengthen our knowledge with many machine learning techniques, theories and tools. Using the MediaPipe and Yolov8 framework, all of the trained models produced results that met our expectation with our aim to go with a rather big number of exercises to work with by detecting between 5 exercises (bicep curl, plank, squat, lunge and sit up) with positive results.

## 2. Contributions and Reflections on the Project

This project boasts its ability to run multiple machine learning models inside of its system without hindering its performance like how other works that are involved in both detecting and recognizing errors of exercises such as 3D Pose Base Feedback for Physical Exercise [[17]](#78h1rrjskwl3). This gives a different approach to this field of work that does not have to use too much computational resources yet still reaches the same level of performance or even above as these works tend to only recognise one style of training of that exercise.

There is also aspect the team need to look back, even though the system we created work well for 5 exercises which is already a huge leap of us comparing to the usual work of detecting only between 4 exercises or less like the Exercise\_Recognition\_A [[2]](#ikqtp6hfa4e0) but we still find our own system to still have some challenges when having to strongly decide what the exercise the user is currently doing when it is still have such weak confidence on detecting the exercises even though it is still able to count the correct exercise it needs to after the user finish their cycle of it.

This research was made to see if we can make a system that is simpler yet is as or more effective than the common way that is usually used in this field as well as to make it be able to simplify the process of detecting both the exercises and the incorrect poses of the exercises at the same time which surpass all the other works before. And we have managed to do those 2 aims really well as the 2 models of ours have nearly similar results spreading from accuracy, processing speed, how well it can handle situations such as actions speed or where the user put their camera, etc. Although there can be things we need to look back as we are still unable to make the model which was made from a different method from the usual methods that are used in this field of work to be significantly better, but to have it run on par with it, we find it a great step for future works like this in the future.

## 3. Limitations and Future Work

Due to limited time, knowledge on web creation and budget. The team only creates a web page on a local host instead of going global on only 1 model we deem most preferable to be shown. With this we have prepared some future ways to fix this issue as well as adding and improving some additional functions:

* Making a web that can accommodate 2 models with 2 different frameworks without having problems relating to their conflicts.
* Extending the dataset by adding more exercises from new ones to old to increase the variety and accuracy of exercise the system can detect.
* Researching on ways to hasten the time it takes to finish processing an input video to get the result.
* Researching on ways to let multiple people in the same capturing screen to be detected individually and get feedback differently to each person without hindering too much to the system performance.

# VIII. References

[1]NgoQuocBao1010. (2023). *Exercise-Correction* [Source code]. GitHub. [Accessed:19/08/2025 https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

[2]Chrisprasanna. (2022). *Exercise\_Recognition\_AI* [Source code]. GitHub.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://github.com/chrisprasanna/Exercise_Recognition_AI/blob/main/README.md>

[3]Jacoo-Zhao. (2022). *3D-Pose-Based-Feedback-For-Physical-Exercises* [Source code]. GitHub.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://github.com/Jacoo-Zhao/3D-Pose-Based-Feedback-For-Physical-Exercises?tab=readme-ov-file>

[4]Ultralytics. (2024). *YOLOv8 models documentation*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://docs.ultralytics.com/vi/models/yolov8/>

[5]LearnOpenCV. (2023). *YOLOv8 object tracking and counting with OpenCV*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://learnopencv.com/yolov8-object-tracking-and-counting-with-opencv>

[6]Roboflow. (2024). *Roboflow: Computer vision made easy*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://roboflow.com/>

[7]YOLOv8.org. (2023). *YOLOv8 label format*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://yolov8.org/yolov8-label-format/>

[8]Ultralytics. (2024). *YOLOv8 performance metrics*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://docs.ultralytics.com/vi/models/yolov8/#performance-metrics>

[9]Google. (2024). *Google Colab*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://colab.google/>

[10]Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library*. arXiv:1912.01703.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)<https://arxiv.org/abs/1912.01703>

[11]OpenCV. (2024). *OpenCV official website*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://opencv.org/>

[12]Kaggle. (2024). *Kaggle Terms of Service*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://www.kaggle.com/terms>

[13]OECD. (2019). *OECD AI principles*.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://oecd.ai/en/ai-principles>

[14]Royal Society. (2018). *Ethics of AI in scientific publishing*. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 376*(2133), 20180085.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://royalsocietypublishing.org/doi/10.1098/rsta.2018.0085>

[15]Google AI Edge. (2023). *MediaPipe* [Source code]. GitHub.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

<https://github.com/google-ai-edge/mediapipe>

[16]Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, É. (2011). *Scikit-learn: Machine learning in Python*. *Journal of Machine Learning Research, 12*, 2825–2830.[Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

[Scikit-learn: Machine Learning in Python](https://jmlr.org/papers/v12/pedregosa11a.html)

[17]Zhao, Z., Kiciroglu, S., Vinzant, H., Cheng, Y., Katircioglu, I., Salzmann, M., & Fua, P. (2022). *3D Pose Based Feedback for Physical Exercises* [Preprint]. arXiv. [Accessed:19/08/2025](https://github.com/NgoQuocBao1010/Exercise-Correction/tree/main)

[https://doi.org/10.48550/arXiv.2208.03257](https://arxiv.org/abs/2208.03257)