Multimodal Interaction

Pages 1–10
a.a. 2022-2023

AI Gym Tracker

Alessandro Di Patria,
Alberto Cotumaccio

Sapienza University of Rome

2^{nd} July 2023

Keywords: Gesture Interaction, Pose Estimation

1 INTRODUCTION

The AI Gym Tracker project represents an innovative web application that harnesses the power of artificial intelligence to track and monitor users' physical activities during their workout sessions. Its main goal is to provide an intuitive and user-friendly user interface that allows users to perform their physical workouts easily, obtaining detailed information about their performance in real time.

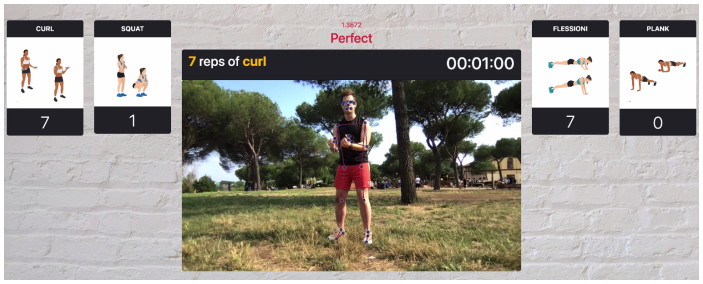

Figure 1. Main usage of AI Gym Tracker application

Physical exercises can be beneficial to the fitness of the human body, but they can also be very harmful if performed incorrectly and too quickly (8). This can lead to a number of risks, including muscle injury, joint overload, and reduced training effectiveness. It is essential to perform exercises with control, focusing on proper technique and appropriate muscle contraction. Correct form and proper rhythm ensure better results and reduce the risk of injury. Accordingly, with this project we also wish to help users perform the exercises with the correct timing to prevent injury, simply using a computer and a webcam as support.

The primary goal of the application is to provide users with an interactive and highly motivating training environment. Using cutting-edge technologies such as artificial intelligence, computer vision for video analysis, and pose recognition, the application is able to count exercise repetitions in real time, while also providing precise and accurate feedback on the speed of execution of poses.

To perform action prediction and process the stream of frames from the webcam, we developed and trained an accurate artificial intelligence model based on recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) neural networks. We created this model through a data collection phase during which we recorded a considerable number of videos of correctly performed exercises, thus creating an accurate custom training dataset. In this way through the use of the powerful Mediapipe library, we extracted the poses performed by the user in real time, determining the joints as a list of skeletal key points of the body.

A distinctive aspect of the application is its flexibility. Users have full control over the duration of training, being able to tailor sessions to their personal needs and goals. At the end of each session, the application provides users with a complete overview of the exercises performed during the workout. In addition, an element of great interest to users concerns the value of calories burned, and we have devoted extensive scientific studies to the development of an accurate formula that takes into account the user's biographical data and their activity performed during the workout. This aspect allows users to have a reliable estimate of the calories consumed, contributing to a better understanding of the results obtained.

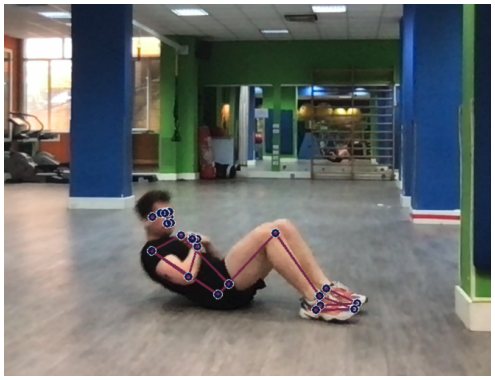

Figure 2. Usage example of AI Gym Tracker

2 MULTIMODALITY

A multimodal web application is an interactive system that integrates different input and output modalities in order to allow users to interact with it in different ways. These modes may include voice input, gestural input, textual input and visual, auditory or tactile output. The main objective of a multimodal web application is to offer a complete and customised user experience, allowing users to use the interaction mode they prefer or that best suits their needs.

2.1 Gestural interaction

The application we developed offers a unique experience thanks to its multi-modality, which allows users to perform physical exercises using the webcam. This innovative mode uses gesture interaction to understand and interpret the movements of the user's entire body. During exercise, the system carefully analyses and interprets the gestures provided as input by users. This multimodal approach creates an interactive and immersive environment, allowing for precise and personalised training. We employed advanced machine learning techniques and, during the training process, the system learned with great precision to recognise possible exercises that are performed during training. This allowed us to create a highly sophisticated and accurate model that, in real time, is able to interpret the movements captured by the webcam and associate them with the specific exercises the user is performing.

2.2 Enabling Technologies (input)

No physical device or object is therefore required in the application to perform the desired movements. Our interface is based on an innovative gestural interaction with perceptual input, which allows users to control the application through intuitive movements and gestures performed in the surrounding space, without the need for any physical contact. This non-physical contact interaction approach offers numerous advantages, including greater convenience and freedom of movement for users, eliminating the need for additional devices or tools.

2.3 System response (output)

We also felt it was of paramount importance to optimally manage the system response, i.e. how the system reacts during user training. During the training phase, users are assumed to be standing at a certain distance from the webcam. Therefore, we considered it crucial to provide audio feedback that alerts users by means of appropriate sounds. For example, a sound is played when an exercise is performed perfectly, or a countdown at the beginning, or even a specific sound effect to signal incorrect user positioning.The sounds were edited and partly realised using music production software (1). However, this does not reduce the importance of the visual output on the screen, which we preferred to keep as simple as possible, displaying information such as the exercise count and timer in a prominent manner.

3 BACKGROUND

In this section, we will provide a broad overview of the technologies adopted during project development. We will start with the Mediapipe library, which provides advanced capabilities for pose detection and skeletal landmarks extraction. Using this data, we are able to obtain accurate information about users' posture during training sessions. Then we will discuss machine learning frameworks such as Tensorflow and Keras that allow us to create and train an advanced model based on recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) neural networks. This sophisticated model analyzes the user's landmarks in real time to make a prediction of the exercise performed. Finally, we will delve into the technology behind our web application. This was built through Python programming language and the Flask library for the back-end side while for the front-end side we used HTML in combination with Javascript and Bootstrap. This combination of technologies allowed us to create an attractive and user-friendly interface.

3.1 Mediapipe

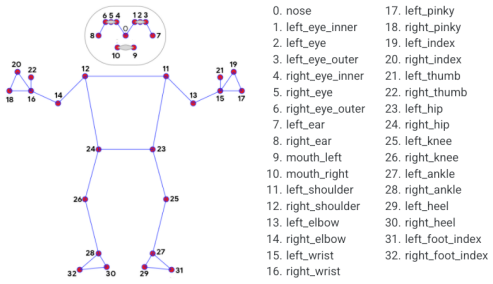

Figure 3. 33 pose landmarks

Mediapipe is an open source library developed by Google Research (15) that provides a flexible and powerful framework for creating machine learning-based multimedia applications. Its main goal is to enable efficient, real-time processing of multimedia data, including video and audio, opening the door to a wide range of innovative applications.

In our project, we exploited one of Mediapipe's strengths, which lies in its ability to analyze video in real time, enabling fast and accurate processing of data from streaming video sources. Using advanced computer vision techniques, Mediapipe can detect and track objects, faces, hands, and other features within videos, enabling a detailed understanding of the actions, movements, and interactions present.

In particular, with regard to human pose detection, Mediapipe is distinguished by the extraction of 33 key points, as shown in Figure 3. The extraction of these landmarks allows precise localization and tracking of the human body in the video, enabling detailed analysis of the movements and poses assumed. Each landmark consists of the following (12):

- **x and y**: landmark coordinates normalized to [0.0, 1.0] for the width and height of the image, respectively.
- **z**: represents the depth of the reference point with the depth at the midpoint of the hips as the origin, and the smaller the value, the closer the reference point is to the camera. The magnitude of z uses approximately the same scale as x.
- **visibility**: a value in [0.0, 1.0] indicating the probability that the reference point is visible (present and not occluded) in the image.

The use of 33 key points for pose detection gives Mediapipe a great ability to understand and interpret the dynamics of the human body in the context of the analyzed video. This is especially useful in applications such as ours, which require accurate motion tracking, such as action recognition, pose-based augmented reality, and the creation of virtual interactions.

Mediapipe's versatility also extends to its ability to handle real-time analysis, enabling immediate response to information extracted from videos. This enables the creation of interactive and responsive applications that can make use of real-time data captured from key points in the pose. This feature is especially valuable for applications such as interactive games, real-time video filters, and surveillance systems based on motion detection.

3.2 Tensorflow - Keras

TensorFlow is an open source machine learning library (6) developed by Google. It is designed to facilitate the creation and training of artificial neural networks on large datasets. TensorFlow offers a wide range of tools and features for creating machine learning models, including optimization algorithms, data visualization tools and parallel processing functions. The framework supports both the definition and training of deep neural networks and the definition of supervised and unsupervised learning algorithms. TensorFlow uses a symbolic programming approach, in which models are defined as computational graphs, allowing greater flexibility and control over the training process.

Keras, on the other hand, is a high-level machine learning library (9) that functions as an application programming interface (API) for TensorFlow. It is designed to simplify the creation and training of neural networks while minimizing the complexity of writing code. Keras provides a set of abstractions that make it easy to define model structure, specify layers and links between them, and configure hyperparameters and optimization algorithms. One of the main features of Keras is its ease of use, making it suitable for both beginners and experts. You can use Keras to create a wide range of machine learning models, including convolutional neural networks, generative neural networks and recurrent neural networks, the latter being our case.

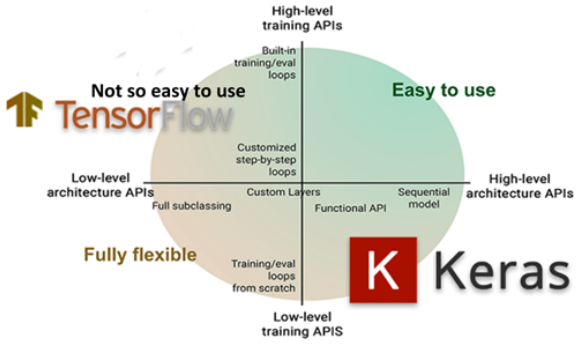

Figure 4. Keras and Tensorflow overview

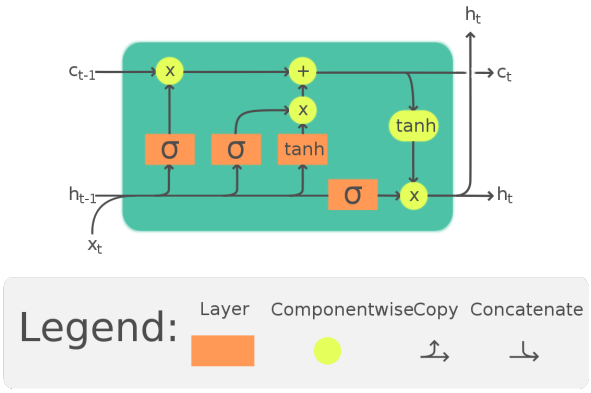

Figure 5. Long short-term memory (LSTM)

3.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs) are two fundamental approaches in the field of neural networks for modeling data sequences.

RNNs Are a type of neural network architecture that introduces the concept of feedback memory (16). Unlike traditional neural networks, in which input and output are treated as independent data sets, RNNs maintain an internal state that retains information about the context of previous observations. This internal state is then used to guide the prediction of the current output. The ability of RNNs to model data sequences, such as natural language or time series, makes them particularly suitable for many applications, such as speech recognition, machine translation and text generation.

However, traditional RNNs suffer from a problem known as the ”Vanishing gradient problem” in which gradients used for learning fade as they are propagated backward through time steps. This makes it difficult for RNNs to maintain long-term memory of relevant information, thus limiting their modeling capabilities. To overcome this problem, long-term memory neural networks (LSTMs) have been introduced.

LSTMs Are a type of RNNs that use a structure of specialized memory units, called ”memory cells” to address the problem of gradient disappearance (13). Memory cells allow LSTMs to maintain and manipulate information over time through controlled storage and forgetting mechanisms. This allows LSTMs to learn long-term dependencies within data sequences, making them particularly effective in modeling complex, long-term relationships.

LSTMs consist of three main components:

- **memory cell**, which stores information over time;
- **input unit**, which regulates the update of the memory cell;
- **an output unit**, which generates the current output based on the internal state of the memory cell

This architecture allows LSTMs to effectively capture long-term dependencies, making them a popular choice for applications that require complex sequence modeling, such as natural language recognition, natural language processing, and text generation.

3.4 Calories formula

In our project, we provide users with statistics on calories burned during exercise sessions. The calculation of calories burned is done accurately, taking into account various factors such as the individual’s gender, weight, and type of exercise performed. To obtain this information, we use the concept of Metabolic Equivalent of Task (MET), which allows us to determine the calories burned while performing different types of physical exercise (7).

MET represents a numerical value that reflects the ratio of a person’s metabolic rate at rest to the metabolic rate while performing a specific physical activity. This value differs according to the sex of the individual, with a value of 4.5 for men and 3.5 for women. Using MET values provides us with a sound scientific basis for estimating caloric consumption during exercise. This approach can be applied generically to estimate energy expended in a variety of physical activities, ensuring a reliable estimate of total caloric consumption.

In addition to the use of MET values, in our approach to calculating calories, we assume that 25 repetitions of an exercise corresponds to the number of calories burned in 1 minute. This assumption allows us to estimate calorie consumption based on the number of repetitions performed for each exercise. Applying a simple proportion, we can calculate calories burned based on the number of repetitions.

$$\sum_{\text{reps}\in\text{Exercises}} \frac{(MET \times 3.5 \times \text{kg} \times \text{reps})}{200 \times 25}$$

3.5 Flask

Flask is a lightweight framework (3) for developing web applications in Python. It is designed to be simple and easy to use, allowing web applications to be created quickly without having to deal with excessive complexity. Flask follows the ”micro” principle and focuses on simplicity and modularity.

One of the main features of Flask is its flexibility. It does not impose a rigid structure or specific architecture for web applications, allowing us to choose the components that best suit our needs. This makes it possible to develop customized and scalable web applications efficiently.

Flask offers a routing system that associates URL requests with function handlers, and this allowed us to easily define different pages and their actions. It also supports the use of templates for dynamic HTML page generation, simplifying the separation of backend logic and presentation.

Another important aspect of Flask is extensibility. Additional functionality can be easily added to the application through the use of extensions. Flask has a wide range of extensions available, covering areas

such as authentication, databases, integration with other Web services, and more.

3.6 Html, Javascript, Bootstrap

The combined use of HTML, JavaScript and Bootstrap has provided us with a solid foundation for creating interactive, aesthetically pleasing and responsive web pages. HTML defines the content structure, JavaScript adds interactive features, and Bootstrap simplifies development with predefined components and consistent styles. This combination of technologies is widely used in modern web development and enables the creation of high-quality user experiences.

HTML (HyperText Markup Language) is a markup language (4) used for structuring and presenting content on the Web. It provides a set of elements and attributes that help define the logical structure of a web page. HTML elements are used to represent different types of content, such as text, images, tables and forms. Attributes provide additional information about the elements and allow their characteristics and behaviors to be specified.

JavaScript is a high-level scripting language (11) supported by most modern browsers that adds interactivity and dynamism to Web pages. We used it to manipulate HTML content by providing it with dynamism, realizing the training session timer. It also allows asynchronous json calls to the server, and this allows us to update in real time the repetitions of the exercises performed by the users.

Bootstrap is an open-source front-end framework for creating responsive and mobile-first web pages (2). It provides a set of predefined components, CSS styles, and JavaScript scripts that helped us simplify web development. With Bootstrap, we could use components such as predefined buttons, forms, and navigation bars, and easily apply responsive styles to adapt web pages to different screen sizes.

4 REQUIREMENTS AND DESIGN

In the context of this final report on the AIGymTracker application, it is critical to examine the requirements needed to ensure an optimal user experience. The application is designed to provide users with intuitive workout exercise tracking and support during physical activity through automatic pose recognition and the use of performance indicators, such as exercise counting, feedback on execution speed, and final calculation of calories burned. In order to achieve these goals, the functional and non-functional requirements of the application are identified below:

Functional requirements

1. **Personal Data Entry**: The application requires users to enter their weight, height, and gender. This data will be used to customize the calculation of statistics and obtain more accurate results.
2. **Automatic pose recognition**: The application must implement a pose recognition algorithm that is reliable, fast, and able to correctly identify different poses during gymnastics exercises.
3. **Real-time exercise tracking**: The application must track the details of exercise sessions in real time, including the number of repetitions performed for each type of exercise, duration of sessions, calories burned, accuracy of running times, and other relevant metrics. This will enable users to monitor progress and evaluate performance.
4. **Calorie Calculation**: The application should provide an accurate calculation of calories burned during exercise, based on metrics such as body metrics and session duration.
5. **Sending statistics via email**: At the end of an exercise session, users can choose to receive a report of statistics via email.

Nonfunctional requirements

1. **High-quality camera**: to ensure accurate pose recognition, the application requires the use of a good-quality camera capable of capturing detailed images of users during gymnastics exercises.
2. **Stable Internet connection**: a stable Internet connection is required while using the application.
3. **Accessibility to a web browser**: the application requires the use of a compatible and up-to-date web browser on the user's device to ensure maximum compatibility and optimal performance.
4. **Ease of use and intuitiveness**: the user interface of the app should be easy to use, with an intuitive design that does not require users to consult complex instructions or manuals. The app should also be accessible to users with little technical experience.

4.1 Use case

In this section we provide an analysis of the use cases of the AIGymTracker application in order to clearly and comprehensively present the interactions between the actors and the system. The main usage scenarios of the application and their use case diagrams will be illustrated. Through the analysis of use cases, it will be possible to understand how AIGymTracker meets users' needs and offers intuitive functionality for monitoring gymnastics exercises. Interaction flows between users and the application will be examined, highlighting actions, system responses, and possible exceptions. Use case diagrams will provide a clear visualization of the relationships between actors and application functionality, offering a visual representation of interactions and possible sequences of actions for users. Use case analysis is an essential tool for understanding how AIGymTracker supports users in monitoring gymnastics exercises by providing a detailed overview of the various functions available.

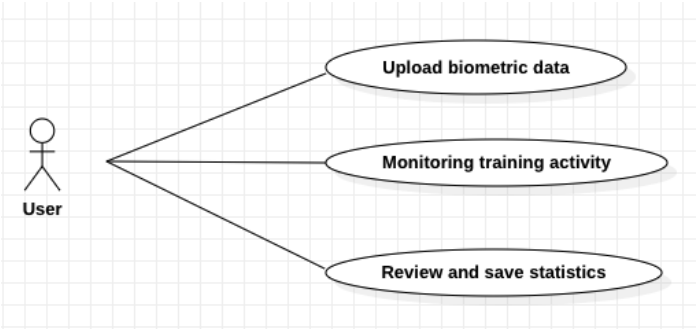

Figure 6. Uml use cases diagram

- **Upload biometric data**: The system must allow the user to upload their biometric data in order to achieve maximum accuracy during the training process.
- **Monitoring training activity**: The user must be able to view in real time all essential parameters for monitoring training activity. These parameters include the number of repetitions for each exercise, the type of exercise being performed, and the speed at which the exercises are being performed.
- **Review and save statistics**: The application must allow the user, at the end of the exercise session, to view and save the overall statistics and calories burned during the activity by sending them via email.

4.2 Sequence diagrams

In the following section we are going to present the sequence diagrams for each usa case of the application.

- **Upload biometric data**: In this activity, there are mainly three entities involved: the user, the upload system, and the application system. The user completes the form by entering his/her biometric data to start the training activity. The upload system guides the user while filling out the form. Next, the biometric data and email address are sent to the application system, which verifies the accuracy of the information provided by the user. If the information entered is valid, the user can start the training. If not, the application system sends an error message to the GUI, which is displayed to the user in the form section.

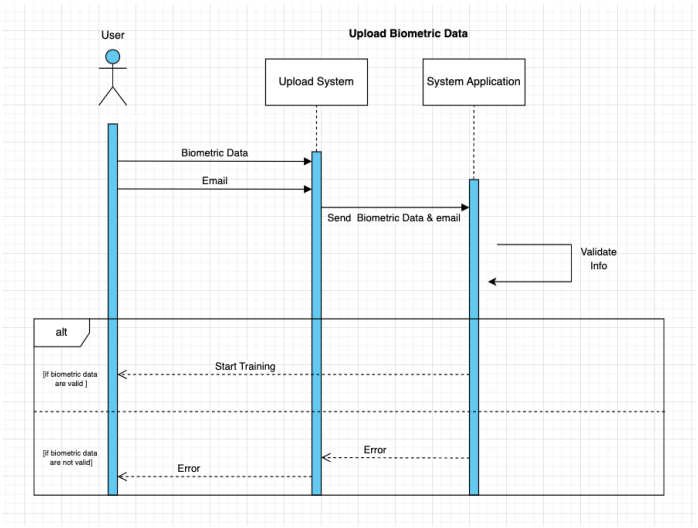

Figure 7. Sequence diagram for Upload biometric data

- **Monitoring training activity**: Three main entities are involved in this activity: the user, the GUI, and the training system. The user initiates the training session using the GUI, which displays information about the status of the exercises. The training system is responsible for recognizing the user's action.

  The training process starts when the user starts the activity and the training system receives the body landmarks sent by the GUI. The training system verifies that all body parts involved in the pose are fully visible. If they are visible, the system predicts the user's exercise, updates the exercise status and sends the predicted exercise to the GUI. Otherwise, if any body part is not visible, the system sends an error message to the GUI.

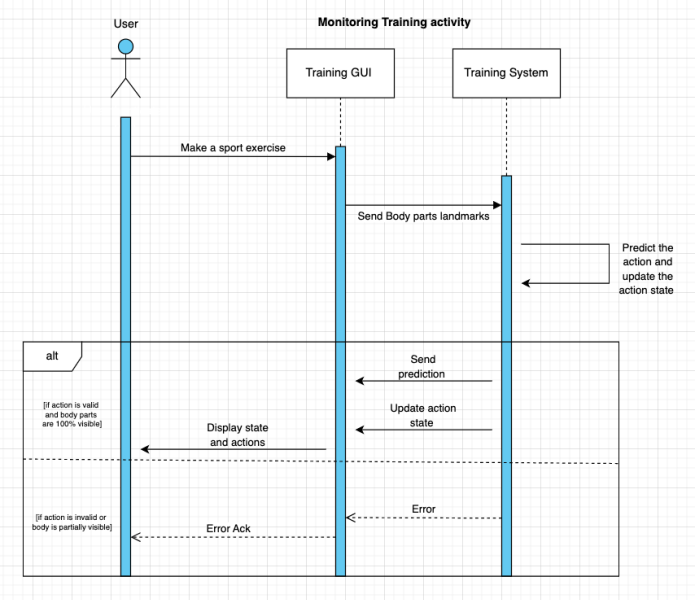

Figure 8. Sequence diagram for monitoring training activity

4.3 Scenario

The following actions help to understand in more detail how the whole application works:

- The user uploads his or her biometric data into a special form.
- The user then starts the training session and can monitor the parameters for each exercise in real time.
- When the user finishes the session, he/she views the workout statistics and can decide whether to save them via an email.

4.4 Prototyping

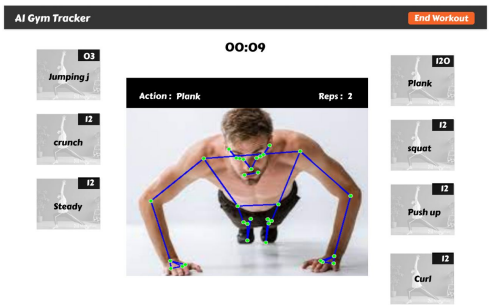

Figure 9. training page prototype

The prototyping phase has been a key element in the development of the web application we are designing. Its main purpose has been to create a visual representation of the application's functionality and user interface, allowing us to assess the look and flow of the user experience before moving on to the actual development phase. This approach allowed us to identify and resolve any usability and interaction issues quickly and efficiently.

To create the prototypes of our three main screens, we used the Figma tool. Figma is a cloud-based user interface (UI) design and prototyping tool that is widely used in the field of digital design (10).

Figma offered us a collaborative and flexible design environment, allowing us to create interactive and realistic interfaces. With its advanced features, we were able to define page layouts, place graphic elements, test different design options, and simulate the flow of navigation between screens.

The importance of this prototyping phase cannot be underestimated. Through the prototypes, we were able to visualize and evaluate the look and flow of the user interface, ensuring an intuitive and pleasant user experience (14). In addition, it allowed us to identify and solve various usability and interaction issues before final implementation, thus reducing development time.

5 IMPLEMENTATION

In this section, we will provide a detailed description of the implementation of the application. In particular, we will explain how the data were collected to train the model and present the training process itself. Next, we will evaluate the trained model using different techniques. In addition, we will explain in detail the approach taken to count the repetitions of each exercise. Finally, we will present the underlying technology of our web application in a clear and understandable way.

5.1 Data explanation and collection

Data collection is a key aspect of the proper training of a machine learning model. In particular, to ensure good performance of the model, it is necessary to have a large amount of high-quality data. In our project, we have the special feature of having created the data in-house, thus ensuring total control over its quality and consistency. Our data collection mainly includes landmarks of photos and videos related to 4 different poses: push-ups,plank, curl, squat and steady(null). The choice of these poses is based on their ease of execution and their importance to physical and muscular well-being.

5.1.1 Data Collection To collect pose landmarks, we created a script in Python. This script uses the MediaPipe libraries to analyze the data in real time and save it. Specifically, the pipeline of our script can be summarized as follows:

- **Setup** We create a folder for each action we want to detect. Within these folders, numbered subfolders will be created, depending on the desired number of repetitions for each action. In our case, we have 60 repetitions for each action, so we will have a total of 60 folders. Within each folder, .npy files containing the landmarks of the poses will be placed. The number of files will depend on the frames of each video (for example, if a video has 30 frames, we will have 30 files since a video is a sequence of images). This methodology allows us to create a labeled dataset since we accurately place the corresponding action in each folder.

Figure 10. The labelled dataset structure

- **Execute the script** The script takes advantage of the Mediapipe library to locate the landmarks of the pose captured by the webcam in real time. Then, these landmarks are saved in numpy files within the specific folder associated with the executed action and the repetition number. In order to avoid physical overload during data collection, a short interval between exercises was included to allow for a short rest period.

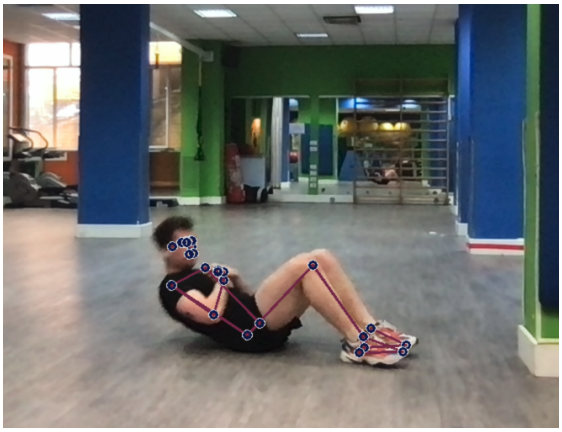

Figure 11. Process of collecting data

5.2 Model Training

Once we have collected all the necessary data, we will employ them in the model training process. As anticipated earlier, the model we will use is an RNN, specifically we will use LSTM (Long Short-Term Memory). This type of model is widely recognized for its ability to handle sequential data and capture long-term dependencies. LSTM networks were developed to overcome the problem of gradient disappearance or explosion, which can occur in traditional recurrent neural networks.

Our model is structured with three LSTM layers, which enable the algorithm to process complex sequences and maintain long-term memory of relevant information. These LSTM layers are followed by densely connected layers, which add additional linear processing capabilities. In particular, the first three layers use the ReLU (Rectified Linear Unit) activation function on the outputs, which allows the introduction of nonlinearities into the model and the learning of complex relationships between input data and desired outputs.

Regarding the last two layers, we chose to use the softmax activation function. This function is commonly used in cases where a multiclass classification problem is addressed. The softmax function allows us to obtain a probability distribution over several classes, assigning each class a value between 0 and 1. This allows us to determine which action or class is most likely based on the characteristics of the input data.

As for the expected input for our model, it has a form of (30, 132). This representation corresponds to the frames of a video and 132 different landmarks parameters. Landmarks are landmarks within an image or sequence of images that represent specific locations of interest. In our case, we have 32 landmarks, each of which is represented by 4 dimensions. These landmarks parameters provide crucial information for identifying and understanding the actions performed.

Once we have completed training the model, we will export its weights to an H5 file. This will allow us to use the trained model in our application to make predictions and classify different actions efficiently and accurately.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 30, 64)            50432

 lstm_4 (LSTM)               (None, 30, 128)           98816

 lstm_5 (LSTM)               (None, 64)                49408

 dense_3 (Dense)             (None, 64)                4160

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 4)                 132

=================================================================
Total params: 205,028
Trainable params: 205,028
Non-trainable params: 0
_________________________________________________________________
```

Figure 12. The structure of a multiclassification model

5.3 Model tuning

Hyperparameter optimization, also known as hyperparameter tuning, represents the process of finding the optimal values for the hyperparameters of a machine learning model in order to improve its performance.

Hyperparameters are model parameters that are not learned during the training phase, but must be specified in advance by the user. In the specific case of the LSTM model used, there are several hyperparameters that could benefit from tuning in order to achieve optimal performance. In the present analysis, we considered the following hyperparameters:

- **Activation** : Activation functions are used to introduce nonlinearities into the LSTM network. The activation function 'ReLU' (Rectified Linear Unit) is commonly adopted, however, other options such as 'tanh' or 'sigmoid' should be explored. The choice of activation function depends on the problem under consideration and the nature of the data.
- **Batch Size**: The batch size, or batch size, determines the number of training samples used in each training step. Using smaller values can result in greater variability in gradient estimates, but also requires more training time.
- **Learning rate**: This hyperparameter defines the rate at which the network updates its parameters. Setting a higher learning rate speeds up the learning process, but it may compromise model convergence (i.e., when the loss stabilizes within an error range around the final value) or even cause the model to diverge.

```
param_grid = {
    'batch_size': [16, 32, 64],
    'activation': ['relu', 'sigmoid'],
    'epochs': [300, 500]
}
```

Figure 13. Parameters for tune the model

The training result shows that the optimal parameters are :
batch size:64, activation function:sigmoid, epochs:500

5.4 Model Evaluation

Evaluation, or assessment, in a machine learning model is a key step in the model development and evaluation process. Its purpose is to measure the performance and quality of the model, based on specific metrics, in order to understand the extent to which the model is able to generalize data and make accurate predictions about new data not used during the training phase.

In the context of a multiclass machine learning model, evaluation is concerned with assessing the performance of the model in classifying more than two classes. In other words, the model is trained to assign an input instance to one of several possible classes. The evaluation can be conducted by using a separate test dataset, which contains samples that were not used during the training phase of the model. In this way, it is possible to evaluate the performance of the model on independent data and measure its ability to generalize. Most of the classification metrics are defined by default for binary cases. To extend these binary metrics to multiclass, several mediation techniques are used. First, a multiclass problem is decomposed into a series of binary problems using the One-vs-One (OVO) or One-vs-Rest (OVR, also called One-vs-All) approach. In essence, the One-vs-Rest strategy converts a multiclass problem into a series of binary tasks for each target class. There are several evaluation metrics that can be used to evaluate the performance of a model. Among the most common metrics are the following:

- **Accuracy**: Measures the percentage of input instances that were classified correctly relative to the total number of instances. However, accuracy may not be representative when classes are unbalanced.
- **Precision**: Measures the proportion of instances classified correctly for a class relative to all instances classified as belonging to that class. This metric is particularly useful when the goal is to minimize false positives.
- **Recall**: Measures the proportion of instances correctly classified for a class compared to all instances actually belonging to that class. This metric is particularly useful when the goal is to minimize false negatives.

  Through the Classification Report we are able to extrapolate for each class the la precision and recall.

	precision	recall	f1-score	support
curl	0.72	0.93	0.81	14
squat	1.00	0.83	0.91	12
push-up	0.92	1.00	0.96	11
null	0.88	0.64	0.74	11
accuracy			0.85	48
macro avg	0.88	0.85	0.85	48
weighted avg	0.87	0.85	0.85	48

Figure 14. Accuracy, precision and recall for each single classes

The results show accurate classification by the model for all classes, achieving an overall accuracy of about 85%.

- **Confusion matrix**: The confusion matrix illustrates the number of instances that were classified correctly or incorrectly for each class. This representation provides a more detailed view of model performance for each class.

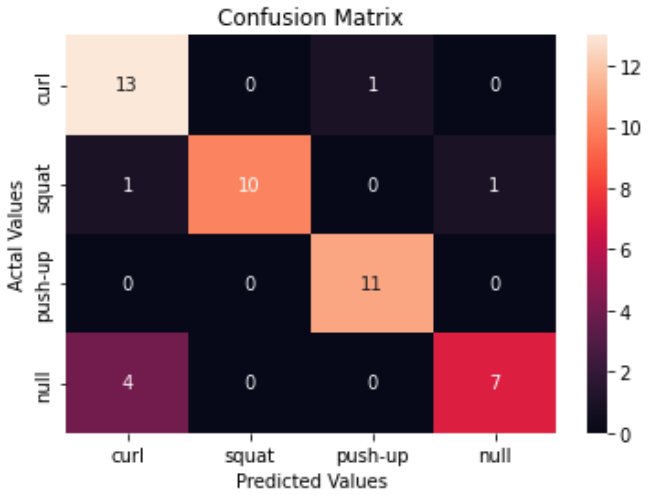

Figure 15. Confusion matrix for the model

The confusion matrix indicates that in most cases the submitted instances were classified correctly. However, in some cases, the action ”Null/Steady” is not classified 100% correctly and is mistakenly mistaken for another action, such as ”curl”.

- **ROC**: The ROC (Receiver Operating Characteristic) curve is a graph widely used in performance evaluation of binary classification models. It represents the relationship between the sensitivity (or True Positive Rate) and specificity (or True Negative Rate) of the model as the classification threshold changes. To better understand the ROC curve, it is necessary to explain the concepts of sensitivity and specificity. Sensitivity measures the model’s ability to correctly identify positive instances, that is, how many times the model correctly classifies an instance as positive compared to the total number of actual positive instances. It is calculated as: Sensitivity = True Positive Rate / (True Positives + False Negatives) Specificity, on the other hand, measures the model’s ability to correctly identify negative instances, that is, how many times the model correctly classifies an instance as negative compared to the total number of actual negative instances. It is calculated as: Specificity = True Negative Rate = True Negatives / (True Negatives + False Positives) The ROC curve is constructed by plotting the True Positive Rate against the False Positive Rate (1 - Specificity)

The results show that for each class, the model has high sensitivity (ability to correctly identify positive cases) and high specificity (ability to correctly identify negative cases), as shown in the figures above left. This represents a satisfactory result.

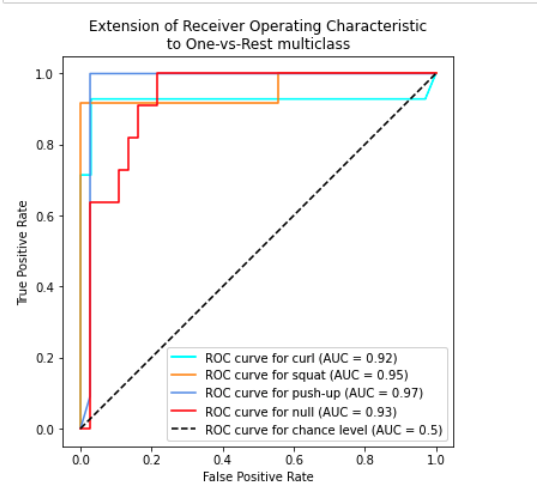

Figure 16. Roc curve for every different class

5.5 Counting Algorithm

Once the ability to correctly estimate a person’s pose has been obtained, it becomes critical to be able to quantify the number of repetitions performed. Through a series of extensive studies, we were able to identify an efficient solution that takes advantage of the angles formed by the joints.

5.5.1 ***compute angle*** We implemented a function that performs the angle calculation by providing three landmarks as input, representing the joints involved in the calculation. These landmarks are treated as three-dimensional vectors to determine the direction and orientation of the motion. To calculate the desired angle, the method of arcotangents is used, which returns the angle between two vectors in the plane. Using this technique, the angle between the vectors formed by (b, c) and (b, a) is calculated. Next, the angle between (b, a) is subtracted from the angle between (b, c) to obtain the desired angle. This value is initially expressed in radians. To represent the angle in degrees, we multiply the value in radians by the conversion factor 180.0 / pi, and take its modulus. Finally, if the calculated angle exceeds 180 degrees, it is converted to an angle between 0 and 180 degrees by subtracting it from 360.

It is important to note that the implementation is based on fundamental mathematical concepts and does not depend directly on the programming language used.

5.5.2 ***Counter Algorithm*** When calculating repetitions, we proceed by estimating the angle formed between the specific joints involved in the exercise movement.

During the detection of an exercise, the concept of ”stage” is introduced, which represents a binary value with two possible states: ”down” and ”up” corresponding to the two phases of the exercise. We use a pair of independent values for each exercise, in order to avoid errors between transitions between different exercises.

The ”down” stage indicates the start of the exercise by the user, while the ”up” stage occurs when the user completes the exercise. Only at the latter time, if the user has performed the stages in the correct order, do we increment the counter related to the exercise in question. It is critical, in fact, to go through both states to ensure the correct execution of the exercise. We can determine the execution time of the exercise easily by calculating the difference between the two timestamps of the two ordered stages.

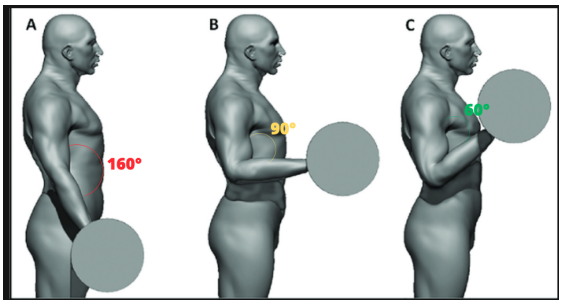

Figure 17. Example of angle calculation for the Curl exercise

5.5.3 ***Speed execution Classification*** The execution time obtained from the algorithm will then be necessary for the classification of the speed of the various actions. In order to obtain more clarity regarding the actual time taken to complete the various exercises, we decided to conduct several field tests in order to obtain a dataset collecting the various execution speeds for each exercise. We then used this data as intervals to classify the execution time into different categories: Perfect, Good, Too Fast.

5.6 Web Application

In this section we describe the web application we developed using the Flask framework, based on the Python programming language, and web pages created with HTML and Bootstrap. The structure of the application includes a Python file named ”app.py,” which represents the core of the application. In this file, the backend of the different HTML pages, the flow management between pages, and the real-time counting algorithm are defined.

The landing page for the platform is the homepage, which consists of a carefully curated HTML page. Here, a brief explanation is given on how the portal works and the main essential information that users need to know before using it. For example, it is indicated that it is necessary to have sufficient space around oneself to perform movements. An important element of this page is the form that users must manually fill out by entering their data (such as email address, age, height, weight, and gender) to start a session. It is important to note that the email address is a mandatory field, as we want to associate it with each training session. The same is true for weight and gender, as these are necessary data for the calculation of calories burned at the end of the workout. Each field in the form is validated to avoid errors and inconsistencies in the data.

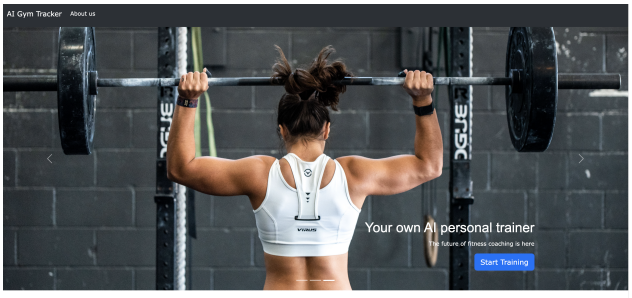

Figure 18. Landing page of AI Gym trainer

If the form is filled out correctly, the user is redirected to the HTML page dedicated to the exercises. Here, through a JavaScript function, a 10-second counter has been implemented that is visible both on the screen and through audio feedback. This allows the user to step away from the computer during this period and have time to stand in front of the webcam. When the 10 seconds expires, the training session officially begins. The screen shows the webcam output in the center, with the user’s body landmarks highlighted. In addition, the counting algorithm described in Section 5.2 is invoked. At the top of the webcam pane is a bar with key information, such as the last exercise performed with the number of repetitions and a timer implemented via JavaScript. This timer increments by one second at a time, and a specific function displays the value formatted in hours, minutes, and seconds on the screen. On the left and right sides of the screen are several cards, one for each exercise available in the platform. Each card shows the name of the exercise, an explanatory image of the exercise, and the total number of repetitions performed (for plank exercises, for example, the time in seconds is shown). These values are generated dynamically from the Python backend via a JSON call. We have also included a dynamic message to alert the user if they are too close to the webcam, thus preventing the algorithm from being invoked correctly. For each exercise performed, feedback is provided on the correct execution in terms of speed, classifying the movement as ”perfect,” ”good,” or ”too fast.” The user can decide independently when to end the workout by simply clicking a button.

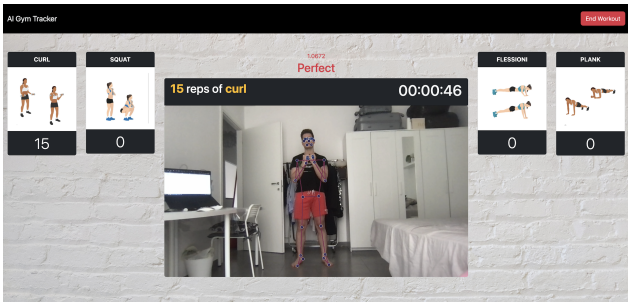

Figure 19. Landing page of AI Gym trainer

At the end of the session, the user is redirected to a new HTML page that shows a final overview of the workout, with the number of repetitions performed for each exercise and the value of calories burned. We are aware that there are different approaches to calculating calories, so to provide transparency we offer users the opportunity to see the formula used for our calculation. There is also a button on this page that optionally allows users to receive a workout report via email. Finally, the user can decide whether to repeat the workout session or return to the homepage.

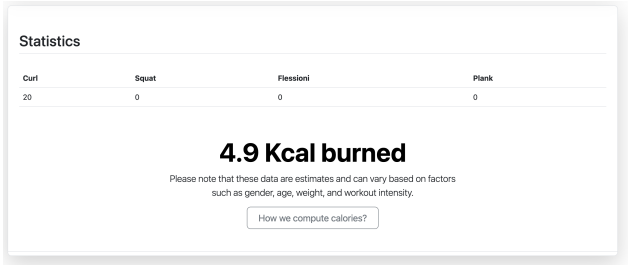

Figure 20. Landing page of AI Gym trainer

6 CONCLUSION AND FUTURE DEVELOPMENT

In conclusion, the AI Gym Tracker project represents an innovative multimodal application that uses artificial intelligence to track and monitor users' physical activities during exercise sessions. The application offers an intuitive and user-friendly user interface, enabling users to perform exercises correctly and obtain detailed information about their performance in real time.

Through the use of advanced technologies such as artificial intelligence, computer vision, and pose recognition, the application is able to count exercise repetitions and provide accurate feedback on the speed at which poses are performed. The artificial intelligence model based on recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) neural networks has demonstrated high accuracy in pose recognition and processing of frame streams from the webcam.

However, the project offers many opportunities for future development. First, uploading the application to a server and making it accessible online could be considered, so as to reach a wider audience.

Also, it would be possible to enrich the application by training the network with more exercises, giving users more flexibility in choosing activities.

Another possible future implementation could be the addition of automatic email functionality to alert users at the end of a training session, providing them with a report of statistics via email. This would allow users to track their progress and performance over time.

To ensure better data management and security, consideration should be given to implementing a database and authentication system for users of the application. This would allow users' personal data to be securely stored and provide a personalized user experience.

We made a demo video on using the application (5).

REFERENCES

[1] Ableton. Available from: https://www.ableton.com/.

[2] Bootstrap. Available from: https://getbootstrap.com/.

[3] Flask documentation. Available from: https://flask.palletsprojects.com.

[4] Html. Available from: https://html.spec.whatwg.org/multipage/.

[5] Demo video of ai gym tracker (2023). Available from: https://youtu.be/IKvSoxlg1Lo.

[6] ABADI, M. AND ET AL. Tensorflow: Large-scale machine learning on heterogeneous systems. (2015). Available from: https://arxiv.org/abs/1603.04467.

[7] AINSWORTH, B. E., HASKELL, W. L., LEON, A. S., JACOBS JR, D. R., MONTOYE, H. J., SALLIS, J. F., AND PAFFENBARGER JR, R. S. Compendium of physical activities: classification of energy costs of human physical activities. *Medicine and Science in Sports and Exercise*, (1993). Available from: https://lh-hsrc.pnu.edu.sa/wp-content/uploads/2018/11/Compendium-of-Physical-Activities-MSSE-2000.pdf.

[8] CHEN, S. AND YANG, R. R. Pose trainer: Correcting exercise posture using pose estimation. (2020). Available from: https://arxiv.org/pdf/2006.11718.pdf.

[9] CHOLLET, F. Keras. (2015). Available from: https://keras.io.

[10] FIGMA. Figma - the collaborative interface design tool. Available from: https://www.figma.com/.

[11] FLANAGAN, D. *JavaScript: The Definitive Guide*. O'Reilly Media (2021). Available from: https://pepa.holla.cz/wp-content/uploads/2016/08/JavaScript-The-Definitive-Guide-6th-Edition.pdf.

[12] GOOGLE. Mediapipe - pose landmarks. Available from: https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md#pose_landmarks.

[13] HOCHREITER, S. AND SCHMIDHUBER, J. Long short-term memory. *Neural computation*, (1997). Available from: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory.

[14] HOUDE, S. AND HILL, C. What do prototypes prototype? In *Handbook of human-computer interaction*. Elsevier (1997). Available from: https://hci.stanford.edu/courses/cs247/2012/readings/WhatDoPrototypesPrototype.pdf.

[15] RESEARCH, G. Mediapipe: A framework for building perception pipelines. https://developers.google.com/mediapipe.

[16] SCHMIDT, R. M. Recurrent neural networks (rnns): A gentle introduction and overview. *Department of Computer Science, Eberhard-Karls-University Tübingen*, (2019). Available from: https://arxiv.org/abs/1912.05911.