CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

❧ 📖 ❧

GRADUATION THESIS IN
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

EXERCISE CORRECTION WITH MACHINE LEARNING

Student: Ngô Hồng Quốc Bảo
Student ID: B1809677
Class: 2019 - 2023 (K44)
Advisor: Prof. Trần Công Án

Can Tho, 12/2022

CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND COMMUNICATION TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

🙎 📖 🙐

**GRADUATION THESIS IN
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)**

EXCERCISE CORRECTION WITH MACHINE LEARNING

Student: Ngô Hồng Quốc Bảo
Student ID: B1809677
Class: 2019 - 2023 (K44)
Advisor: Prof. Trần Công Án

Can Tho, 12/2022

INSTRUCTOR'S COMMENTS

Can Tho, ………………. 2022

(Instructor sign and write full name)

ACKNOWLEDGMENT

I cannot express enough thanks to my instructors/teachers from the College of Information and Communication technology for their continued support and encouragement. My sincere thanks especially go to Prof. Trần Công Án for his guidance and advices throughout the development of this project.

My completion of this project could not have been accomplished without the support of my classmates. The supports and advices from them have always been the biggest motivation for me during the whole process. Their encouragement when the times got rough are much appreciated and duly noted.

Last but not least, I would like to thank my parents who helped me a lot in gathering different information, collecting data and guiding me, despite their busy schedules, they had always been there with me from time to time in making project.

Can Tho, December 2022

Student

Ngô Hồng Quốc Bảo

ABSTRACT

Fitness is becoming an important part of human life as it brings many beneficial to personal health. However, exercises can also be ineffective and could be dangerous if they are performed incorrectly by the performer. Proper form when doing anything physical is important, but it is especially important when working out or training. Not only can proper form lower your risk of injury but it also allows you to move efficiently, increase your performance, and enables you to have a full range of motion. In my project, I use machine learning to provides detailed analysis and recommendation on the exercises' performer to improve their form.

In addition, with deep learning and computer vision are being strongly researched and improve by the day. Especially the development of Mediapipe by Google, an open-source framework to "build word-class machine learning solutions" that provides cornerstone Machine Learning models for common tasks like hand tracking, posture detection, ... Make use of the power of Mediapipe's pose detection, this project "Exercise Correction with Machine Learning" is built in order to analyze, detect and classifying the forms of fitness exercises. The experimental results show that the algorithm proposed in this paper can effectively identify correct and incorrect forms that are performed in an exercise.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1. PROBLEM DESCRIPTION

The advancements in artificial intelligence and computational power, computer vision technology has taken a huge leap toward integration in our daily lives. Computer vision is the field of computer science that focuses on creating digital systems that can process, analyze, and make sense of visual data (images or videos) in the same way that humans do. The concept of computer vision is based on teaching computers to process an image at a pixel level and understand it. Technically, machines attempt to retrieve visual information, handle it, and interpret results through special software algorithms.

In addition, nowadays, exercise is proven to be extremely important for people's daily lives. Being physically active can improve your brain health, help manage weight, reduce the risk of disease, strengthen bones and muscles, and improve your ability to do everyday activities. Everyone can experience the health benefits of physical activity regardless of their age, gender, ethnicity, shape ... When it comes to exercises quality is more important than quantity. How an exercise is performed may mean the difference between going harder and getting sidelined. Anyone the more experienced can benefit from some occasional form feedback. Perfecting form will boost performance, conserve energy and reduce injuries over time.

However, for many people, especially beginners or people who mainly do their exercise by themselves, a noticeable amount of them perform these exercises improper form due to the lack of awareness or instruction. With improper form, the performers risk straining/injuring their body rather than training it. One thing is for sure, they are not going to see results if they are sidelined with a frustrating injury.

This thesis is intended to build machine learning models for different exercises with the help of Mediapipe framework. For each exercise, the correspond model will detect whether if the exercise is performed properly or not. The trained model will be applied to build a web application where users can provide their workout videos to get feedbacks on their form.

2. PURPOSE OF THE STUDY

The purpose of this study is to develop 4 machine learning models for 4 of the most home exercises which each model can detect any form of incorrect movement while a person is performing a correspond exercise. In addition, a web application that utilize the trained models, will be built in other to analyze and provide feedbacks on workout videos.

3. LIMITATION AND SCOPE

This study researches deep learning knowledge such as computer vision, neural network and Mediapipe framework. The study also uses Python programming language, Open CV library for image processing; Sci-kit learn library and Keras library for building machine learning model; Vue.js framework and Django framework for the web application that utilizes the trained model for feedbacks on incorrect form to exercise videos.

4. GENERAL APPROACH

- Research on which popular exercises which commonly improperly perform.
- Research on which technology to choose that is suitable to solve the problem.
- Collect and process data of the chosen exercises.
- Train and evaluate model for each exercise.
- Build a web application utilize all the trained models.
- Evaluate the final model and test the web application.

5. CRITERIA FOR STUDY SUCCESS

- Successfully build 4 models for 4 exercises which can detect proper and improper form of each exercise.
- Successfully build a web application which apply the trained models for providing feedbacks and recommendation for users bases on their exercise videos.
- Earn knowledge on deep learning and computer vision topics. Apply Mediapipe framework to real life application.

6. OUTLINE OF THE STUDY

- INTRODUCTION: Introduce the problem and the general concepts of the topic, such as expectation, limit, content, etc.
- CHAPTER 1: AN OVERVIEW OF TECHNOLOGY AND LIBRARY USED. Present related theoretical content about computer vision, MediaPipe framework and other technology used in the study.
- CHAPTER 2: METHODOLOGY. Present general methods for detecting errors of the chosen exercises, go into detail on each exercise and an overview of the web application.
- CHAPTER 3: MODEL EVALUATION. Present evaluation metrics and evaluation results for all trained models.
- CHAPTER 4: TESTING APPLICATION. Building web application evaluation scenarios and evaluation results after experience.

7. RELATED WORKS

In March 2018, a study "Pose Trainer: Correcting Exercise Posture using Pose Estimation" is published by Steven Chen and Richard Yang (both from Standford University) [5] with the intention to use machine learning in fitness exercise for helping prevent injuries and im-prove the quality of people's workouts with just a computer and a webcam. The report of the study introduces Pose Trainer, an end-to-end computer vision application that uses pose estimation, visual geometry, and machine learning to provide personalized feedback on fitness exercise form. The study worked with 4 different exercises (bicep curl, front raise, shoulder shrug and shoulder press) recording training videos for each, and use both geometric heuristic algorithms to provide personalized feed-back on specific exercise improvements, as well as machine learning algorithms to automatically determine posture correctness using only labeled input videos.

CONTENTS

CHAPTER 1: AN OVERVIEW OF TECHNOLOGY AND LIBRARY USED

1. COMPUTER VISION

Computer vision is the field of computer science that focuses on creating digital systems that can process, analyze, and make sense of visual data (images or videos) in the same way that humans do. The concept of computer vision is based on teaching computers to process an image at a pixel level and understand it. Technically, machines attempt to retrieve visual information, handle it, and interpret results through special software algorithms.

There is a lot of research being done in the computer vision field, but it's not just research. Real-world applications demonstrate how important computer vision is to endeavors in business, entertainment, transportation, healthcare and everyday life. A key driver for the growth of these applications is the flood of visual information flowing from smartphones, security systems, traffic cameras and other visually instrumented devices.

- Google Translate lets users point a smartphone camera at a sign in another language.
- IBM is applying computer vision technology with partners like Verizon to bring intelligent AI to the edge, and to help automotive manufacturers identify quality defects before a vehicle leaves the factory.
- The development of self-driving vehicles relies on computer vision to make sense of the visual input from a car's cameras and other sensors. It's essential to identify other cars, traffic signs, lane markers, pedestrians, bicycles and all of the other visual information encountered on the road.

2. MEDIAPIPE FRAMEWORK
2.1. Definition

MediaPipe is an open-source framework for building pipelines to perform computer vision inference over arbitrary sensory data such as video or audio. Using MediaPipe, such a perception pipeline can be built as a graph of modular components. MediaPipe was built for machine learning (ML) teams and software developers who implement production-ready ML applications, or students and researchers who publish code and prototypes as part of their research work.

In computer vision pipelines, those components include model inference, media processing algorithms, data transformations, etc. Sensory data such as video

streams enter the graph, and perceived descriptions such as object-localization or face-keypoint streams exit the graph.
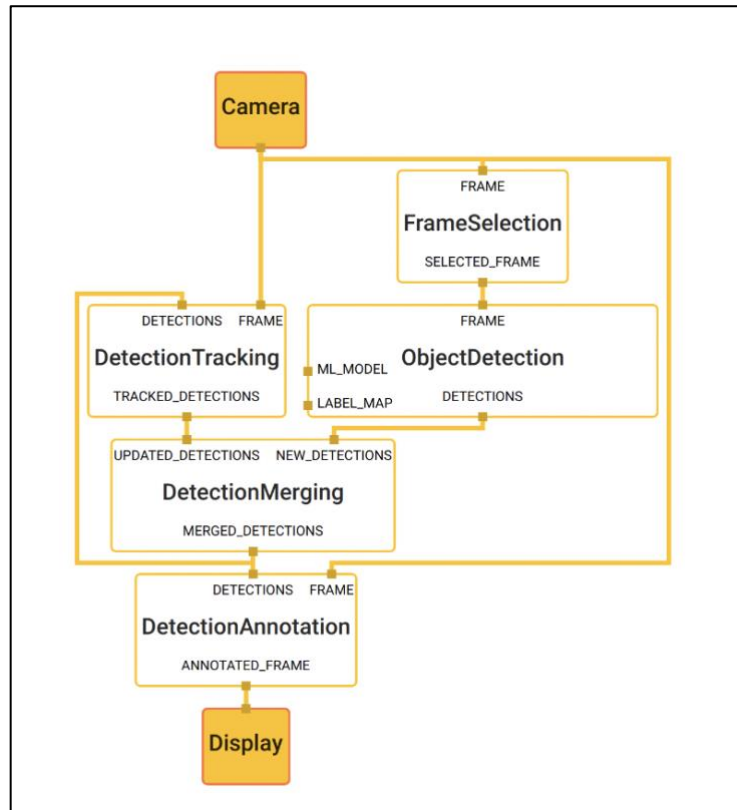
Figure 1: Object detection pipeline in MediaPipe

The main use case for MediaPipe is rapid prototyping of perception pipelines with inference models and other reusable components. MediaPipe also facilitates the deployment of perception technology into demos and applications on a wide variety of different hardware platforms. MediaPipe enables incremental improvements to perception pipelines through its rich configuration language and evaluation tools.

2.2. MediaPipe Architecture

MediaPipe allows a developer to prototype a pipeline incrementally. A pipeline is defined directed graph of components where each component is a *Calculator*. The graph is specified using a *GraphConfig protocol* buffer and then run using a Graph object.

In the graph, the calculators are by data, *Streams*. Each represents a time-series of data, *Packets*. Together, the calculators and streams define a data-flow graph. The packets which flow across the graph are collated by their timestamps within the time-series.

- *Packet*: basic data unit in MediaPipe. A packet consists of a numeric timestamp and a shared pointer to an immutable payload.
- *Stream*: each node in the graph is connected to another node through a stream. A stream carries a sequence of packet whose timestamps must be monotonically increasing.
- *Calculators*: implemented as each node of a graph. The bulk of graph execution happens inside its calculator.
- *Graph*: all processing takes places within the context of a Graph. A graph contains a collection of nodes joined by directed connections along which packets can flow.
- *GraphConfig*: is a specification that describes the topology and functionality of a MediaPipe graph. All the necessary configurations of the node, such its type, inputs and outputs must be described in the specification. Description of the node can also include several optional fields, such as node-specific options, input policy and executor.
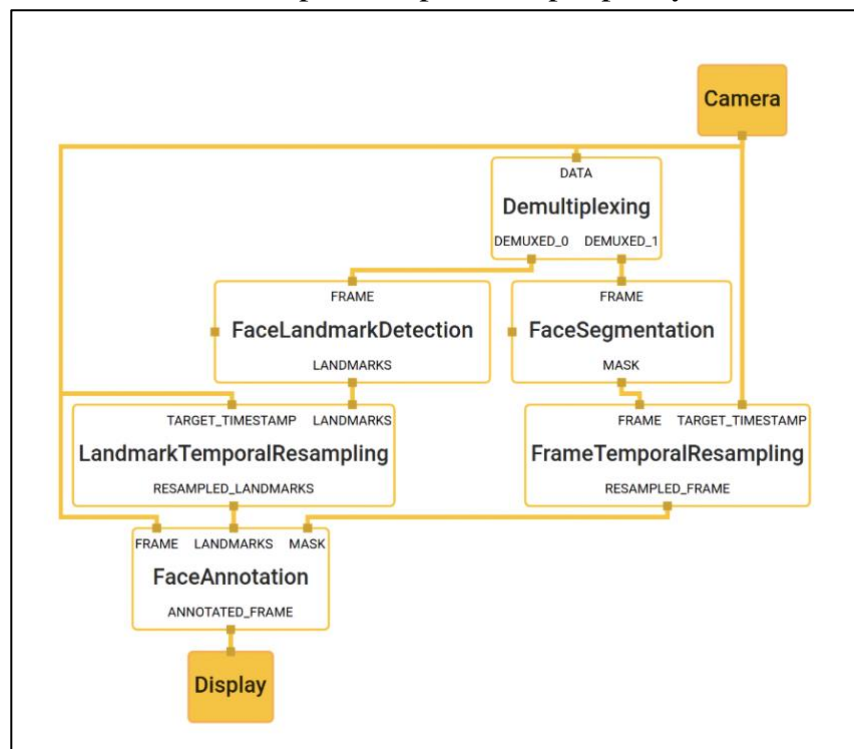

Figure 2: Face landmark detection and segmentation using MediaPipe

3. MEDIAPIPE POSE
3.1. Overview and Machine Learning pipeline

MediaPipe Pose is a ML solution for high-fidelity body pose tracking, inferring 33 3D landmarks and background segmentation mask on the whole body from RGB video frames utilizing our BlazePose research that also powers the ML

Kit Pose Detection API. Current state-of-the-art approaches rely primarily on powerful desktop environments for inference, whereas our method achieves real-time performance on most modern mobile phones, desktops/laptops, in python and even on the web.
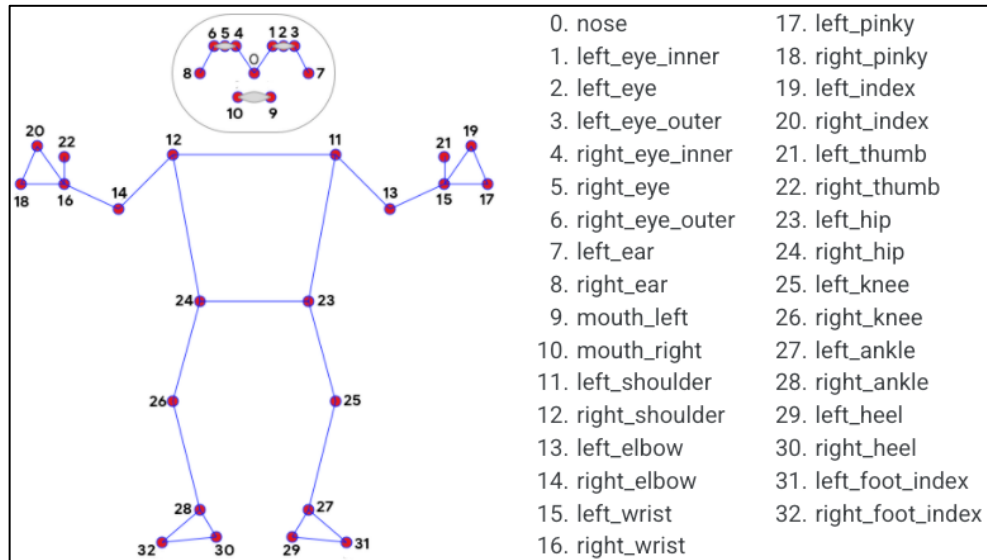
Figure 3: The 33 landmarks model in MediaPipe Pose predicts

The solution utilizes a two-step detector-tracker ML pipeline, proven to be effective in MediaPipe Hands and MediaPipe Face Mesh solutions. Using a detector, the pipeline first locates the person/pose region-of-interest (ROI) within the frame. The tracker subsequently predicts the pose landmarks and segmentation mask within the ROI using the ROI-cropped frame as input. Note that for video use cases the detector is invoked only as needed, i.e., for the very first frame and when the tracker could no longer identify body pose presence in the previous frame. For other frames the pipeline simply derives the ROI from the previous frame's pose landmarks.

Figure 4: MediaPipe pose detection's pipeline

3.2. BlazePose detector model

For real-time performance of the full ML pipeline consisting of pose detection and tracking models, each component must be very fast, using only a few milliseconds per frame. To accomplish this, it is known that the strongest signal to the neural network about the position of the torso is the person's face (due to its high-contrast features and comparably small variations in appearance). Ther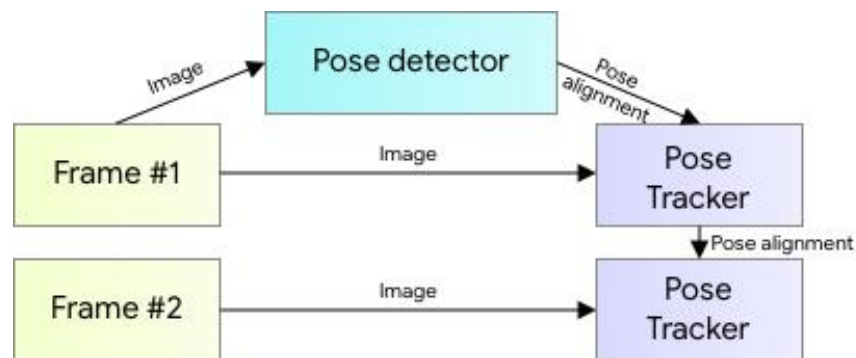efore, BlazePose is a fast and lightweight pose detector by making the strong (yet for many mobile and web applications valid) assumption that the head should be visible for single-person use case.

Therefore, the detector is inspired by the lightweight BlazeFace model, used in MediaPipe Face Detection, as a proxy for a person detector. It explicitly predicts two additional virtual keypoints that firmly describe the human body center, rotation and scale as a circle. Inspired by Leonardo's Vitruvian man, the model predicts the midpoint of a person's hips, the radius of a circle circumscribing the whole person, and the incline angle of the line connecting the shoulder and hip midpoints.
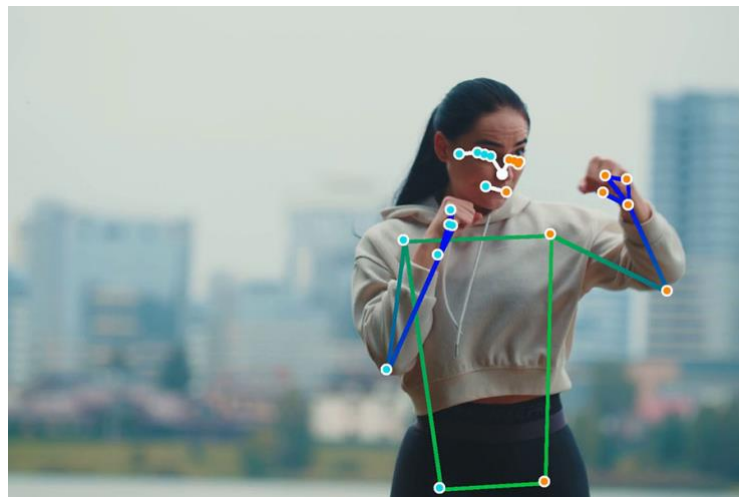
Figure 5: Example of pose detection by MediaPipe

3.3. MediaPipe Pose Output

MediaPipe Pose returns a list of 33 pose landmarks. Each landmark consists of the following properties:

- *x* and *y*: Landmark coordinates normalized to [0.0, 1.0] by the image width and height respectively.
- *z:* Represents the landmark depth with the depth at the midpoint of hips being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of *z* uses roughly the same scale as *x*.

- *visibility*: A value in *[0.0, 1.0]* indicating the likelihood of the landmark being visible (present and not occluded) in the image.

4. OTHER TECHNOLOGY AND LIBRARIES USED
4.1. Open CV

OpenCV is an open-source software library for computer vision and machine learning. The OpenCV full form is open-source computer vision library. It was created to provide a shared infrastructure for applications for computer vision and to speed up the use of machine perception in consumer products.

Gary Bradsky invented OpenCV in 1999 and soon the first release came in 2000. This library is based on optimized C / C++ and supports Java and Python along with C++ through an interface. The library has more than 2500 optimized algorithms, including an extensive collection of computer vision and machine learning algorithms, both classic and state-of-the-art. Using OpenCV it becomes easy to do complex tasks such as identify and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D object models, generate 3D point clouds from stereo cameras, stitch images together to generate an entire scene with a high-resolution image and many more.

4.2. Scikit-learn

Scikit-learn is a free software library for the Python programming language that provides a collection of algorithms for machine learning and data mining. It features various classification, regression and clustering algorithms including support vector machines, random forests, boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. It is licensed under the BSD license.

4.3. Keras

Keras is an open-source deep learning framework for python. It has been developed by an artificial intelligence researcher at Google named Francois Chollet. Leading organizations like Google, Square, Netflix, Huawei and Uber are currently using Keras. This tutorial walks through the installation of Keras, basics of deep learning, Keras models, Keras layers, Keras modules and finally conclude with some real-time applications.

Keras has nice guiding principles: modularity, minimalism, extensibility, and Python-nativeness. Keras also has out-of-the-box implementations of common network structures. It's fast and easy to get a convolutional neural network up and running.

4.4. Vue.js

Vue is an open-source progressive framework that is designed to be incrementally adoptable, as the core library is focused around the view layer only. Vue.js is primarily used to build web interfaces and one-page applications. In saying that, it can also be applied to both desktop and mobile app development thanks to the HTML extensions and JS base working in tandem with an Electron framework – making it a heavily favored frontend tool.

4.5. Django

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support. Due to its nature of written in Python, it would make it easier to deploy machine learning based projects.

CHAPTER 2: METHODOLOGY

1. EXERCISES SELECTION AND ERROR DETERMINATION

First and foremost, the initial step of this thesis is to decide on which exercises to train machine learning model for form correction. Due to the limitation of time for the thesis, only 4 exercises are picked. Each exercise needs to meet the following criteria:

- An exercise must be a common exercise.
- An exercise is popular among people who workouts at home or workouts mostly by themselves. Therefore, it would raise the possibility that the exercise would be performed improperly.
- An exercise must contain at least 2 popular errors which impact the inefficiency of the exercise.

Depends on the above criteria, I decided on 4 exercises: *bicep curl*, *basic plank*, *basic quat* and *lunge*.

Figure 6: A person doing lunge

After decided on 4 exercises, for each exercise, at least 2 popular errors for that exercise would be determined and then develop strategy to detect those errors (The detail of each error will be going in depth in page 24). Here are the determined common errors for each exercise:

- *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 and knee placement too wide/tight.
- *Lunge*: Knee over toe while going down.

2. DATA COLLECTING

2.1. Self-collected data

Due to the lack of videos or dataset online that recorded human doing exercises both in a proper or improper way, the majority of self-collected videos were either recorded by myself, my friends or my family. In total, there are 5 contributors for this dataset. For each exercise, each contributor is required to:

- Record 2 videos with different camera angles when the contributor performs the exercise correctly.
- Record 2 videos with different camera angles for each determined error of the exercises.

For each video that provided by each contributor, the video has to meet these following criteria:

- Minimum duration: depends on exercise. With exercises which does not require much action such as plank, the minimum required duration of a video would be 30 seconds. However, with exercises which involve more movements such as squat, the contributors are required to perform the exercise at least 15 reps for each video. (More detail in …)
- Video is recorded in environment with enough light so that the whole body of a person is in good visibility.
- For each exercise, video needs to show clearly the important joints or movement which important to the exercise.
- The contributor needs to be on frame throughout the video.

After collecting the videos, depends on the exercise, they are divided into multiple classes:

- "proper-form": Videos of a contributor perform the exercise correctly.
- "error-name": Each video of a contributor performs the exercise correctly will be in a separate class.

```
.
└─ exercise/
   ├─ proper-form/
   ├─ error-form-1/
   └─ error-form-2/
```
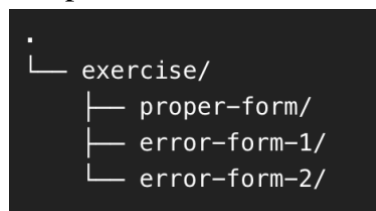
Figure 7: Directory tree structure of collected videos example

2.2. Public Dataset from Kaggle

With an exercise such as *Plank*, as there is not much movement during the exercise, I'm able to find a dataset from an open database from Kaggle. In short, Kaggle is an online community platform for data scientists and machine learning enthusiasts.

The found dataset is about many yoga poses but the very well-known ones are the downward dog pose, goddess pose, tree pose, plank pose and the warrior pose. The dataset contains 5 folders for 5 poses, each folder contains images of people correctly doing the correspond pose.

For the purpose of this thesis, only the folder contains the images of people properly doing plank is chosen. There are 266 image files in that folder, I handpicked all the images that represent a basic plank and discard the reset. In conclusion, there are 30 images which are arranged to the proper form class for *basic plank*.

3. SIMPLE ERROR DETECTION

For some simple errors (for example, the feet placement error in squat), the detection method is either measuring the distance/angle between different joints during the exercise with the coordinate outputs from MediaPipe Pose.

a. Distance Calculation

Assume there are 2 points with the following coordinates: Point 1 $(x1, y1)$ and Point 2 $(x2, y2)$, below is the formula to calculate the distance between 2 points.

$$distance = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

b. Angle Calculation

Assume there are 3 points with the following coordinates: Point 1 $(x1, y1)$, Point 2 $(x2, y2)$ and Point 3 $(x3, y3)$, below is the formula to calculate the angle created by 3 points.

$$angle\_in\_radian = \arctan2(y3 - y2, x3 - x2) - arctan2(y1 - y2, x1 - x2)$$
$$angle\_in\_degree = (angle\_in\_rad * 180)/\Pi$$

4. MODEL TRAINING FOR ERROR DETECTION
4.1. Data processing

The general approach of processing data from collected videos for training model is illustrated in Figure 8: Data processing below.
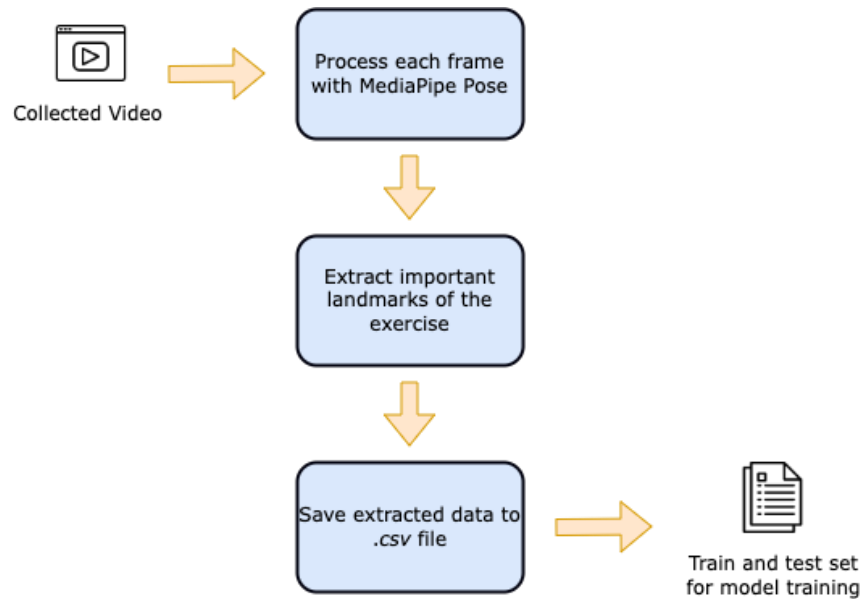
Figure 8: Data processing

4.1.1. Detecting important landmarks

For each exercise, there will be different poses/body's position, therefore it is essential to identify which parts (shoulder, hip, …) of a body are contribute to the exercise. The important landmarks identified for each exercise are utilized to extract body part's position while exercising using MediaPipe.


```
    # Determine important landmarks for plank
    IMPORTANT_LMS = [
        "NOSE",
        "LEFT_SHOULDER",
        "RIGHT_SHOULDER",
        "RIGHT_ELBOW",
        "LEFT_ELBOW",
        "RIGHT_WRIST",
        "LEFT_WRIST",
        "LEFT_HIP",
        "RIGHT_HIP",
    ]
```

Figure 9: Example of important landmarks for plank exercise

In addition, every landmark's property such as *x, y, z* and *visibility*, will be flatten to 4 headers for an *csv* file. For example, with the *NOSE* landmarks, the 4 headers would be: NOSE_x, NOSE_y, NOSE_z and NOSE_v (detail mention in page 15 - MediaPipe Pose Output). Therefore, for each exercise, an empty *csv* file will be initialized. The first header is the "label" which contain a class for each datapoint, the rest of file's headers are all the important headers with their properties flatten.

4.1.2. Extract data from video to file using OpenCV and MediaPipe

With every exercise, there are videos which are separated into different classes. With each video, every frame that matches a certain form of an exercise would be:

- Pose estimation: Process the video using OpenCV then utilize MediaPipe Pose to produce a list of coordinate predictions of all keypoint locations, and their corresponding prediction confidence. Configuration options for Medipipe Pose:
  - `MIN_DETECTION_CONFIDENCE`: Minimum confidence value (`[0.0, 1.0]`) from the person-detection model for the detection to be considered successful. Default to `0.5`.
  - `MIN_TRACKING_CONFIDENCE`: Minimum confidence value (`[0.0, 1.0]`) from the landmark-tracking model for the pose landmarks to be considered tracked successfully, or otherwise person detection will be invoked automatically on the next input image. Setting it to a higher value can increase robustness of the solution, at the expense of a higher latency. Default to `0.5`.

```
cap = cv2.VideoCapture("/path/to/video")

with mp_pose.Pose(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as pose:
        while cap.isOpened():
                ret, image = cap.read()
                image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
                results = pose.process(image)
```

- For every frame with the list of predicted landmarks, the important landmarks for a correspond exercise would be extracted in append as a row to the csv file with the label column matches the class of the video.
- The collected data which is saved in a csv file would be split to train and evaluate model. 80% of the data will be used for model training and the remaining 20% will be used for model evaluation.

4.2. Model Training

There are 2 methods used in this thesis for model training. For each exercise, the models trained for each method will be compared and the best model will be chosen.

- Classification with Scikit-learn.
- Building a Neural Network for classification with Keras.

4.2.1. Classification with Scikit-learn

The machine learning algorithms used for model training are: Decision Tree/Random Forest (RF), K-Nearest Neighbors (KNN), C-Support Vector (SVC), Logistic Regression classifier (LR) and Stochastic Gradient Descent classifier (SGDC). After the training session, each result from each algorithm will be evaluate with metrics such as: precision, recall, accuracy and F1 score. The algorithm which provides the best results will be selected. Below is the sample code for this section.

```
algorithms = [("LR", LogisticRegression()),
              ("SVC", SVC()),
              ('KNN, KNeighborsClassifier()),
              ("SGDC", SGDClassifier()),
              ('RF', RandomForestClassifier()),]
for name, model in algorithms:
        trained_model = model.fit(X_train, y_train)
```

4.2.2. Neural Network for classification with Keras

With Keras, composing a neural network by creating layers and linking them together. In this project, one type of layer called a fully connected or Dense layer. In Keras, this is defined by the `keras.layers.Dense` class.

A dense layer has a number of neurons, which is a parameter which can be chosen when the layer is created. When connecting the layer to its input and output layers every neuron in the dense layer gets an edge (i.e. connection) to all of the input neurons and all of the output neurons.

Occasionally, *Dropout* layers are utilized in the model's architecture. The term "dropout" refers to dropping out the nodes (input and hidden layer) in a neural network. The nodes are dropped by a dropout probability of p. Dropout layer is used to combat the problem of overfitting. In overfitting, a unit may change in a way that fixes up the mistakes of the other units, using dropout, it prevents these units to fix up the mistake of other units, thus preventing co-adaptation, as in every iteration the presence of a unit is highly unreliable. Therefore, by randomly dropping a few units (nodes), it forces the layers to take more or less responsibility for the input by taking a probabilistic approach.

For each training session, there are 4 models' architectures which would be experiment with. Depends on the evaluation metrics for each model, the model that yield the best results will be chosen.

- Neural network with 3 Dense layers.
- Neural network with 5 Dense layers.
- Neural network with 5 Dense layers and 2 Dropout layers.
- Neural network with 7 Dense layers.

5. ERROR DETECTION PROCESS IN DEPTH FOR EACH EXERCISES

The previous section discusses about the general approach, each step to build models for the 4 exercises, this section will address in detail on each exercise. neurons and all of the output neurons.

5.1. Bicep Curl
5.1.1. Basic technique, errors description and important landmarks

The biceps curl is a highly recognizable weight-training exercise that works the muscles of the upper arm, and to a lesser extent, those of the lower arm. The exercise has to be performed while standing, involve around the arms moving up and down while holding a dumbbell or barbell.

There are 3 popular errors of bicep curl that will be targeted in this thesis:

- Loose upper arm: when an arm moves upward during the exercise, the upper arm is moving instead of staying still.
- Weak peak contraction: when an arm moves upward, it does not go high enough therefore not put enough contraction to the bicep.
- Lean too far back: the performer's torso leans back and fore during the exercise for momentum.

In my research and exploration, the important MediaPipe Pose landmarks for this exercise are: nose, left shoulder, right shoulder, right elbow, left elbow, right wrist, left wrist, right hip and left hip.

5.1.2. Errors detection method
a. Loose upper arm

Can be detected by calculating the angle between the elbow, shoulder and the shoulder's projection on the ground. Through my research, if the angle is over 40 degrees, the movement will be classified as a "loose upper arm" error.

b. Weak peak contraction

Can be detected by calculating the angle between the wrist, elbow and shoulder when the performer's arm is coming up. Through my research, if the angle is more than 60 degrees before the arm comes down, the movement will be classified as a "weak peak contraction" error.

c. Lean too far back

Due to its complexity, machine learning is used for this error. Videos of contributors perform with and without this error are used as data for training model. Figure 11 is a visual graph represent the number of frames gathered from the videos and their classes. There are a total of 15372 images, in which, there are 8238 samples (53.6%) belong to class *correct posture (C)* and 7134 (46.4%) samples belong to class *lean back (L)*. Every datapoint in the dataset is consist of 37 columns which are formed based on its classed and the important landmarks.
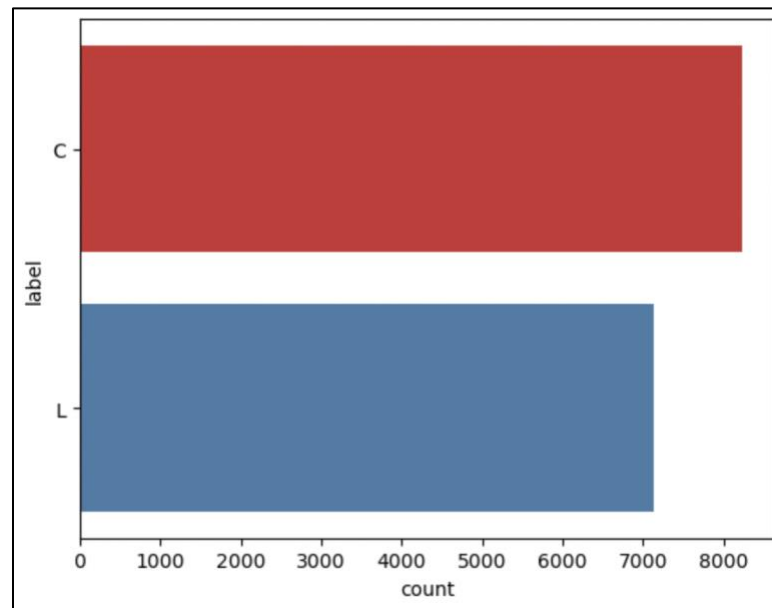
Figure 10: Class balance of Bicep Curl's dataset

As mention earlier in page 22 - Model Training a combination of Scikit-learn machine learning algorithms and 4 neural network architectures is used for model training. The results of the experiment in training mentioned model are shown in the table below.

Table I. Model training experiments for Bicep Curl's error

Model	Neural Network	Precision	Recall	Accuracy
KNN	False	0.975	0.968	0.972
7 layers	True	0.972	0.962	0.967
5 layers	True	0.963	0.949	0.955
SVC	False	0.93	0.934	0.932
RF	False	0.947	0.925	0.931
7 layers with dropout	True	0.936	0.924	0.93
3 layers	True	0.939	0.92	0.929
LR	False	0.792	0.738	0.762
SGDC	False	0.712	0.715	0.715

5.2. Basic Plank
5.2.1. Basic technique, errors description and important landmarks
The plank, or planking, is an exercise that involves your core muscles, improving your strength, balance and endurance. To perform a plank, lying on the

ground with the elbows in line with the shoulder and the feet shoulder width apart, Push body up bearing the weight on the forearms and feet, body is kept straight.

There are 3 popular errors of basic plank that will be targeted in this thesis:

- High lower back: while performing the exercise, instead of keeping the lower back straight, it is raised too high.
- Low lower back: while performing the exercise, instead of keeping the lower back straight, it is brought down too low.

In my research and exploration, the important MediaPipe Pose landmarks for this exercise are: nose, left shoulder, right shoulder, right elbow, left elbow, right wrist, left wrist, right hip, left hip, right knee, left knee, right ankle, left ankle, right heel, left heel, right foot index and left foot index.

5.2.2. Errors detection method

Videos of contributors perform in both 3 stages (proper form, high lower back and low lower back) of this exercise are used for to build a machine learning model. Figure 11 is a visual graph represent the number of frames gathered from the videos and their classes. There are a total of 28623 images, in which, there are 9630 samples (33.6%) belong to class *correct form (C)*, 8982 samples (31.4%) belong to class *high back (H)* and 10011 (35%) samples belong to class *low back (L)*. Every datapoint in the dataset consists of 69 columns which are formed based on its classed and the important landmarks.
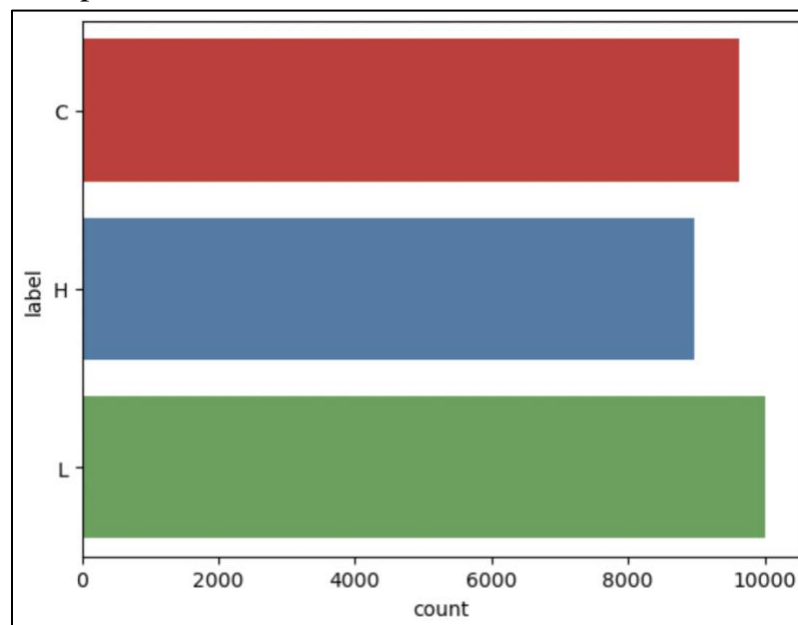

Figure 11: Class balance of Basic Plank's dataset

The results of the experiment in training mentioned model are shown in the table below (detail mentioned in page 22 - Model Training).

Table II. Model training experiments for Plank's error detection

Model	Neural Network	Precision	Recall	Accuracy
LR	False	0.996	0.996	0.996
7 layers with dropout	True	0.994	0.994	0.994
SVC	False	0.987	0.987	0.987
SGDC	False	0.981	0.981	0.981
KNN	False	0.955	0.949	0.949
5 layers	True	0.934	0.93	0.93
7 layers	True	0.935	0.924	0.924
RF	False	0.922	0.899	0.899
3 layers	True	0.869	0.848	0.848

5.3. Basic squat
5.3.1. Basic technique, errors description and important landmarks

A squat is a strength exercise in which the trainee lowers their hips from a standing position and then stands back up. During the descent of a squat, the hip and knee joints flex while the ankle joint dorsiflexes; conversely the hip and knee joints extend and the ankle joint plantarflexes when standing up.

There are 3 popular errors of basic squat that will be targeted in this thesis:
- Feet placement: Feet placement is extremely important in squat. The 2 feet should place in a way that the width of 2 feet is around the width of 2 shoulders.
- Knee placement: Knee placement is not only important but could be dangerous if perform incorrectly with heavy load. During the "down" stage of the exercise, the knee should be open out to wider than the feet width.

In my research and exploration, the important MediaPipe Pose landmarks for this exercise are: left shoulder, right shoulder, right hip, left hip, right knee, left knee, right ankle and left ankle.

5.3.2. Stage detection method

In contrast with other exercises, there are 2 stages when performing squat, "up" and "down" stage. Because it is important for error detection to discriminate stage of the squat, a model is trained for this task.

Videos of contributors perform proper form of a squat of this exercise are used for to build a machine learning model. Figure 12 is a visual graph represent the number of frames gathered from the videos and their classes. There are a total of

374 images, in which, there are 188 samples (50.25%) belong to class *up* and 186 (49.75%) samples belong to class *down*. Every datapoint in the dataset consists of 37 columns which are formed based on its classed and the important landmarks.
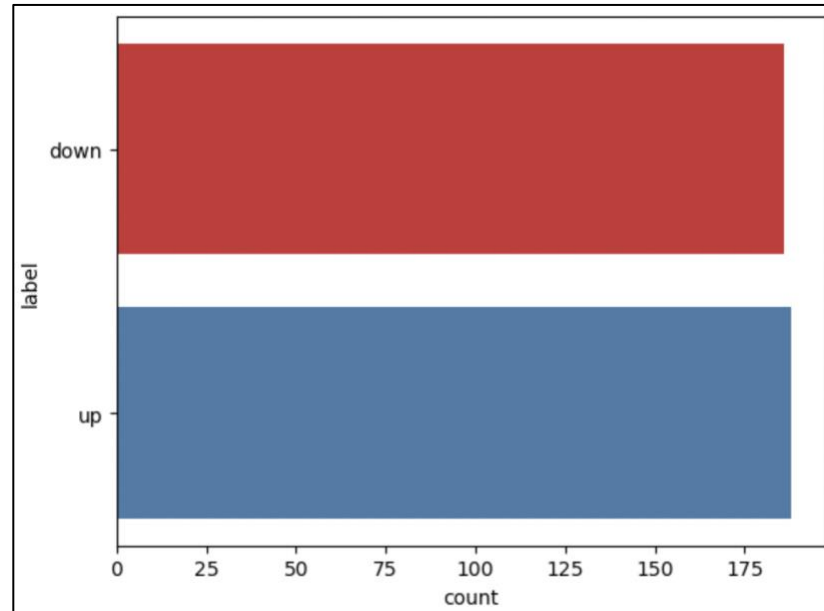

Figure 12: Class balance of Squat's dataset

The results of the experiment in training mentioned model are shown in the table below (detail mentioned in page 22 - Model Training).

Table III. Model training experiments for Squat's stage

Model	Neural Network	Precision	Recall	Accuracy
LR	False	0.994	0.994	0.994
5 layers	True	0.994	0.993	0.993
SGDC	False	0.993	0.993	0.993
3 layers	True	0.99	0.992	0.992
KNN	False	0.985	0.985	0.985
7 layers	True	0.985	0.982	0.982
7 layers with dropout	True	0.981	0.985	0.985
SVC	False	0.978	0.977	0.977
RF	False	0.254	0.504	0.504

5.3.3. Errors detection method

a. Feet placement

Can be detected by calculate ratio between the distance of 2 feet and the distance of 2 shoulders. To precisely choose the correct ratio, videos of contributors perform proper form of a squat are analyzed. In that respect, 851 datapoints are gathered.
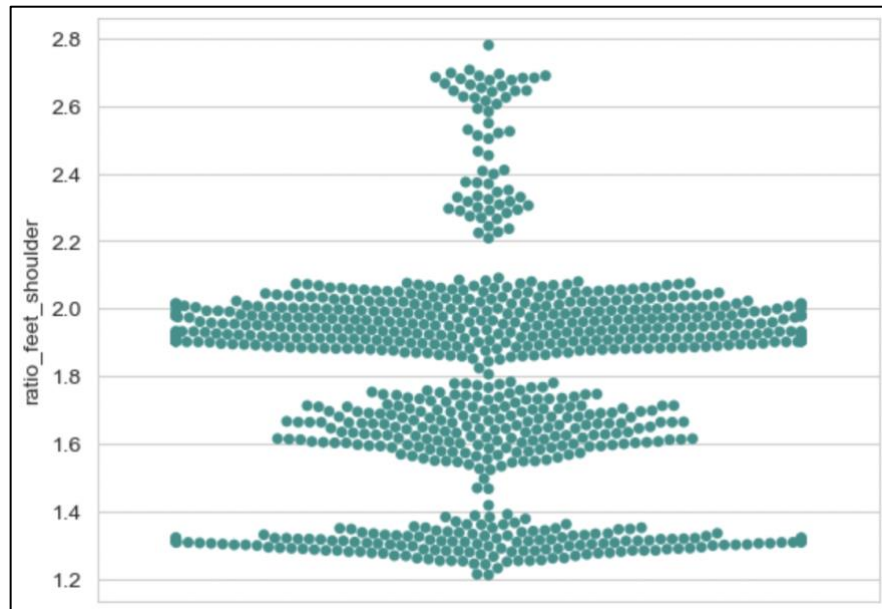

Figure 13: Graph representing the ratio between feet width and shoulder width in squat

Figure 13 is a visual graph represents the ratio between the feet width and shoulder width calculated from 851 data points. In conclusion, assume that x is the ratio between feet width and shoulder width:

- $1.2 \leq x \leq 2.8$: correct foot placement
- $x < 1.2$: foot placement is too tight
- $x > 2.8$: foot placement is too wide

b. Knee placement

Can be detected by calculating ratio between the distance of 2 knee and 2 feet. Similar to the previous error, videos of contributors are analyzed to determine a correct threshold. Due to the dynamic movement of the knee during the exercise, the calculated ratio from the data will be separate into 3 stages: up, middle and down.
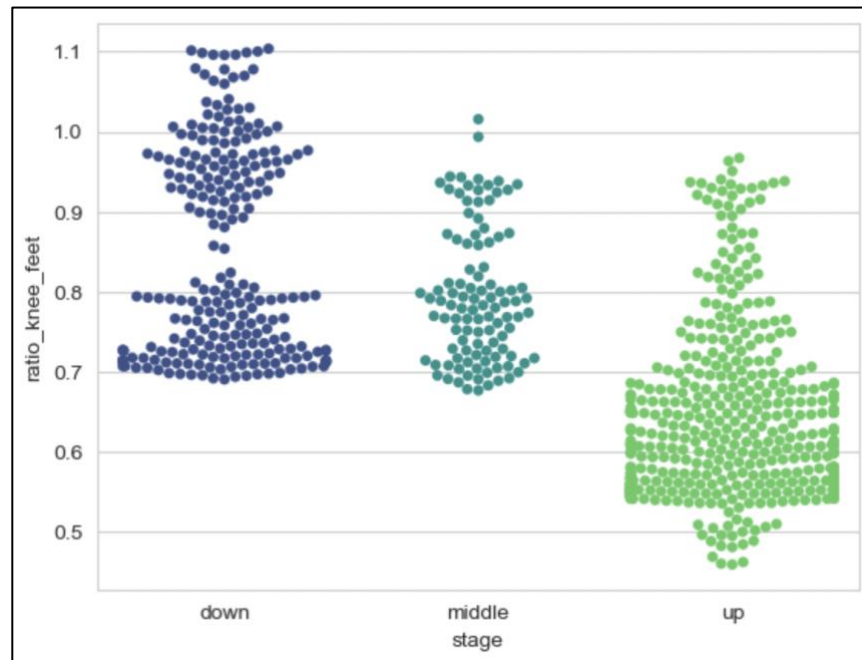
Figure 14: Graph representing the ratio between knee width and feet width in each squat stage

Figure 14 above is a visual graph represents the ratio between the knee width and feet width. In conclusion, assume that x is the ratio between feet width and shoulder width:

- At "down" stage:
    - $0.7 \leq x \leq 1.1$: correct knee placement
    - $x < 0.7$: knee placement is too tight
    - $x > 1.1$: knee placement is too wide
- At "middle" stage:
    - $0.7 \leq x \leq 1$: correct knee placement
    - $x < 0.7$: knee placement is too tight
    - $x > 1$: knee placement is too wide
- At "up" stage:
    - $0.5 \leq x \leq 1.1$: correct knee placement
    - $x < 0.5$: knee placement is too tight
    - $x > 1.1$: knee placement is too wide

5.4. Lunge
5.4.1. Basic technique, errors description and important landmarks

A lunge can refer to any position of the human body where one leg is positioned forward with knee bent and foot flat on the ground while the other leg is positioned behind.

There are 2 popular errors of lunge that are targeted in this thesis:
- Knee angle: while the body is in a low position, the angle of 2 knees should be around 90 degrees. The knee's angle too wide or too tight could be an indication of the performer takes too big or too narrow of a step while performing lunge.
- Knee over toe: while the body is in a low position, especially with heavier weight, the performer usually leans too much forward cause the knee to go past the toe. This popular error can result in a knee injury or an imbalance for the performer.

In my research and exploration, the important MediaPipe Pose landmark for lunge are: nose, left shoulder, right shoulder, left knee, right knee, right hip, left hip, left ankle, right ankle, left heel, right heel, left foot index and right foot index.

5.4.2. Error detection methods

a. Knee angle

Can be detected by calculating the angle of the left and right knee. To precisely choose the correct lower and upper thresholds for this error, videos of contributors perform correct form of the exercise are analyzed. In that respect, 9906 datapoints of correct knee angles are gathered.

Figure 15 is a visual graph represent the angles of right and left knee while properly performing lunge. In conclusion from the graph, the angle of left/right knee during the low position of a lunge should be in between *60 and 135 degrees*.
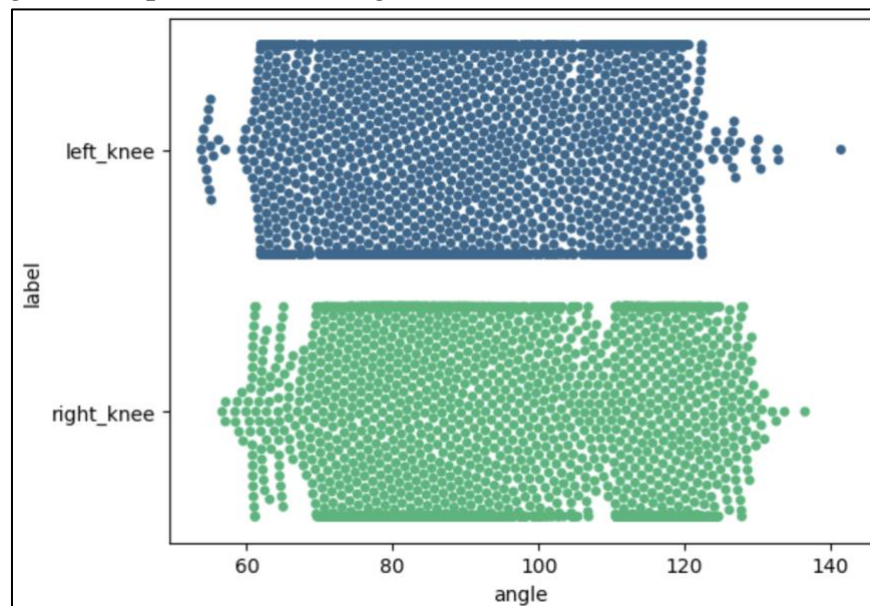

Figure 15: Graph representing the angles of right and left knee in correct lunge position

b. Knee over toe

Videos of contributors perform in both 2 stages (proper form and knee-over-toe form) of this exercise are used for to build a machine learning model. Figure 16 is a visual graph represent the number of frames gathered from the videos and their classes. There are a total of 17907 images, in which, there are 8793 samples (49.1%) belong to class *correct form (C)* and 9114 samples (51.9%) belong to class *knee-over-toe (L)*. Every datapoint in the dataset consists of 53 columns which are formed based on its classed and the important landmarks.
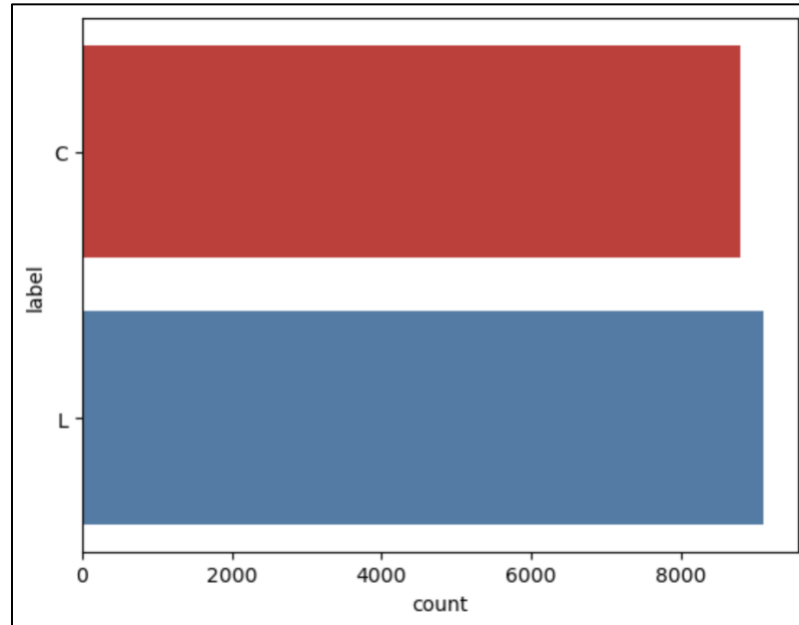

Figure 16: Class balance of Lunge's dataset

The results of the experiment in training mentioned model are shown in the table below (detail mentioned in page 22 - Model Training).

Table IV. Model training experiments for Lunge's error

Model	Neural Network	Precision	Recall	Accuracy
LR	False	0.973	0.972	0.972
SGDC	False	0.961	0.956	0.956
3 layers	True	0.937	0.928	0.928
7 layers with dropout	True	0.892	0.865	0.865
RF	False	0.855	0.842	0.842
5 layers	True	0.861	0.831	0.831
KNN	False	0.768	0.765	0.765
7 layers	True	0.838	0.773	0.773
SVC	False	0.752	0.752	0.72

6. WEB APPLICATION FOR EXERCISE CORRECTION

The primary purpose of the web application is to applied and showcase every trained model to catch errors and provide details instruction to fix the errors from exercise videos. Vue.js and Django are the tools used to build this web application.

Due to the shortage of time and demonstration purpose, the main and only feature of the web is for user to upload his/her exercise video, the system will analyze the it and provide any related statistics (rep counter, …) or tackle any errors found and suggest advices to fix those errors.

The trained models entirely employ by the Django server. The server will provide an API for client to upload video file, receive analyzed statistic and analyzed video files. The Figure 17 below is the illustrated process for a video analyzing by the system.
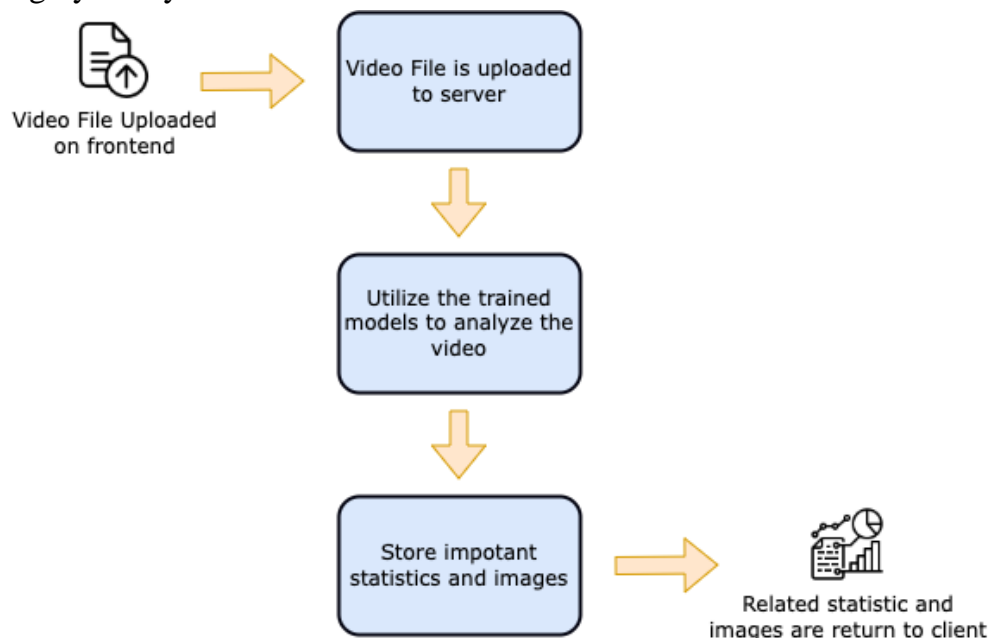

Figure 17: Web server - Video analyzing process

For each video, here is the process of analyze it from the server:
- Error detection with trained models: Depends on the exercise type, the correspond model will be chosen for the error detection. Every error will be recorded in order to return back to the client. Moreover, any frame which has a detected error will be saved.
- Video and images saving: The full analyzed video is also saved with detail drawing of landmarks and any other statistics. An API is also provided for client to stream the analyzed video.

To sum up, the approach steps to complete the model have been presented in detail and in detail. At the same time, the construction of a web application to demonstrate the predictive ability of the model was also presented. The next section will focus on the process of evaluating and testing the quality of the mentioning model and web application.

CHAPTER 3: MODEL EVALUATION

This section will give an objective assessment of all the trained models, discuss about the performance of the models trained by the mentioned approach in the previous section and explain some factors affecting the evaluation results and also present the testing process.

1. EVALUATION METRICS

Evaluation metrics are used to measure the quality of the statistical or machine learning model. Evaluating machine learning models or algorithms is essential for any project. There are many different types of evaluation metrics available to test a model. Below are the metrics used to evaluate trained models from this study. this thesis:

- *Confusion matrix*: provides a detailed overview of the classification. For a better performing model True Positive (TP), True Negative (TN) must be high and False Negative (FN), False Positive (FP) should be low as possible.
    - 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 positives to the total predicted positives.

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

- *Recall*: tells the proportion that the model is accurately classifying the true positives. It is also called Sensitivity.

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

- *F1 score*: defined as the harmonic mean of precision and recall. The higher the precision and recall, the higher the F1-score.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

- *F1 score curve*: a graph shows the tradeoff between F1 score and different threshold. By observing the changes of F1 score, the curve can help to find the optimal threshold for a model.
- *ROC curve (receiver operating characteristic curve)*: a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate (TPR) and False Positive Rate (FPR).
  - $TPR = TP/(TP + FN)$
  - $FPR = FP/(FP + TN)$
- *AUC (area under the ROC curve)*: measures the entire two-dimensional area underneath the entire ROC curve. AUC provides an aggregate measure of performance across all possible classification thresholds.

2. EVALUATION RESULTS
2.1. Bicep Curl – Lean Too Far Back

According to the metrics from the models training experiments mentioned shown in Table I. Model training experiments for Bicep Curl's error. The best model for this error is the K-Nearest Neighbors (KNN) model. As shown below, the model provides decent results.

	Correct form	Lean too far back
Correct	338	1
Lean too far back	16	249

Table V. Bicep Curl error - Confusion matrix

From the above matrix, here are the others metrics:

	Precision	Recall	F1 Score
Correct	0.955	0.997	0.975
Lean too far back	0.996	0.94	0.966
Average	0.976	0.968	0.971

Table VI. Bicep Curl error - Evaluation results

The choice of confidence threshold affects the evaluation and detection results of the model, choosing the appropriate confidence threshold will give better result. Observing the F1 curve and ROC curve below can help with choosing the

optimal confidence threshold. From the F1 curve chart, the lines between classes are closed to each other, and the threshold which yield the best F1 score is around 0.5.
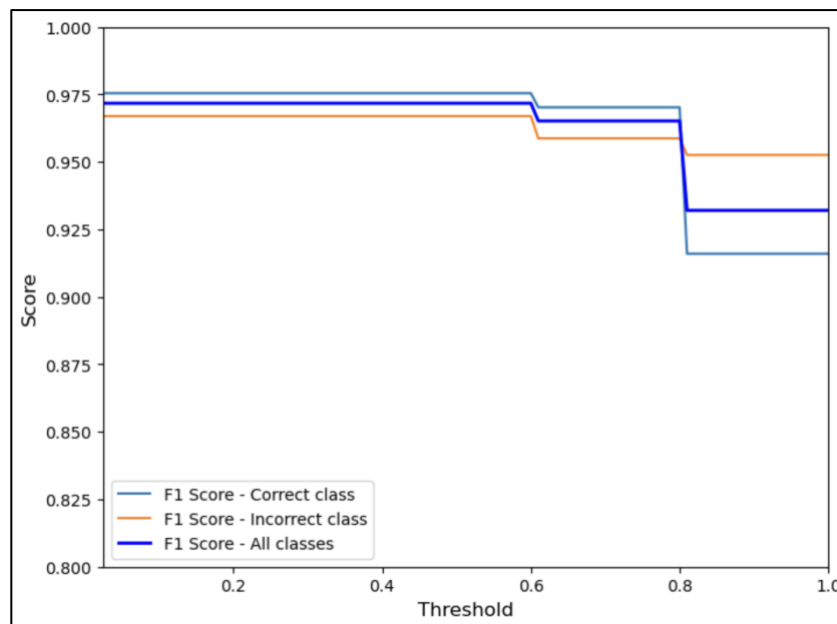
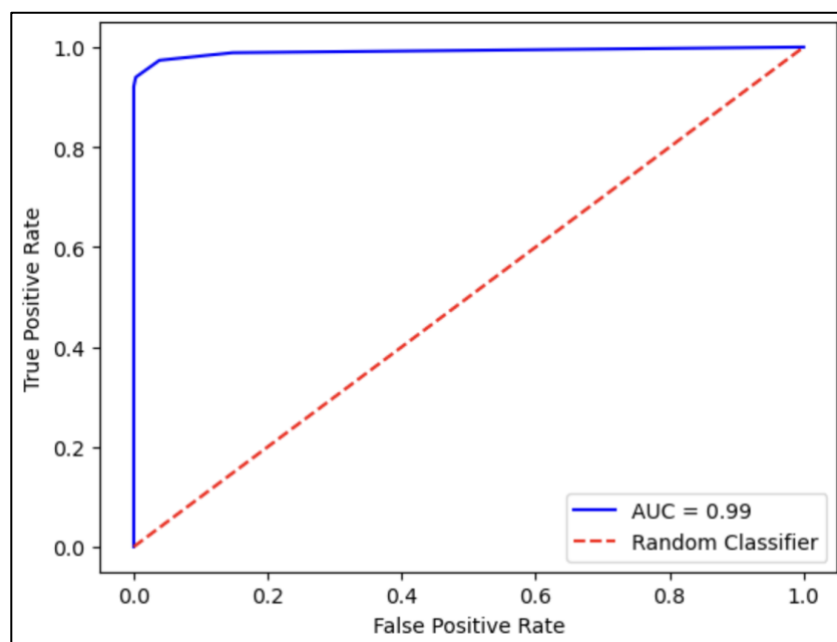

Figure 18: Bicep curl error - F1 curve

Figure 19: Bicep curl error - ROC curve

2.2. Basic plank errors

According to the metrics from the models training experiments mentioned shown in Table II. Model training experiments for Plank's error detection. The best

model for this error is the Logistic Regression (LR) model. As shown below, the model provides decent results.

	Correct	High back	Low back
Correct	234	0	0
High back	1	240	0
Low back	2	0	233

Table VII. Plank error - Confusion matrix

From the above matrix, here are the others metrics:

	Precision	Recall	F1 Score
Correct	0.987	1.0	0.994
High back	1.0	0.996	0.998
Low back	1.0	0.991	0.996
Average	0.996	0.996	0.996

Table VIII. Plank error - Evaluation results

Observing the F1 curve and ROC curve below can help with choosing the optimal confidence threshold. From the F1 curve chart, the lines between classes are closed to each other, and the threshold which yield the best F1 score is around 0.6.
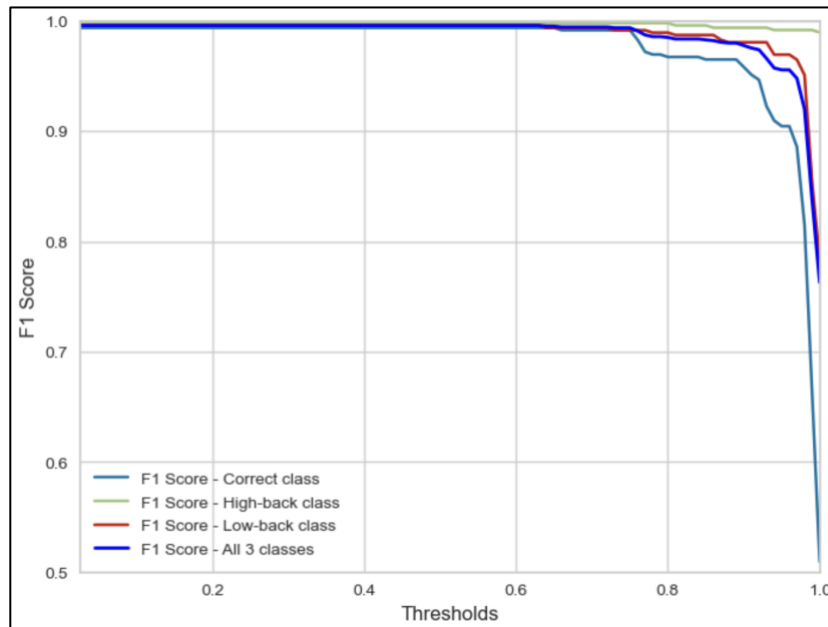
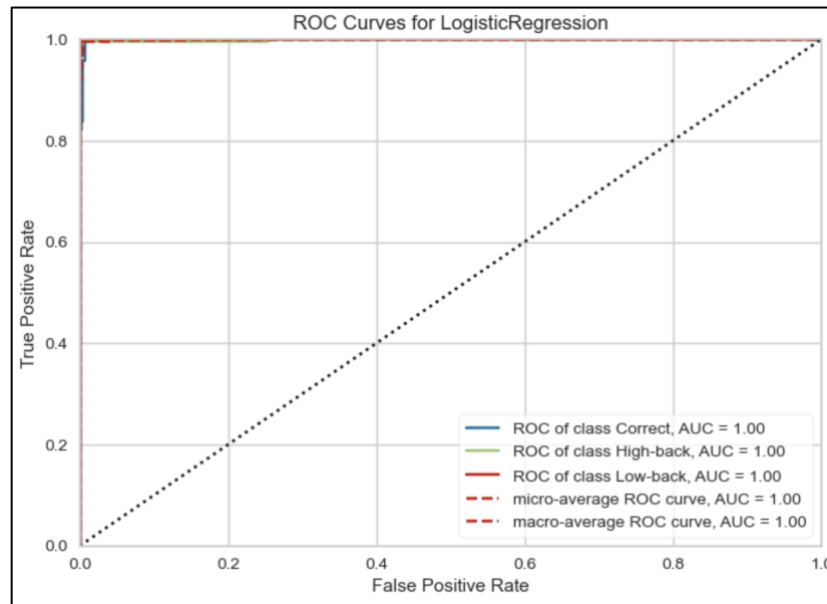

Figure 20: Plank error - F1 curve

Figure 21: Plank error - ROC curve

2.3. Basic squat stage

According to the metrics from the models training experiments mentioned shown in Table III. Model training experiments for Squat's stage. The best model for stage detection is the Logistic Regression (LR) model. As shown below, the model provides decent results.

	Up	Down
Up	428	2
Down	3	420

Table IX: Squat stage - Confusion matrix

	Precision	**Recall**	**F1 Score**
Down	0.993	0.995	0.994
Up	0.995	0.993	0.995
Average	0.994	0.994	0.995

Table X: Squat stage - Evaluation results

Observing the F1 curve and ROC curve below can help with choosing the optimal confidence threshold. From the F1 curve chart, the lines between classes are closed to each other, and the threshold which yield the best F1 score is around 0.5.
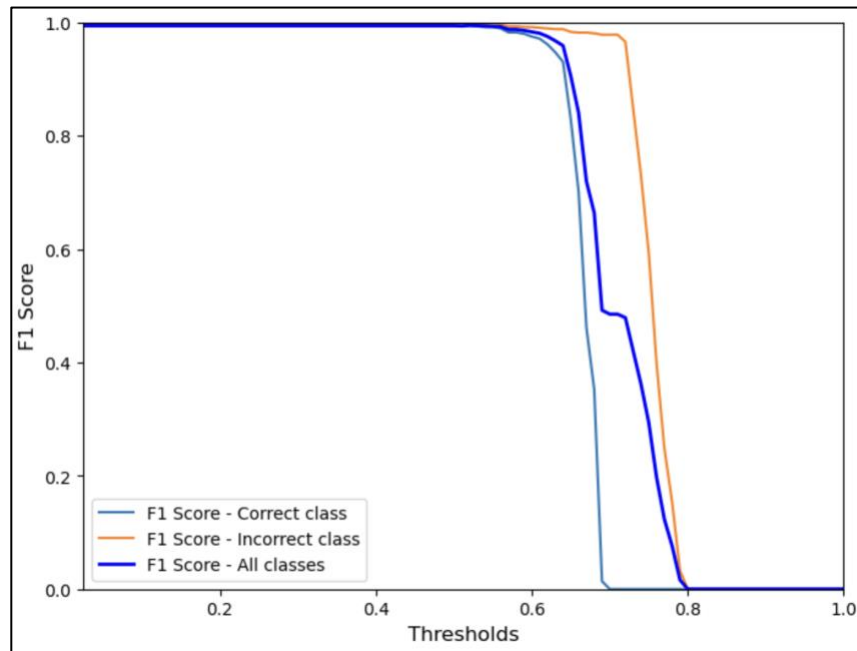
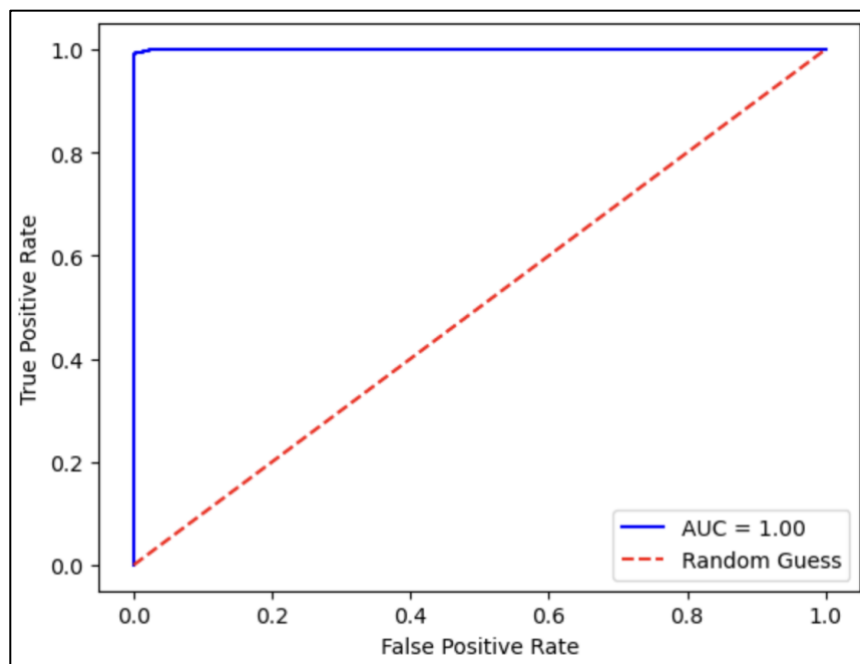*Figure 22: Squat stage - F1 score curve*

Figure 23: Squat stage - ROC curve

2.4. Lunge error – Knee over toe

According to the metrics from the models training experiments mentioned shown in Table IV. Model training experiments for Lunge's error. The best model for this error is the Logistic Regression (LR) model. As shown below, the model provides decent results.

	Correct	Knee over toe
Correct	545	1
Knee over toe	30	531

Table XI. Lunge error - Confusion matrix

	Precision	Recall	F1 Score
Correct	0.998	0.947	0.972
Knee over toe	0.948	0.998	0.972
Average	0.972	0.972	0.972

Table XII. Lunge error - Evaluation results

Observing the F1 curve and ROC curve below can help with choosing the optimal confidence threshold. From the F1 curve chart, the lines between classes are closed to each other, and the threshold which yield the best F1 score is around 0.5.
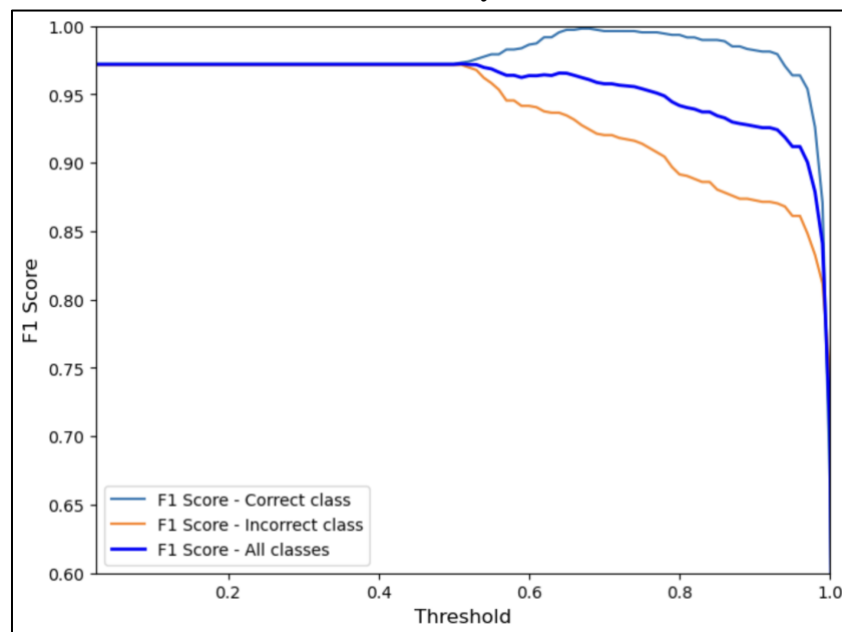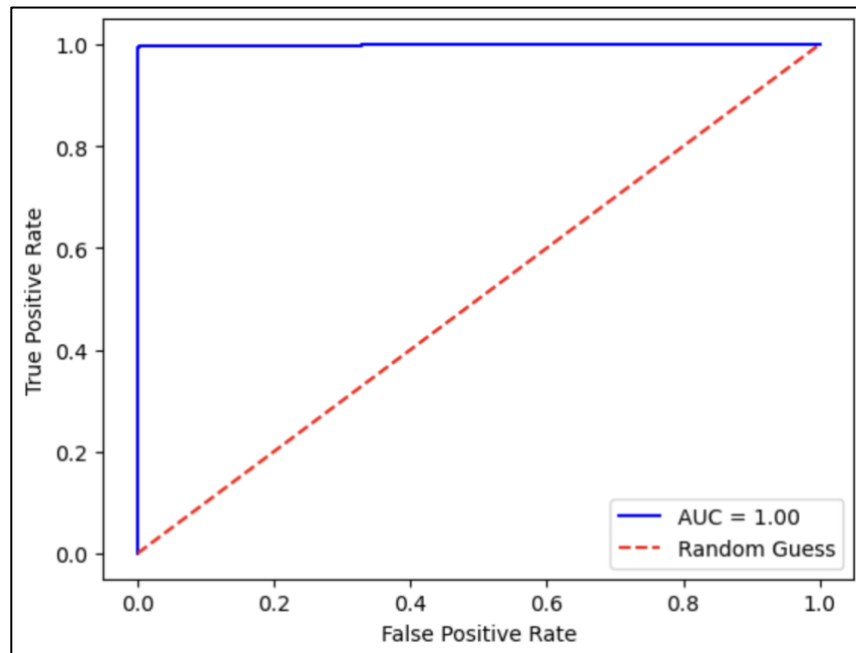

Figure 24: Lunge error - F1 score curve

Figure 25: Lunge error - ROC curve

CHAPTER 4: TESTING APPLICATION

1. TEST CRITERIA

In order to avoid errors while deploying the application and making sure to application works as expected, the black box testing method was used to conduct system testing with the criteria as:

- Test success criteria: test results meet requirements in specification document, test results must return desired results and have success status or return error with clear and understandable message.
- Test failure criteria: actual test results differ from the expected results noted in the specification document or an unmentioned error occurs.

2. TEST ENVIRONMENT

The results of tests can be influenced by many objective and subjective factors such as lighting, distance, angle, … Therefore, in order to reduce to minimum these factors, the tests were conducted under the following conditions for every exercise:

- *Distance*: Tester must maintain a distance in that the whole body should be visible to camera. However, the distance also must be closed enough to reserve the resolution.
- *Angle*: Depends on the exercise, every important joint or landmark which is essential to correctly perform must be visible to the camera.
- *Lighting*: Lighting in testing videos must be maintained at medium level which reserves the visibility of the tester.

Test system information:

- Hardware:
  - o Personal computer with stable internet connection.
  - o Apple M1, RAM 16GB, SSD 256GB.
- Software:
  - o Operating System: macOS Ventura version 13.0.1.
  - o Google Chrome version 107.0.5304.121 (arm64).

3. TEST FEATURES

The table below displays all the features that were tested with their expected results. For each exercise, an uploaded video correspond to that exercise must contain the correct form and all the incorrect forms.

Table XIII. Test - Features require to be tested.

No	Feature	Description	Expected Results
1	Analyzed Upload Exercise Video	− User uploads an exercise video. − User chooses a type of exercise. − System displays statistic gathered from the video. − System displays a full analyzed video. − If errors are found, system displays list of errors caught from the video.	− Successfully upload video. − Successfully display analyzed statistics. − Successfully display analyzed video. − Successfully display images of error if there are ones.

From the list of features or functions that need to be tested listed in Table XIII, test scenarios were formed for each function. Each case will specify input data, expected results, actual results along with evaluation comments. The test scenarios for each function are described in Table XIV below.

Table XIV. Test features' scenario

Code	Scenario	Input	Expected output	Actual Output
QB-01	Detected incorrect Bicep Curl forms.	Uploaded Video	− Display Bicep Curl's counter. − Display images of the error along with its timestamp. − Display full video with detail analysis.	As expected
QB-02	Detected incorrect Plank forms.	Uploaded video	− Display images of the error along with its timestamp. − Display full video with detail analysis.	As expected

Code	Scenario	Input	Expected output	Actual Output
QB-03	Detected incorrect Squat forms.	Uploaded Video	– Display Squat's counter. – Display images of the error along with its timestamp. – Display full video with detail analysis.	As expected
QB-04	Detected incorrect Lunge forms.	Uploaded Video	– Display Lunge's counter. – Display images of the error along with its timestamp. – Display full video with detail analysis.	As expected

In conclusion, the application successfully meets the test criteria as seen from the test results. Below are some of the frames which are detected as error from the tests.

Figure 26: Test QB-01. Bicep Curl error detection

Figure 27: Test QB-02. Plank error detection

Figure 28: Test QB-03. Squat error detection

Figure 29: Test QB-04. Lunge error detection

It can be seen that the efficiency of object detector in this study is quite high when apply to the web-based application, so it can be said that this system successfully meets the set goals. The next section will discuss about any shortcoming and future implementations of the system.

CONCLUSION

1. FINAL RESULTS

Throughout the process of researching and working on this thesis, I was able to expose myself to a lot of machine learning theory, technology and tools. With the help of MediaPipe Pose framework, all the trained models yielded results that meet my expectation. All models can helps detecting or classifying error in 4 exercises (bicep curl, basic plank, basic squat and lunge) with highly effective results. The models are also evaluated with positive outcome.

However, it is worth noticing that the majority of the data (both train set and test set) is my own self-recorded videos, which definitely affected the training process of the models and can be the reasons for their high accuracy.

2. FUTURE WORKS

Due to limited time and resources, I mainly focused in developing and finalizing the basic and most important errors of each exercise. In other hand, I also prepare some future paths to improve the system in the future:

- Expanding the datasets with other contributors' videos to remove the models' bias.
- Researching on more common error for the study's 4 exercises.
- Researching on more popular exercises which also being commonly perform incorrectly.

REFERENCES

[1]   Alexandra Merisoiu, "Lunges – Common Mistakes And Correct Technique", 25/12/2015. Available: https://themerisoiutechnique.com/2015/12/25/lunges-common-mistakes-and-correct-technique/.

[2]   Anilkumar, Ardra, Athulya KT, Sarath Sajan, and Sreeja KA. "*Pose Estimated Yoga Monitoring System.*" Available at SSRN 3882498 (2021).

[3]   Babara Gibson, "WHY FORM IS IMPORTANT IN WORKING OUT", 29/07/2016. Available: https://www.fitness19.com/why-form-is-important-in-working-out/.

[4]   Bojana Galic, "6 Biceps Curl Mistakes That Make This Exercise Way Less Effective", 13/10/2019. Available: https://www.livestrong.com/article/13722123-biceps-curl-mistakes.

[5]   Chen, Steven, and Richard R. Yang. "*Pose Trainer: correcting exercise posture using pose estimation.*" arXiv preprint arXiv:2006.11718 (2020).

[6]   Deven Rathore, "Building A Video Streaming App With Nuxt.js, Node And Express", 13/04/2021. Available: https://www.smashingmagazine.com/2021/04/building-video-streaming-app-nuxtjs-node-express/.

[7]   Eric Bugera, "Are You Making These 5 Squat Mistakes? Here's How To Fix Them", 21/09/2021. Available: https://barbend.com/squat-mistakes/.

[8]   Jakub Czakon, "F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose?", 14/11/2022. Available: https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc.

[9]   Jason BrownLee, "How to Use ROC Curves and Precision-Recall Curves for Classification in Python", 31/08/2018. Available: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python.

[10]   Josyula, Rohit, and Sarah Ostadabbas. "*A Review on Human Pose Estimation.*" arXiv preprint arXiv:2110.06877 (2021).

[11] Kukil, "Building a Poor Body Posture Detection and Alert System using MediaPipe", 08/03/2022. Available: https://learnopencv.com/building-a-body-posture-analysis-system-using-mediapipe/.

[12] Lipton, Zachary C., Charles Elkan, and Balakrishnan Naryanaswamy. "Optimal thresholding of classifiers to maximize F1 measure." In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 225-239. Springer, Berlin, Heidelberg, 2014.

[13] Lugaresi, Camillo, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang et al. "*Mediapipe: A framework for perceiving and processing reality.*" In Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR), vol. 2019. 2019.

[14] Lugaresi, Camillo, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang et al. "*Mediapipe: A framework for building perception pipelines.*" arXiv preprint arXiv:1906.08172 (2019).

[15] Siddharth M, "Performing multi-class Classification on FIFA Dataset Using Keras",   14/07/2021.   Available: https://www.analyticsvidhya.com/blog/2021/07/performing-multi-class-classification-on-fifa-dataset-using-keras/.

[16] Tiffany Ayuda, "3 Common Plank Mistakes (And How to Avoid Them)", 22/10/2022. Available: https://dailyburn.com/life/fitness/how-to-do-a-plank/.