Kafr El Sheikh University

Faculty of Computers & Information

Bioinformatics Department

GymEye

By

Samira Waheed Hegazy

Mahmoud Ayman El Ghalban

Ahmed Mohamed Ramadan

Aya Abdul Hakeem El Tahawy

Mohamed Essam Abo Gad

Mustafa Reda Barkat

Under Supervision of

Associate Professor Ahmed Elashry

Information Systems Lecturer

Information Systems Department

Faculty of Computer & information Sciences

Kafr El Sheikh University

Team members' contacts

Name	Contact number	Email
Samira Waheed Hegazy	+201030015594	Samirawaheed2000@gmail.com
Mahmoud Ayman El Ghalban	+201554165944	Mahmoudayman9112000@gmail.com
Ahmed Mohamed Ramadan	+201002267826	Ahmedmramadan33@gmail.com
Aya Abdul Hakeem El Tahawy	+201143424759	Aya497694@gmail.com
Mohamed Essam Abo Gad	+201124039596	Mohamed.e2048@gmail.com
Mustafa Reda Barkat	+201066327835	Barkatmustafa16@gmail.com

Kafr El Sheikh University

Faculty of Computers & Information

Bioinformatics Department

GymEye

June 2023

Acknowledgements

All praise and thanks to ALLAH for providing us the ability to complete this work and for anyone provided us with help and support.

We would like to express our gratitude *to Prof Dr.Ahmed Elashry* for his great effort with us. He gave us good notes that more helped to do this project. We would like to thank his for his care to represent something good. We hope to make his proud.

We would like to thank *Eng. Alaa Sehsaah* in a computer science department who gave us more from her knowledge and experience. For her continuous support in implementation. She always present whenever we need her help.

Finally, we would thank our family and all people who gave us support and encouragement, especially our friends for shearing their knowledge.

Abstract

It is important while performing specific exercise to know the best and right poses that help building a good healthy body and keep good fitness otherwise, you will suffer pain and injuries and take long time to recover from the side effects of ignoring the right instructions and poses of exercises. Also, it is costly to afford a personal coach or trainer to help practicing exercise in right way. So, we decided to build an exercise pose correction application that detects specific errors in videos recorded by user and suggests the right pose or way to perform that exercise, displaying instructions and images clarifying available exercises in the system. Also, we will introduce some articles related to performing exercises and gym workouts that help people understand the life of workouts and fitness. We used deep learning in building models that recognize and detect bad poses in the user`s recorded video by using pose estimation.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

L_CLASS	Lumbar Class: [means 0 or 1 correct or not**.]**
T_CLASS	Torso Class: [The same as Lumbar.]
L_PROB	Lumbar Probability: [Means the probability of being the right class.]
T_PROB	Torso Probability: [The same as Lumbar.]
LC	Lumbar Correct
LE	Lumbar Error
TC	Torso Correct
TE	Torso Error
UNK	Unknown
C	for correct back pose
L	is for Lean back pose
R_PC	Right Peak Contraction.
L_PC	Left Peak Contraction
R_LUA	Right Loose upper arm
L_LUA	Left Loose Upper Arm

Chapter 1

Introduction

1.1 Problem Definition

Detecting errors in gym exercise execution and providing feedback on it is crucial for preventing injuries and maximizing muscle gain. However, feedback from personal trainers is a costly option and hence used only sparingly-typically only a few days a month, just enough to learn the basic form. We believe that an automated computer vision-based workout form assessment (e.g., in the form of an app) would provide a cheap and viable substitute for personal trainers to continuously monitor users' workout form when their trainers are not around. Ours, on the other hand, is the first work to tackle the problem of workout form assessment distinctly in complex, real-world gym scenarios, where, people generally record themselves using ubiquitous cellphone cameras that they place somewhere in the vicinity; which results in large variances in terms of camera angles, alongside clothing styles, lighting, and occlusions due to gym equipment (barbells, dumbbells, racks).

These environmental factors combined with the subtle nature of workout errors (refer to Fig. 1.1) and the convoluted, uncommon poses that people go through while exercising, cause major challenges for OTS pose estimators (refer to Fig. 1.1), and consequently, workout form errors cannot be reliably detected from pose. We propose to replace the error-prone pose estimators with our more robust domain knowledge-informed self-supervised representations that are sensitive to pose and motion, learned from unlabeled videos — helps in avoiding annotation efforts. Towards those ends, our contributions are as follows.

Figure 1.1 Concepts:

(a) Errors of small magnitude generally occurring in workout form: Good column shows correct posture/execution (knees should be outwards), while the Bad column shows erroneous form during exercising.

Figure 1.1-Exersise good & bad form

(b) Examples of failures of off-the-shelf 2D and 3D-pose estimators in real-world gym scenarios (compare the discrepancies in pose estimation with the magnitude of the errors to be detected). We tackle the problem of detecting errors in workout form. To do so more accurately, we replace the error-prone pose estimators with our more robust fitness domain-oriented representations learnt using self-supervision.

The following statistics show the rate of injuries in gyms:

1. Exercise equipment accounted for approximately 409,000 injuries in 2021, making it the highest category of sports and recreation injuries.
2. Of those injuries, 76,078 were sustained by individuals between the ages of 15 and 24.
3. A study published in 2013 found a rate of 3.1 injuries for every 1,000 hours spent doing Cross Fit training.

4. A Salt Lake City hospital treated 52 individuals between May 2011 and November 2012 for injuries sustained at local jump gyms, with some patients requiring multiple medical visits.
5. Emergency rooms across the United States treat more than 10,000 individuals each day for injuries resulting from sports, recreation, and exercise.

These statistics highlight the significant problem of injuries sustained during exercise, particularly in gym settings, and demonstrate the need for effective strategies to promote safe exercise practices and prevent injuries.

1.2 Objectives

- Mobile application that enable the trainers to explanatory video of how to perform the exercise correctly
- Application provides the trainee with the ability to upload a video of himself while performing the exercise
- System analyzes the videos uploaded by the user and gives him results about the correctness of his performance of the exercise
- Application provides some sports and health articles
- System provides all these functions through mobile application and website
- Limit the injures occur because of the incorrect exercises that practice without coach lead them.

1.3 Stakeholders

Users are anyone who wants to improve his/her performance for various types of exercise. The users should be able to display their exercises by attached video that clarify his/her performance whether the user is doing right or wrong. And also he

can know his/her body needs for essential body nutrients that can help him/her to be healthier generally and specially to be fit to practice his exercises better.

The user should be able to do the following functions:

- Uploading a video for his/her exercises.
- Giving advice through his data to know his/her body needs for essential body nutrients.
- Reading some sports and health articles.
- Receiving the results and some advice about the correctness of his performance of the exercise after uploading the video.
- Displaying an explanatory video of how to perform the exercises correctly that he/she wants to practice.
- Submitting suggestions and complaints that will be received and dealt by the administrator.

Also Administrator is also one of our system users who have many responsibilities.

The administrator should be able to do the following functions:

- Monitoring the site through the dashboard.
- Adding and editing sports and health articles on the site.
- Monitoring system performance and troubleshooting issues.
- Ensuring security and efficiency of IT infrastructure.
- Ensure security through access controls, backups and firewalls.
- Upgrade systems with new releases and models.
- Receiving and dealing suggestions and complaints from the users.

1.4 Scope

This system is designed for individuals seeking to improve their exercise performance across a range of activities. Users can easily display their exercises by attaching videos that clearly show their performance, allowing them to identify areas for improvement and optimize their workouts.

Additionally, the system provides personalized recommendations for essential nutrients that can support overall health and fitness, helping users to achieve their goals and perform their exercises with greater success. Overall, this system is ideal for anyone looking to enhance their exercise experience and achieve better results.

1.5 Constrains

- The mobile app must be compatible with both Android and IOS devices.
- The app must be able to handle a large volume of user-generated video content and data.
- Users need good lighting in the place which does his exercises.
- The app's algorithms for analyzing video content must be accurate and reliable.
- Users need to have good internet network.
- Users will have access to a mobile device with a camera and internet connection.
- Users will have a basic understanding of how to use a mobile app and record video content.
- The app's video analysis capabilities will be dependent on the quality of the video content provided by users.

1.6 Time Plan

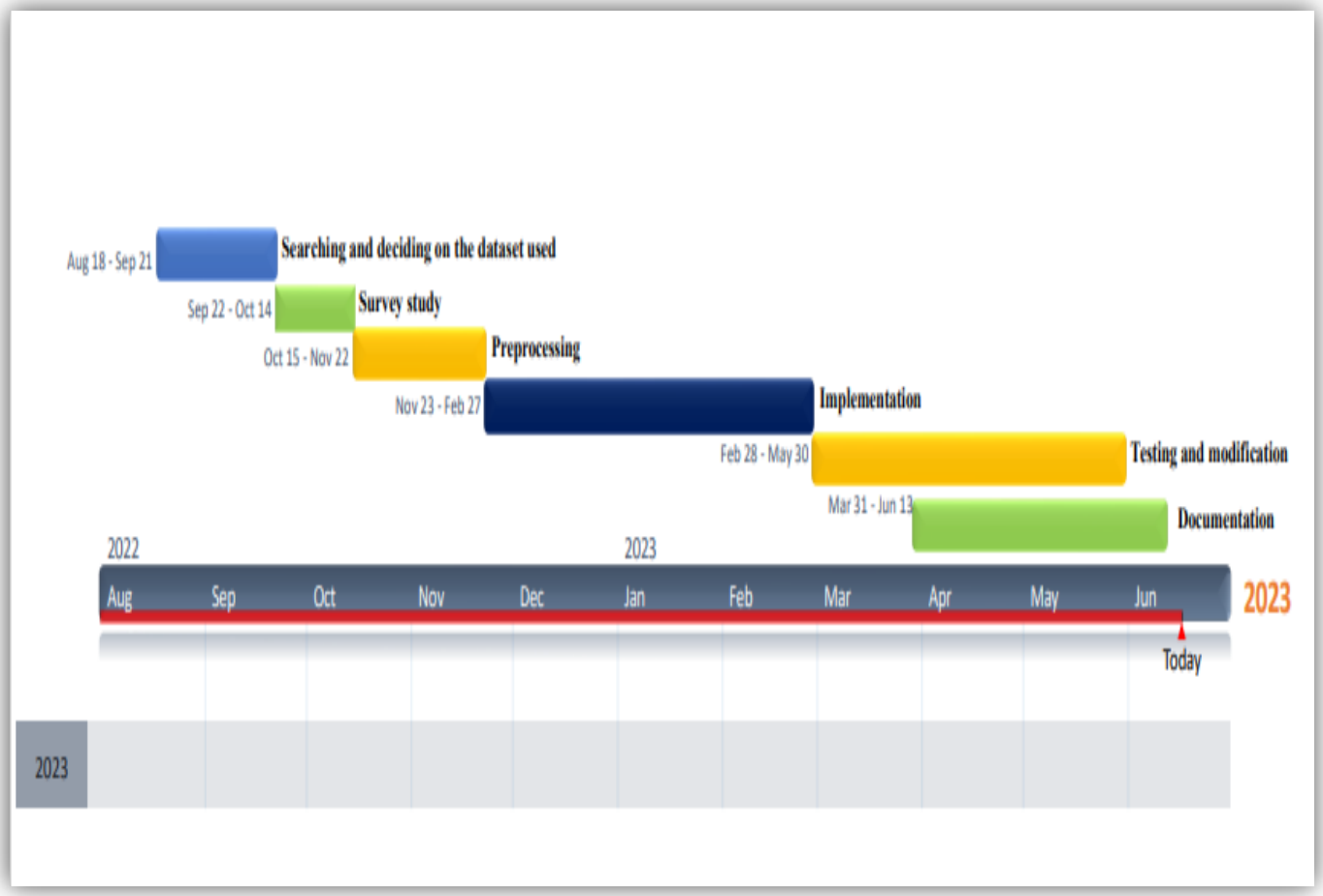


Figure 1.2- Time plan

1.7 Documentation Outline

Our Documentation is organized as follows:

Chapter 2:

A detailed description of the project field, description and comparison of existing similar applications.

Chapter 3:

System requirements, user requirements, also detect project budget and the list of risks the project have.

Chapter 4:

System Architecture, the use case diagram, data flow, activity, sequence, state, and class diagram. Also contain the UI/UX of the application.

Chapter 5:

Implementation of several models and the trials made and results that come out of them.

Chapter 6:

Testing the software for publishing.

Chapter 7:

Conclusion and future work.

References:

Websites, articles and papers used to support our work.

Chapter 2

Literature Review

2.1 Description of detection and classification filed

The detection and classification tasks are mainly related to computer vision, ML and DL field which we used neural network technique to obtain the desired result.

Computer vision is a scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do.

2.1.1 How dose classification work?

Classification is a process of categorizing a given set of data into classes; it can be performed on both structured and unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

Classification Model: The model predicts or draws a conclusion to the input data given for training; it will predict the class or category for the data.

2.1.2 How dose detection work?

Iterating over the problem of localization plus classification we end up with the need for detecting and classifying multiple objects at the same time. Object detection is the problem of finding and classifying a variable number of objects on an image (Frames) that takes from the video. The important difference is the "variable" part. In contrast with problems like classification, the output of object detection is variable in length, since the number of objects detected may change from video to video.

2.1.3 What is machine learning?

Machine learning (ML) is the study of algorithms and mathematical models that computer systems use to progressively improve their performance on a specific task. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task.

Machine learning algorithms are used in the applications of email filtering, detection of network intruders, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers.

The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics. Machine learning tasks are classified into several broad categories.

In supervised learning, the algorithm builds a mathematical model of a set of data that contains both the inputs and the desired outputs. For example, if the task were determining whether an image contained a certain object, the training data for a supervised learning algorithm would include images/ videos with and without that object (the input), and each image would have a label (the output) designating whether it contained the object.

In special cases, the input may be only partially available, or restricted to special feedback. IN unsupervised learning, the algorithm builds a mathematical model of a set of data which contains only inputs and no desired outputs.

Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories. Dimensionality reduction is the process of reducing the number of "features", or inputs, in a set of data.

Unsupervised learning algorithms are used to find structure in the data, like grouping or clustering of data points. Unsupervised learning can discover patterns in the data, and can group the inputs into categories, as 18 | P a g e in feature learning. Dimensionality reduction is the process of reducing the number of "features", or inputs, in a set of data.

2.1.4 What is Deep learning?

It is an extension of machine learning, and machine learning is the study of techniques that let machines carries out various tasks without being explicitly programmed to do so.

Machine learning systems have three principal components: an input, a node/neuron, and an output. The input is the data that is being fed into the machine learning system, and the node/neuron represents one type of mathematical algorithm that will manipulate this information.

Finally, the output of the system is the network's decision or inference about the data after it has been manipulated by the algorithm. These three components represent a simple neural network. A deep neural network is the term applied to many simple neural networks that have been linked together.

Deep neural networks have multiple layers, where the output of the first layer becomes the input of the second layer, and the output of the second layer becomes the input of a third layer, and so on.

A neural network operates by analyzing the relevant features of the input data, detecting patterns in that data, and then making predictions about that data or similar data. The deeper a neural network is, the more layers it has, the more complex the pattern it can distinguish. The term "deep learning" just refers to using deep neural networks to analyze data, discern the relevant patterns in the data, and make predictions about the data.

2.1.5 What is the Different between ML and DL?

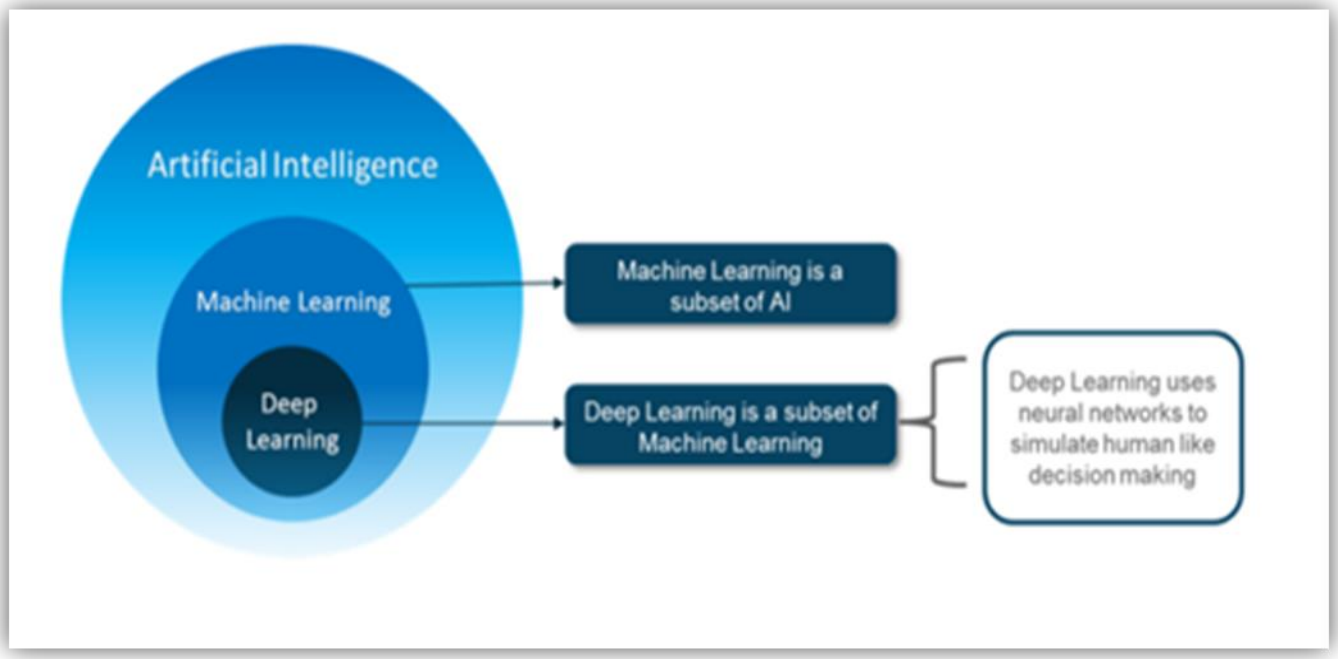


Figure 2.1-Artificial Intelligence

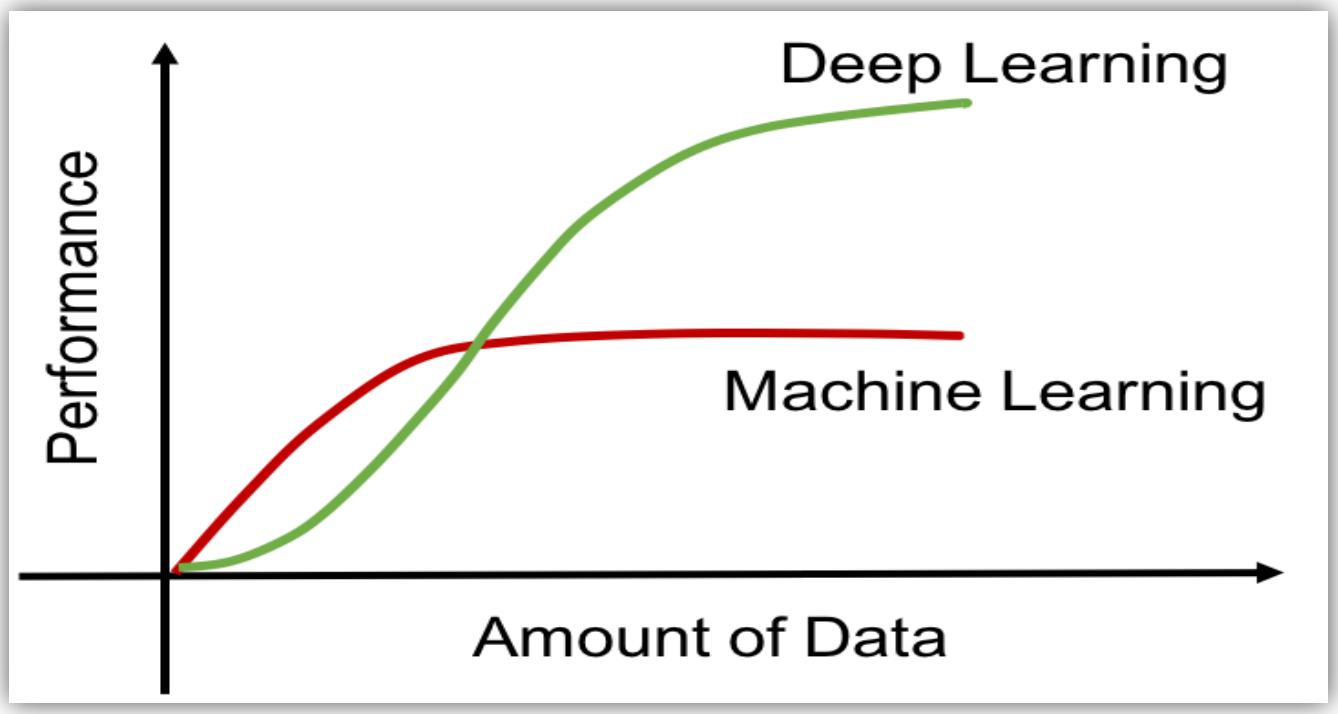


Figure 2.2- Amount of data in DL and ML

As one can see, Deep Learning is subset of Machine Learning while Machine Learning is part of AI. Artificial Intelligence is a technique which enables machines to mimic human behavior. Ultimate aim of AI is to make intelligent machines that can perform human behavior and take own smart decision.

Machine Learning is sub-part of AI that uses statistical methods that enable machines to improve with experience. Deep Learning is sub-part of ML that make use of Neural Networks (similar to neurons in human being) to simulate human brain like behavior. But why do we need deep learning?

The reason is ML algorithms can't play longer in higher dimension and/or higher number of observations data. Another issue with ML is one has to fine-tune the number of parameters. In case of deep learning, neural network will decide on its own about important features. Basically deep learning mimics the way our brain tend to learn from previous experience.

2.2 Related work

There are many solutions available that use machine learning and computer vision to provide feedback on exercise form and technique.

These solutions all aim to help users improve their exercise form and technique through the use of technology. Here are a few examples with their pros and cons:

2.2.1 Forme.ai

- **<u>introduction</u>**

This platform uses computer vision to analyze the user's form during exercises and provide real-time feedback on areas for improvement.

- **Advantages**

- Real-time feedback during exercise.
- Customized recommendations based on the user's goals and abilities.
- Provides a comprehensive analysis of each exercise.
- Offers video demonstrations for each exercise.

- **Disadvantages**

- Limited number of exercises available.
- May not be suitable for more complex exercises or movements.

2.2.2 Mirror

- **Introduction**

Mirror is an interactive home gym that uses computer vision to provide feedback on form and technique during exercises.

- **Advantages**

- Provides a full-body workout in one machine.
- Offers personalized workout recommendations based on user's goals and preferences.
- Real-time feedback on form and technique during exercise.
- Wide variety of exercises and workouts available.

- **Disadvantages**

- Expensive compared to other solutions.
- May not be suitable for more advanced athletes or bodybuilders

2.2.3 Kinetisense

- **Introduction**

This platform uses motion capture technology to assess the user's movement during exercises and provide personalized feedback on areas for improvement.

- **Advantages**

- Provides a comprehensive analysis of the user's movements and form Offers personalized feedback and recommendations based on the user's abilities and goals. Can be used in a variety of settings, including physical therapy and sports performance training
- Can be used with a wide range of exercises and movements.

- **Disadvantages**

- Requires specialized equipment and software.
- May be more difficult to set up and use than other solutions.
- Can be expensive compared to other solutions.

2.2.4 Push:

- **Introduction**

This app uses machine learning to analyze the user's performance during exercises and provide personalized feedback and recommendations for improvement.

- **Advantages**

- Offers real-time feedback during exercise.
- Provides personalized recommendations based on the user's performance and goals.
- Can be used with a wide range of exercises and movements.
- Offers video demonstrations for each exercise.

- **Disadvantages**

- May not be suitable for more advanced athletes or bodybuilders.
- Limited number of exercises available.

2.2.5 GymCam:

- **Introduction**

This platform uses computer vision to track the user's movements during exercises and provide real-time feedback on form and technique.

- **Advantages**

- Provides real-time feedback during exercise.
- Can be used with a wide range of exercises and movements.
- Offers video demonstrations for each exercise.

- **Disadvantages**

- May not be suitable for more complex exercises or movements.
- Limited number of exercises available.

These solutions all aim to help users also our project could take a similar approach by incorporating machine learning and computer vision to analyze the user's movements and provide feedback on form and technique.

2.3 Comparison of previous solutions

Solutions	Cons	Props
Forme.ai	Real-time feedback, customized recommendations, comprehensive analysis, video demonstrations.	Limited exercises available, not suitable for complex movements.
Mirror	Full-body workout, personalized recommendations, real-time feedback, wide variety of exercises.	Expensive, not suitable for advanced athletes.
Kinetisense	Comprehensive analysis, personalized feedback, can be used in different settings, wide range of exercises.	Requires specialized equipment and software, difficult to set up, expensive.
Push	Real-time feedback, personalized recommendations, wide range of exercises, video demonstrations.	Limited exercises available, not suitable for advanced athletes.
GymCam	Real-time feedback, wide range of exercises, video demonstrations.	Limited exercises available, not suitable for complex movements.

Table 2.1- Previous solutions

Chapter 3

Planning and Requirements

3.1 Development requirements

The operating environment for the system is as listed below:

- Operating system: Android and IOS.
- Database: MySQL database.
- Platform: Flutter for building mobile application UI, ASP flask for backend development.
- Client/Server System.

3.2 Cost & budget

There is no cost as all needs for the development is available, but we will need an enough budget for the marketing plan.

3.3 Functional requirements

3.3.1 User requirements

3.3.1.1 The ability to create an account and login:

- Users should be able to create a new account by providing their basic information, such as name, email, and password.
- The system should verify the email address provided by the user to prevent fake accounts.
- Users should be able to log in to their accounts using their email and password.
- The system should provide the option for users to reset their passwords if they forget them.

3.3.1.2 The ability to browse articles on sports and health:

- Users should be able to browse articles related to sports and health on the platform.
- The system should provide search functionality to help users find specific articles.
- Users should be able to filter articles by category, date, or popularity.

3.3.1.3 The ability to watch a video showing the correct performance of exercise:

- Users should be able to watch videos showing the correct performance of different types of exercises.
- The system should provide search functionality to help users find specific exercise videos.
- Users should be able to filter exercise videos by category, difficulty level, or popularity.

3.3.1.4 The ability to upload sports and health articles through the admin only:

- The admin should be able to upload new articles related to sports and health on the platform.
- The admin should be able to edit or delete existing articles.
- The admin should be able to review and approve user-submitted articles before publishing them on the platform.

3.3.1.5 Make the user able to upload a video while performing the exercise:

- Users should be able to record a video while performing an exercise using their mobile device.

- Users should be able to upload the recorded video to the platform for analysis.

3.3.1.6 The ability to keep the results of video analysis and browse them later:

- The system should store the results of video analysis for each user.
- Users should be able to browse their previous analysis results.

3.3.1.7 The ability to receive the results of the video analysis to see how many percent his performance was correct:

- The system should analyze the uploaded video and provide the user with feedback on their exercise performance.
- The feedback should include a percentage score indicating how correct the exercise performance was.

3.3.2 System requirements

3.3.2.1 The ability to create an account and login:

- The system should provide a user registration form to collect user information.
- The system should store user information in a database.
- The system should provide a login form for users to enter their email and password.
- The system should authenticate users by verifying their email and password.

3.3.2.2 The ability to browse articles on sports and health:

- The system should provide a web page where users can browse articles.
- The system should fetch article data from a database.

- The system should provide a search functionality to search for articles.
- The system should provide a filter functionality to filter articles by category, date, or popularity.

3.3.2.3 The ability to watch a video showing the correct performance of exercise:

- The system should provide a web page where users can watch exercise videos.
- The system should fetch exercise video data from a database.
- The system should provide a search functionality to search for exercise videos.
- The system should provide a filter functionality to filter exercise videos by category, difficulty level, or popularity.

3.3.2.4 The ability to upload sports and health articles through the admin only:

- The system should provide an admin dashboard to manage articles.
- The system should allow the admin to upload new articles, edit existing ones, and delete articles.
- The system should store article data in a database.
- The system should allow the admin to review and approve user-submitted articles before publishing them.

3.4 Nonfunctional requirements

3.4.1 Performance requirements

- The application should optimize the video speed to minimize its size and improve performance.
- To minimize memory consumption, the application should not save the videos uploaded by the users.
- The application should save the final results of exercises for future reference.
- The application should provide an easy way to track the user's weight and calories burned, to meet the various needs of users.
- The application should be easy to use and provide a speedy response to ensure a better user experience.

3.4.2 Security requirement

- Implement secure user authentication and authorization to ensure that only authorized users can access and use the app.
- Use encryption to protect user data, such as login credentials and video analysis results, both in transit and at rest.
- Implement measures to prevent unauthorized access, modification, or deletion of user data and app resources.
- Conduct regular security testing and updates to identify and mitigate potential vulnerabilities.

3.4.3 Logs requirement

- Implement logging of all user actions, including account creation, login, article browsing, and video uploads, and analysis results.
- Store logs securely and ensure they are tamper-evident to detect and prevent any unauthorized modifications.
- Use logs for troubleshooting, performance monitoring, and security auditing purposes.
- Define a retention policy for logs to balance the need for data retention with data privacy and storage constraints.

3.4.4 Errors exception and handling requirement

- Validating user inputs during registration and login to ensure the required information is entered and in the correct format.
- Handling errors related to video uploading, such as loss of internet connection or unstable connection.
- Verifying that the uploaded video is the correct one before processing to avoid errors in training results.
- Handling errors caused by incorrect user position or camera angles during video recording to ensure accurate results.
- Keeping a log of all errors that occur within the system for future reference.
- Assigning the responsibility of handling errors to the admin to minimize user impact.
- Displaying error messages that explain the nature of the error and provide guidance on how to resolve it.

3.4.5 Safety Requirements

- Ensure the application is compliant with all safety regulations and standards for health and fitness applications.
- Provide clear instructions and warnings to users about any potential risks or hazards associated with the exercises or workouts featured in the application.
- Ensure that all exercises and workouts are appropriate for the user's level of fitness and health status.
- Include safety guidelines for users to follow while performing the exercises, such as proper form and technique, warm-up and cool-down procedures, and rest periods.
- Provide safety features to prevent accidents, such as setting time limits for workouts and detecting signs of fatigue or overexertion.
- Include emergency protocols and contact information for medical professionals in case of an injury or medical emergency.
- Regularly update and maintain the application to address any safety issues or concerns that may arise.

3.4.6 Software Quality Attributes

- **Code quality:** The code should be well-structured and follow best practices to ensure that it is easy to maintain and update. Additionally, the code should be designed to be reusable to avoid duplication of effort and reduce the time needed for future updates.
- **Bug-free:** The application should undergo thorough testing to ensure that it is free of bugs and errors. This will ensure that the application is stable and reliable for users.

- **Security:** The application should be designed with a high degree of security to ensure that user data is protected. This includes using encryption, secure login processes, and protection against potential vulnerabilities.
- **Speed:** The application should be designed for efficient performance, with fast response times for common user actions such as logging in and browsing content. The application should also be optimized to minimize memory and bandwidth usage to ensure fast uploading and viewing of videos.
- **User experience:** The application should be designed with the user in mind, with a clear and intuitive interface that is easy to use. The application should also be designed to be accessible to users with disabilities, to ensure that everyone can use it.

3.4.7 Business Rules

The business rules that should be considered in the system requirements are:

- Only registered users are allowed to upload articles or videos.
- Only the admin is allowed to upload articles on sports and health.
- Users can only upload videos of their own performance during exercise.
- The uploaded videos should be of reasonable length and size.
- The system should provide accurate results for video analysis.
- Users should not be able to modify or delete their results after they have been saved.
- Users should be able to access their saved results at any time.
- The system should have a mechanism to prevent unauthorized access to user data.

- The system should comply with relevant data protection laws and regulations.
- The system should have a mechanism to prevent spam or irrelevant content from being uploaded.
- The system should have a mechanism to monitor and remove inappropriate content.
- Users should be able to provide feedback on the system and report any issues they encounter.

Chapter 4

Analysis and Design

Development Approach

4.1 Scrum Development Approach

Based on the requirements we suggested using an agile development approach, specifically Scrum.

4.1.1 Introduction

Scrum is an iterative and incremental framework for software development that emphasizes collaboration, flexibility, and customer satisfaction. It involves breaking down the project into smaller, more manageable chunks called "sprints," which typically last two to four weeks.

During each sprint, a cross-functional team works together to deliver a working increment of the product. At the end of the sprint, the team reviews the work they have done, and then plans the next sprint based on feedback from the customer and any changes in priorities.

4.1.2 Why Incremental Development?

- Scrum is a flexible, adaptable, and collaborative project management framework.
- Scrum involves breaking down projects into small, manageable tasks completed in short iterations.
- Scrum allows for quick feedback, continuous improvement, and the ability to adapt to changes.
- Scrum teams include a Scrum Master, Product Owner, and team of developers.

- Scrum includes sprint planning, daily Scrum, and sprint review and retrospective meetings.
- Scrum provides a structured approach for managing complex projects.
- Scrum may not be appropriate for all projects and should be evaluated based on project requirements and stakeholder expectations.

4.1.3 Scrum Approach Problems and Solutions

- **Problem:** Lack of clarity in project goals or backlog items.
- ✓ **Solution:** we prioritized the backlog items and provided clear definitions of the project goals as a team.
- **Problem:** Difficulty in estimating the time required for completing backlog items.
- ✓ **Solution:** we used historical information to estimate the time required for completing backlog items.
- **Problem:** Poor communication or collaboration between team members.
- ✓ **Solution:** we holded daily Scrum meetings to keep everyone aligned, and facilitate communication and collaboration between team members.
- **Problem:** Inability to adapt to changes or unexpected events.
- ✓ **Solution:** The Scrum framework is designed to allow for quick feedback and continuous improvement, which should enable the team to adapt to changes or unexpected events as they arise.

4.2 Functional decomposition Diagram (FFD)

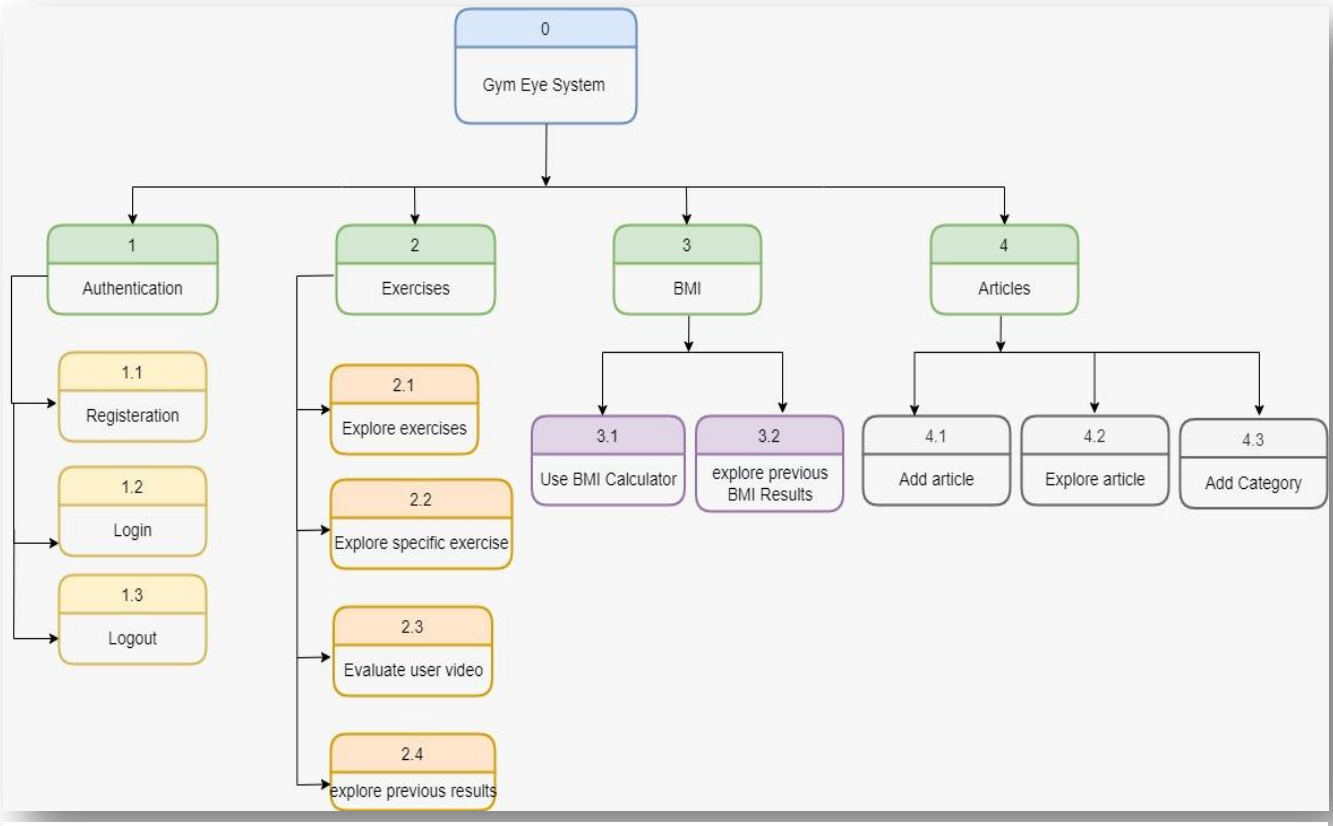


Figure 4.1- FFD diagram

4.3 Data flow Diagram (DFD)

- **Context diagram**

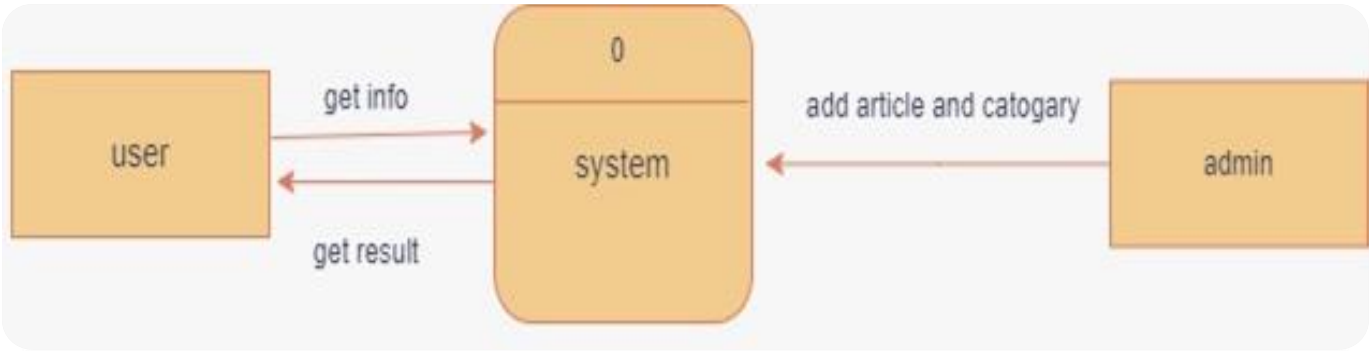


Figure 4.2-1 DFD Context diagram

- **Level 0 diagram**

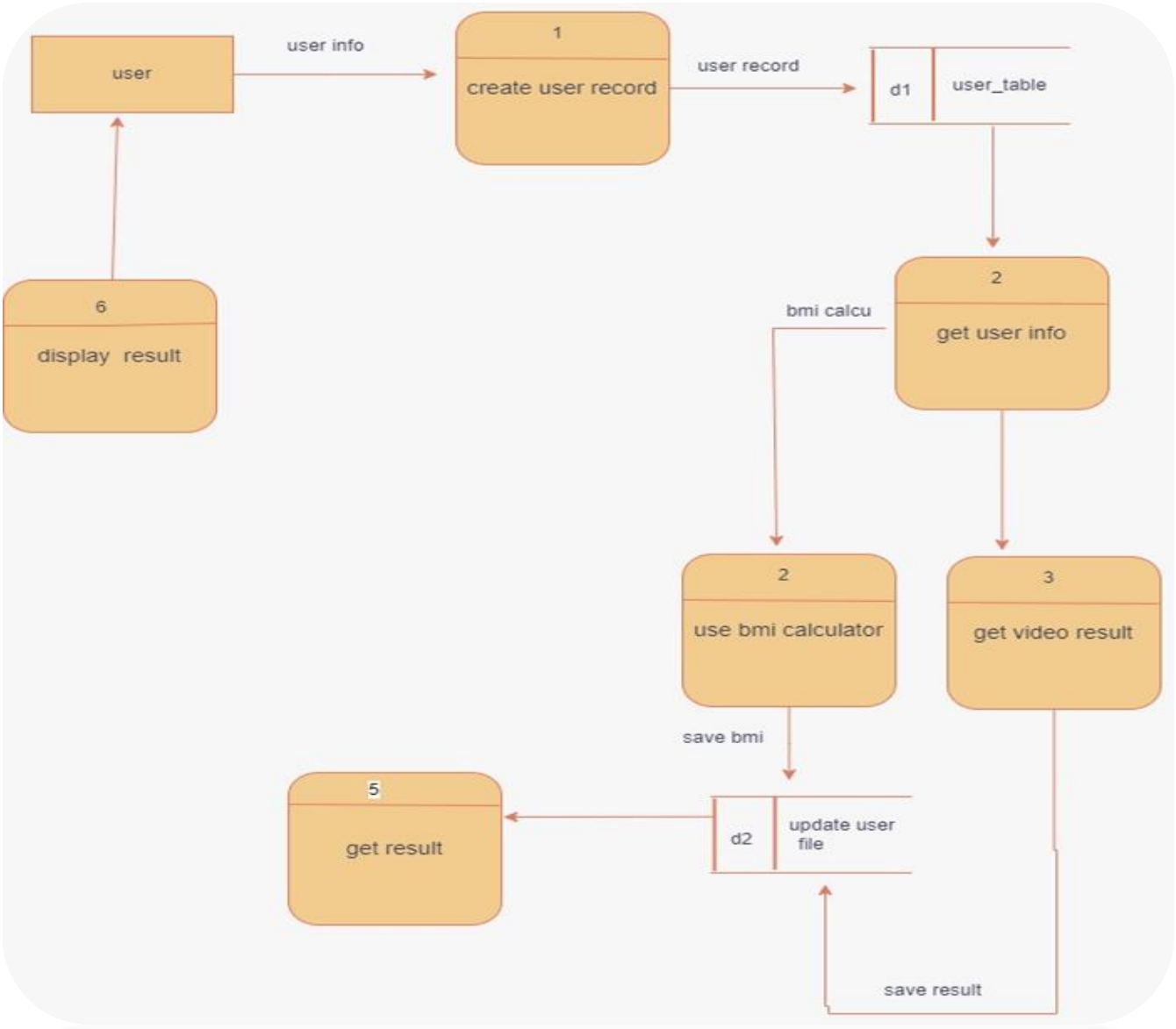


Figure 4.2-2 DFD level 0 diagram

4.4 UML Diagrams

UML is a way of visualizing a software program using a collection of diagrams. The current UML standards call for different types of diagrams; these diagrams are organized into two distinct groups:

4.4.1 Behavioral UML diagrams

- **Entity relationship Diagram (ERD)**

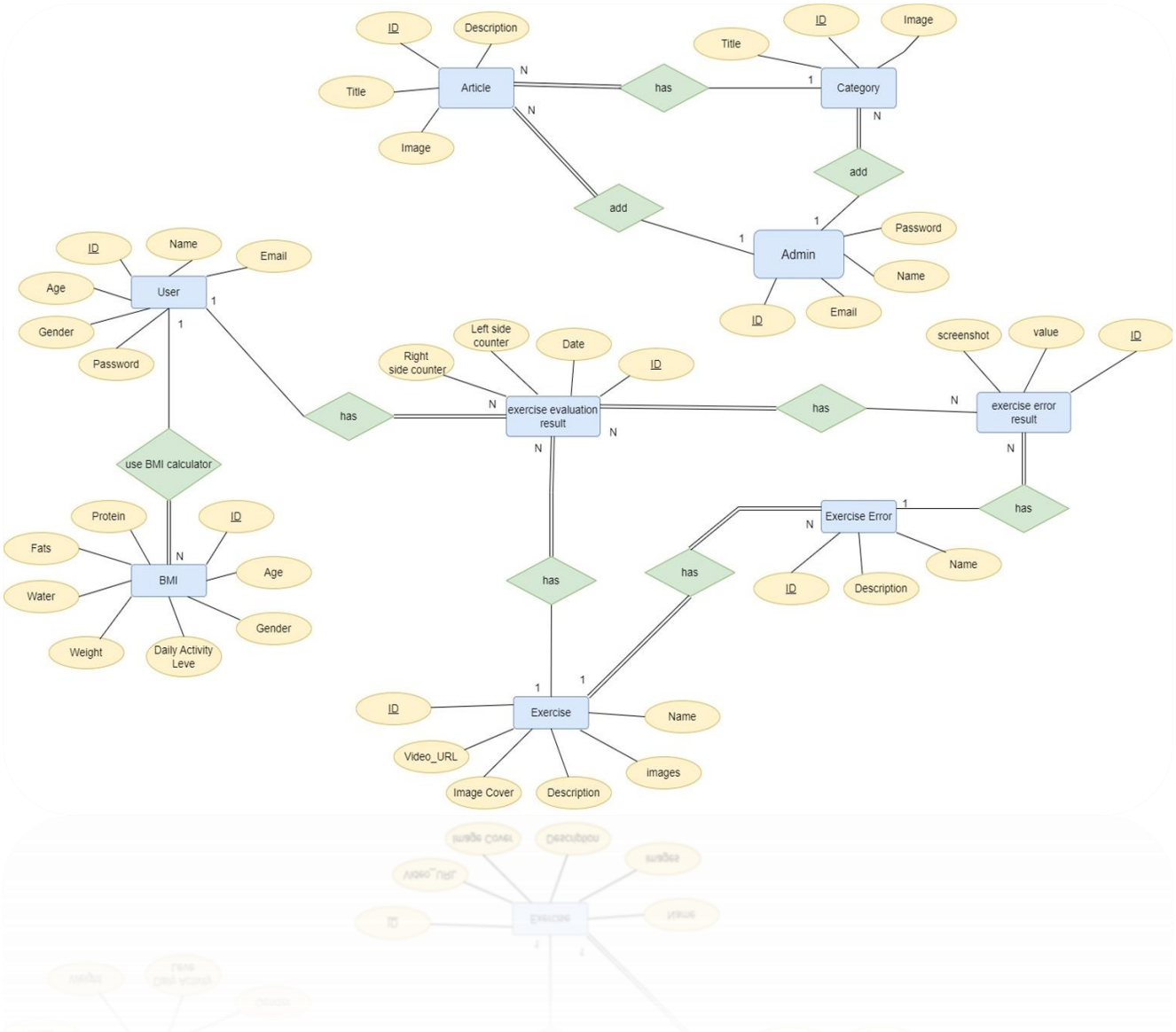


Figure 4.3- Entity relationship diagram

- **Schema Diagram**

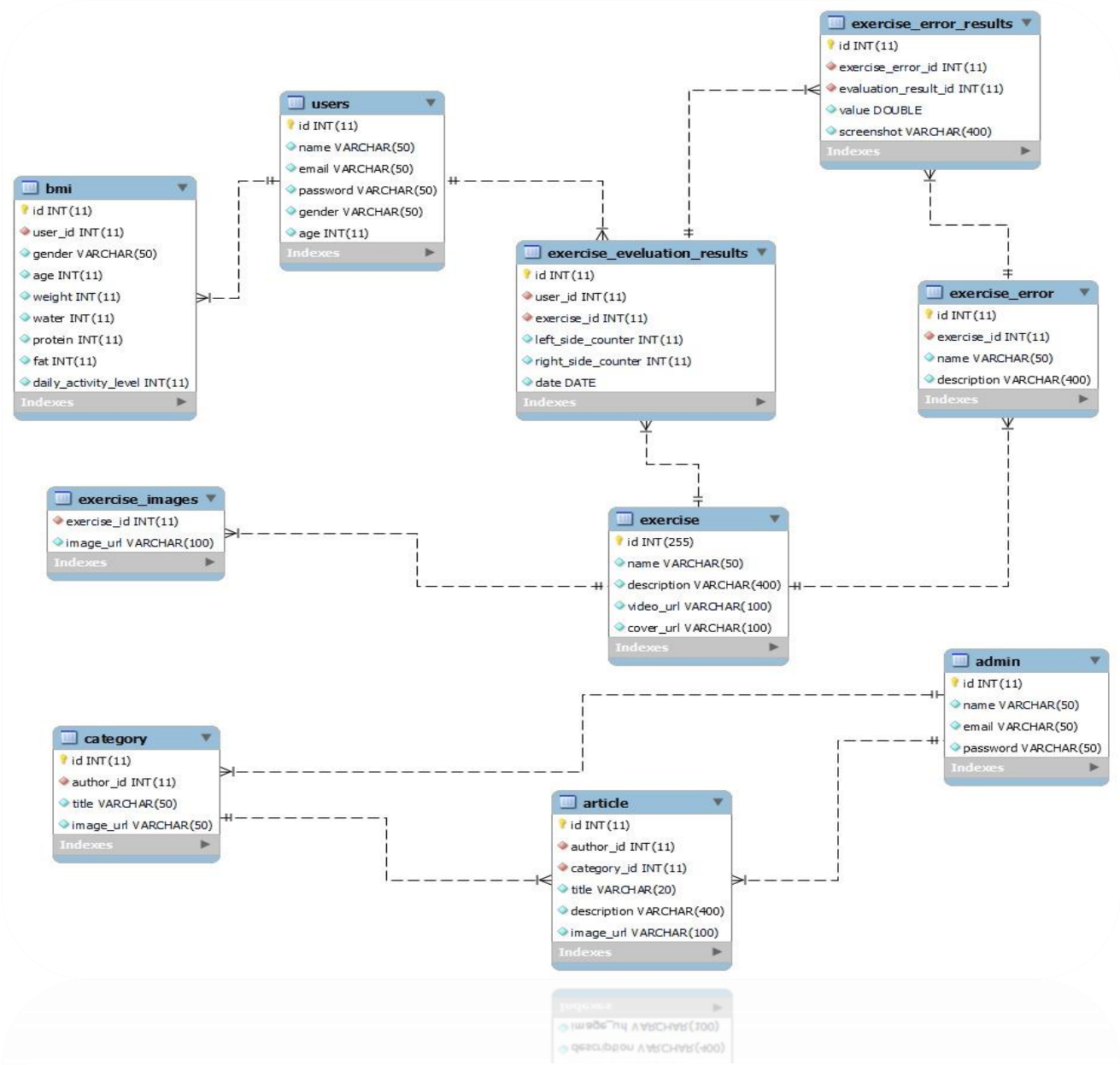


Figure 4.4- Schema diagram

- **Activity Diagram**

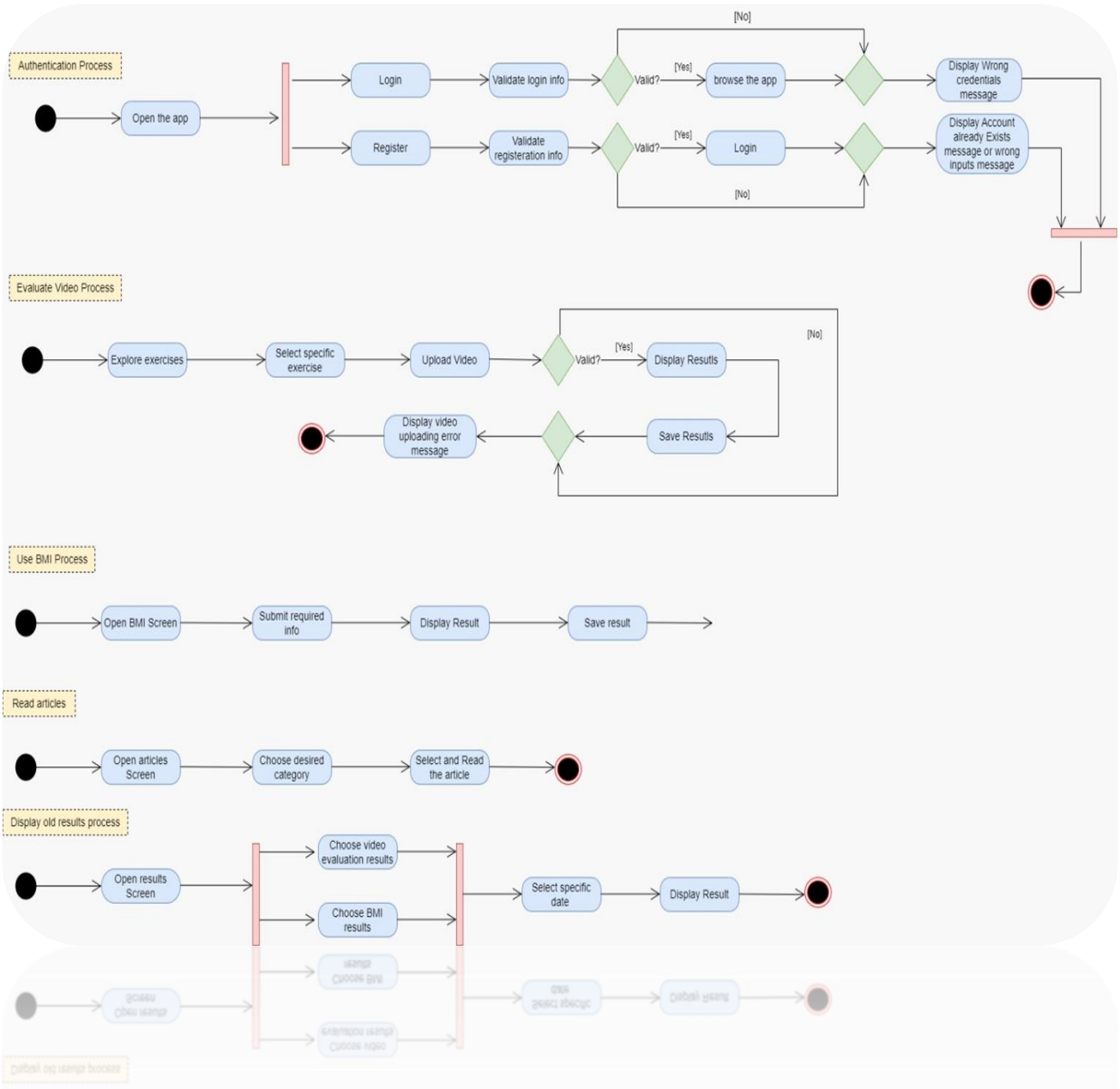


Figure 4.5- Activity diagram

- **Sinariou Diagram**

  - ✓ **Register Scenario:**

Use Case Name: register	**ID: 01**	**Priority: High**
Actor: user		
Description: The ability of the user to create an account on the system		
Trigger: The user decides to start caring for his body and health. **Type:** ☑ External ☐ Temporal		
Preconditions: 1. User information must be verified.		
Steps: 1. The user browses the main page. 2. The user click on register. 3. The user fill registration form. 4. The user click on submit to create his account.		
Post conditions: - The user now will be able to login using his email and password. - The user now will be able to use all system features.		
Exceptions: - User's personal information might be incorrect.		
Assumptions: - User has successfully created an account and managed to enter correct personal information.		

Table 4.1- Register scenario

✓ **Login Scenario:**

Use Case Name: login	**ID:** 02	**Priority:** High
Actor: user		
Description: The ability of the user to login on the system using his account		
Trigger: The user decides to use specific feature in the system. **Type:** ☑ External ☐ Temporal		
Preconditions: 1. User information must have an account on the system.		
Steps: 1. The user browse the main page. 2. The user click on login. 3. The user fill his credentials. 4. The user click on login button.		
Post conditions: - The user now will be able to use desired system features.		
Exceptions: - User's credentials might be incorrect. - User might not have an account on the system.		
Assumptions: - User has successfully created logged into the systems.		

Table 4.2- Login scenario

✓ **Evaluate exercise Scenario:**

Use Case Name: Evaluate exercise	**ID:** 03	**Priority:** High
Actor: user		
Description: Allow the trainee to test a video of him performing a specific exercise and see if his performance is correct.		
Trigger: When the trainee wants to test a video in which he performs an exercise to see how correct it is. **Type:** ☑ External ☐ Temporal		
Preconditions: 1. The user is logged in. 2. The video should not be longer than 30 seconds.		
Steps: 1. The user selects one of the exercises. 2. The user watches a video and illustrations of how to perform the exercise correctly. 3. The user uploads the video.		
Post conditions: - The user will be able to view a report on this video and know the correctness of his performance of the exercise. - The user can view these reports at a later time.		
Exceptions: - The user can upload a video in which he performs a different exercise than the one he chose.		
Assumptions: - The user has now successfully uploaded the video and received a report on his exercise.		

Table 4.3- Evaluate exercise scenario

✓ **Use BMI Scenario:**

Use Case Name: Use BMI	**ID: 04**	**Priority: Medium**
Actor: trainee		
Description: Allowing the user to calculate his body's daily needs of appropriate nutrients.		
Trigger: When the trainee wants to calculate his body's daily needs of appropriate nutrients to reach a specific goal. **Type:** ☑ External ☐ Temporal		
Preconditions: 1. The user is logged in. 2. User information must be verified.		
Steps: 1. The user fills in the required information.		
Post conditions: - After that, the user receives a report showing the body's daily needs for nutrients. - The user can view these reports at a later time.		
Exceptions: - The user may enter incorrect information, which will result in incorrect results.		
Assumptions: - The user has successfully received a report showing the body's daily needs for nutrients.		

Table 4.4- Use BMI scenario

✓ **Show Evaluation or BMI results Scenario:**

Use Case Name: Show evaluation or BMI results **ID: 05**	**Priority: Medium**
Actor: user	
Description: Allowing the user to browse his previous reports	
Trigger: When the trainee wants to browse his previous reports and compare between them to evaluate his progress. **Type:** ☑ External ☐ Temporal	
Preconditions: **1.** The user is logged in. **2.** The user has already tested one of the videos before or calculated his body's needs for nutrients	
Steps: 1. Go to the reports screen. 2. Choose the type of reports.	
Post conditions: - Old reports have been restored.	
Exceptions: - User can try to restore reports without having previous reports.	
Assumptions: - The user has successfully received all previous reports.	

Table 4.5- Show evaluation or BMI results scenario

✓ **Admin Adds category Scenario:**

Use Case Name: Admin adds category	**ID: 06**	**Priority: Low**
Actor: admin		
Description: Admin adds a category for articles		
Trigger: When admin wants to add new category for articles in the system **Type:** ☑ External ☐ Temporal		
Preconditions: 1. The admin must be logged in.		
Steps: 1. The admin goes to articles screen. 2. The admin adds a new category.		
Post conditions: - Admin can now add new articles to added category.		
Exceptions: - Admin has written existed category title.		
Assumptions: - The admin has successfully added a new category.		

Table 4.6- Admin adds category scenario

- **Use case Diagram**

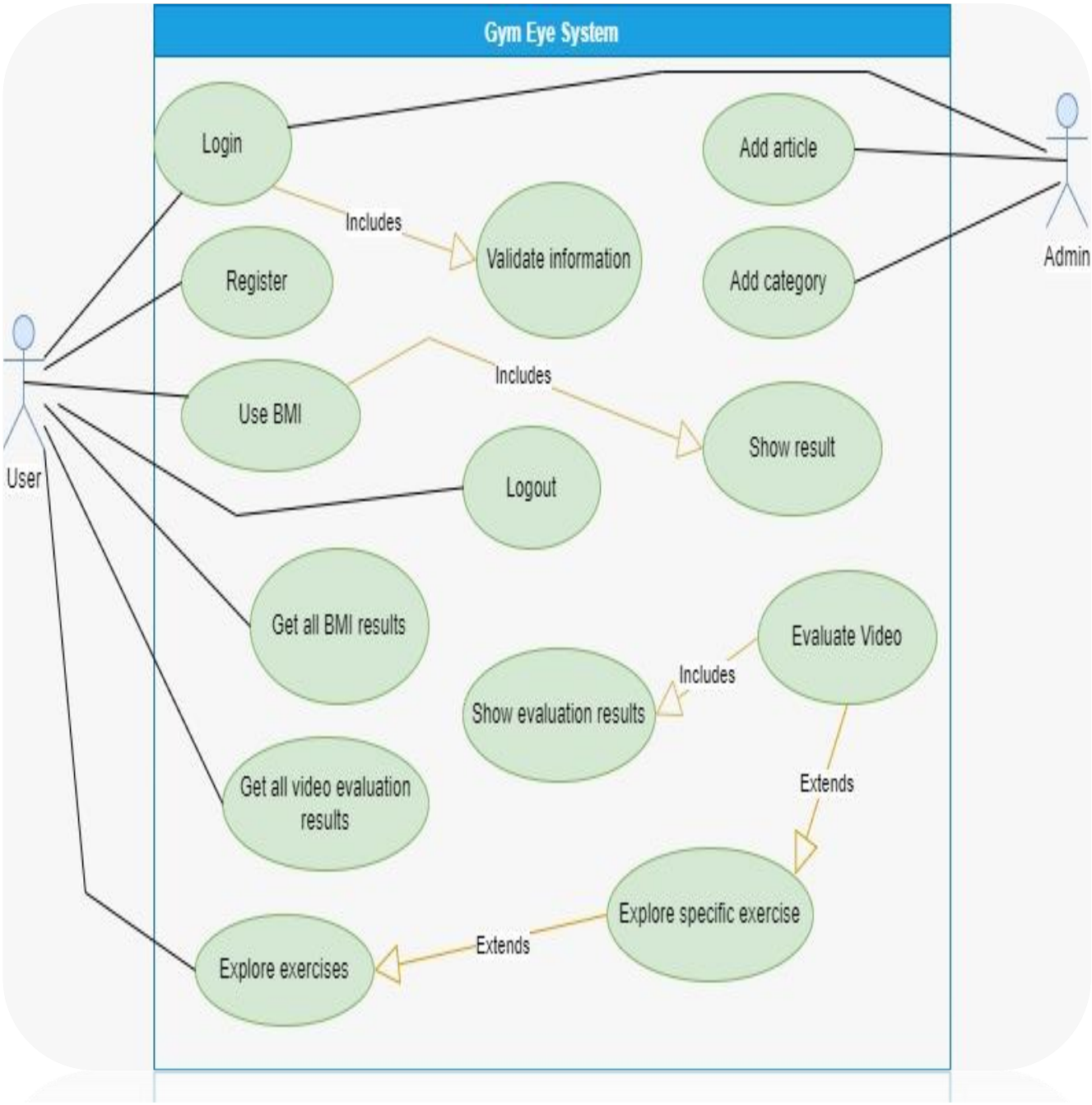


Figure 4.6- Use case diagram

- **Sequence Diagram**

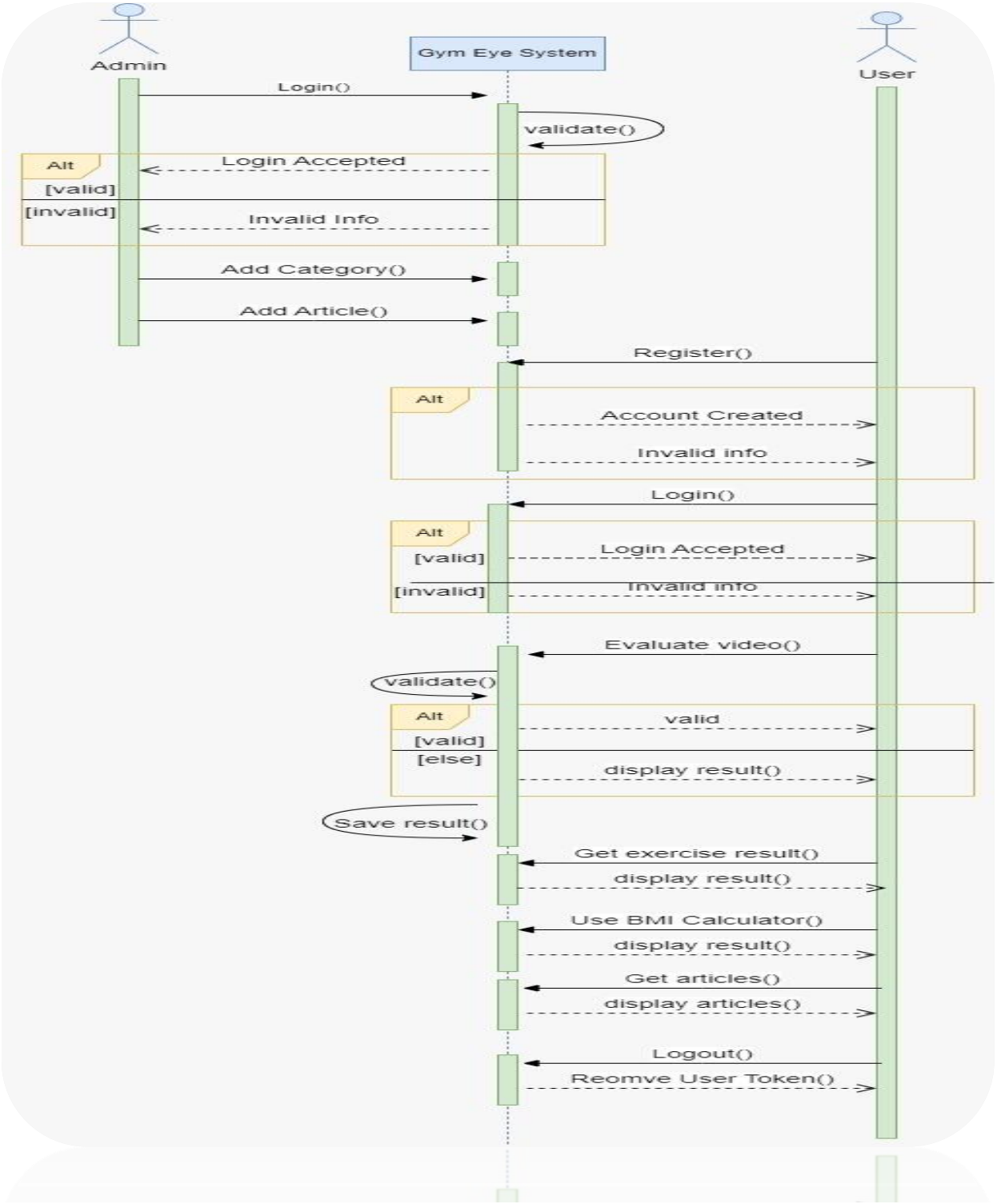


Figure 4.7- Sequence diagram

4.4.2 Structural UML diagrams

- **Class Diagram**

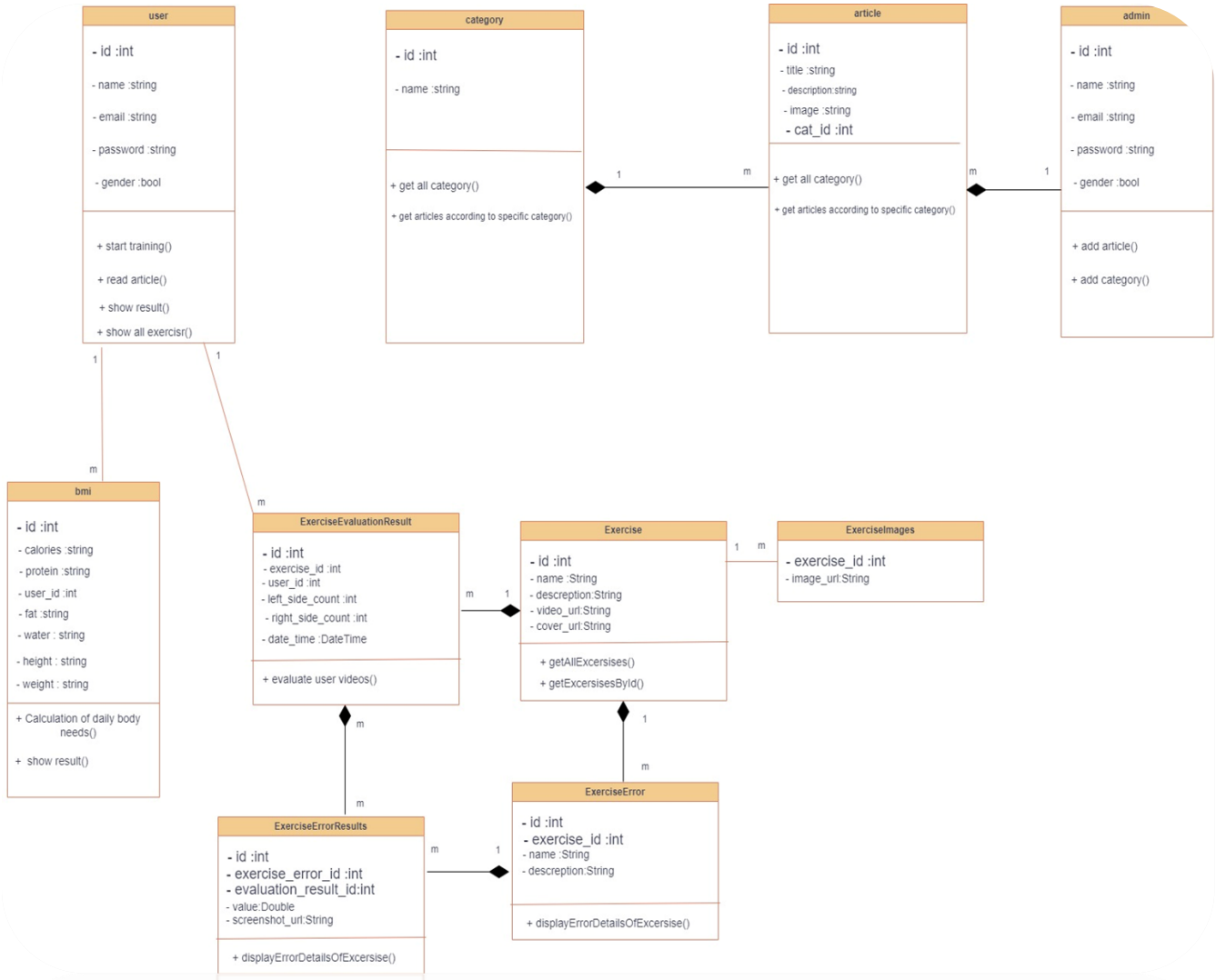


Figure 4.8- Class diagram

4.5 Flow Chart

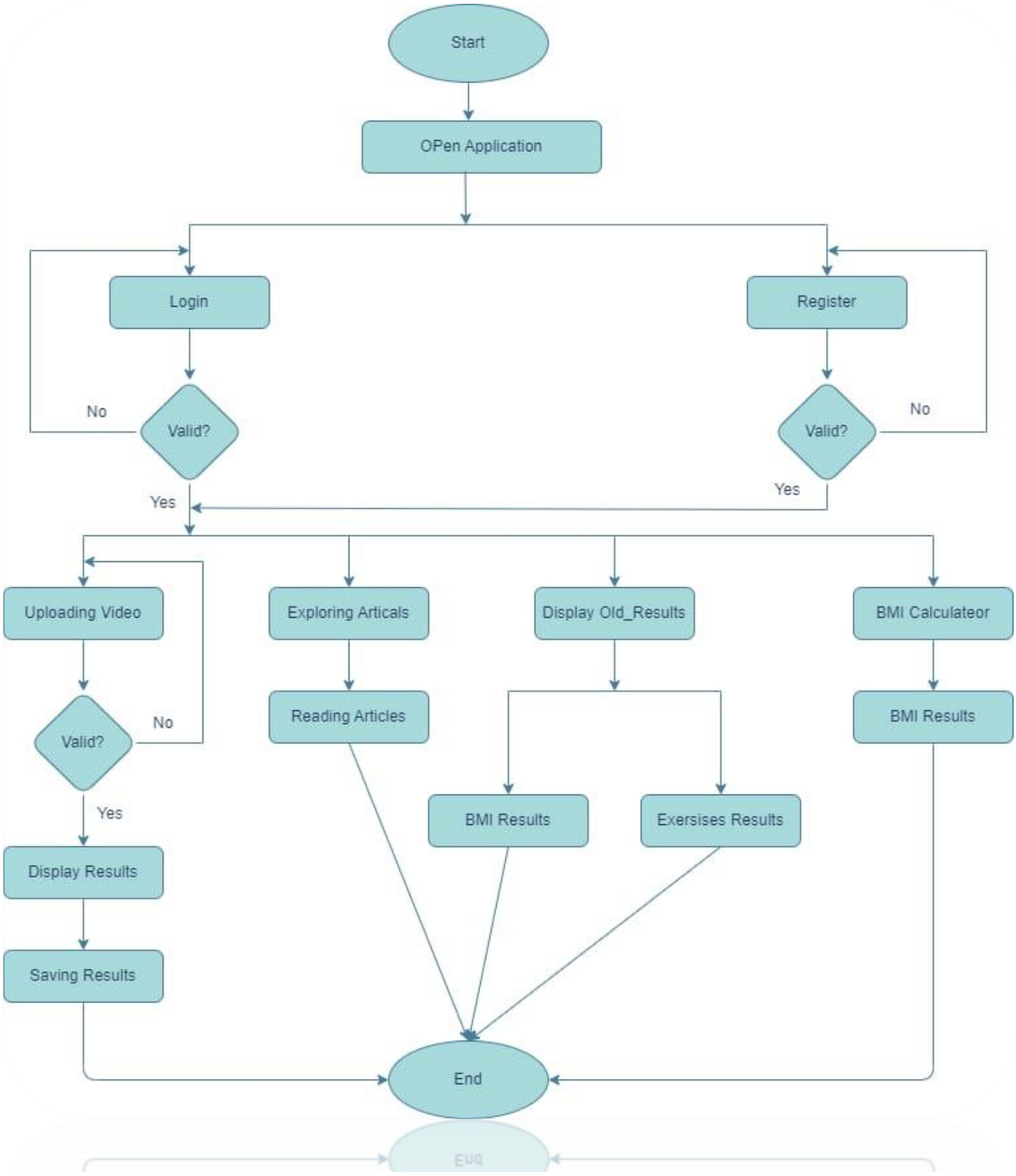


Figure 4.9- Flow chart diagram

4.6 Software application design

4.6.1 UI in mobile application

- **Resigester screen**

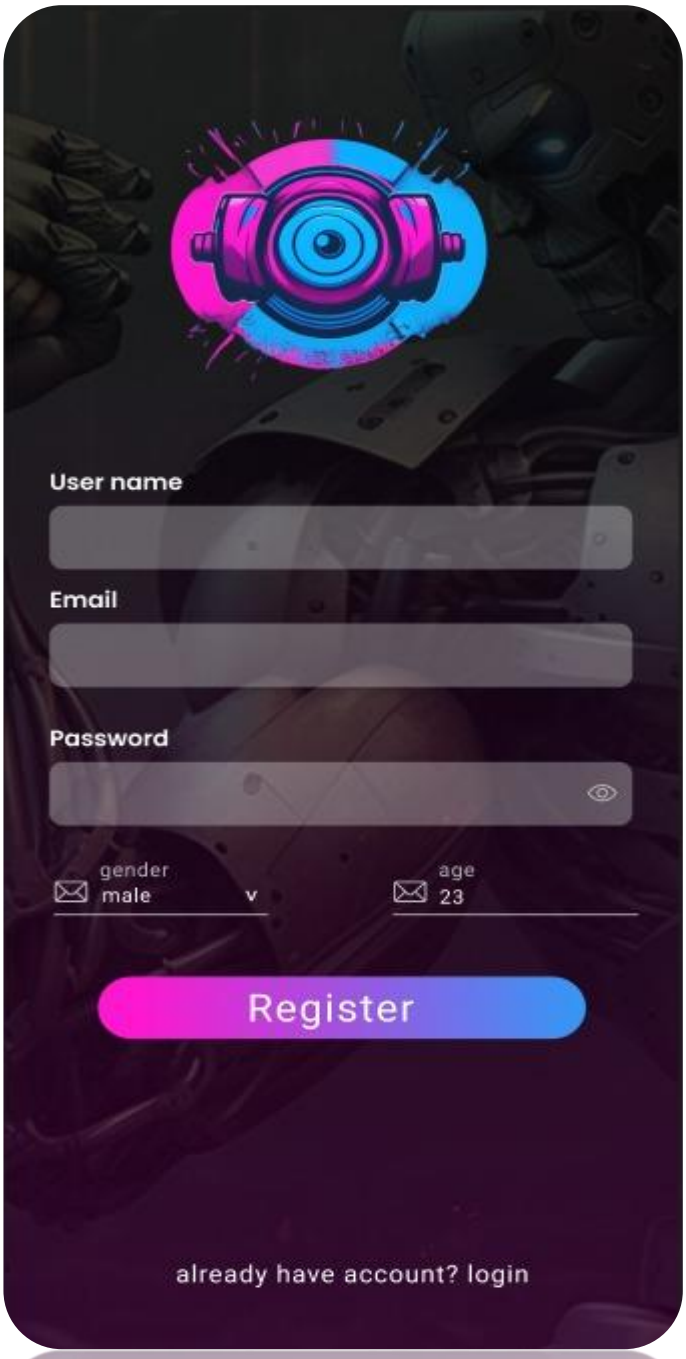


Figure 4.10—Resister screen

- **Login screen**

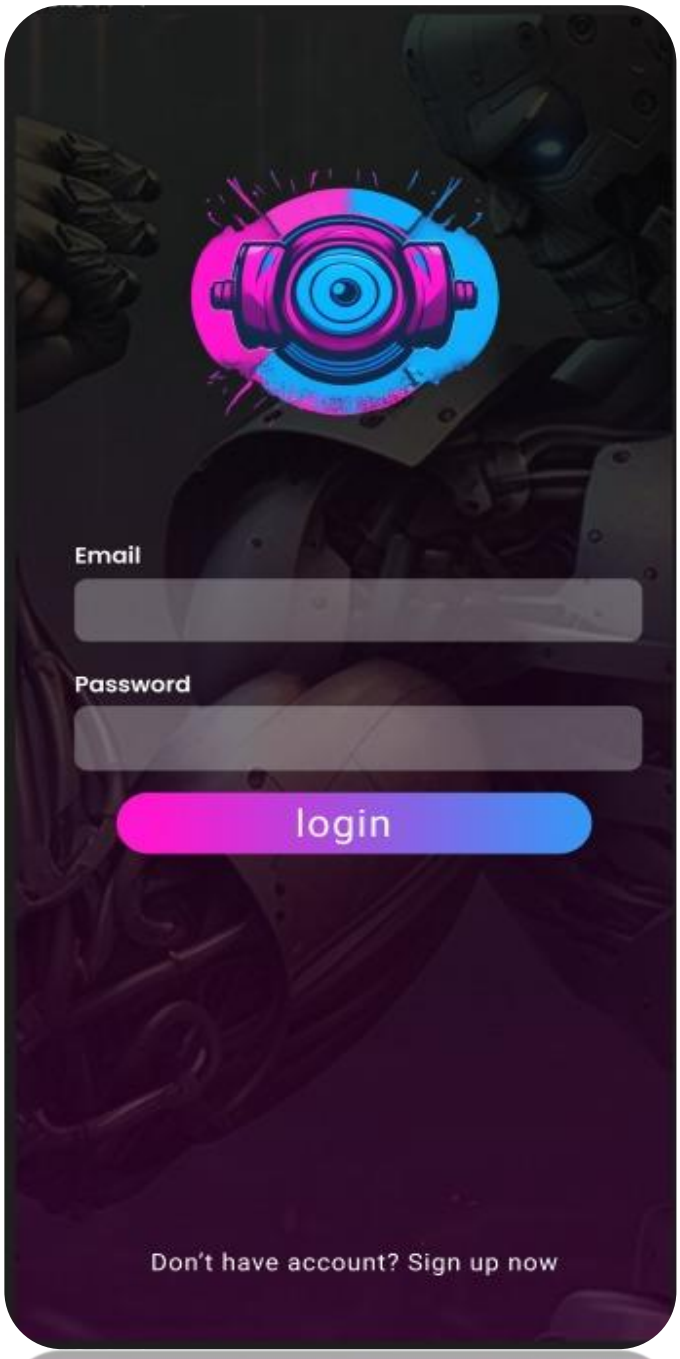


Figure 4.11- Login screen

- **Home screen**

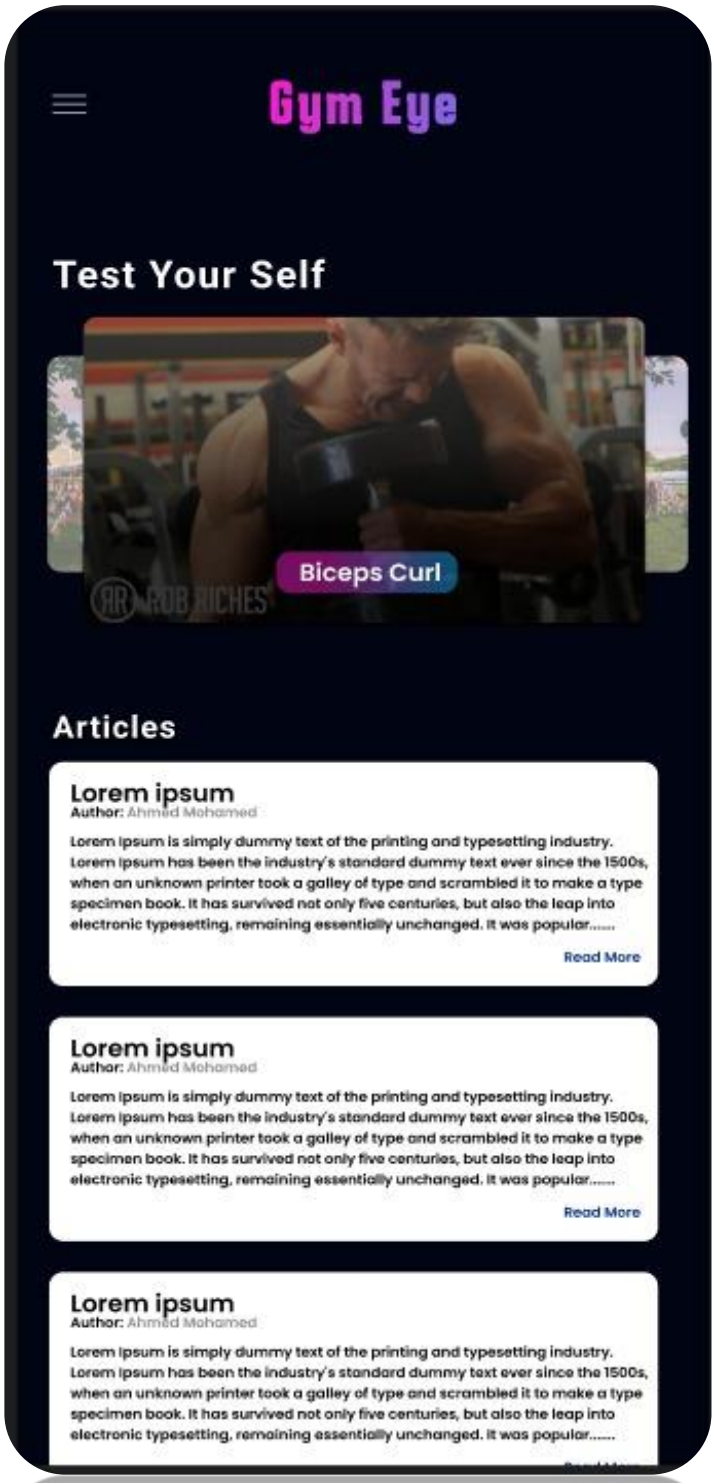


Figure 4.12- Home screen

- **Home menu screen**

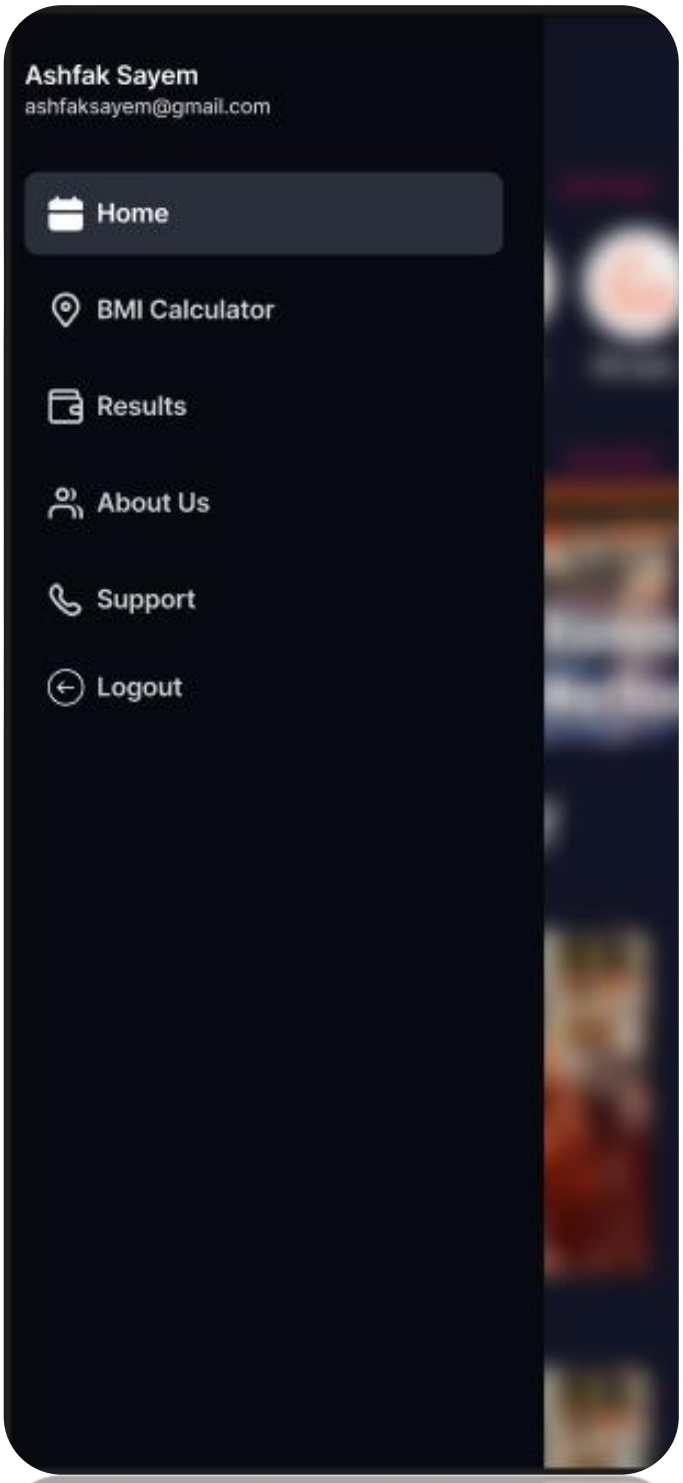


Figure 4.13- Home menu screen

- **BMI Calculator screen**

Figure 4.14- BMI Calculator screen

- **BMI report screen**

Figure 4.15- BMI report screen

- **BMI Results screen**

Figure 4.16- BMI Results screen

- **Exercises Results screen**

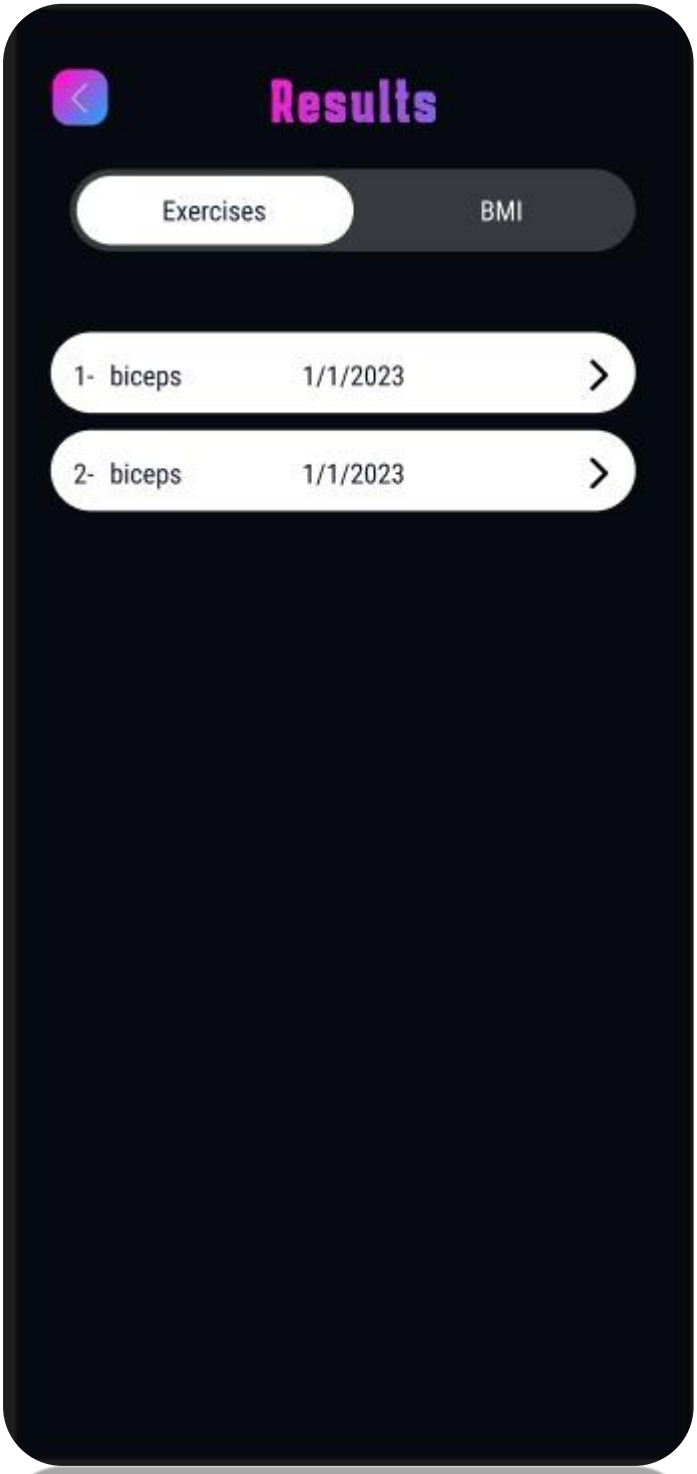


Figure 4.17- Exercises Results screen

- **Report Details screen**

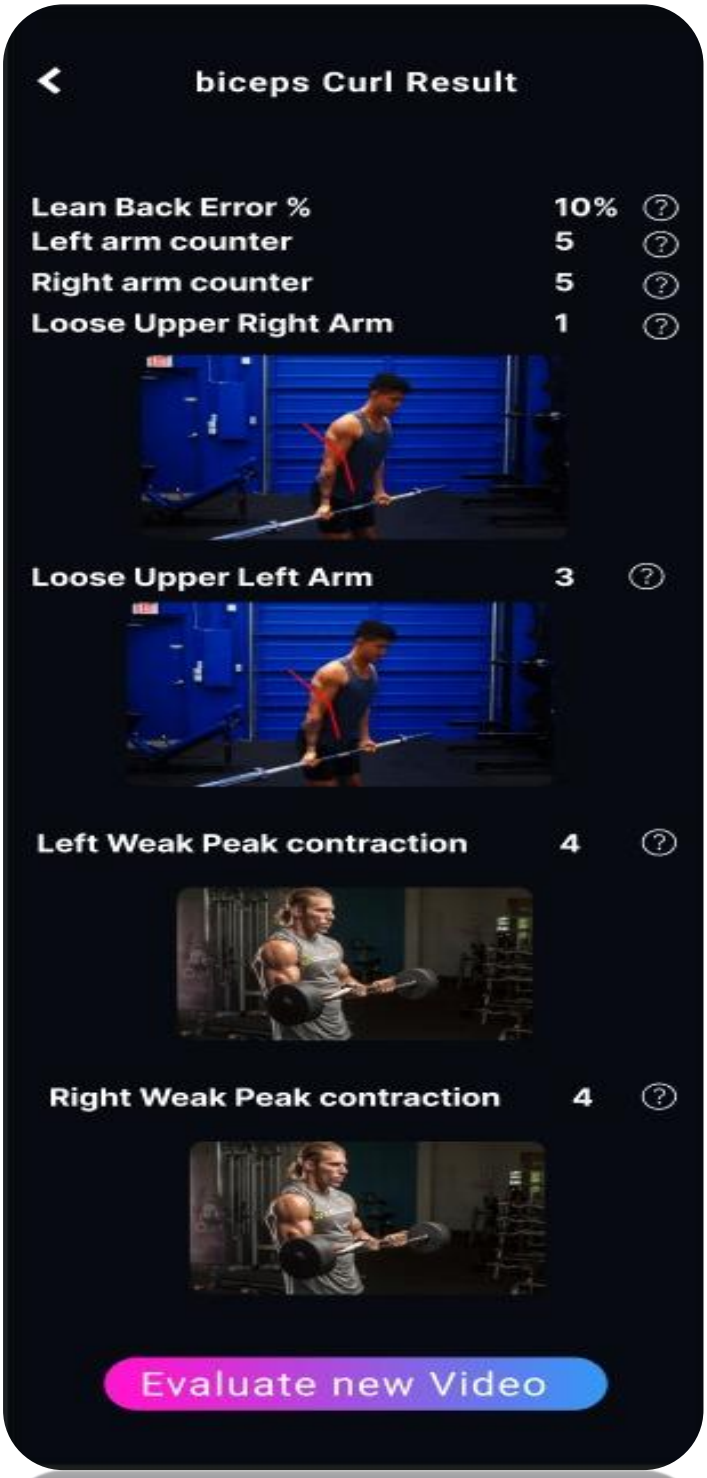


Figure 4.18- Report details screen

- **Training Details screen**

Figure 4.19- Training details screen

- **Article screen**

Figure 4.20- Article screen

4.7 External Interface Requirements

4.7.1 User Interfaces

- The app should have a clean and intuitive user interface that is easy to navigate.
- Users should be able to easily record and upload video content of them performing exercises.
- The app should provide clear and concise feedback to users on the correctness of their exercise technique.

4.7.2 Hardware Interfaces

- The app will require access to the user's device camera and microphone to record video content and audio feedback.
- The app will require an internet connection to upload and analyze video content.

4.7.3 Software Interface.

- The app's algorithms for analyzing video content must be integrated with the mobile app's software architecture.
- The app should be able to access and use external data sources to provide personalized nutrition recommendations.

4.7.4 Communications Interfaces

- The app will require a stable and reliable internet connection to communicate with external servers for video analysis and data retrieval.
- The app should provide users with clear and concise notifications and alerts related to their exercise technique and nutrition recommendations.

Chapter 5

Implementation

Machine learning models

5.1 Squat Exercise model

There are two models that detect two errors in squat performance. Every model for one error. The errors are knees forward and knees inward.

Knees Forward: When performing a squat, it is important to maintain proper alignment of the knees. The knees should be in line with the toes and should not move too far forward beyond the toes. However, the knees forward error occurs when the knees move too far forward beyond the toes during the squat. This can place excessive stress on the knees and can lead to injury over time. Here is a picture illustrating the error:

Figure 5.1- Knees Forward Error

Knees Inward: This occurs when the knees collapse inward towards each other during the squat, instead of staying in line with the toes. This can place excessive stress on the knees and can also lead to injury over time. Here is a picture of the error:

Figure 5.2- Knees Inward Error

Here is the correct form for this exercise:

Figure 5.3- Squat Exercise

5.2 Biceps Exercise model

There are three models that detect three errors in biceps performance. The errors are lean back, loose upper arm, weak peak contraction error.

Lean Back: the performer's torso leans back and fore during the exercise for momentum. Here there is a focus on back movement generally.

Loose Upper Arm: when an arm moves upward during the exercise, the upper arm is moving instead of staying still. As here there is a focus on back movement.

Figure 5.4- Lean Back Error

Weak Peak Contraction: when an arm moves upward, it does not go high enough therefore not put enough contraction to the bicep. This is focus on the arm movement.

The lean back error is detected using the deep learning model while the two other errors are detected using some calculations as following:

Figure 5.5- Loose Upper Arm Error

Loose upper arm: Can be detected by calculating the angle between the elbow, shoulder and the shoulder's projection on the ground. If the angle is over 40 degrees, the movement will be classified "loose upper arm" error.

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

Figure 5.6- Weak Peak Contraction Error

Here is the correct form biceps exercise:

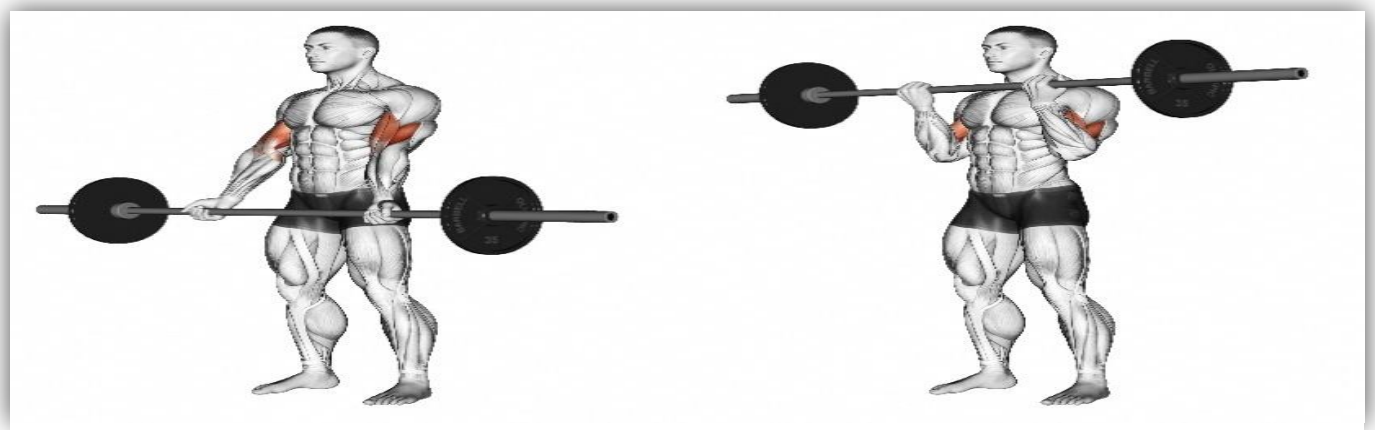


Figure 5.7- Biceps Exercise

5.3 Barbell Row Exercise model

Here, there are two errors that are detected by two models. The errors are lumbar and torso error.

Lumbar error: Lumbar error occurs when the lower back (lumbar spine) is rounded or arched excessively during the barbell row exercise. This can place excessive stress on the lower back and increase the risk of injury.

Figure 5.8- Lumbar Error

Torso Error: Torso error occurs when the torso is not kept stable and straight during the barbell row exercise. This can cause the body to sway or twist, increasing the risk of injury to the lower back and other muscles.

Figure 5.9- Torso Error

Here is the correct form in this exercise:

Figure 5.10- Barbell Row Exercise

5.4 System Architecture

The train data are preprocessed and then are fed to learning the model in order to predict the labels or detect the defects found in the image

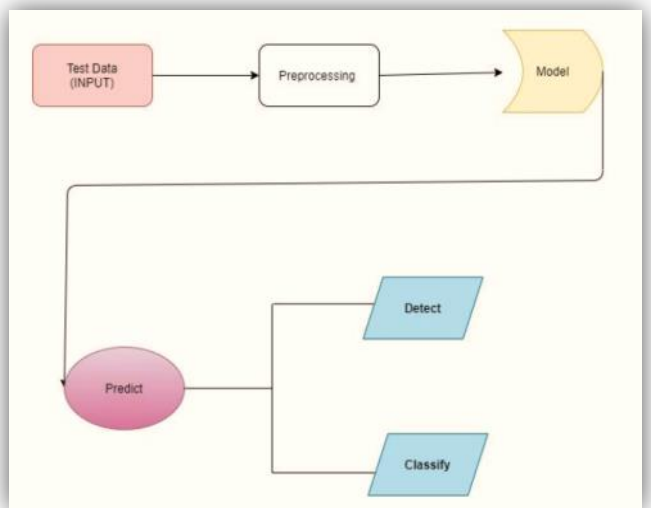


Figure 5.11- System Architecture

Input: Video or an image or multiple images.

Output: a label in case of classification or an image in case of detection.

Preprocessing step: is done to make the images "Frames" quality better by using one of these techniques:

- Sharpening the images.
- Using histogram equalization.
- Using one or more.
- Using adaptive histogram equalization.

5.5 Architecture of models used:

We used different types of models in this project for training data and classification or prediction is a feedforward neural network built using the Keras Sequential API.

5.5.1 Feedforward neural network

This is a type of artificial neural network in which the information flows in only one direction, from the input layer to the output layer, without any loops or feedback connections. It is also known as a multilayer perceptron (MLP) because it consists of multiple layers of perceptron's or artificial neurons.

5.5.2 Structure of FNN

In a feedforward neural network, each layer receives input from the previous layer and passes its output to the next layer until the final output is produced by the output layer. The input layer receives the input data, which is transformed by the hidden layers through a series of nonlinear transformations until the output is generated by the output layer.

Each neuron in a feedforward neural network performs a weighted sum of the inputs and applies an activation function to the result to produce its output. The weights and biases of the neurons are learned during the training process using an optimization algorithm, such as stochastic gradient descent, to minimize the error between the predicted output and the true output.

5.5.3 Purpose of FNN

Feedforward neural networks can be used for a variety of tasks, including classification, regression, and pattern recognition. They have been used successfully in many applications, such as image and speech recognition, natural language processing, and financial forecasting.

5.5.4 Cons of FNN

One of the key advantages of feedforward neural networks are:

- Their ability to model complex nonlinear relationships between inputs and outputs.
- They can be prone to over fitting if the number of hidden layers or neurons is too large or if the training data is insufficient.
- Regularization techniques, such as dropout and weight decay, can be used to prevent over fitting.

5.5.5 The general architecture of the neural network:

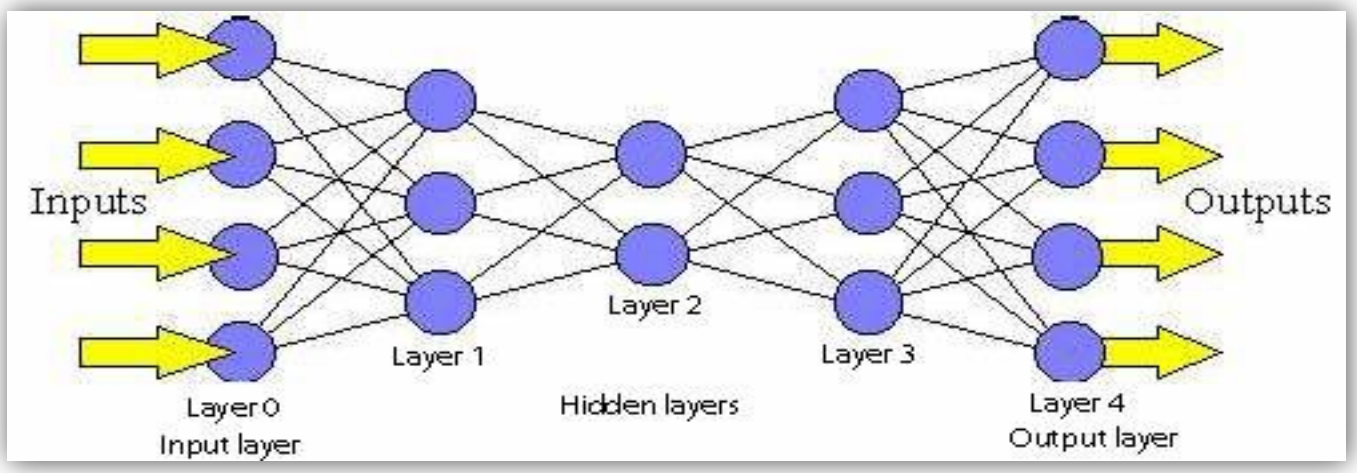


Figure 5.12- Architecture of neural network

5.6 Keras Sequential

This is a high-level API for building neural network models in Keras. It allows you to easily create a linear stack of layers, where the output of one layer is fed as the input to the next layer. This makes it easy to define and train a wide range of neural network architectures, from simple feedforward networks to more complex models with multiple inputs and outputs.

We have used four neural networks architectures in the training phase to detect the best architecture to be used in prediction stage in each exercise. The four architectures are as following:

- Three layers model.
- Five layers model
- Seven layers model
- Seven layers model with dropout layers

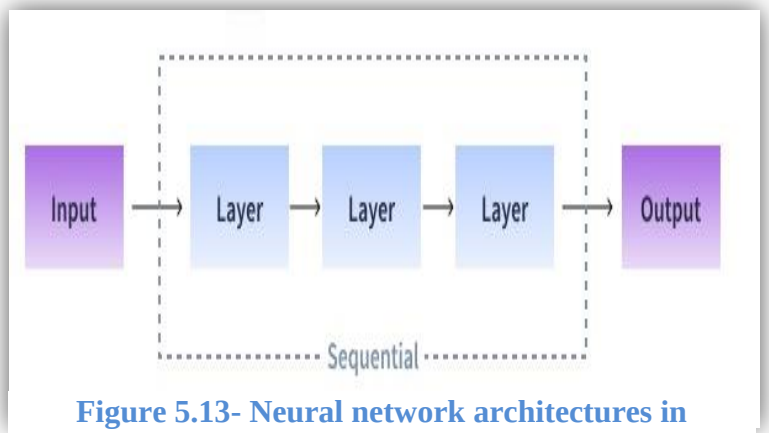


Figure 5.13- Neural network architectures in prediction stage

5.7 Dropout

This is a regularization technique used in neural networks to prevent over fitting, which occurs when a model becomes too complex and starts to memorize the training data rather than generalizing to new data (making prediction to new data). Dropout works by randomly dropping out some of the units or neurons in a layer during training, which forces the remaining units to learn more robust and generalizable features.

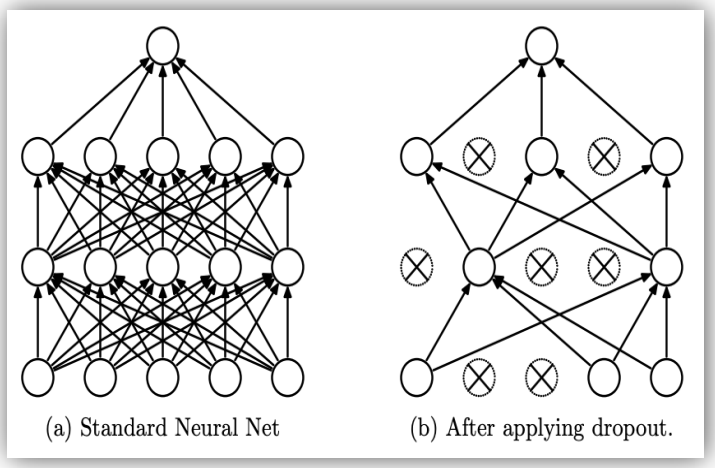


Figure 5.14- Dropout

A dropout layer is a layer in a neural network that applies dropout regularization. Typically, dropout layers are added between fully connected layers in a neural network. During training, a dropout layer randomly sets a fraction of the units in the previous layer to zero with a certain probability (usually around 0.5).

The remaining units are then scaled up by a factor of 1/ (1-p) to compensate for the dropped-out units, so that the overall magnitude of the input to the next layer remains roughly the same.

The dropout layer is then removed during inference, and the full network is used to make predictions on new, unseen data.

We also used an input scaling process which is the process of transforming the input data of a machine learning model to a common scale or distribution. The goal is to ensure that each input feature has a similar range and distribution, which can

help the model learn the underlying patterns and relationships in the data more accurately.

Input scaling is particularly important for models that use distance-based measures or gradient-based optimization algorithms, such as neural networks. The most common method for input scaling is standardization, which transforms the data to have a mean of 0 and a standard deviation of 1.

This is achieved by subtracting the mean of each feature from the data and then dividing by its standard deviation. The resulting scaled values have a mean of 0 and a standard deviation of 1. Standardization is often preferred over normalization because it is less sensitive to outliers and preserves the shape of the distribution.

Here is a description for architecture selected as the best model for each exercise error:

1. **Knees Forward Error – Squat Exercise:**

- This Model has 7 layers, including an input layer, 5 hidden layers, and an output layer. The input layer is assumed to have 36 input units
- Each of the hidden layers has a different number of units and uses the rectified linear unit (ReLU) activation function. The ReLU activation function is commonly used in deep learning models because it is computationally efficient and has been shown to be effective in reducing the risk of vanishing gradients during back propagation.
- The output layer has 2 units, indicating that this is a binary classification problem. The softmax activation function is used in the output layer to ensure that the output values are normalized such that they sum to 1 and can be interpreted as probabilities.

To summarize, the model architecture has the following layers:

1. Input layer - 36 units
2. Hidden layer 1: 36 units, ReLU activation function
3. Hidden layer 2: 416 units, ReLU activation function
4. Hidden layer 3: 192 units, ReLU activation function
5. Hidden layer 4: 64 units, ReLU activation function
6. Hidden layer 5: 384 units, ReLU activation function
7. Hidden layer 6: 256 units, ReLU activation function
8. Output layer: 2 units, softmax activation function

2. Knees Inward Error – Squat Exercise:

- The model has 3 layers, including an input layer, a hidden layer, and an output layer.
- The input layer is assumed to have 36 input units (i.e., the number of units in the first hidden layer).
- Each of the hidden layers has a different number of units and uses the rectified linear unit (ReLU) activation function. The ReLU activation function is commonly used in deep learning models because it is computationally efficient and has been shown to be effective in reducing the risk of vanishing gradients during back propagation.
- The output layer has 2 units, indicating that this is a binary classification problem. The softmax activation function is used in the output layer to ensure that the output values are normalized such that they sum to 1 and can be interpreted as probabilities.

The model architecture has the following layers:

1. Input layer - 36 units
2. Hidden layer 1: 36 units, ReLU activation function

3. Hidden layer 2: 480 units, ReLU activation function
4. Output layer: 2 units, softmax activation function

3. Lean Back Error – Biceps Exercise:

- The model has 7 layers, including an input layer, five hidden layers, and an output layer.
- The input layer is assumed to have 36 input units (i.e., the number of units in the first hidden layer).
- Each of the hidden layers has a different number of units and uses the rectified linear unit (ReLU) activation function. The ReLU activation function is commonly used in deep learning models because it is computationally efficient and has been shown to be effective in reducing the risk of vanishing gradients during back propagation.
- The output layer has 2 units, indicating that this is a binary classification problem. The softmax activation function is used in the output layer to ensure that the output values are normalized such that they sum to 1 and can be interpreted as probabilities.

The model architecture has the following layers:

1. Input layer - 36 units
2. Hidden layer 1: 36 units, ReLU activation function
3. Hidden layer 2: 448 units, ReLU activation function
4. Hidden layer 3: 96 units, ReLU activation function
5. Hidden layer 4: 320 units, ReLU activation function
6. Hidden layer 5: 256 units, ReLU activation function
7. Hidden layer 6: 256 units, ReLU activation function
8. Output layer: 2 units, softmax activation function

4. Torso Error – Barbell Row Exercise:

- The model has 3 layers, including an input layer, a hidden layer, and an output layer.
- The input layer is 52 units (i.e., the number of units in the first hidden layer).
- Each of the hidden layers has a different number of units and uses the rectified linear unit (ReLU) activation function. The ReLU activation function is commonly used in deep learning models because it is computationally efficient and has been shown to be effective in reducing the risk of vanishing gradients during back propagation.
- The output layer has 2 units, indicating that this is a binary classification problem. The softmax activation function is used in the output layer to ensure that the output values are normalized such that they sum to 1 and can be interpreted as probabilities.

The model architecture has the following layers:

1. Input layer – 52 units.
2. Hidden layer 1: 52 units, ReLU activation function
3. Hidden layer 2: 352 units, ReLU activation function
4. Output layer: 2 units, softmax activation function

5.8 Lumbar Error – Barbell Row Exercise:

- The model has 7 layers, including an input layer, four hidden layers with dropout, and an output layer.
- The input layer is assumed to have 52 units input units (i.e., the number of units in the first hidden layer).
- Each of the hidden layers (except for the input and output layers) has a different number of units, uses the rectified linear unit (ReLU) activation function, and includes a dropout layer. Dropout is a regularization technique that randomly

drops out (i.e., sets to zero) a fraction of the input units during training, which can help prevent over fitting.

- The output layer has 2 units, indicating that this is a binary classification problem. The softmax activation function is used in the output layer to ensure that the output values are normalized such that they sum to 1 and can be interpreted as probabilities.

To summarize, the model architecture has the following layers:

1. Input layer - 52 units
2. Hidden layer 1: 52 units, ReLU activation function, with dropout
3. Hidden layer 2: 448 units, ReLU activation function, with dropout
4. Hidden layer 3: 0 units, no activation function or dropout (this layer effectively does nothing and could be removed)
5. Hidden layer 4: 352 units, ReLU activation function, with dropout
6. Hidden layer 5: 0 units, no activation function or dropout (this layer effectively does nothing and could be removed)
7. Hidden layer 6: 480 units, ReLU activation function, with dropout
8. Output layer: 2 units, softmax activation function

Then we will explain some terms us used:

1. ReLU:

- This is a popular activation function used in neural networks. It is a simple, computationally efficient function that is easy to implement and has been shown to work well in many applications.
-

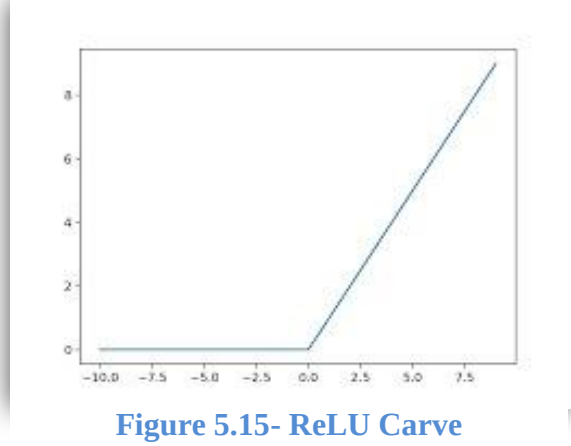


Figure 5.15- ReLU Carve

- The ReLU function is defined mathematically as: **f(x) = max (0, x)** In other words, the output of the function is equal to the input if the input is positive, and zero otherwise. This means that ReLU introduces a nonlinear "rectification" into the network, allowing it to model more complex functions than a purely linear model.
- One advantage of ReLU over other activation functions (such as the sigmoid) is that it does not saturate for large positive or negative inputs. This means that the gradient of the function is always nonzero, which can help prevent the vanishing gradient problem that can occur in deep neural networks.

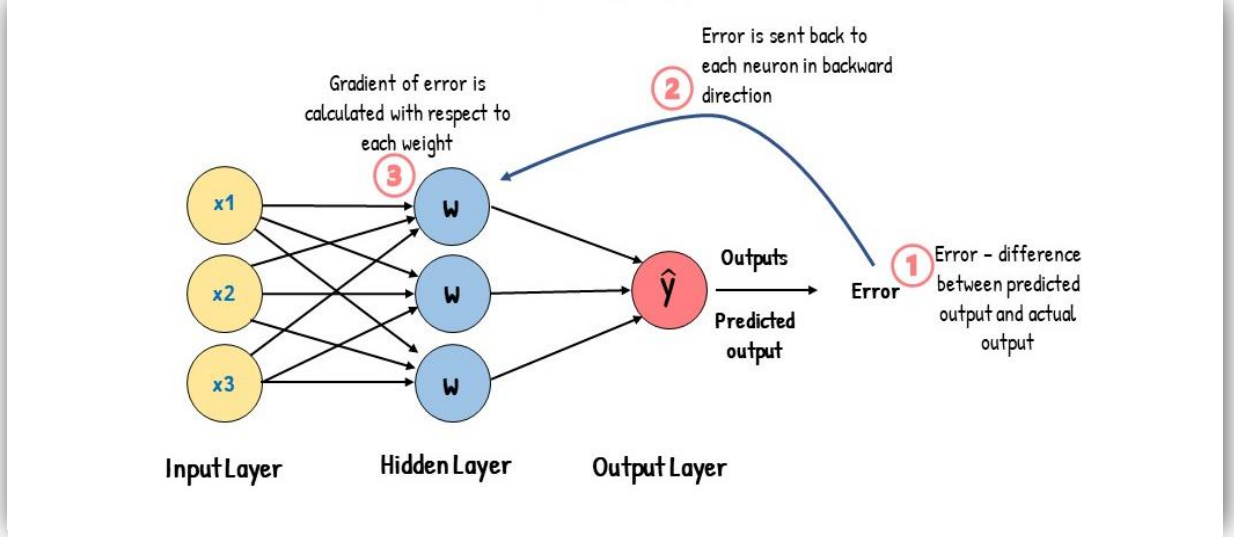


Figure 5.16- Structure of ReLU

- Another advantage of ReLU is that it is computationally efficient to evaluate, since it only involves a simple comparison and a max operation.

2. Back propagation:

- Back propagation is an algorithm used in neural networks to train the model by adjusting the weights and biases of the neurons. It is a form of supervised learning, where the network is trained on a labeled dataset with known inputs and outputs.

- The basic idea of back propagation is to start with a random initialization of the weights and biases in the network, and then iteratively adjust them in order to minimize the difference between the predicted outputs of the network and the true outputs in the training data. This is done by computing the gradient of the loss function with respect to the weights and biases, and using this gradient to update the weights and biases in the opposite direction of the gradient.

The back propagation algorithm consists of two phases:

1. **Forward propagation:** In this phase, the input data is fed into the network, and the output of each neuron is computed by applying the activation function to the weighted sum of its inputs. This process is repeated for each layer in the network, until the final output is produced.
2. **Backward propagation:** In this phase, the error between the predicted output and the true output is computed, and then propagated backwards through the network to compute the gradient of the loss function with respect to the weights and biases. This is done using the chain rule of calculus, which allows the error to be decomposed into smaller components that can be attributed to each neuron in the network. The gradient is then used to update the weights and biases of the neurons in the opposite direction of the gradient, using an optimization algorithm such as stochastic gradient descent.

- The back propagation algorithm is repeated for multiple epochs (i.e., passes through the training data) until the model converges to a set of weights and biases that minimize the loss function on the training data. At this point, the model can be used to make predictions on new, unseen data.

3. Softmax:

- This is an activation function commonly used in the output layer of neural networks for multiclass classification problems. It takes a vector of real-valued inputs and produces a probability distribution over the possible classes.

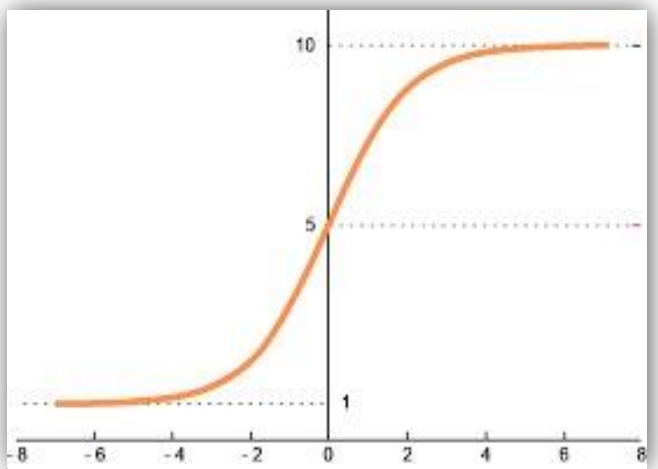


Figure 5.17- Softmax carve

- The softmax function works by exponentiation each input value and then normalizing the resulting values so that they sum to one. Mathematically, the softmax function is defined as:
    - ✓ Softmax (z_i) = exp (z_i) / sum (exp (z_j)) for j = 1 to K.
    - ✓ Where z_i is the i-th input value, K is the number of classes, and the sum is taken over all K input values.
- The output of the softmax function is a vector of K probabilities, where each element represents the probability of the input belonging to a particular class. The output probabilities are always between 0 and 1, and they sum to 1 (since they are normalized).
- The softmax function is useful for multiclass classification problems because it allows the network to output a probability distribution over the possible classes, rather than just a single prediction. This can be useful for tasks such as image classification, where there may be multiple possible classes that the input could belong to.
- During training, the softmax function is typically used in conjunction with the cross-entropy loss function, which measures the difference between the predicted probability distribution and the true distribution of the training labels.

The goal of training is to adjust the weights and biases in the network so that the predicted probabilities match the true probabilities as closely as possible.

5.8 Datasets

The dataset used in training the models for each error is the **Fitness-AQA** (Fitness Action Quality Assessment). This dataset is covering three different exercises: Squat, Barbell Row, and Overhead Press. Seven different types of exercise errors are covered. Unlabeled data is also provided to facilitate self-supervised learning. We just used the dataset of squat and barbell row exercise according to our time scheduling and requirements.

The dataset contains sets of videos and images for each exercise and also some json files that split the dataset into training, testing and validation sets. Also, there were some json files that indicate which video has specific error or not and we will talk in details about how we prepared the dataset for training down below

Here is the reference homepage of the dataset which indicates how it has been collected and built: https://github.com/ParitoshParmar/Fitness-AQA

This is the reference of full paper of the dataset: https://arxiv.org/abs/2202.14019

Another two datasets we used in Biceps Curl exercise are datasets provided by Kaggle users:

- **The first dataset**: contains videos for three exercises, one of them is Biceps Curl. The dataset is separated into segmented dataset which is collection of short videos for some seconds and original dataset without segmentation, but we used the segmented one as it is prepared well for training. This link is the reference for the dataset: https://www.kaggle.com/datasets/jiunn1998/workout-exercise

- **The second dataset:** contains videos of people doing workouts including biceps curl exercise for one rep at least meaning it is less than 5 seconds almost. This dataset mostly sourced from YouTube. Here is the reference for this dataset: https://www.kaggle.com/datasets/hasyimabdillah/workoutfitness-video

Also, our team has recorded some videos for using them in training, testing, and validation for biceps and barbell row workout. Then we will explain how we prepared the dataset for features extraction and then training stage:

1. **Lean Back Error - Biceps Exercise:**

We collected the required videos from all datasets and then separated them into training and testing dataset with ratio of 75% training to 25% testing and kept some videos less than ten videos for validation.

2. **Knees Forward and Knees Inward Errors – Squat Exercise:**

The JSON files provided in the dataset contain key value pairs in the form of dictionary. Each error has a JSON file indicating which video has error and which has not. The key is the name of the video of the workout and the value is a list containing the duration the error happened in if there is error and if not, the list is empty as following:

{45823_6": [[0.65, 2.78]], "45844_1": []}

As we are evaluating the movement in every frame and not just the movement in duration of the video, we used just the keys to extract the names of videos of required exercise and copied videos that have knees inward error in specific folder along with another folder that include videos empty of knees inward error. And so

the knees forward then we separated each error folder into training and testing with the same ratio above 75% to 25% and some videos for validation.

Finally here you can see how the dataset looked like in final stage:

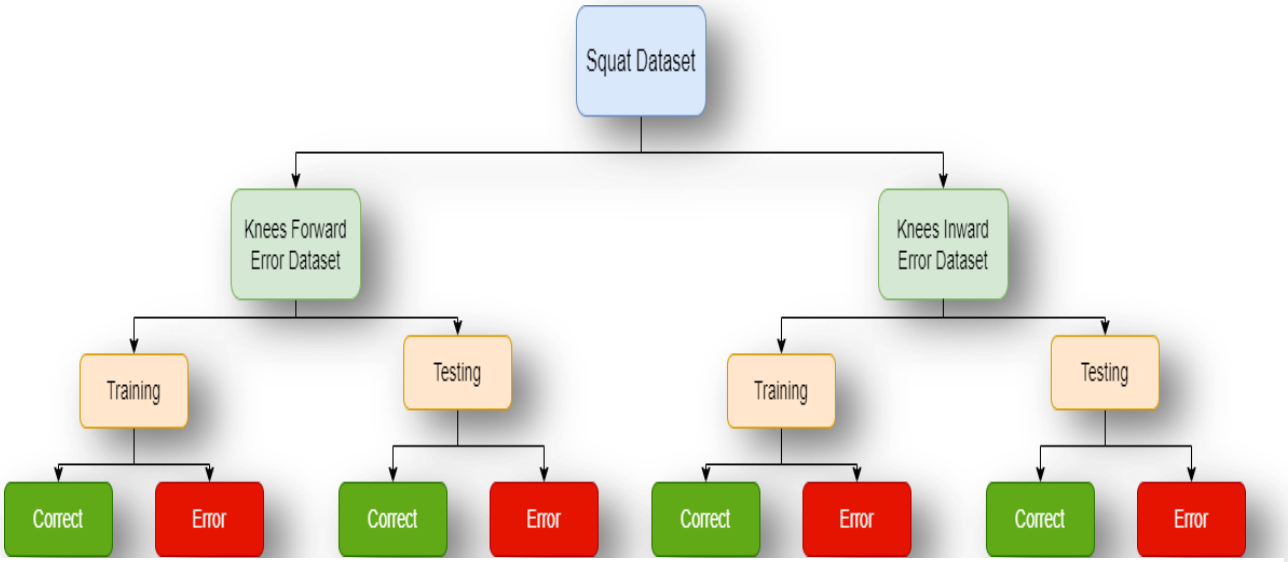


Figure 5.18- Dataset structure in squat exercise

3. Lumbar and Torso Errors – Barbell Row Exercise:

The preparation process is the same as squat except for the dataset was set of images not videos and the form of JSON files for each error looks like following:

{"57228_3_5": 0, "57009_4_12": 1,}

The key is the name of image and the value is 0 or 1 which 0 indicates that there is no error in the image and 1 indicates error existence.

Also, it was reversed as we divided the images into Lumbar and Torso using the JSON files then, we divided each error dataset into correct and error then, each correct and error dataset into training and testing with the same ration above.

Here how the dataset looked like in final stage:

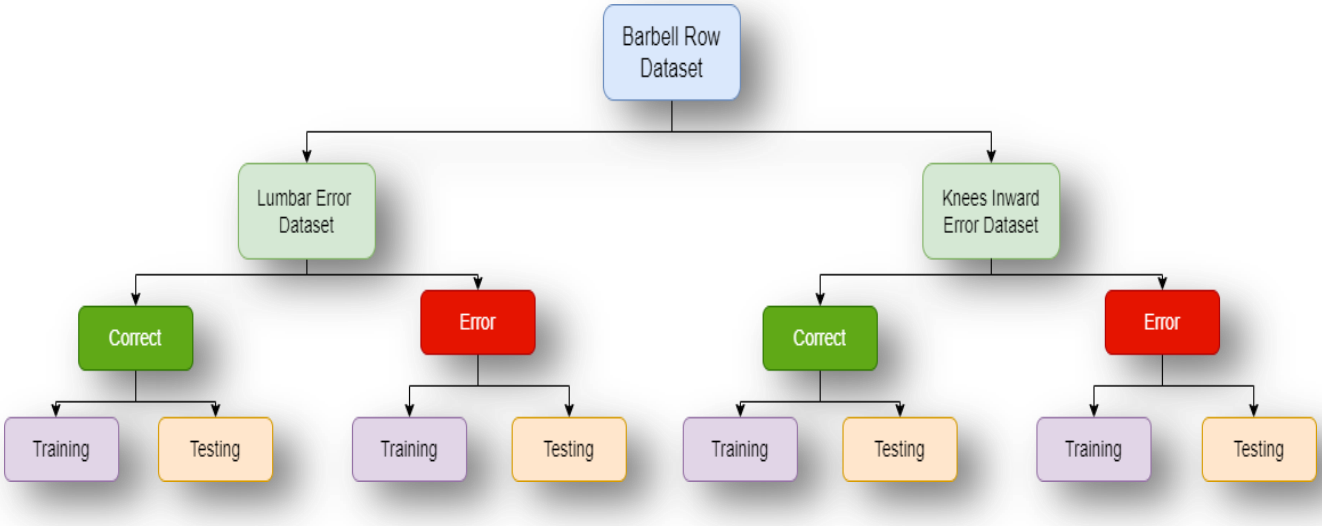


Figure 5.19- Dataset Structure in barbell row exercise

The lean back error preparation process differs from squat errors dataset and barbell dataset as the videos included performance of the wrong form and there was not correct form.

Here are the snippets of code used to extract dataset from JSON files:

This is for lumbar data extracting and the rest of errors are similar in the approach except for JSON file form:

Finally (all) dictionary does not have all the data names as they are too many.

```
all={"56067_3_51": 0, "54105_1_11": 0, "55676_6_13": 0, "55110_8_46": 0, "55912_4_20": 0, "55143_2_27": 0,
"53792_9_10": 0, "53252_3_31": 0, "56161_5_32": 0, "53377_1_54": 0, "55787_3_57": 0, "56609_4_14": 0,
"54324_11_22": 0, "56040_3_58": 0, "55485_4_16": 0, "54906_2_7": 0, "54938_2_2": 0, "57119_1_1": 0, "55657_4_14":
0, "57218_20_34": 0, "56225_3_41": 0, "55400_1_29": 0, "54543_5_31": 0, "57672_4_19": 0, "53569_2_4": 0}
import shutil
import os
for v in all:

        if(all[v]== 0 ):
            shutil.copy('E:/FCI
Bio/GP/Dataset/Paper/2/BarbellRow/Labeled_Dataset/barbellrow_images_raw/'+v+'.jpg','E:/FCI
Bio/GP/Dataset/Paper/2/BarbellRow/Labeled_Dataset/correct' )
        elif(all[v] == 1):
            shutil.copy('E:/FCI
Bio/GP/Dataset/Paper/2/BarbellRow/Labeled_Dataset/barbellrow_images_raw/'+v+'.jpg','E:/FCI
Bio/GP/Dataset/Paper/2/BarbellRow/Labeled_Dataset/lumbar' )
```

Figure 5.20- JSON

5.9 Dependencies

Here, we will go in details with dependencies and packages we used to build the model:

5.9.1 Mediapipe:

- Mediapipe is an open-source framework developed by Google that provides a pipeline for building real-time multimedia applications. It's designed to be flexible and modular, allowing developers to easily build and test new components or modify existing ones. Mediapipe is also optimized for mobile devices, making it ideal for developing applications that require low-latency, high-performance processing.
- One of the most popular features of Mediapipe is its pose estimation component, which uses machine learning to detect and track the movement of human bodies in real-time. This feature is particularly useful for applications such as fitness tracking, virtual try-on, and gesture recognition.
- MediaPipe's pose estimation component is based on a deep neural network architecture that is trained on large datasets of human pose data. The network takes input from a camera or video stream and outputs a set of key points that represent the location of different body parts, such as the shoulders, elbows, and knees.
- The Pose Landmarker uses a series of models to predict pose landmarks. The first model detects the presence of human bodies within an image frame, and the second model locates landmarks on the bodies.

The following models are packaged together into a downloadable model bundle:

- Pose detection model: detects the presence of bodies with a few key pose landmarks.
- Pose Landmarker model: adds a complete mapping of the pose. The model outputs an estimate of 33 3-dimensional pose landmarks.

This bundle uses a convolutional neural network similar to MobileNetV2 and is optimized for on-device, real-time fitness applications. Here are the 33 pose landmarks detected by Mediapipe:

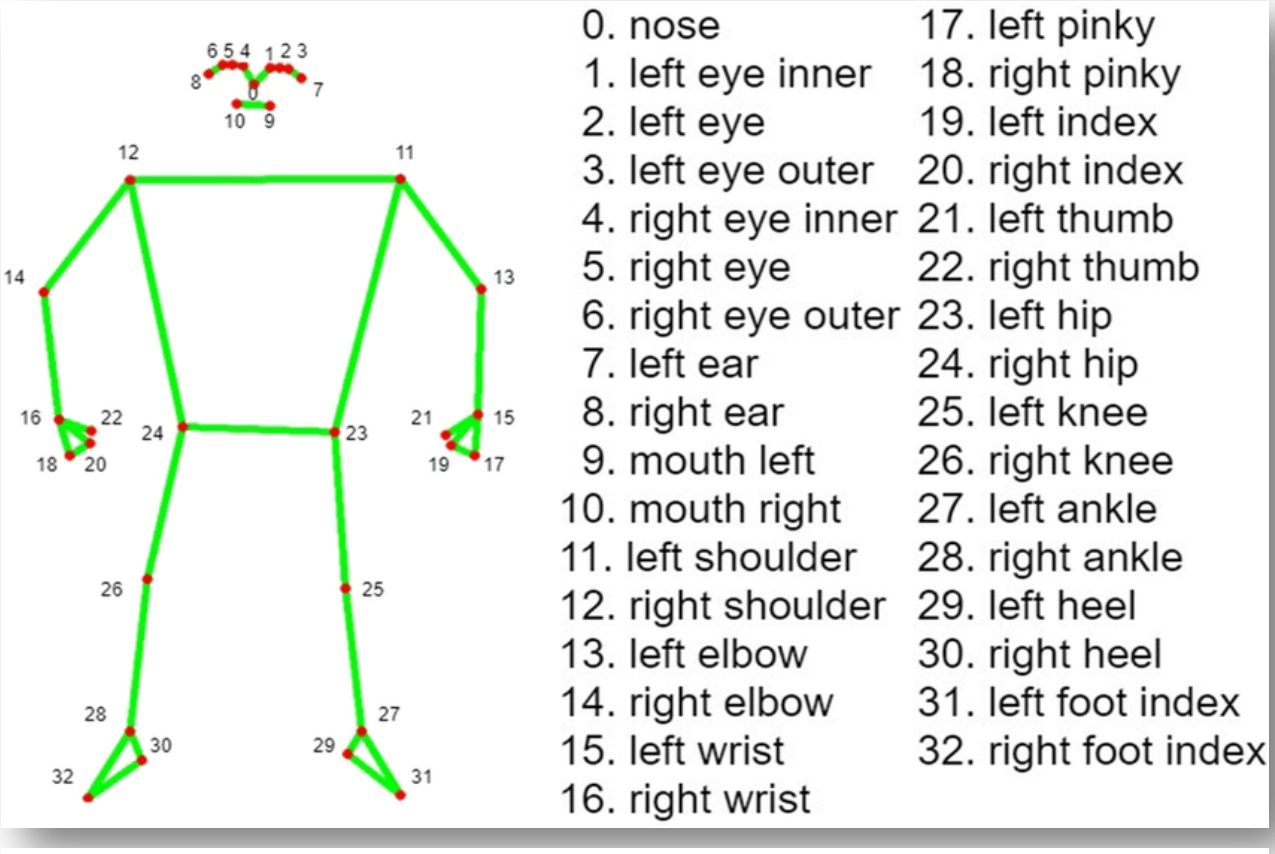


Figure 5.21- Pose landmarks that detected by Mediapipe

5.9.2 OpenCV:

- Is an open-source computer vision and machine learning library that provides a set of tools and algorithms for processing and analyzing images and videos.
- OpenCV is written in C++, but it also provides interfaces for other programming languages, including Python, Java, and MATLAB. It includes a wide range of modules that cover various areas of computer vision; including image and video processing, object detection and tracking, machine learning, and deep learning.
- One of the most popular features of OpenCV is its ability to perform real-time image and video processing. It provides a set of efficient algorithms for tasks such as feature detection, image segmentation, and object recognition, which can be used to build a wide range of applications, including surveillance systems, autonomous vehicles, and augmented reality.
- OpenCV also includes a variety of pre-trained models for object detection and recognition, such as the Haar Cascade classifier, which can detect faces, eyes, and other objects in images and videos. It also provides support for deep learning frameworks such as **TensorFlow** and **PyTorch**, allowing developers to use pre-trained models and build their own custom models for specific tasks.

Here is an example of using OpenCV and Mediapipe:

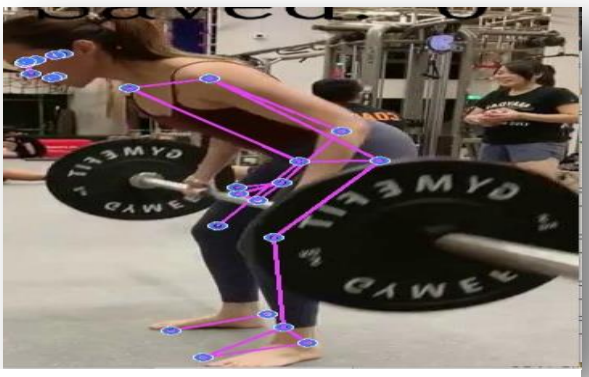


Figure 5.22- OpenCV and Mediapipe Example

5.9.3 Numpy:

- This is s an open-source Python library for scientific computing and numerical analysis. It provides a set of powerful tools for working with multi-dimensional arrays and matrices, as well as a variety of mathematical functions for performing complex numerical operations.

- NumPy is widely used in scientific computing, data analysis, and machine learning because of its efficient array processing capabilities. It provides a high-level interface for working with arrays and matrices, allowing developers to perform complex operations with ease.
- One of the key features of NumPy is its ability to perform vectorized operations, which allows for faster and more efficient processing of large arrays.

5.9.4 Pandas:

- This is an open-source library for data manipulation and analysis in Python. It provides a set of powerful tools for working with structured data, such as tabular data and time-series data, and supports a variety of data formats, including CSV, Excel, SQL databases, and JSON.

- One of the key features of Pandas is its ability to work with tabular data in the form of "DataFrame" objects. A DataFrame is a two-dimensional table-like data structure that consists of rows and columns, with each column representing a different variable or feature. Pandas provides a set of

functions for importing, cleaning, and transforming data in this format, as well as functions for performing statistical analysis and data visualization.

- Here is an example of using pandas to describe a csv file containing landmarks data which is used then in training:

```
df = describe_dataset("./train.csv")
```

```
Headers: ['label', 'nose_x', 'nose_y', 'nose_z', 'nose_v', 'left_shoulder_x', 'left_shoulder_y', 'left_shoulder_z', 'left_shoul
der_v', 'right_shoulder_x', 'right_shoulder_y', 'right_shoulder_z', 'right_shoulder_v', 'right_elbow_x', 'right_elbow_y', 'righ
t_elbow_z', 'right_elbow_v', 'left_elbow_x', 'left_elbow_y', 'left_elbow_z', 'left_elbow_v', 'right_wrist_x', 'right_wrist_y',
'right_wrist_z', 'right_wrist_v', 'left_wrist_x', 'left_wrist_y', 'left_wrist_z', 'left_wrist_v', 'left_hip_x', 'left_hip_y',
'left_hip_z', 'left_hip_v', 'right_hip_x', 'right_hip_y', 'right_hip_z', 'right_hip_v']
Number of rows: 979
Number of columns: 37

Labels:
C    557
L    422
Name: label, dtype: int64

Missing values: False

Duplicate Rows : 0
```

Figure 5.23- Pandas Example

5.9.5 Seaborn:

- This is an open-source Python visualization library based on Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics, and is particularly useful for exploring and visualizing relationships between variables in large datasets.

- Seaborn provides a variety of functions for creating different types of plots, including scatter plots, line plots, bar plots, histograms, and heatmaps. It also includes specialized functions for visualizing statistical relationships, such as regression plots, joint plots, and pair plots.

Here is an example of how we used seaborn:

```
sns.countplot(y='label', data=test_df, palette="Set1")
<AxesSubplot:xlabel='count', ylabel='label'>
```

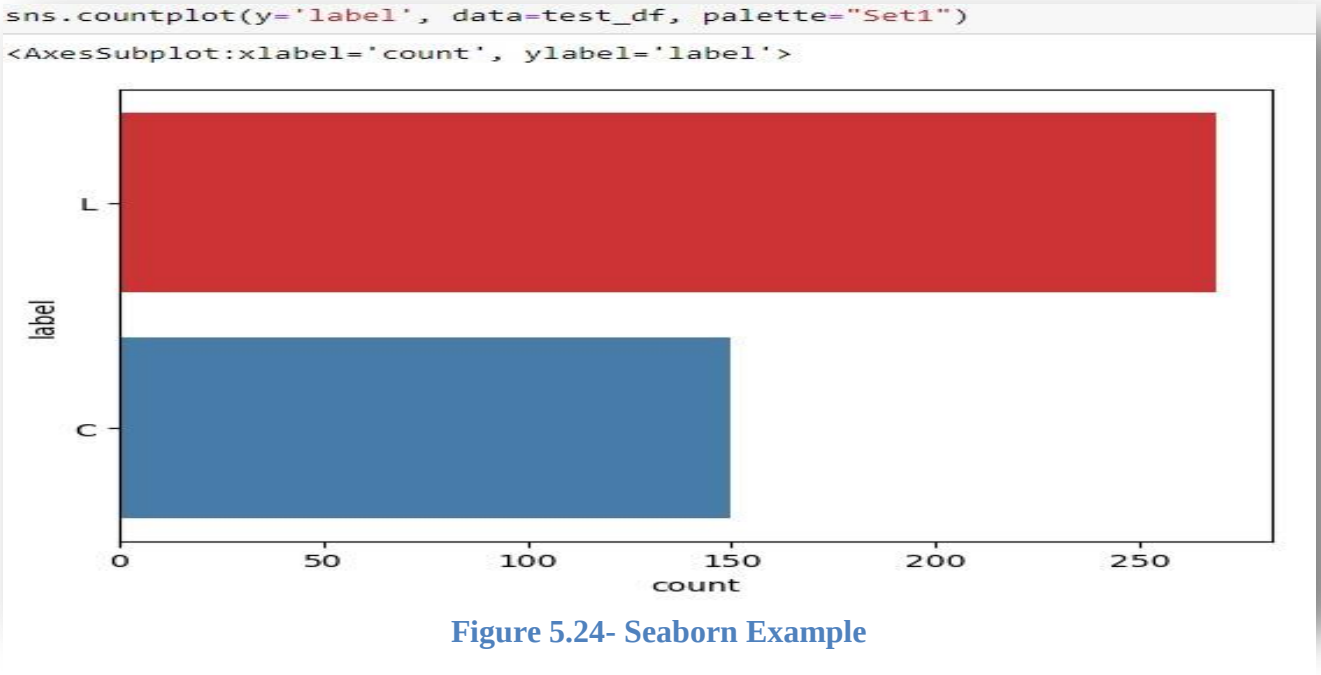


Figure 5.24- Seaborn Example

5.9.6 CSV:

- This is a built-in Python library that provides functionality for working with comma-separated values (CSV) files. A CSV file is a plain text file that stores tabular data, with each row representing a record and each column representing a field or attribute of that record.
- The csv module provides functions for reading and writing CSV files, as well as for manipulating the data within them. It supports a variety of CSV dialects and can handle different types of delimiters, such as commas, tabs, and semicolons.

Here is an example of using CSV to add columns to a csv file for collecting training or testing data:

```
with open(dataset_path, mode="w", newline="") as f:
    csv_writer = csv.writer(f, delimiter=",", quotechar='"', quoting=csv.QUOTE_MINIMAL)
    csv_writer.writerow(HEADERS)
```

Figure 5.25- Example for using CSV to add columns to CSV file

5.9.7 OS:

- This is a built-in Python library that provides a way to interact with the operating system. It provides a set of functions for working with files, directories, and other operating system-related tasks.
- Some of the common tasks that can be performed using the OS module include:
    - ✓ **File and directory operations:** The OS module provides functions for working with files and directories, such as creating, deleting, moving, and renaming files and directories.
    - ✓ **Environment variables:** The OS module provides functions for working with environment variables, such as setting and getting environment variables.
    - ✓ **Process management:** The OS module provides functions for managing processes, such as launching new processes, waiting for a process to terminate, and getting the process ID of the current process.
    - ✓ **File system information:** The OS module provides functions for getting information about the file system, such as the current working directory, the list of files in a directory, and the size and modification time of a file.
    - ✓ **Platform-specific functionality:** The OS module provides functions that are specific to different operating systems, such as functions for working with Windows registry keys on Windows systems.

Here is an example of using OS Module to read dataset from directory:

```
FOLDER_PATH = "E:/FCI Bio/GP/Dataset/Paper/2/BarbellRow/Labeled_Dataset/Barbell Row Dataset/lumbar/correct/train/"
picture_files = [os.path.join(FOLDER_PATH, f) for f in os.listdir(FOLDER_PATH) if os.path.isfile(os.path.join(FOLDER_PATH, f))]
print(f"Total pictures: {len(picture_files)}")
```

Figure 5.26- OS module to read dataset

5.9.8 Pickle:

- This is a Python library that provides a way to serialize and deserialize Python objects. It is commonly used for saving the state of a program or for transferring data between different processes or systems.
- The process of serializing a Python object involves converting it into a stream of bytes that can be stored in a file or sent over a network. The pickle library provides functions for serializing Python objects into a binary format that can be easily saved to a file or transmitted over a network.
- The process of deserializing a Python object involves converting the stream of bytes back into a Python object. The pickle library provides functions for deserializing Python objects from a binary format to their original Python object.
- We used this library to save models and input scalers as pickle files and load them in training and prediction processes.

Here is an example of using pickle library to load an input scaler:

```
with open("./model/input_scaler_lumbar.pkl", "rb") as f2:
    sc = pickle.load(f2)
```

Figure 5.27- Pickle library using

5.9.9 Sklearn (Scikit-learn):

- This is an open-source Python library for machine learning. It provides a set of tools and algorithms for data preprocessing, feature selection, model selection, and model evaluation, as well as utilities for working with large datasets and for data visualization.
- Scikit-learn are built on top of NumPy, SciPy, and Matplotlib, and provide a consistent and easy-to-use interface for machine learning tasks. It includes a wide range of algorithms for supervised and unsupervised learning, including linear and logistic regression, decision trees, random forests; support vector machines (SVMs), k-nearest neighbors (k-NN), clustering, and dimensionality reduction.
- One of the key features of Scikit-learn is its ability to handle both structured and unstructured data. It provides tools for working with tabular data, text data, and image data, and includes a variety of feature extraction and transformation functions for preparing data for machine learning algorithms.

We used this library mainly in generating reports and metrics to evaluate models. Here is an example:

- A confusion matrix that is used to evaluate the performance of a model by comparing predicted and actual class labels.

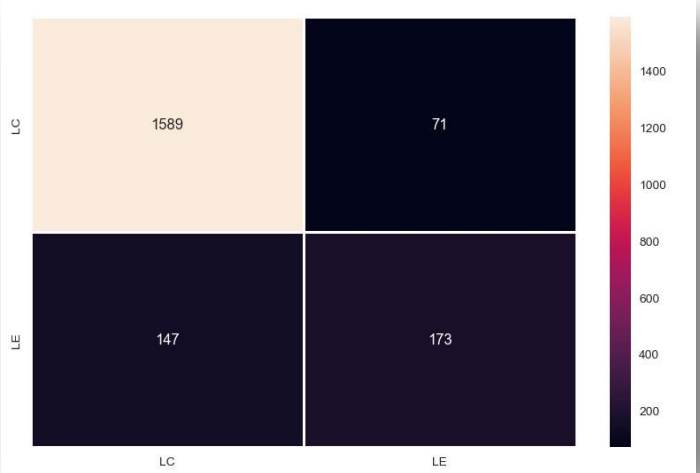


Figure 5.28- Sklearn to evaluate models

5.9.10 Keras:

- Keras is an open-source deep learning framework written in Python that provides a high-level API for building and training neural networks. It is designed to be user-friendly, modular, and extensible, and is built on top of other deep learning libraries such as TensorFlow, Microsoft Cognitive Toolkit, and Theano.
- Keras provides a simple and intuitive interface for building and training deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and multi-layer perceptrons (MLPs). It supports a wide range of activation functions, loss functions, and optimization algorithms, and provides tools for data preprocessing, data augmentation, and model evaluation.

Here is an example of the result of using keras models in training:

	Model	Precision Score	Recall Score	F1 score	Confusion Matrix
0	7_layers_with_dropout_lumbar	[0.959, 0.856]	[0.977, 0.765]	[0.968, 0.808]	[[977, 23], [42, 137]]
1	7_layers_lumbar	[0.96, 0.836]	[0.973, 0.771]	[0.966, 0.802]	[[973, 27], [41, 138]]
2	5_layers_lumbar	[0.95, 0.826]	[0.973, 0.715]	[0.961, 0.766]	[[973, 27], [51, 128]]
3	3_layers_lumbar	[0.954, 0.786]	[0.964, 0.737]	[0.959, 0.761]	[[964, 36], [47, 132]]

Figure 5.29- Keras models in training

5.9.11 Matplotlib:

- This is a Python library for creating static, interactive, and animated visualizations in Python. It is widely used in data analysis and scientific research for creating high-quality plots, charts, and other types of visualizations.
- Matplotlib provides a wide range of functions for creating different types of plots, including line plots, scatter plots, bar plots, histograms, pie charts,

and more. It also provides tools for customizing plot elements such as titles, labels, legends, and color maps.

- One of the key features of Matplotlib is its ability to create publication-quality plots with a high degree of customization. It provides a variety of built-in styles and themes that can be easily applied to plots, as well as functions for customizing the appearance of plot elements such as fonts, colors, and line styles.

Here is an example of using it in evaluation process:

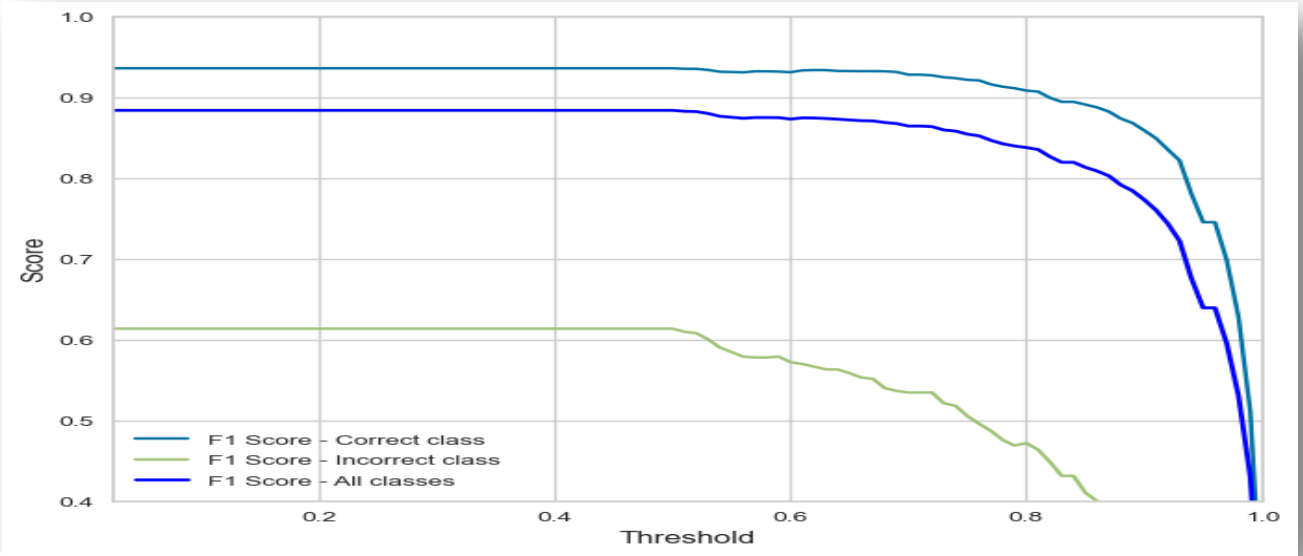


Figure 5.30- Matplotlib evaluation carve

5.9.12 TensorFlow:

- TensorFlow is an open-source machine learning framework developed by Google. It provides a set of tools for building and training deep neural networks for a wide range of tasks, including image recognition, natural language processing, and reinforcement learning.

- TensorFlow provides a flexible and scalable platform for building and training deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and deep belief networks (DBNs). It includes a wide range of activation functions, loss functions, and optimization algorithms, and provides tools for data preprocessing, data augmentation, and model evaluation.
- We used load model function provided by the keras module that allows you to load and restore a saved Keras model from a file.

5.10 Splitting Features and Target

- First, the features that we are interested in and required for training and testing process are the important landmarks that are used mostly in a specific exercise.
- The target is the class of error movement whether the performance of the movement in a specific frame is wrong and has error or correct.
- After detecting the class in every frame, we calculate the ratio of errors to correct movements and evaluate the whole performance of user`s video.
- Next, we will deep through the process of extracting features in general and preparing it for training with code figures for biceps exercise indicating the steps of extracting features (We will get enough with one exercise code figures as the rest exercises pass through the same processes but we will mention the differences and put the code figures of other exercises if needed):

Figure 5.31- Splitting Features and Target

- ✓ First, we detect the important landmarks to be used for collecting data required for training process.

```
# Determine important landmarks for Squat row exercise errors (knees forward and knees inward)
IMPORTANT_LMS = [
"NOSE",
"LEFT_SHOULDER",
"RIGHT_SHOULDER",
"LEFT_HIP",
"RIGHT_HIP",
"LEFT_KNEE",
"RIGHT_KNEE",
"LEFT_ANKLE",
"RIGHT_ANKLE",
]
```

Figure 5.32- Detecting important landmarks

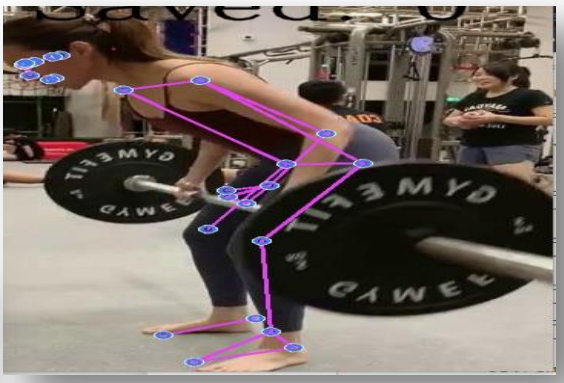
- ✓ Barbell Row Landmarks
- ✓ Squat Landmarks

After that we display the videos or the images dataset of the specific exercise in a specific directory and use Mediapipe pose models, the pose detection model to detect bodies or human existence in the video or the image and pose Landmarker for mapping the required landmarks. Then, we manually pressing a key from the keyboard to extract the landmarks of the required movement that indicate whether the user is performing wrong or right and save them in a csv file with a label

```
FOLDER_PATH = "E:/FCI Bio/GP/Dataset/Paper/2/BarbellRow/Labeled_Dataset/Barbell Row Dataset/lumbar/correct/train/"
picture_files = [os.path.join(FOLDER_PATH, f) for f in os.listdir(FOLDER_PATH) if
os.path.isfile(os.path.join(FOLDER_PATH, f))]
print(f"Total pictures: {len(picture_files)}")

DATASET_PATH = "./train.csv"
saved_counts = 0

init_csv(DATASET_PATH)

with mp_pose.Pose(min_detection_confidence=0.7, min_tracking_confidence=0.5) as pose:
    index = 0

    while True:
        if index == len(picture_files):
            break

        file_path = picture_files[index]

        image = cv2.imread(file_path)

        # Flip image horizontally for more data
        image = cv2.flip(image, 1)

        # Reduce size of a frame
        if image.shape[1] > 500:
            image = cv2.resize(image, (500, int(image.shape[0] * 500 / image.shape[1])))

        # Recolor image from BGR to RGB for mediapipe
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        results = pose.process(image)

        # Recolor image from BGR to RGB for mediapipe
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Draw landmarks and connections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
mp_drawing.DrawingSpec(color=(244, 117, 66), thickness=2, circle_radius=4), mp_drawing.DrawingSpec(color=(245, 66,
230), thickness=2, circle_radius=2))

        # Display the saved count
        cv2.putText(image, f"Saved: {saved_counts}", (20, 20), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 0, 0), 2,
cv2.LINE_AA)

        cv2.imshow("CV2", image)

        k = cv2.waitKey(1) & 0xFF

        if k == ord('d'):
            index += 1
        # Press C to save as correct form
        elif k == ord('1'):
            export_landmark_to_csv(DATASET_PATH, results, "lumbar_correct")
            saved_counts += 1
        # Press L to save as lumbar error
        elif k == ord("e"):
            export_landmark_to_csv(DATASET_PATH, results, "lumbar_error")
            saved_counts += 1
        # Press L to save as torsor angle error
        elif k == ord("t"):
            export_landmark_to_csv(DATASET_PATH, results, "torso_correct")
            saved_counts += 1
        elif k == ord("i"):
            export_landmark_to_csv(DATASET_PATH, results, "torso_error")
            saved_counts += 1
        elif k == ord("f"):
            index += 1
            os.remove(file_path)
        elif k == ord("q"):
            break
        else:
            continue


    # Close cv2 window
    cv2.destroyAllWindows()
```

Figure 5.33- Barbell row and squat landmarks

indicating the classification of these landmarks` movement whether it is correct or wrong.



label	nose_x	nose_y	nose_z	nose_v
lumbar_e	0.150119	0.189877	-0.12382	0.998912
lumbar_e	0.098437	0.216002	-0.12302	0.996727
lumbar_e	0.944771	0.162811	0.001672	0.999623
lumbar_e	0.93643	0.16199	-0.1565	0.999865
lumbar_e	0.925159	0.167002	-0.15073	0.999875
lumbar_e	0.926586	0.154271	-0.17289	0.99978
lumbar_e	0.959023	0.252627	-0.30131	0.999893
lumbar_e	0.879162	0.322513	-0.10121	0.999696
lumbar_e	0.895498	0.284718	0.070397	0.9998
lumbar_e	0.88466	0.211407	-0.07826	0.999352
lumbar_e	0.877615	0.187701	-0.0021	0.998943
lumbar_e	0.88009	0.206292	-0.27159	0.999315
lumbar_e	0.897672	0.230077	0.007154	0.996478
lumbar_e	0.901956	0.334453	-0.05741	0.997807
lumbar_e	0.948966	0.189853	-0.43121	0.999899
lumbar_e	0.9305	0.206687	-0.36413	0.999959
lumbar_e	0.94724	0.213261	-0.46198	0.999984
lumbar_e	0.947533	0.215528	-0.1811	0.999983
lumbar_e	0.879655	0.128428	-0.52795	0.999987
lumbar_e	0.946241	0.217857	-0.42788	0.99999

The other exercises differ in the type of error and labels recorded when pressing keys. Here is how the video or image displayed:

The information of landmarks extracted is the x, y, z positions of the landmark and the visibility of the landmark which is between 0 or 1and indicates the confidence that the landmark is visible in the image. A visibility of 0 indicates that the landmark is not visible, while a visibility of 1 indicates that the landmark is fully visible.

After that, we describe the csv dataset to know the content of the information we have and how many class labels we have and the number of each label instances.

```
def describe_dataset(dataset_path: str):
    '''
    Describe dataset
    '''

    data = pd.read_csv(dataset_path)
    print(f"Headers: {list(data.columns.values)}")
    print(f'Number of rows: {data.shape[0]} \nNumber of columns: {data.shape[1]}\n')
    print(f"Labels: \n{data['label'].value_counts()}\n")
    print(f"Missing values: {data.isnull().values.any()}\n")

    duplicate = data[data.duplicated()]
    print(f"Duplicate Rows : {len(duplicate.sum(axis=1))}")

    return data
```

Figure 5.34- Description CSV dataset

The describe dataset function:

```
df = describe_dataset("./train.csv")

Headers: ['label', 'nose_x', 'nose_y', 'nose_z', 'nose_v', 'left_shoulder_x', 'left_shoulder_y', 'left_shoulder_z', 'left_shoul
der_v', 'right_shoulder_x', 'right_shoulder_y', 'right_shoulder_z', 'right_shoulder_v', 'right_elbow_x', 'right_elbow_y', 'righ
t_elbow_z', 'right_elbow_v', 'left_elbow_x', 'left_elbow_y', 'left_elbow_z', 'left_elbow_v', 'right_wrist_x', 'right_wrist_y',
'right_wrist_z', 'right_wrist_v', 'left_wrist_x', 'left_wrist_y', 'left_wrist_z', 'left_wrist_v', 'left_hip_x', 'left_hip_y',
'left_hip_z', 'left_hip_v', 'right_hip_x', 'right_hip_y', 'right_hip_z', 'right_hip_v']
Number of rows: 979
Number of columns: 37

Labels:
C    557
L    422
Name: label, dtype: int64

Missing values: False

Duplicate Rows : 0
```

Figure 5.35- Description Dataset function

The result of describe dataset function:

In summary, here how are the features extracting process goes down:

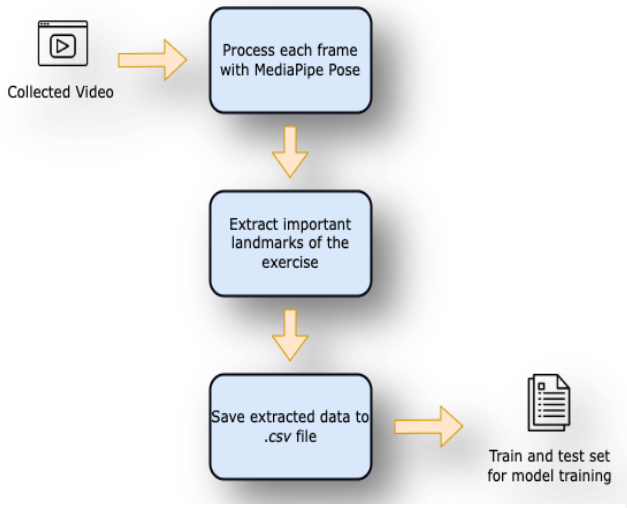


Figure 5.36- Result of describe dataset function

The target here is to classify the movement in each frame whether it is correct or not so, here are the targets in each model.

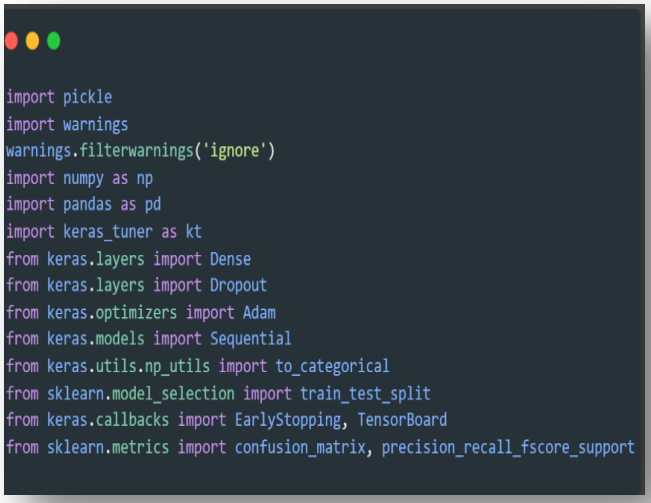
- In Biceps: the target is to classify back movement as lean back error or correct movement.
- In Squat: to classify knees movement whether it is forward error in the down stage or not and also whether it is inward error or not while the user getting to the up stage and holding too heavy weights.
- In Barbell Row: classify whether the status of lumbar part of the back is correct or wrong and the status of the torso correct or wrong.

5.11Training

The training process involves iteratively adjusting the parameters of a neural network in order to minimize the difference between the predicted outputs of the network and the true outputs. This is typically done using a technique called backpropagation, which involves computing the gradient of the loss function with respect to each parameter in the network and then adjusting the parameters in the direction of the negative gradient.

Here is how we went through training process which is the same for all models:

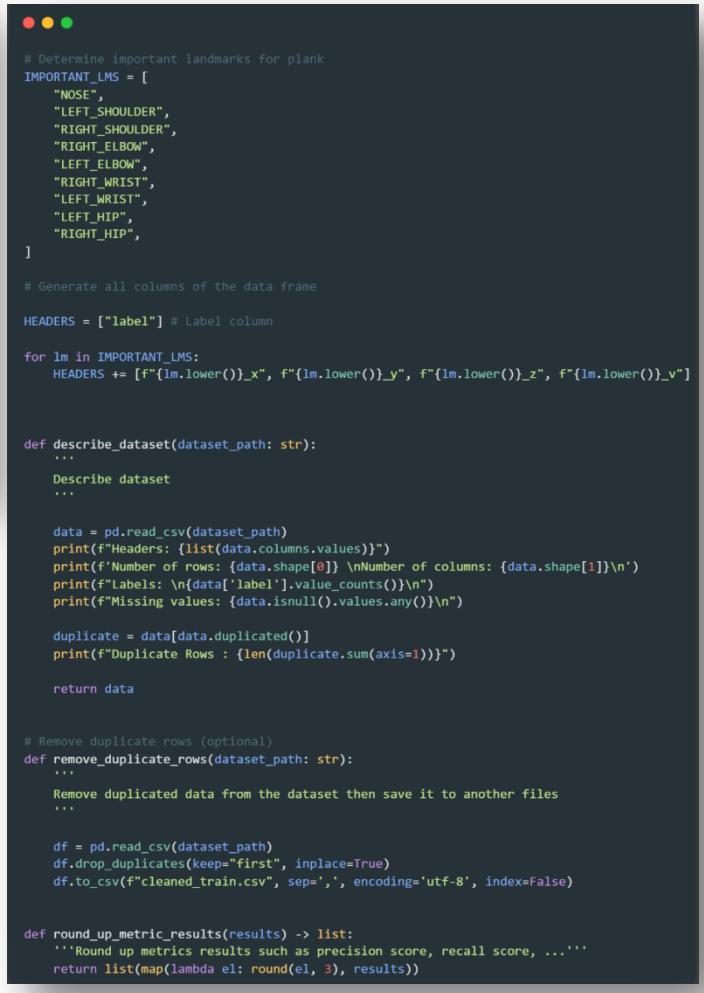
- First, we define the required dependencies for training process.



```
import pickle
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
import keras_tuner as kt
from keras.layers import Dense
from keras.layers import Dropout
from keras.optimizers import Adam
from keras.models import Sequential
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping, TensorBoard
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support
```

Figure 5.37- Definition of required dependencies for training process

- Second, we define the required landmarks and functions such as describe_datatset as we mentioned before in splitting features and target section.



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

# Generate all columns of the data frame

HEADERS = ["label"] # Label column

for lm in IMPORTANT_LMS:
    HEADERS += [f"{lm.lower()}_x", f"{lm.lower()}_y", f"{lm.lower()}_z", f"{lm.lower()}_v"]



def describe_dataset(dataset_path: str):
    '''
    Describe dataset
    '''

    data = pd.read_csv(dataset_path)
    print(f"Headers: {list(data.columns.values)}")
    print(f'Number of rows: {data.shape[0]} \nNumber of columns: {data.shape[1]}\n')
    print(f"Labels: \n{data['label'].value_counts()}\n")
    print(f"Missing values: {data.isnull().values.any()}\n")

    duplicate = data[data.duplicated()]
    print(f"Duplicate Rows : {len(duplicate.sum(axis=1))}")

    return data


# Remove duplicate rows (optional)
def remove_duplicate_rows(dataset_path: str):
    '''
    Remove duplicated data from the dataset then save it to another files
    '''

    df = pd.read_csv(dataset_path)
    df.drop_duplicates(keep="first", inplace=True)
    df.to_csv(f"cleaned_train.csv", sep=',', encoding='utf-8', index=False)


def round_up_metric_results(results) -> list:
    '''Round up metrics results such as precision score, recall score, ...'''
    return list(map(lambda el: round(el, 3), results))
```

- After that, we categorize the label column of the DataFrame which is the csv file of landmarks and labels by replacing any occurrences of the string of specific label with the integer value 0 and any occurrences of the string of specific label with the integer value 1 and so on if the labels are more than that.

```
# load dataset
df = describe_dataset("./train.csv")

# Categorizing label
df.loc[df["label"] == "C", "label"] = 0
df.loc[df["label"] == "L", "label"] = 1
```

Figure 5.38- Categorize the label column of DataFrame

Categorizing labels of lean back error in biceps exercise.

- Then, we load a saved input scaler object from a file, standardize the input features of the dataset using the scaler, split the dataset into training and testing sets, and define an early stopping criterion for training a machine learning model.

```
with open("./model/input_scaler.pkl", "rb") as f:
sc = pickle.load(f)

Standard Scaling of features
x = df.drop("label", axis = 1)
x = pd.DataFrame(sc.transform(x))

y = df["label"]

y_cat = to_categorical(y)

x_train, x_test, y_train, y_test = train_test_split(x.values, y_cat, test_size=0.2, random_state=1234)

stop_early = EarlyStopping(monitor='val_loss', patience=3)

Final Results
final_models = {}
```

Figure 5.39- Categorizing labels of lean back error in biceps exercise

- The next is to define a few functions for describing the architecture of the feedforward neural network, selecting the best model using Keras Tuner, and building a specific type of model architecture. We then use Keras Tuner to search for the best hyper parameters for the model architecture and trains the model using the best hyper parameters found.

The models used for training are 4 models as we mentioned before in the models` architecture section, 3 layers, 5 layers, 7 layers, and 7 layers with dropout layer then, we select the best model of them with best results and scores.

```
def describe_model(model):
'''
Describe Model architecture
'''
print(f"Describe models architecture")
for i, layer in enumerate(model.layers):
number_of_units = layer.units if hasattr(layer, 'units') else 0

python
Copy
    if hasattr(layer, "activation"):
        print(f"Layer-{i + 1}: {number_of_units} units, func: ", layer.activation)
    else:
        print(f"Layer-{i + 1}: {number_of_units} units, func: None")
def get_best_model(tuner):
'''
Describe and return the best model found from keras tuner
'''
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
best_model = tuner.hypermodel.build(best_hps)

stylus
Copy
describe_model(best_model)

print("\nOther params:")
ignore_params = ["tuner", "activation", "layer", "epoch"]
for param, value in best_hps.values.items():
    if not any(word in param for word in ignore_params):
        print(f"{param}: {value}")

return best_model
def model_31_builder(hp):
model = Sequential()
model.add(Dense(36, input_dim = 36, activation = "relu"))

routeros
Copy
hp_activation = hp.Choice('activation', values=['relu', 'tanh'])
hp_layer_1 = hp.Int('layer_1', min_value=32, max_value=512, step=32)
hp_learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])

model.add(Dense(units=hp_layer_1, activation=hp_activation))
model.add(Dense(2, activation = "softmax"))

model.compile(optimizer=Adam(learning_rate=hp_learning_rate), loss="categorical_crossentropy", metrics =
["accuracy"])

return model
tuner_31 = kt.Hyperband(
model_31_builder,
objective='val_accuracy',
max_epochs=10,
directory='keras_tuner_dir',
project_name='keras_tuner_demo',
)
tuner_31.search(x_train, y_train, validation_data=(x_test, y_test), epochs=10, callbacks=[stop_early])

model_31 = get_best_model(tuner_31)
model_31.fit(x_train, y_train, epochs=100, batch_size=10, validation_data=(x_test, y_test), callbacks=[stop_early])

final_models["3_layers"] = model_31
```

Figure 5.40- Selecting the best model with the best results and scores

The next step is evaluating the performance of the trained deep learning models stored in the final models dictionary using the test dataset. Then, we print a description of each model's architecture, and then evaluate each model's performance by computing the confusion matrix, precision, recall, and F1 score. It stores the results in a list called train_set_results, sorts the results by F1 score in descending order, and displays the results in a Pandas DataFrame.

```
for name, model in final_models.items():
print(f"{name}: ", end="")
describe_model(model)
print()

train_set_results = []

for name, model in final_models.items():
# Evaluate model
predict_x = model.predict(x_test, verbose=False)
y_pred_class = np.argmax(predict_x, axis=1)
y_test_class = np.argmax(y_test, axis=1)

reasonml
Copy
cm = confusion_matrix(y_test_class, y_pred_class, labels=[0, 1])
(p_score, r_score, f_score, _) = precision_recall_fscore_support(y_test_class, y_pred_class, labels=[0, 1])

train_set_results.append(( name, round_up_metric_results(p_score), round_up_metric_results(r_score),
round_up_metric_results(f_score), cm ))
train_set_results.sort(key=lambda k: sum(k[3]), reverse=True)
pd.DataFrame(train_set_results, columns=["Model", "Precision Score", "Recall Score", "F1 score", "Confusion
Matrix"])
```

Figure 5.41- Displaying the results in a Pandas DataFrame

Here are the results for this code snippet:

```
3_layers: Describe models architecture
Layer-1: 36 units, func:  <function relu at 0x00000208FEB779D0>
Layer-2: 448 units, func:  <function relu at 0x00000208FEB779D0>
Layer-3: 2 units, func:  <function softmax at 0x00000208FEB70F70>

5_layers: Describe models architecture
Layer-1: 36 units, func:  <function relu at 0x00000208FEB779D0>
Layer-2: 256 units, func:  <function tanh at 0x00000208FEB77D30>
Layer-3: 320 units, func:  <function tanh at 0x00000208FEB77D30>
Layer-4: 480 units, func:  <function tanh at 0x00000208FEB77D30>
Layer-5: 2 units, func:  <function softmax at 0x00000208FEB70F70>

7_layers_with_dropout: Describe models architecture
Layer-1: 36 units, func:  <function relu at 0x00000208FEB779D0>
Layer-2: 448 units, func:  <function tanh at 0x00000208FEB77D30>
Layer-3: 0 units, func: None
Layer-4: 192 units, func:  <function tanh at 0x00000208FEB77D30>
Layer-5: 0 units, func: None
Layer-6: 128 units, func:  <function tanh at 0x00000208FEB77D30>
Layer-7: 2 units, func:  <function softmax at 0x00000208FEB70F70>

7_layers: Describe models architecture
Layer-1: 36 units, func:  <function relu at 0x00000208FEB779D0>
Layer-2: 448 units, func:  <function relu at 0x00000208FEB779D0>
Layer-3: 96 units, func:  <function relu at 0x00000208FEB779D0>
Layer-4: 320 units, func:  <function relu at 0x00000208FEB779D0>
Layer-5: 256 units, func:  <function relu at 0x00000208FEB779D0>
Layer-6: 256 units, func:  <function relu at 0x00000208FEB779D0>
Layer-7: 2 units, func:  <function softmax at 0x00000208FEB70F70>
```

Figure 5.42- Result of code snippet

	Model	Precision Score	Recall Score	F1 score	Confusion Matrix
0	3_layers	[1.0, 1.0]	[1.0, 1.0]	[1.0, 1.0]	[[113, 0], [0, 83]]
1	5_layers	[1.0, 1.0]	[1.0, 1.0]	[1.0, 1.0]	[[113, 0], [0, 83]]
2	7_layers_with_dropout	[0.991, 1.0]	[1.0, 0.988]	[0.996, 0.994]	[[113, 0], [1, 82]]
3	7_layers	[0.991, 1.0]	[1.0, 0.988]	[0.996, 0.994]	[[113, 0], [1, 82]]

Then, we repeat the categorizing process, scaling, and evaluating the models with the test dataset and then saving the best model in an extension of h5 file.

```
# load dataset
test_df = describe_dataset("./test.csv")

# Categorizing label
test_df.loc[test_df["label"] == "C", "label"] = 0
test_df.loc[test_df["label"] == "L", "label"] = 1

# Standard Scaling of features
test_x = test_df.drop("label", axis = 1)
test_x = pd.DataFrame(sc.transform(test_x))

test_y = test_df["label"]

# # Converting prediction to categorical
test_y_cat = to_categorical(test_y)

test_set_results = []

for name, model in final_models.items():
    # Evaluate model
    predict_x = model.predict(test_x, verbose=False)
    y_pred_class = np.argmax(predict_x, axis=1)
    y_test_class = np.argmax(test_y_cat, axis=1)

    cm = confusion_matrix(y_test_class, y_pred_class, labels=[0, 1])
    (p_score, r_score, f_score, _) = precision_recall_fscore_support(y_test_class, y_pred_class, labels=[0, 1])

    test_set_results.append(( name, round_up_metric_results(p_score), round_up_metric_results(r_score),
round_up_metric_results(f_score), cm ))

test_set_results.sort(key=lambda k: k[1] + k[2] + k[3], reverse=True)
pd.DataFrame(test_set_results, columns=["Model", "Precision Score", "Recall Score", "F1 score", "Confusion
Matrix"])

final_models["7_layers"].save("C:/Users/Alrowad/Exercise-correction/biceps_model/model/bicep_dp.h5")
```

Figure 5.43- Testing process code

Here are the results of testing process:

	Model	Precision Score	Recall Score	F1 score	Confusion Matrix
0	7_layers_with_dropout	[0.629, 0.827]	[0.713, 0.766]	[0.669, 0.795]	[[107, 43], [63, 206]]
1	7_layers	[0.625, 0.905]	[0.867, 0.71]	[0.726, 0.796]	[[130, 20], [78, 191]]
2	5_layers	[0.58, 0.897]	[0.867, 0.651]	[0.695, 0.754]	[[130, 20], [94, 175]]
3	3_layers	[0.576, 0.756]	[0.553, 0.773]	[0.565, 0.765]	[[83, 67], [61, 208]]

Figure 5.44- Results of testing process

Note that these results and the previous ones are for biceps models. The rest exercises pass through the same processes so, we do not need to deep through their training.

5.12Prediction

The prediction process involves passing input data through a series of layers, each containing a set of learnable parameters that transform the input data into a more useful representation. The output of the last layer is then passed through an activation function to produce the final prediction.

Here how the prediction process goes down in biceps exercise which is the same for squat and barbell:

- First thing is to define the required dependencies, landmarks and functions used for prediction process.

```
import mediapipe as mp
import cv2
import numpy as np
import pandas as pd
import datetime

import pickle

import warnings
warnings.filterwarnings('ignore')

# Drawing helpers
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose

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

# Generate all columns of the data frame

HEADERS = ["label"] # Label column

for lm in IMPORTANT_LMS:
    HEADERS += [f"{lm.lower()}_x", f"{lm.lower()}_y", f"{lm.lower()}_z", f"{lm.lower()}_v"]
```

Figure 5.45- Required dependencies, landmarks and functions used for prediction process

```python
def rescale_frame(frame, percent=50):
    '''
    Rescale a frame from OpenCV to a certain percentage compare to its original frame
    '''
    width = int(frame.shape[1] * percent/ 100)
    height = int(frame.shape[0] * percent/ 100)
    dim = (width, height)
    return cv2.resize(frame, dim, interpolation =cv2.INTER_AREA)


def calculate_angle(point1: list, point2: list, point3: list) -> float:
    '''
    Calculate the angle between 3 points
    Unit of the angle will be in Degree
    '''
    point1 = np.array(point1)
    point2 = np.array(point2)
    point3 = np.array(point3)

    # Calculate algo
    angleInRad = np.arctan2(point3[1] - point2[1], point3[0] - point2[0]) - np.arctan2(point1[1] - point2[1],
point1[0] - point2[0])
    angleInDeg = np.abs(angleInRad * 180.0 / np.pi)

    angleInDeg = angleInDeg if angleInDeg <= 180 else 360 - angleInDeg
    return angleInDeg


def extract_important_keypoints(results, important_landmarks: list) -> list:
    '''
    Extract important keypoints from mediapipe pose detection
    '''
    landmarks = results.pose_landmarks.landmark

    data = []
    for lm in important_landmarks:
        keypoint = landmarks[mp_pose.PoseLandmark[lm].value]
        data.append([keypoint.x, keypoint.y, keypoint.z, keypoint.visibility])

    return np.array(data).flatten().tolist()
```

- The next thing is to build a Biceps pose analysis class or squat analysis or barbell analysis class that performs analysis on human bicep poses using the output of a pose estimation model. The class has a __init__ method that initializes various thresholds for the pose analysis, along with some other parameters. The class also has a get joints method that checks for the visibility of bicep pose joints and gets their coordinates, and an analyze pose method that analyzes the pose by detecting errors and counting bicep curls.

```python
class BicepPoseAnalysis:
    def __init__(self, side: str, stage_down_threshold: float, stage_up_threshold: float,
peak_contraction_threshold: float, loose_upper_arm_angle_threshold: float, visibility_threshold: float):
        # Initializing thresholds
        self.stage_down_threshold = stage_down_threshold
        self.stage_up_threshold = stage_up_threshold
        self.peak_contraction_threshold = peak_contraction_threshold
        self.loose_upper_arm_angle_threshold = loose_upper_arm_angle_threshold
        self.visibility_threshold = visibility_threshold

        self.side = side
        self.counter = 0
        self.stage = "down"
        self.is_visible = True
        self.detected_errors = {
            "LOOSE_UPPER_ARM": 0,
            "PEAK_CONTRACTION": 0,
        }

        # Params for loose upper arm error detection
        self.loose_upper_arm = False

        # Params for peak contraction error detection
        self.peak_contraction_angle = 1000
        self.peak_contraction_frame = None

    def get_joints(self, landmarks) -> bool:
        '''
        Check for joints' visibility then get joints coordinate
        '''
        side = self.side.upper()

        # Check visibility
        joints_visibility = [ landmarks[mp_pose.PoseLandmark[f"{side}_SHOULDER"].value].visibility,
landmarks[mp_pose.PoseLandmark[f"{side}_ELBOW"].value].visibility, landmarks[mp_pose.PoseLandmark[f"
{side}_WRIST"].value].visibility ]

        is_visible = all([ vis > self.visibility_threshold for vis in joints_visibility ])
        self.is_visible = is_visible

        if not is_visible:
            return self.is_visible

        # Get joints' coordinates
        self.shoulder = [ landmarks[mp_pose.PoseLandmark[f"{side}_SHOULDER"].value].x,
landmarks[mp_pose.PoseLandmark[f"{side}_SHOULDER"].value].y ]
        self.elbow = [ landmarks[mp_pose.PoseLandmark[f"{side}_ELBOW"].value].x, landmarks[mp_pose.PoseLandmark[f"
{side}_ELBOW"].value].y ]
        self.wrist = [ landmarks[mp_pose.PoseLandmark[f"{side}_WRIST"].value].x, landmarks[mp_pose.PoseLandmark[f"
{side}_WRIST"].value].y ]

        return self.is_visible
```

Figure 5.46- Building Biceps pose analysis class

```
    def analyze_pose(self, landmarks, frame):
        '''
        - Bicep Counter
        - Errors Detection
        '''
        self.get_joints(landmarks)

        # Cancel calculation if visibility is poor
        if not self.is_visible:
            return (None, None)

        # * Calculate curl angle for counter
        bicep_curl_angle = int(calculate_angle(self.shoulder, self.elbow, self.wrist))
        if bicep_curl_angle > self.stage_down_threshold:
            self.stage = "down"
        elif bicep_curl_angle < self.stage_up_threshold and self.stage == "down":
            self.stage = "up"
            self.counter += 1

        # * Calculate the angle between the upper arm (shoulder & joint) and the Y axis
        shoulder_projection = [ self.shoulder[0], 1 ] # Represent the projection of the shoulder to the X axis
        ground_upper_arm_angle = int(calculate_angle(self.elbow, self.shoulder, shoulder_projection))

        # * Evaluation for LOOSE UPPER ARM error
        if ground_upper_arm_angle > self.loose_upper_arm_angle_threshold:
            # Limit the saved frame
            if not self.loose_upper_arm:
                self.loose_upper_arm = True
                # save_frame_as_image(frame, f"Loose upper arm: {ground_upper_arm_angle}")
                self.detected_errors["LOOSE_UPPER_ARM"] += 1
        else:
            self.loose_upper_arm = False

        # * Evaluate PEAK CONTRACTION error
        if self.stage == "up" and bicep_curl_angle < self.peak_contraction_angle:
            # Save peaked contraction every rep
            self.peak_contraction_angle = bicep_curl_angle
            self.peak_contraction_frame = frame

        elif self.stage == "down":
            # * Evaluate if the peak is higher than the threshold if True, marked as an error then saved that frame
            if self.peak_contraction_angle != 1000 and self.peak_contraction_angle >=
self.peak_contraction_threshold:
                # save_frame_as_image(self.peak_contraction_frame, f"{self.side} - Peak Contraction:
{self.peak_contraction_angle}")
                self.detected_errors["PEAK_CONTRACTION"] += 1

            # Reset params
            self.peak_contraction_angle = 1000
            self.peak_contraction_frame = None

        return (bicep_curl_angle, ground_upper_arm_angle)
```

- After that, we define some validation video paths, load input scaler and load the detection or prediction model.
- Analyze bicep curls in a video using pose estimation. It starts by initializing some constants and an instance of the BicepPoseAnalysis class for both the left and right arms. The code then enters a loop that reads frames from

```
from tensorflow.keras.models import load_model

VIDEO_PATH1 = "./dataset/val/video_2023-04-14_19-29-09.mp4"
VIDEO_PATH2 = "./dataset/val/video_2023-04-14_19-29-38.mp4"
VIDEO_PATH3 = "./dataset/val/video_2023-04-14_19-29-47.mp4"
VIDEO_PATH4 = "./dataset/val/VID_20230419_021302.mp4"
VIDEO_PATH5 = "./dataset/val/VID_20230419_021426.mp4"
VIDEO_PATH6 = "./dataset/val/VID_20230419_021551.mp4"
VIDEO_PATH7 = "./dataset/val/VID_20230419_021654.mp4"
VIDEO_PATH8 = "./dataset/val/VID_20230419_021802.mp4"
VIDEO_PATH9 = "./dataset/val/VID_20230419_021924.mp4"
VIDEO_PATH10 = "./dataset/val/VID_20230419_022013.mp4"

# Load input scaler
with open("./model/input_scaler.pkl", "rb") as f:
    input_scaler = pickle.load(f)


# Load model
DL_model = load_model("./model/7_layers.h5")
```

Figure 5.47- Analyzing biceps curls in video using pose estimation

a video using OpenCV and performs pose estimation on each frame using the mp_pose.Pose module.

- If a human is detected in the frame, the analyze pose method of the BicepPoseAnalysis class is called for both the left and right arms to analyze their bicep curl angles and upper arm angles. The code then extracts key points from the frame and makes a prediction using a deep learning model. The predicted class and probability are displayed along with the counter for each arm and any detected errors such as loose upper arm or peak contraction.
- The same process goes for squat and barbell with different errors and results.

```
cap = cv2.VideoCapture(VIDEO_PATH2)

VISIBILITY_THRESHOLD = 0.65

# Params for counter
STAGE_UP_THRESHOLD = 90
STAGE_DOWN_THRESHOLD = 120

# Params to catch FULL RANGE OF MOTION error
PEAK_CONTRACTION_THRESHOLD = 60

# LOOSE UPPER ARM error detection
LOOSE_UPPER_ARM = False
LOOSE_UPPER_ARM_ANGLE_THRESHOLD = 40

# STANDING POSTURE error detection
POSTURE_ERROR_THRESHOLD = 0.95
posture = 0

# Init analysis class
left_arm_analysis = BicepPoseAnalysis(side="left", stage_down_threshold=STAGE_DOWN_THRESHOLD,
stage_up_threshold=STAGE_UP_THRESHOLD, peak_contraction_threshold=PEAK_CONTRACTION_THRESHOLD,
loose_upper_arm_angle_threshold=LOOSE_UPPER_ARM_ANGLE_THRESHOLD, visibility_threshold=VISIBILITY_THRESHOLD)

right_arm_analysis = BicepPoseAnalysis(side="right", stage_down_threshold=STAGE_DOWN_THRESHOLD,
stage_up_threshold=STAGE_UP_THRESHOLD, peak_contraction_threshold=PEAK_CONTRACTION_THRESHOLD,
loose_upper_arm_angle_threshold=LOOSE_UPPER_ARM_ANGLE_THRESHOLD, visibility_threshold=VISIBILITY_THRESHOLD)

with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, image = cap.read()

        if not ret:
            break

        # Reduce size of a frame
        image = rescale_frame(image, 50)
        # image = cv2.flip(image, 1)

        video_dimensions = [image.shape[1], image.shape[0]]

        # Recolor image from BGR to RGB for mediapipe
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        results = pose.process(image)

        if not results.pose_landmarks:
            print("No human found")
            continue

        # Recolor image from BGR to RGB for mediapipe
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Draw landmarks and connections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
mp_drawing.DrawingSpec(color=(244, 117, 66), thickness=2, circle_radius=2), mp_drawing.DrawingSpec(color=(245, 66,
230), thickness=2, circle_radius=1))
```

Figure 5.48- Deferent errors and results for biceps and barbell exercises

```
        # Make detection
        try:
            landmarks = results.pose_landmarks.landmark

            (left_bicep_curl_angle, left_ground_upper_arm_angle) =
left_arm_analysis.analyze_pose(landmarks=landmarks, frame=image)
            (right_bicep_curl_angle, right_ground_upper_arm_angle) =
right_arm_analysis.analyze_pose(landmarks=landmarks, frame=image)

            # Extract keypoints from frame for the input
            row = extract_important_keypoints(results, IMPORTANT_LMS)
            X = pd.DataFrame([row, ], columns=HEADERS[1:])
            X = pd.DataFrame(input_scaler.transform(X))

            # Make prediction and its probability
            prediction = DL_model.predict(X)
            predicted_class = np.argmax(prediction, axis=1)[0]
            prediction_probability = round(max(prediction.tolist()[0]), 2)

            if prediction_probability >= POSTURE_ERROR_THRESHOLD:
                posture = predicted_class

            # Visualization
            # Status box
            cv2.rectangle(image, (0, 0), (500, 40), (245, 117, 16), -1)

            # Display probability
            cv2.putText(image, "RIGHT", (15, 12), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str(right_arm_analysis.counter) if right_arm_analysis.is_visible else "UNK", (10,
30), cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

            # Display Left Counter
            cv2.putText(image, "LEFT", (95, 12), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str(left_arm_analysis.counter) if left_arm_analysis.is_visible else "UNK", (100,
30), cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

            # * Display error
            # Right arm error
            cv2.putText(image, "R_PC", (165, 12), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str(right_arm_analysis.detected_errors["PEAK_CONTRACTION"]), (160, 30),
cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
            cv2.putText(image, "R_LUA", (225, 12), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str(right_arm_analysis.detected_errors["LOOSE_UPPER_ARM"]), (220, 30),
cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

            # Left arm error
            cv2.putText(image, "L_PC", (300, 12), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str(left_arm_analysis.detected_errors["PEAK_CONTRACTION"]), (295, 30),
cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
            cv2.putText(image, "L_LUA", (380, 12), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str(left_arm_analysis.detected_errors["LOOSE_UPPER_ARM"]), (375, 30),
cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)

            # Lean back error
            cv2.putText(image, "LB", (460, 12), cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
            cv2.putText(image, str("C" if posture == 0 else "L") + f" ,{predicted_class},
{prediction_probability}", (440, 30), cv2.FONT_HERSHEY_COMPLEX, 0.3, (255, 255, 255), 1, cv2.LINE_AA)


            # * Visualize angles
            # Visualize LEFT arm calculated angles
            if left_arm_analysis.is_visible:
                cv2.putText(image, str(left_bicep_curl_angle), tuple(np.multiply(left_arm_analysis.elbow,
video_dimensions).astype(int)), cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE_AA)
                cv2.putText(image, str(left_ground_upper_arm_angle), tuple(np.multiply(left_arm_analysis.shoulder,
video_dimensions).astype(int)), cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1, cv2.LINE_AA)


            # Visualize RIGHT arm calculated angles
            if right_arm_analysis.is_visible:
                cv2.putText(image, str(right_bicep_curl_angle), tuple(np.multiply(right_arm_analysis.elbow,
video_dimensions).astype(int)), cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 0), 1, cv2.LINE_AA)
                cv2.putText(image, str(right_ground_upper_arm_angle),
tuple(np.multiply(right_arm_analysis.shoulder, video_dimensions).astype(int)), cv2.FONT_HERSHEY_COMPLEX, 0.5, (255,
255, 0), 1, cv2.LINE_AA)

        except Exception as e:
            print(f"Error: {e}")

        cv2.imshow("CV2", image)

        # Press Q to close cv2 window
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

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

Here are the results of prediction process during building the model for the biceps exercise:

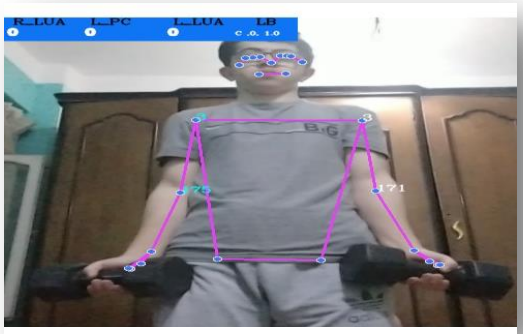


Figure 5.49- Results of prediction process in biceps model

Here are the results expected from every model after building the entire project:

Biceps Exercise:

- Left arm counter
- Right arm counter
- Detected right arm errors and left arm errors
    - ✓ Loose Upper arm counter and image for the error
    - ✓ Weak Peak contraction counter and image for the error
- Lean back percent.

Squat Exercise:

- Ratio of errors to correct movements for knees forward errors.
- Ratio of errors to correct movements for knees inward errors.
- Forward error image.
- Inward error image.
- Left knee counter and right knee counter

Barbell Row Exercise:

- Left and right arm counter
- Lumbar error ratio.
- Torso error ratio.
- Lumbar error image and torso error image.

5.13Evaluation

Once the model has been trained, it is important to evaluate its performance. This involves testing the model on the validation set to determine its accuracy and other metrics, such as precision, recall, and F1 score.

Here how the evaluation process goes down in all three models we built:

- First, we import the required dependencies and packages and load the trained models and the input scaler for the evaluation process:
- Second, we define some important functions such as describe_datatset, round_up_metric_result, categorize labels, and scale features:

```
import pickle
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from yellowbrick.classifier import ROCAUC
from sklearn.metrics import roc_curve, auc
from tensorflow.keras.models import load_model
from keras.utils.np_utils import to_categorical
from sklearn.metrics import precision_score, accuracy_score, f1_score, recall_score, confusion_matrix

warnings.filterwarnings('ignore')

# Load all sklearn models
with open("./model/all_sklearn.pkl", "rb") as f:
    sklearn_models = pickle.load(f)

# Load all deep learning models
three_model = load_model("C:/Users/Alrowad/Exercise-correction/bicpes_model/model/3_layers.h5")
five_model = load_model("C:/Users/Alrowad/Exercise-correction/bicpes_model/model/5_layers.h5")
seven_model = load_model("C:/Users/Alrowad/Exercise-correction/bicpes_model/model/7_layers.h5")
sevenD_model = load_model("C:/Users/Alrowad/Exercise-correction/bicpes_model/model/7_layers_with_dropout.h5")

# Load input scaler
with open("./model/input_scaler.pkl", "rb") as f:
    sc = pickle.load(f)
```

Figure 5.50- Evaluation process in three models

```python
def describe_dataset(dataset_path: str):
    '''
    Describe dataset
    '''

    data = pd.read_csv(dataset_path)
    print(f"Headers: {list(data.columns.values)}")
    print(f'Number of rows: {data.shape[0]} \nNumber of columns: {data.shape[1]}\n')
    print(f"Labels: \n{data['label'].value_counts()}\n")
    print(f"Missing values: {data.isnull().values.any()}\n")

    duplicate = data[data.duplicated()]
    print(f"Duplicate Rows : {len(duplicate.sum(axis=1))}")

    return data


def round_up_metric_results(results) -> list:
    '''Round up metrics results such as precision score, recall score, ...'''
    return list(map(lambda el: round(el, 3), results))

# load dataset
test_df = describe_dataset("./test.csv")

# Categorizing label
test_df.loc[test_df["label"] == "C", "label"] = 0
test_df.loc[test_df["label"] == "L", "label"] = 1

# Standard Scaling of features
test_x = test_df.drop("label", axis = 1)
test_x = pd.DataFrame(sc.transform(test_x))

test_y = test_df["label"].astype('int')

# # Converting prediction to categorical
test_y_cat = to_categorical(test_y)
```

- Then, we evaluate the performance of trained deep learning models on a test dataset. We load each model from a saved path and uses it to predict the output for the test data, then calculate several evaluation metrics including precision, recall, accuracy, and F1 score, as well as the confusion matrix. The evaluation results are stored in a pandas DataFrame for further analysis.

```python
test_set_results = []

model_paths = ["./model/3_layers.h5",
               "./model/5_layers.h5",
               "./model/7_layers.h5",
               "./model/7_layers_with_dropout.h5"]

for path in model_paths:
    model = load_model(path)

    # Evaluate model
    predict_x = model.predict(test_x, verbose=False)
    y_pred_class = np.argmax(predict_x, axis=1)
    y_test_class = np.argmax(test_y_cat, axis=1)

    cm = confusion_matrix(y_test_class, y_pred_class, labels=[0, 1, 2])
    p_score = precision_score(y_test_class, y_pred_class, average="weighted")
    a_score = accuracy_score(y_test_class, y_pred_class)
    r_score = recall_score(y_test_class, y_pred_class, average="weighted")
    f1_score_result = f1_score(y_test_class, y_pred_class, average="weighted")

    test_set_results.append(( path,  p_score, r_score, a_score, f1_score_result, cm ))

dp_eval = pd.DataFrame(test_set_results, columns=["Model Path", "Precision Score", "Recall Score", "Accuracy
Score", "F1 Score", "Confusion Matrix"])

dp_eval
```

Figure 5.51- Store the evaluation results in pandas DataFrame

Here is the explanation of evaluation metrics used in the evaluation process:

- Confusion matrix: A confusion matrix is a table that summarizes the predicted and true class labels for a classification model. It shows the number of true positives, true negatives, false positives, and false negatives, and can be used to calculate other evaluation metrics.
    - **True positives (TP):** The number of samples that are correctly classified as positive (i.e., the model predicts positive and the true label is also positive).
    - **True negatives (TN):** The number of samples that are correctly classified as negative (i.e., the model predicts negative and the true label is also negative).
    - **False positives (FP**): The number of samples that are incorrectly classified as positive (i.e., the model predicts positive but the true label is actually negative).
    - **False negatives (FN):** The number of samples that are incorrectly classified as negative (i.e., the model predicts negative but the true label is actually positive).
- Precision score: Precision is a measure of the proportion of correctly classified positive samples among all samples that were classified as positive. It is calculated as the ratio of true positives to the sum of true positives and false positives.
- Recall score: Recall is a measure of the proportion of correctly classified positive samples among all samples that are actually positive. It is calculated as the ratio of true positives to the sum of true positives and false negatives.

- Accuracy score: Accuracy is a measure of the proportion of correctly classified samples among all samples. It is calculated as the ratio of the number of correct predictions to the total number of predictions.
- F1 score: The F1 score is a weighted average of precision and recall, and provides a single measure of a model's performance. It is calculated as 2 * (precision * recall) / (precision + recall), and takes values between 0 and 1, with higher values indicating better performance.

Here are the results of evaluation metrics of all trained models in every exercise error:

- Biceps Exercise – Lean back Error:

	Model Path	Precision Score	Recall Score	Accuracy Score	F1 Score	Confusion Matrix
0	./model/3_layers.h5	0.691934	0.694511	0.694511	0.693078	[[83, 67, 0], [61, 208, 0], [0, 0, 0]]
1	./model/5_layers.h5	0.783923	0.727924	0.727924	0.733145	[[130, 20, 0], [94, 175, 0], [0, 0, 0]]
2	./model/7_layers.h5	0.804898	0.766110	0.766110	0.770925	[[130, 20, 0], [78, 191, 0], [0, 0, 0]]
3	./model/7_layers_with_dropout.h5	0.756463	0.747017	0.747017	0.750039	[[107, 43, 0], [63, 206, 0], [0, 0, 0]]

Figure 5.52- Biceps Exercise – Lean back Error

- Squat Exercise – Knees Forward Error:

	Model Path	Precision Score	Recall Score	Accuracy Score	F1 Score	Confusion Matrix
0	./model/3_layers.h5	0.602010	0.622970	0.622970	0.607478	[[298, 590, 0], [362, 1275, 0], [0, 0, 0]]
1	./model/5_layers.h5	0.629988	0.631683	0.631683	0.630805	[[414, 474, 0], [456, 1181, 0], [0, 0, 0]]
2	./model/7_layers.h5	0.589884	0.613465	0.613465	0.596088	[[278, 610, 0], [366, 1271, 0], [0, 0, 0]]
3	./model/7_layers_with_dropout.h5	0.608150	0.630891	0.630891	0.612817	[[292, 596, 0], [336, 1301, 0], [0, 0, 0]]

Figure 5.53- Squat Exercise – Knees Forward Error

- Squat Exercise – knees Inward Error:

	Model Path	Precision Score	Recall Score	Accuracy Score	F1 Score	Confusion Matrix
0	./model/3_layers.h5	0.809437	0.781678	0.781678	0.794174	[[2463, 442, 0], [292, 165, 0], [0, 0, 0]]
1	./model/5_layers.h5	0.795888	0.776026	0.776026	0.785347	[[2477, 428, 0], [325, 132, 0], [0, 0, 0]]
2	./model/7_layers.h5	0.798234	0.773052	0.773052	0.784659	[[2458, 447, 0], [316, 141, 0], [0, 0, 0]]
3	./model/7_layers_with_dropout.h5	0.799332	0.792980	0.792980	0.796083	[[2540, 365, 0], [331, 126, 0], [0, 0, 0]]

Figure 5.54- Squat Exercise – knees Inward Error

- Barbell Row Exercise – Torso Error:

	Model Path	Precision Score	Recall Score	Accuracy Score	F1 Score	Confusion Matrix
0	./model/3_layers_torso.h5	0.932213	0.935076	0.935076	0.933452	[[2161, 68, 0], [91, 129, 0], [0, 0, 0]]
1	./model/5_layers_torso.h5	0.917976	0.928134	0.928134	0.919361	[[2187, 42, 0], [134, 86, 0], [0, 0, 0]]
2	./model/7_layers_torso.h5	0.915594	0.926092	0.926092	0.917692	[[2182, 47, 0], [134, 86, 0], [0, 0, 0]]
3	./model/7_layers_with_dropout_torso.h5	0.929763	0.935484	0.935484	0.931533	[[2176, 53, 0], [105, 115, 0], [0, 0, 0]]

Figure 5.55- Barbell Row Exercise – Torso Error

- Barbell Row Exercise – Lumbar Error:

	Model Path	Precision Score	Recall Score	Accuracy Score	F1 Score	Confusion Matrix
0	./model/3_layers_lumbar.h5	0.862959	0.863131	0.863131	0.863045	[[1525, 135, 0], [136, 184, 0], [0, 0, 0]]
1	./model/5_layers_lumbar.h5	0.869608	0.877273	0.877273	0.872309	[[1567, 93, 0], [150, 170, 0], [0, 0, 0]]
2	./model/7_layers_lumbar.h5	0.872175	0.873737	0.873737	0.872925	[[1540, 120, 0], [130, 190, 0], [0, 0, 0]]
3	./model/7_layers_with_dropout_lumbar.h5	0.880833	0.884848	0.884848	0.882532	[[1561, 99, 0], [129, 191, 0], [0, 0, 0]]

Figure 5.56- Barbell Row Exercise – Lumbar Error

- The next step is to make predictions on a test dataset. The predicted values are converted to discrete labels using a threshold of 0.5, and the precision score, recall score, and F1 score are calculated for the predicted labels compared to the true labels. The resulting scores are printed.

```
best_model01 = seven_model
y_predictions = best_model01.predict(test_x)

# convert continuous values to discrete labels using a threshold of 0.5
y_predictions_labels = (y_predictions >= 0.5).argmax(axis=-1)

p_score = precision_score(test_y, y_predictions_labels, labels=[0, 1], average=None)
r_score = recall_score(test_y, y_predictions_labels, labels=[0, 1], average=None)
f1_score_result = f1_score(test_y, y_predictions_labels, labels=[0, 1], average=None)

p_score, r_score, f1_score_result

(0.726 + 0.796) / 2
```

Figure 5.57- Test data predictions

Here is the result of the previous code for biceps exercise explained:

```
(array([0.625     , 0.90521327]),
 array([0.86666667, 0.71003717]),
 array([0.72625698, 0.79583333]))

(0.726 + 0.796) / 2

0.761
```

Figure 5.58- Result of biceps exercise

- p_score: The precision score is an array with two values, one for each class label (0 and 1). The first value (0.625) represents the precision for label 0, and the second value (0.90521327) represents the precision for label 1.
- r_score: The recall score is an array with two values, one for each class label (0 and 1). The first value (0.86666667) represents the recall for label 0, and the second value (0.71003717) represents the recall for label 1.
- f1_score_result: The F1 score is an array with two values, one for each class label (0 and 1). The first value (0.72625698) represents the F1 score for label 0, and the second value (0.79583333) represents the F1 score for label 1.
- To get an overall measure of performance for the model, we can calculate the weighted average of the F1 score for both classes. This is done by multiplying each F1 score by the number of samples in each class, summing the results, and dividing by the total number of samples. In this case, assuming that both classes have the same number of samples, the calculation would be:
    - (0.726 + 0.796) / 2 = 0.761. This means that the model has an average F1 score of 0.761, which is a measure of its overall performance on the test dataset.

Note: The processes are the same for all models so we will get enough with one model for code figures.

Here is the result of prediction code for knees forward error explained:

```
(array([0.47586207, 0.71359517]),
 array([0.46621622, 0.72144166]),
 array([0.47098976, 0.71749696]))

(0.470 + 0.717) / 2

0.5934999999999999
```

Figure 5.59- Result of prediction code for knees forward error

- p_score: The precision score is an array with two values, one for each class label (0 and 1). The first value (0.47586207) represents the precision for label 0, and the second value (0.71359517) represents the precision for label 1.
- r_score: The recall score is an array with two values, one for each class label (0 and 1). The first value (0.46621622) represents the recall for label 0, and the second value (0.72144166) represents the recall for label 1.
- f1_score_result: The F1 score is an array with two values, one for each class label (0 and 1). The first value (0.47098976) represents the F1 score for label 0, and the second value (0.71749696) represents the F1 score for label 1.
- To get an overall measure of performance for the model, we can calculate the weighted average of the F1 score for both classes. This is done by multiplying each F1 score by the number of samples in each class, summing the results, and dividing by the total number of samples. Assuming that both classes have the same number of samples, the calculation would be:
    - (0.470 + 0.717) / 2 = 0.5935. This means that the model has an average F1 score of 0.5935, which is a measure of its overall performance on the test dataset.

Here is the result of prediction code for knees inward error explained:

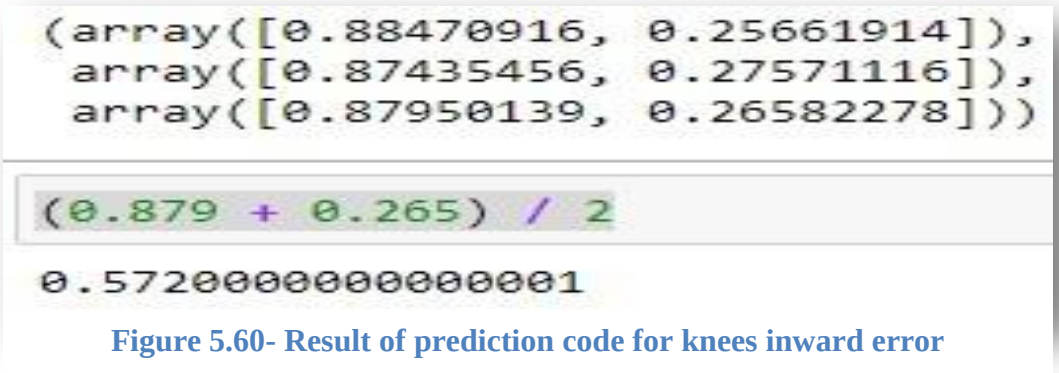


Figure 5.60- Result of prediction code for knees inward error

- p_score: The precision score is an array with two values, one for each class label (0 and 1). The first value (0.88470916) represents the precision for label 0, and the second value (0. 25661914) represents the precision for label 1.
- r_score: The recall score is an array with two values, one for each class label (0 and 1). The first value (0. 87435456) represents the recall for label 0, and the second value (0. 27571116) represents the recall for label 1.
- f1_score_result: The F1 score is an array with two values, one for each class label (0 and 1). The first value (0. 87950139) represents the F1 score for label 0, and the second value (0. 26582278) represents the F1 score for label 1.
- To get an overall measure of performance for the model, we can calculate the weighted average of the F1 score for both classes. This is done by multiplying each F1 score by the number of samples in each class, summing the results, and dividing by the total number of samples. Assuming that both classes have the same number of samples, the calculation would be:
    - (0.879 + 0.265) / 2= 0.5720000000000001. This means that the model has an average F1 score of 0.5720000000000001, which is a measure of its overall performance on the test dataset.

Here is the result of prediction code for torso error explained:

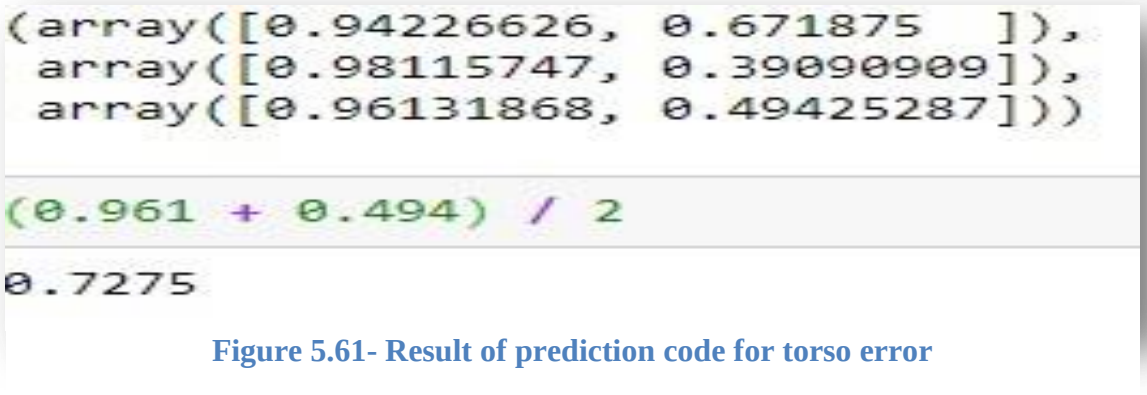


Figure 5.61- Result of prediction code for torso error

- p_score: The precision score is an array with two values, one for each class label (0 and 1). The first value (0.94226626) represents the precision for label 0, and the second value (0.671875) represents the precision for label 1.
- r_score: The recall score is an array with two values, one for each class label (0 and 1). The first value (0.98115747) represents the recall for label 0, and the second value (0.39090909) represents the recall for label 1.
- f1_score_result: The F1 score is an array with two values, one for each class label (0 and 1). The first value (0.96131868) represents the F1 score for label 0, and the second value (0.49425287) represents the F1 score for label 1.
- To get an overall measure of performance for the model, we can calculate the weighted average of the F1 score for both classes. This is done by multiplying each F1 score by the number of samples in each class, summing the results, and dividing by the total number of samples. Assuming that both classes have the same number of samples, the calculation would be:
    - (0.961 + 0.494) / 2 = 0.7275. This means that the model has an average F1 score of 0.7275, which is a measure of its overall performance on the test dataset.

Here is the result of prediction code for lumbar error explained:

```
(array([0.91532258, 0.70901639]),
 array([0.95722892, 0.540625  ]),
 array([0.93580683, 0.61347518]))

(0.936 + 0.613) / 2

0.7745
```

Figure 5.62- Result of prediction code for lumbar error

- p_score: The precision score is an array with two values, one for each class label (0 and 1). The first value (0.91532258) represents the precision for label 0, and the second value (0.70901639) represents the precision for label 1.

- r_score: The recall score is an array with two values, one for each class label (0 and 1). The first value (0.95722892) represents the recall for label 0, and the second value (0.540625) represents the recall for label 1.
- f1_score_result: The F1 score is an array with two values, one for each class label (0 and 1). The first value (0.93580683) represents the F1 score for label 0, and the second value (0.61347518) represents the F1 score for label 1.
- To get an overall measure of performance for the model, we can calculate the weighted average of the F1 score for both classes. This is done by multiplying each F1 score by the number of samples in each class, summing the results, and dividing by the total number of samples. Assuming that both classes have the same number of samples, the calculation would be:
    - (0.936 + 0.613) / 2= 0.7745. This means that the model has an average F1 score of 0.7745, which is a measure of its overall performance on the test dataset.

The next step after making prediction on test set is to draw confusion matrix:

```
deep_cm = eval_df[ eval_df["Model Path"] == './model/7_layers.h5' ]["Confusion Matrix"].values[0]

cm_array_df = pd.DataFrame(deep_cm, index=["C", "L", "None"], columns=["C", "L", "None"])

fig, ax = plt.subplots(figsize=(8,4))
sns.heatmap(cm_array_df, linewidths=1, annot=True, ax=ax, fmt='g')
```

Figure 5.63- Drawing confusion matrix

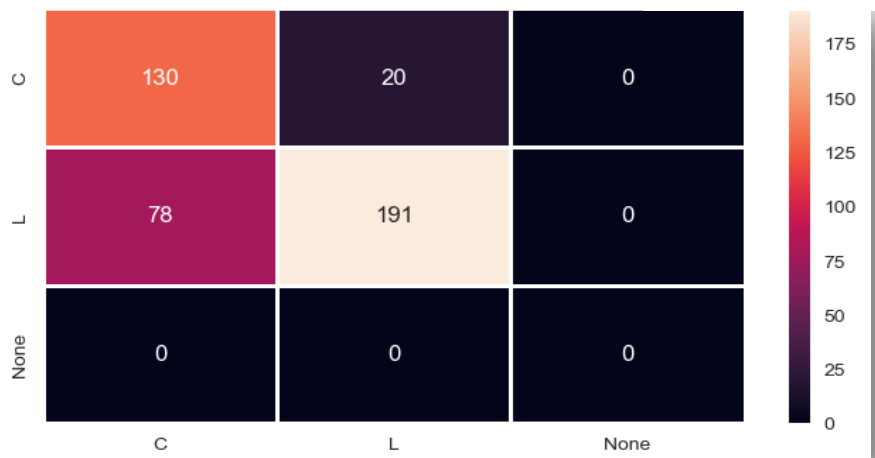


Figure 5.64- Confusion matrix for biceps model

Here is the confusion matrix for biceps model:

Confusion Matrix of Knees Forward error:

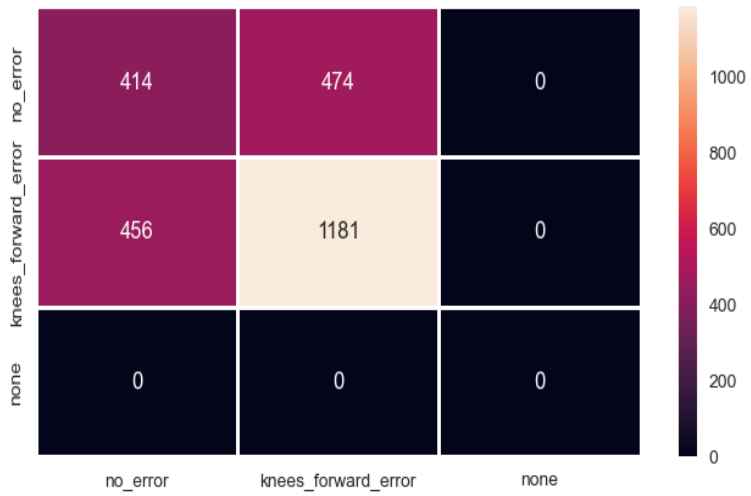


Figure 5.65- Confusion Matrix of knees forward error

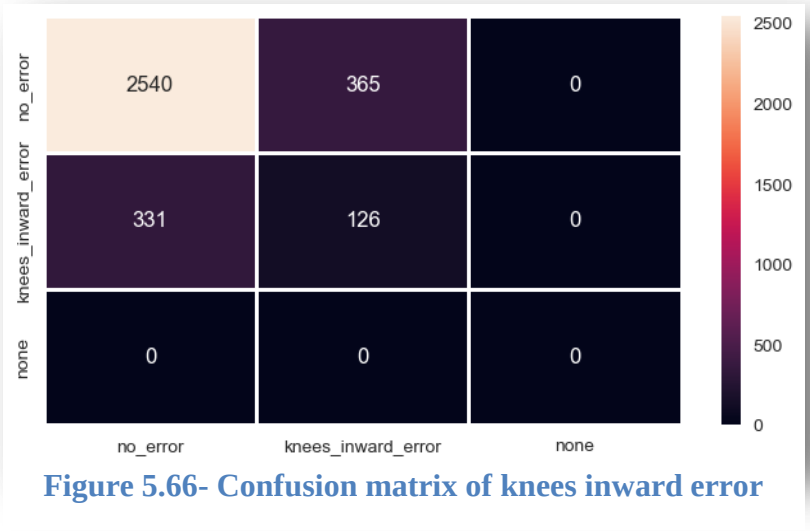


Figure 5.66- Confusion matrix of knees inward error

Confusion Matrix of Knees Inward Error:

Confusion Matrix of Lumbar Error:

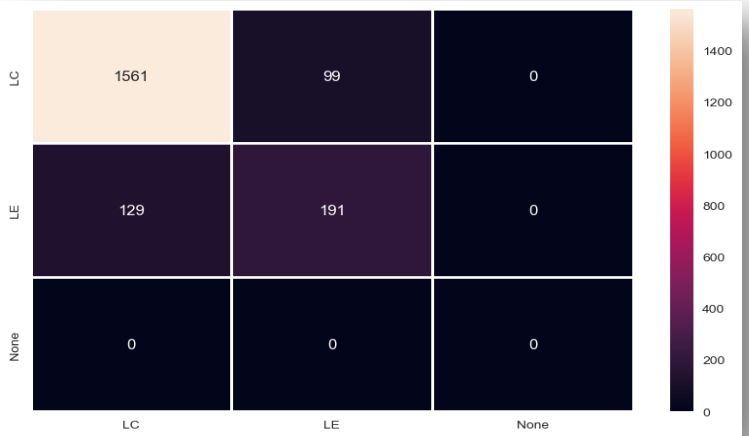


Figure 5.67- Confusion matrix of knees inward error

Confusion Matrix of Torso Error:

Figure 5.68- Confusion matrix of torso error

The next step is to define two functions: to labels and calculate_correlation_score_confidence. The former returns predicted labels, taking into account a confidence threshold, while the latter predicts labels using a pre-trained model and calculates F1 scores for each class label and weighted F1 scores for different predicted label sets obtained using different confidence thresholds.

```python
def to_labels(y_pred, y_pred_proba, threshold):
    '''Return prediction taking confidence threshold into account'''
    results = []

    for index, predicted_class in enumerate(y_pred):
        prediction_probabilities = y_pred_proba[index]
        class_prediction_probability = round(prediction_probabilities[np.argmax(prediction_probabilities)], 2)

        results.append(predicted_class if class_prediction_probability >= threshold else -1)

    return results


def calculate_correlation_score_confidence(test_x, test_y):
    '''Calculate correlation between Precision score/Recall score/F1 score and confidence threshold'''
    y_predictions = best_model.predict(test_x)
    y_predict_proba = best_model.predict_proba(test_x)

    thresholds = list(np.arange(0, 1.01, 0.01))

    f1_score_results = []

    for threshold in thresholds:
        true_predictions = to_labels(y_predictions, y_predict_proba, threshold)
        f1_s = list(f1_score(test_y, true_predictions, labels=[0, 1], average=None))
        all_class_f1 = f1_score(test_y, true_predictions, labels=[0, 1, 2], average="weighted")
        f1_s.append(all_class_f1)
        f1_score_results.append(f1_s)

    return thresholds, f1_score_results
```

Figure 5.69- Labels and calculate_correlation_score_confidence

Here is the F1 Scores plot for Lean back error:

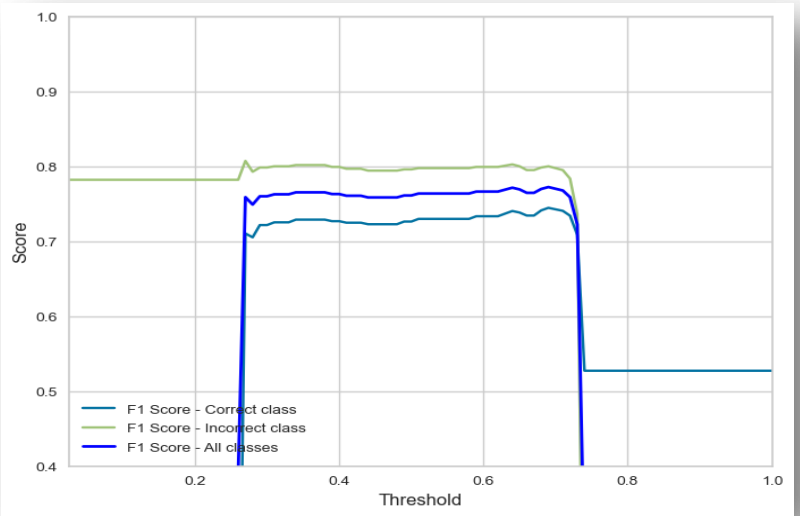


Figure 5.70- F1 score plot for lean back error

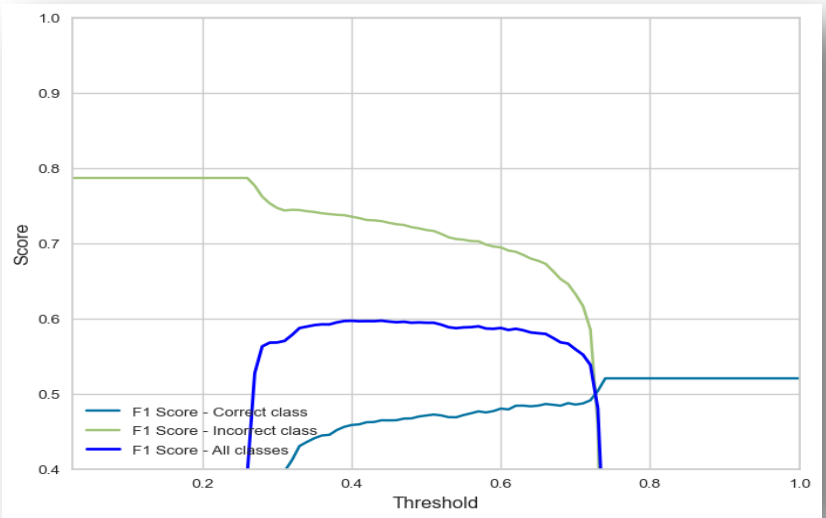


Figure 5.70- F1 score for knees forward error

Here is the F1 Scores plot for Knees Forward error:

Here is the F1 Scores plot for Knees Inward error:

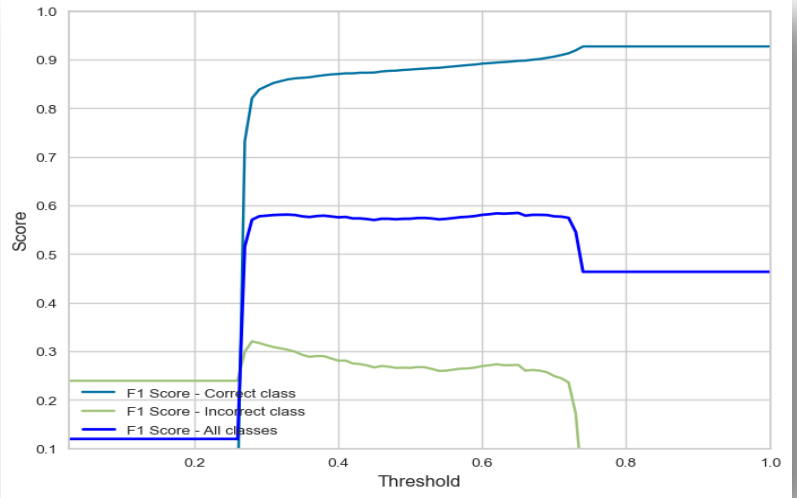


Figure 5.71- F1 score plot for knees inward error

Here is the F1 Scores plot for Lumbar error:

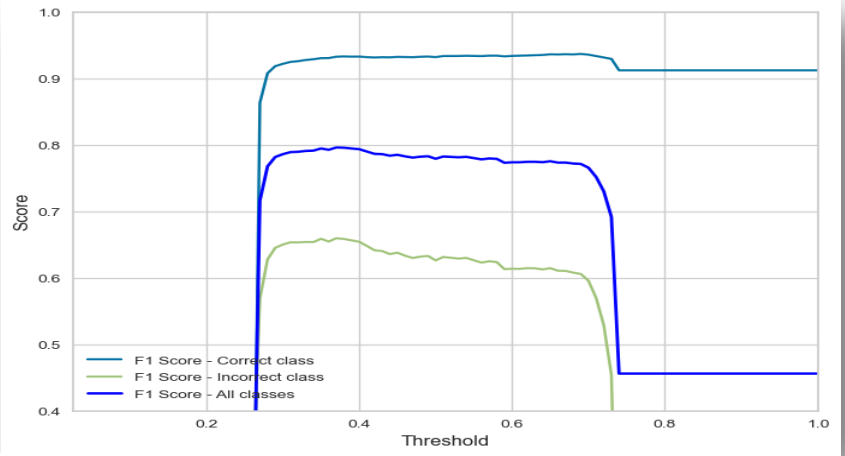
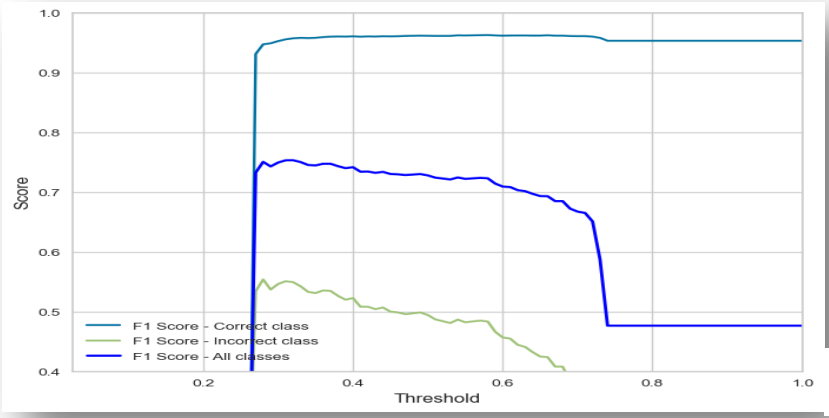


Figure 5.72- F1 score plot for lumbar error



Here is the F1 Scores plot for Torso error:

Figure 5.73- F1 score plot for torso error

ROC Curve:

A ROC (Receiver Operating Characteristic) curve is a graphical representation of the performance of a binary classification model at different classification thresholds.

The ROC curve plots the true positive rate (TPR) or sensitivity (i.e. the proportion of positive instances that are correctly classified as positive) against the false positive rate (FPR) or 1-specificity (i.e. the proportion of negative instances that are incorrectly classified as positive) at different

The final step is calculating the Receiver Operating Characteristic (ROC) curve for a binary classification model best_model01 using the roc_curve() function from the sklearn.metrics module. The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) for different classification thresholds. The area under the curve (AUC) is also calculated using the auc () function from the same module.

```
# calculate the fpr and tpr for all thresholds of the classification
probs = best_model01.predict(test_x)
preds = probs[:,1]
fpr, tpr, threshold = roc_curve(test_y, preds)
roc_auc = auc(fpr, tpr)

optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = threshold[optimal_idx]
print(f"Optimal Threshold: {optimal_threshold}")

# method I: plt
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.plot([0, 1], [0, 1],'r--', label="Ra")
plt.legend(loc=4)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Figure 5.74- ROC curve for binary classification model

Here is the ROC Curve Plot for Lean back error:

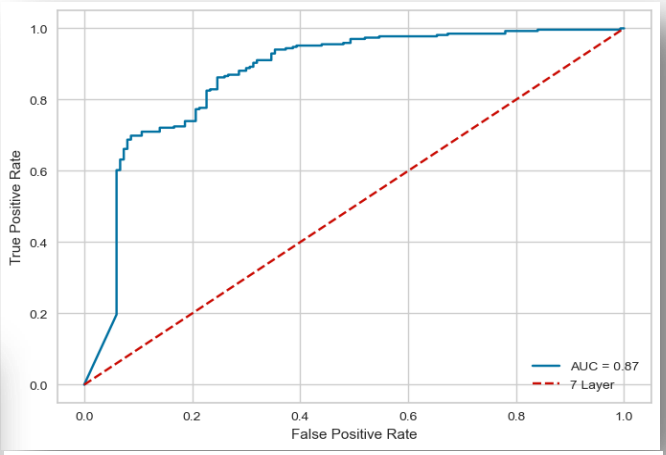


Figure 5.75- ROC curve plot for lean back error

Here is the ROC Curve Plot for Knees Forward error:

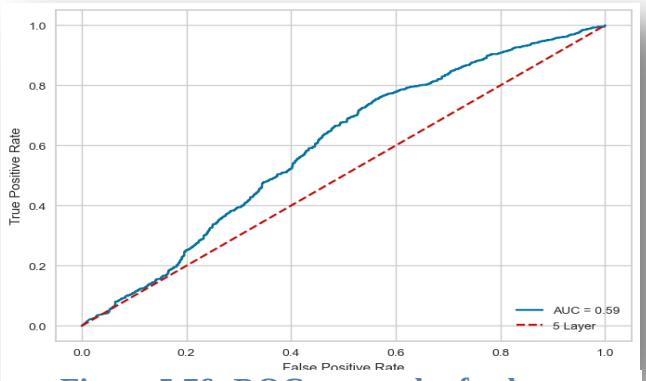


Figure 5.76- ROC curve plot for knees forward error

Here is the ROC Curve Plot for Knees Inward error:

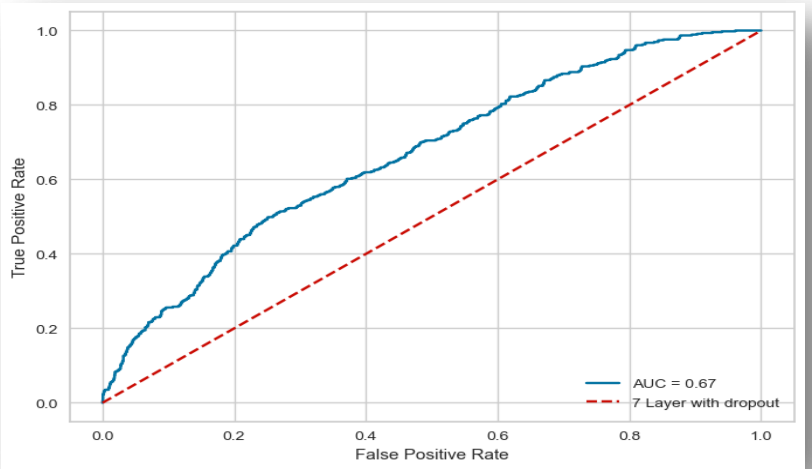


Figure 5.77- ROC curve plot for knees inward error

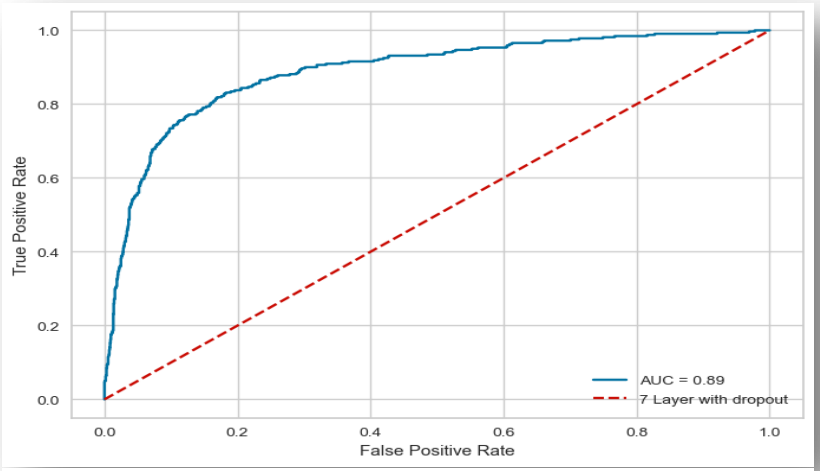


Figure 5.78- ROC curve plot for lumbar error

Here is the ROC Curve Plot for Lumbar error:

Here is the ROC Curve Plot for Torso error:

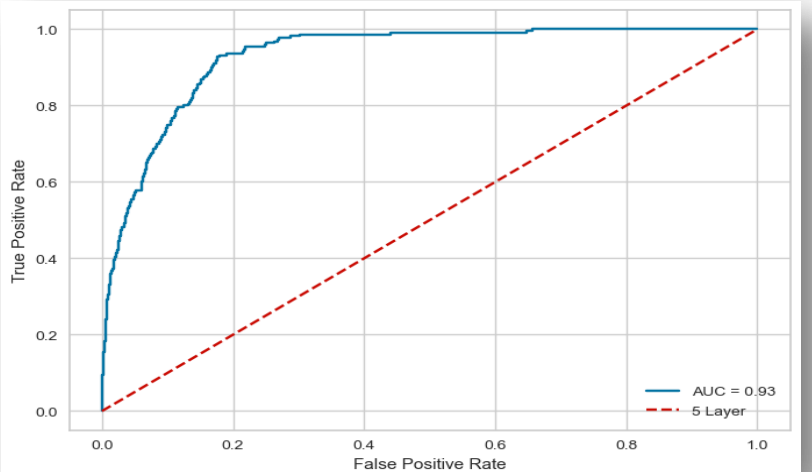


Figure 5.79- ROC curve plot for torso error

Chapter 6

Software Testing

6.1 Introduction

- **Purpose of the document:** To report the results of the testing phase for GymEye system.
- **Scope:** All test cases executed during the testing phase.

6.2 Summary of Results

Overall results of the testing phase:

- ✓ Number of test cases executed: 8
- ✓ Number of test cases passed: 7
- ✓ Number of test cases failed: 1
- ✓ Number of test cases blocked: 0

Summary of issues and defects identified during the testing phase:

The number of defects was identified during the testing phase related to GymEye application

6.3 Detailed Test Results

Test Case ID	Description	Preconditions	Expected Result	Actual Result	Status	Comments
1	Verify that a new user can register and create an account successfully	None	The new user should be able to register and create an account successfully	Pass	Closed.	No issues were found during this test case.

2	Verify that existing users can log in to their accounts with the correct credentials	A user account has been created.	The existing user should be able to log in to their account with the correct credentials.	Pass	Closed.	No issues were found during this test case.
3	Verify that the system prevents unauthorized access to restricted pages.	A user account has been created.	The system should prevent unauthorized access to restricted pages.	Pass	Closed.	No issues were found during this test case.
4	Verify that the system can handle multiple logins attempts and block users after a certain number of failed attempts.	A user account has been created.	The system should be able to handle multiple logins attempts and block users after a certain number of failed attempts.	**Failed**	Open.	The system prevents the user to login if he doesn't enter the correct data, but it doesn't block him.
5	**Verify that the user can choose a specific**	The user is logged in to the system.	The user should be able to select	Pass	Closed.	No issues were found during this test case.

	exercise and view the results of the exercise		specific exercise to train and show result.			
	Make sure that the video does not exceed a certain length	The user is logged in to the system and has started an exercise.	**The system should prevent user from uploading long video**	Pass	Closed.	No issues were found during this test case.
7	Verify that the system can handle different types of exercise	The user is logged in to the system and has started an exercise.	The system should provide different type of exercise.	Pass	Closed.	No issues were found during this test case.
8	Verify that the system is able to evaluate a person's exercise once the video has been uploaded	The user is logged in to the system and has started an exercise.	The system should automatically send the results as soon as the video is uploaded	**pass**	closed.	The video content might be another content instead of required exercise.

Table 6.1- Test Results

6.4 Conclusion

Summary of the testing phase and results:

7 out of 8 test cases passed.

Recommendations for further improvements or modifications to the system:

- ✓ Fix the defects identified during the testing phase.
- ✓ Add additional test cases to cover more scenarios and edge cases.
- ✓ Use different testing approaches and test data to avoid Pesticide Paradox.
- ✓ Early testing.

Chapter 7

Conclusion and Future Improvements

7.1 Conclusion

In this project, we managed to build five models for five different errors or poses in three workouts which are squat exercise, barbell row exercise, and biceps exercise. We used feedforward neural network to build 4 different architectures with different layers and test each of them on the dataset to check which one will give best results to use then in prediction process. We also managed to detect some errors by calculating the angles of landmarks or body positions required to recognize these errors and calculate counters for the three exercises. The features were extracted using Mediapipe pose estimation and processed for training process using some tuning operations that prepare data well for deep learning training process. We managed also to provide images and descriptions that clarify the nature of the errors done by users and provide some articles that will be helpful for the user. Also, there is a BMI calculator that can help the user organize his body nutrients and manage to build a well fitted body.

7.2 Future improvements

In coming versions in future, there are some improvements to be considered:

- New exercises will be added with all common and possible errors happen while performing them.
- We will add and describe common injuries and pains happen due to wrong poses and ignoring right instructions.
- We might add categories for exercises such as categories for each muscle trainings such as chest trainings, back, triceps, biceps and so on.
- Also, there might be categories for type of exercises like cardio or weightlifting.

- There might be some plans which are common between gyms for beginners to follow.
- We will use better techniques for building and training models like CNN (Convolutional Neural Network), RNN (Recurrent Neural Network) and LSTM (Long Short – Term Memory).
- We will use better datasets or might try to build our own dataset that contains most correct and wrong poses in most exercises
- We will try to add methods for health monitoring and suggesting better solutions on how to take care of your health based on these monitoring results.

References

References

1. Benaim, S., Ephrat, A., Lang, O., Mosseri, I., Freeman, W.T., Rubinstein, M., Irani, M., Dekel, T.: Speednet: Learning the speediness in videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9922–9931 (2020)

2. Cao, Z., Hidalgo, G., Simon, T., Wei, S.E., Sheikh, Y.: Openpose: realtime multiperson 2d pose estimation using part affinity fields. IEEE transactions on pattern analysis and machine intelligence 43(1), 172–186 (2019)

3. Chen, S., Yang, R.R.: Pose trainer: correcting exercise posture using pose estimation. arXiv preprint arXiv:2006.11718 (2020)

4. Chen, X., Pang, A., Yang, W., Ma, Y., Xu, L., Yu, J.: Sportscap: Monocular 3d human motion capture and fine-grained understanding in challenging sports videos. arXiv preprint arXiv:2104.11452 (2021)]

5. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15750–15758 (2021)

6. Chopra, S., Hadsell, R., LeCun, Y.: Learning a similarity metric discriminatively, with application to face verification. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). vol. 1, pp. 539–546. IEEE (2005)

7. Doughty, H., Mayol-Cuevas, W., Damen, D.: The pros and cons: Rank-aware temporal attention for skill determination in long videos. In: Proceedings of the

IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7862–7871 (2019)

8. Du, C., Graham, S., Depp, C., Nguyen, T.: Assessing physical rehabilitation exercises using graph convolutional network with self-supervised regularization. In: 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). pp. 281–285. IEEE (2021)

9. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv:1803.07728 (2018)

10. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet? In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 6546–6555 (2018)

11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

12. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. science 313(5786), 504–507 (2006)

13. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: International workshop on similarity-based pattern recognition. pp. 84–92. Springer (2015)

14. Honari, S., Constantin, V., Rhodin, H., Salzmann, M., Fua, P.: Unsupervised learning on monocular videos for 3d human pose estimation. arXiv preprint arXiv:2012.01511 (2020)

15. Hyvarinen, A., Morioka, H.: Unsupervised feature extraction by time-contrastive learning and nonlinear ica. Advances in Neural Information Processing Systems 29, 3765–3773 (2016)

16. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(7), 1325–1339 (jul 2014)

17. Jenni, S., Meishvili, G., Favaro, P.: Video representation learning by recognizing temporal transformations. In: European Conference on Computer Vision. pp. 425– 442. Springer (2020)

18. Jing, L., Yang, X., Liu, J., Tian, Y.: Self-supervised spatiotemporal feature learning via video rotation prediction. arXiv preprint arXiv:1811.11387 (2018)

19. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: Computer Vision and Pattern Regognition (CVPR) (2018)

20. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al.: The kinetics human action video dataset. arXiv preprint arXiv:1705.06950 (2017)

21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

22. Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In: ICCV (2019)

23. Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y.: Learning hierarchical invariant spatiotemporal features for action recognition with independent subspace analysis. In: CVPR 2011. pp. 3361–3368. IEEE (2011)

24. Li, J., Bhat, A., Barmaki, R.: Improving the movement synchrony estimation with action quality assessment in children play therapy. In: Proceedings of the 2021 International Conference on Multimodal Interaction. pp. 397–406 (2021)

25. Liu, D., Li, Q., Jiang, T., Wang, Y., Miao, R., Shan, F., Li, Z.: Towards unified surgical skill assessment. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9522–9531 (2021)

26. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: unsupervised learning using temporal order verification. In: European Conference on Computer Vision. pp. 527–544. Springer (2016)

27. Ogata, R., Simo-Serra, E., Iizuka, S., Ishikawa, H.: Temporal distance matrices for squat classification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)

28. Pan, J.H., Gao, J., Zheng, W.S.: Action assessment by joint relation graphs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)

29. Park, T., Zhu, J.Y., Wang, O., Lu, J., Shechtman, E., Efros, A., Zhang, R.: Swapping autoencoder for deep image manipulation. Advances in Neural Information Processing Systems 33, 7198–7211 (2020)

30. Parmar, P., Morris, B.: Action quality assessment across multiple actions. In: 2019 IEEE winter conference on applications of computer vision (WACV). pp. 1468– 1476. IEEE (2019)

31. Parmar, P., Morris, B.T.: Measuring the quality of exercises. In: 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). pp. 2241–2244. IEEE (2016)

32. Parmar, P., Reddy, J., Morris, B.: Piano skills assessment. arXiv preprint arXiv:2101.04884 (2021)

33. Parmar, P., Tran Morris, B.: Learning to score olympic events. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 20–28 (2017)

34. Parmar, P., Tran Morris, B.: What and how well you performed? a multitask learning approach to action quality assessment. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 304–313 (2019)

35. Pirsiavash, H., Vondrick, C., Torralba, A.: Assessing the quality of actions. In: European Conference on Computer Vision. pp. 556–571. Springer (2014)

36. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)

37. Rhodin, H., Salzmann, M., Fua, P.: Unsupervised geometry-aware representation for 3d human pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 750–767 (2018)

38. Roditakis, K., Makris, A., Argyros, A.: Towards improved and interpretable action quality assessment with self-supervised alignment. In: The 14th PErvasive Technologies Related to Assistive Environments Conference. pp. 507–513 (2021)

39. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual

recognition challenge. International journal of computer vision 115(3), 211–252 (2015)

40. Ryali, C.K., Schwab, D.J., Morcos, A.S.: Characterizing and improving the robustness of self-supervised learning through background augmentations. arXiv preprint arXiv:2103.12719 (2021)

41. Sardari, F., Paiement, A., Hannuna, S., Mirmehdi, M.: Vi-net—view-invariant quality of human movement assessment. Sensors 20(18), 5258 (2020)

42. Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., Brain, G.: Time-contrastive networks: Self-supervised learning from video. In: 2018 IEEE international conference on robotics and automation (ICRA). pp. 1134–1141. IEEE (2018)

43. Sigurdsson, G.A., Gupta, A., Schmid, C., Farhadi, A., Alahari, K.: Actor and observer: Joint modeling of first and third-person videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7396–7404 (2018)

44. Tang, Y., Ni, Z., Zhou, J., Zhang, D., Lu, J., Wu, Y., Zhou, J.: Uncertainty-aware score distribution learning for action quality assessment. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9839– 9848 (2020)

45. Tao, L., Paiement, A., Damen, D., Mirmehdi, M., Hannuna, S., Camplani, M., Burghardt, T., Craddock, I.: A comparative study of pose representation and dynamics modelling for online motion quality assessment. Computer vision and image understanding 148, 136–152 (2016)

46. Wang, J., Jiao, J., Bao, L., He, S., Liu, Y., Liu, W.: Self-supervised spatio-temporal representation learning for videos by predicting motion and appearance statistics. In: CVPR. pp. 4006–4015 (2019)

47. Wang, J., Jiao, J., Liu, Y.H.: Self-supervised video representation learning by pace prediction. In: European conference on computer vision. pp. 504–521. Springer (2020)

48. Wang, T., Wang, Y., Li, M.: Towards accurate and interpretable surgical skill assessment: A video-based method incorporating recognized surgical gestures and skill levels. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 668–678. Springer (2020)

49. Xu, C., Fu, Y., Zhang, B., Chen, Z., Jiang, Y.G., Xue, X.: Learning to score figure skating sport videos. IEEE transactions on circuits and systems for video technology 30(12), 4578–4590 (2019)

50. Xu, D., Xiao, J., Zhao, Z., Shao, J., Xie, D., Zhuang, Y.: Self-supervised spatiotemporal learning via video clip order prediction. In: Computer Vision and Pattern Recognition (CVPR) (2019)

51. Yu, X., Rao, Y., Zhao, W., Lu, J., Zhou, J.: Group-aware contrastive regression for action quality assessment. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7919–7928 (2021)

52. Zeng, L.A., Hong, F.T., Zheng, W.S., Yu, Q.Z., Zeng, W., Wang, Y.W., Lai, J.H.: Hybrid dynamic-static context-aware attention network for action assessment in long videos. In: Proceedings of the 28th ACM International Conference on Multimedia. pp. 2526–2534 (2020)

53. Y. Zhu, C. Tang, H. Liu and P. Huang, "End-Face Localization and Segmentation of Steel Bar Based on Convolution Neural Network," in IEEE Access, vol. 8, pp. 74679-74690, 2020, doi: 10.1109/ACCESS.2020.2989300.

54. Wen Chen, Yiping Gao, Liang Gao, Xinyu Li, A New Ensemble Approach based on Deep Convolutional Neural Networks for Steel Surface Defect classification,Procedia CIRP,Volume 72,2018,Pages 1069-1072,ISSN 2212-8271

55. F. A. Saiz, I. Serrano, I. Barandiarán and J. R. Sánchez, "A Robust and Fast Deep Learning-Based Method for Defect Classification in Steel Surfaces," 2018 International Conference on Intelligent Systems (IS), Funchal - Madeira, Portugal, 2018, pp. 455-460, doi: 10.1109/IS.2018.8710501.