ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

Diseño de un framework para la extracción y análisis de puntos clave basado en Deep Learning para la mejora de la técnica empleada en actividades físicas.

Autor:
Ignacio Hernández Bas

Director del proyecto:
Álvaro Clemente Verdú

03-06-2022

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Diseño de un framework para la extracción y análisis de puntos clave basado en Deep Learning para la mejora de la técnica empleada en actividades físicas.

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2022/23 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Ignacio Hernández Bas Fecha: 03/06/2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Álvaro Clemente Verdú Fecha: 03/06/2023

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

TRABAJO DE FIN DE GRADO

Diseño de un framework para la extracción y análisis de puntos clave basado en Deep Learning para la mejora de la técnica empleada en actividades físicas.

Autor:
Ignacio Hernández Bas

Director del proyecto:
Álvaro Clemente Verdú

03-06-2022

DISEÑO DE UN FRAMEWORK PARA LA EXTRACCIÓN Y ANÁLISIS DE PUNTOS CLAVE BASADO EN DEEP LEARNING PARA LA MEJORA DE LA TÉCNICA EMPLEADA EN ACTIVIDADES FÍSICAS

Autor: Hernández Bas, Ignacio.
Director: Clemente Verdú, Álvaro.
Entidad Colaboradora: BigML, Inc

RESUMEN DEL PROYECTO

Las ramas del *Deep Learning y Pose Detection* se están desarrollando con rapidez, pero a pesar de esto su aprovechamiento es muy complejo. Para facilitar su uso, se ha desarrollado una aplicación que permite la detección y análisis de la técnica realizada por el usuario durante la ejecución de diferentes actividades físicas. A partir de la creación de un *dataset y* la evaluación del sistema sobre este conjunto de datos*,* se han obtenido resultados satisfactorios que muestran el gran potencial que posee la detección de poses en el ámbito del deporte.

Palabras clave*: Deep Learning, Pose Detection, keypoints*

1. Introducción

Durante los últimos años, la rama del *Deep Learning* propia del aprendizaje computacional (*Machine Learning*) ha aumentado en popularidad considerablemente gracias al constante desarrollo de las tecnologías y a la aparición de nuevas técnicas e investigaciones. Deep Learning se define como el conjunto de técnicas las cuales se basan en algoritmos que tratan de replicar la estructura y funcionamiento del cerebro humano a partir de la creación de Redes Neuronales. Entre las aplicaciones más populares se encuentra su uso para el desarrollo de automóviles que circulen de forma autónoma, la creación de modelos de lenguaje y de Procesamiento de Lenguaje Natural (NLP) empleadas por herramientas tales como ChatGPT [1] o para el procesamiento y clasificación de imágenes.

Del avance en técnicas de procesamiento de imágenes, surge la necesidad de la clasificación de eventos, tales como los movimientos o poses realizados por un individuo. La rama encargada de llevar a cabo estas detecciones recibe el nombre de *Pose Detection*, empleando técnicas donde los ordenadores tratan de replicar la forma en la que el ser humano asimila la información a través de los ojos. *Pose Detection* se centra en el diseño y creación de modelos que permitan localizar los puntos clave o *keypoints* de la entidad que está realizando la acción a analizar (por ejemplo, un humano). Estos puntos clave hacen referencia a las partes del cuerpo detectadas por el modelo de detección de pose, el cual devuelve las coordenadas en el espacio de cada uno de estos puntos en un instante de tiempo determinado.

Una vez se obtienen dichas coordenadas del modelo, otros sistemas pueden utilizar estos resultados y combinarlos con otros tipos de información, tal como el contexto o actividad que se está realizando, para obtener una visión más enriquecida de la acción empleada por el usuario.

A pesar de la existencia de aplicaciones que ya incorporan técnicas de detección de pose tales como la detección de posibles errores en la realización de posturas de yoga [2] o el análisis del swing empleado por un golfista [3], estas se limitan a realizar análisis sencillos que aportan una información limitada al usuario y en su mayoría, estos están solamente enfocados al análisis de un rango específico de acciones.

2. Definición del proyecto

Este proyecto define la creación de un sistema el cual permita acercar al usuario múltiples de las funcionalidades y herramientas relativas a la detección y análisis de poses en el ámbito del deporte y del ejercicio físico. Por lo tanto, se va a diseñar una aplicación de detección y análisis de actividades físicas, en la cual se le facilite al usuario la información relativa a sus movimientos realizados y los detalles para llevar a cabo las correcciones necesarias en el caso de que no los esté ejecutando de manera correcta en función del tipo de ejercicio que haya realizado.

Este proyecto esta principalmente enfocado en la detección de ejercicios físicos que un usuario podría realizar ya sea en su hogar o en el gimnasio y con la posibilidad de que pueda emplear diferentes objetos tales como pesas o barras para llevar a cabo su actividad.

3. Descripción del sistema

El sistema creado busca reducir el nivel de comprensión necesario para poder hacer uso de los beneficios de la detección de poses a partir de la creación de una herramienta de gran usabilidad que permita detectar con eficacia y precisión los eventos que suceden durante la realización de una actividad física. Para poder llevar a cabo esta función, el sistema se divide en tres bloques fundamentales, los cuales se muestran representados en la figura 1.

- **Aplicación Web**: gracias a la interfaz web creada en se permite al usuario interaccionar con el sistema permitiendo que este pueda cargar sus respectivos vídeos para su procesado, análisis y representación de resultados.
- **Procesado y detección de poses**: sección del sistema encargada de realizar la ingesta del vídeo introducido por el usuario, llevando a cabo la detección de la pose del usuario en cada una de las imágenes (*frames*) que componen el vídeo. Dicha detección será realizada a partir del modelo de Pose Detection de código abierto *MoveNet [4]*.
- **Análisis y obtención de eventos**: una vez ya se obtienen todas las imágenes del vídeo procesado con todas sus poses detectadas, se realiza el análisis de cada una de estas poses en función del tipo de ejercicio físico que el usuario ha ejecutado. Cada tipo de ejercicio supone que se deben de aplicar una serie de reglas determinadas para alcanzar la pose objetivo que se considera como correcta. Esto se realiza a partir de la agrupación de una serie de reglas generales, las cuales engloban el movimiento y posición de un conjunto de ciertas partes del cuerpo como las piernas, brazos o el torso.

Gracias a esto se dispone de un sistema adaptable que permite incorporar nuevas actividades físicas sin la necesidad de generar reglas específicas en función de la actividad que se desee incorporar a la aplicación. Tras el análisis, se lleva a cabo la síntesis de los resultados en eventos únicos que aporten información de utilidad al usuario donde se definen las acciones que se han realizado en el vídeo.

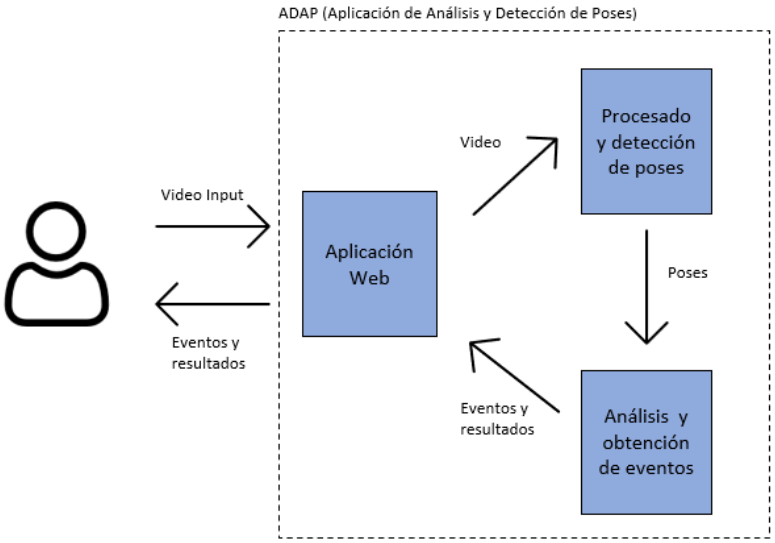

Figura 1 Secciones principales del sistema

4. Resultados

Tras la implementación del sistema, es importante evaluar el comportamiento del sistema, determinando si las predicciones realizadas por la sección de análisis se están realizando de forma correcta.

Figura 2 Pose objetivo alcanzada

Figura 3 Pose objetivo no alcanzada

Las figuras 2 y 3 muestran ejemplos de las imágenes resultantes del proceso de análisis y detección de un vídeo. Estas representaciones sirven como indicadores para el usuario sobre el rendimiento obtenido en la actividad. Las diferentes partes del cuerpo se resaltan en color verde o magenta, según si se encuentran dentro o fuera del rango objetivo de la pose que se debe alcanzar en función de la actividad.

Mediante la creación de un *dataset* donde se ha llevado a cabo el etiquetado de múltiples actividades físicas diferentes, se ha podido medir cuanto de preciso es el sistema a la hora de llevar a cabo las detecciones de las diferentes actividades físicas.

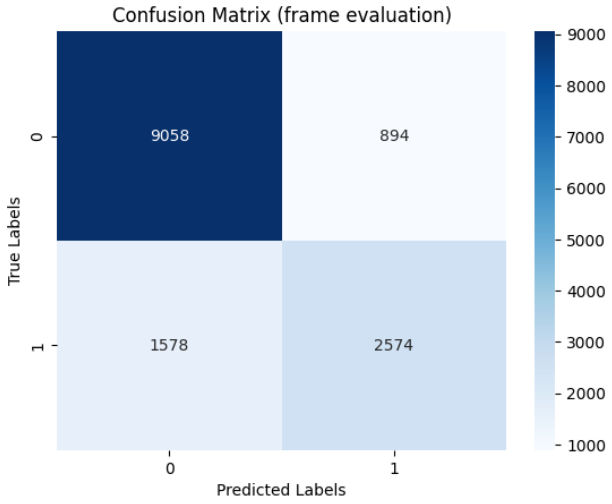

Figura 4 Matriz de confusión resultado de la predicción

Métrica	Valor
Accuracy	0.82
Precision	0.74
Recall	0.61
F1 Score	0.67

Tabla 1 Métricas de evaluación obtenidas

La figura 4 muestra la matriz de confusión resultado de la comparación entre todas las imágenes etiquetadas y las imágenes predichas por el sistema. Los resultados del *accuracy* muestran que más del 80% de los *frames* etiquetados se predicen correctamente, otras métricas como la *precision* indica que el 74% de las veces que se detecta una pose sin errores, esta también se había calificado como correcta.

Por otro lado, el *recall* supera el 60%, lo cual refleja que el sistema por lo general detecta correctamente los frames con una pose correcta, pero existen casos que el sistema no detecta. Ante estos resultados se puede afirmar que el funcionamiento del sistema es satisfactorio, pero presenta un posible margen de mejora en sus predicciones.

5. Conclusiones

Gracias a la creación de este proyecto, se ha podido presentar al usuario las utilidades que presenta la detección de poses a través del deporte, produciendo que se abra una nueva puerta a una innovadora forma de hacer ejercicio.

Los resultados obtenidos corroboran el buen funcionamiento del sistema, pero es de gran importancia mencionar que la cantidad de datos que se poseen y la variabilidad entre estos es determinante a la hora de llevar a cabo una evaluación. A mayor número de datos y situaciones que se poseen, más información se puede obtener de la evaluación de los datos predichos, obteniendo mejores conclusiones acerca del funcionamiento del sistema. Por lo que se recomienda tomar el *dataset* establecido como un punto de partida útil que permite concretar la precisión actual que presenta el sistema.

Respecto a los posibles trabajos futuros, sería interesante la composición de un dataset especializado en el marco de trabajo del proyecto, el cual presente una extensión mucho mayor y concentre numerosas actividades físicas, realizadas por una gran cantidad de personas y desde un mayor número perspectivas distintas que las que presenta el conjunto de datos actual.

Por otro lado, se recomienda la investigación de métodos de *Pose Detection* que incorporen múltiples cámaras a la hora de realizar la grabación de la actividad. A pesar de que MoveNet es un modelo que aporta buenos resultados con el uso de una sola cámara, un análisis en tres dimensiones de la acción podría suponer una mejora en la detección de los puntos clave del usuario y por lo tanto obtener una mayor precisión en el análisis de las actividades físicas.

6. Referencias

[1] ABDULLAH, Malak; MADAIN, Alia; JARARWEH, Yaser. ChatGPT: Fundamentals, applications and social impacts. En *2022 Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 2022. p. 1-8.

[2] Ajay Chaudhari et al. YOG-GURU: REAL-TIME YOGA POSE CORRECTION SYSTEM USING DEEP LEARNING METHODS. URL:https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9509937.

[3] Marc Marais and Dane Brown. Golf Swing Sequencing Using Computer Vision. URL: https://link.springer.com/chapter/10.1007/978-3-031-04881-4_28

[4] Google. MoveNet: modelo de detección de pose ultrarrápido y preciso. URL https://www.tensorflow.org/hub/tutorials/movenet

FRAMEWORK DESIGN FOR KEYPOINT EXTRACTION AND ANALYSIS BASED ON DEEP LEARNING FOR THE IMPROVEMENT OF THE TECHNIQUE USED IN PHYSICAL ACTIVITIES

Author: Hernández Bas, Ignacio.
Supervisor: Clemente Verdú, Álvaro.
Collaborating Entity: BigML, Inc

ABSTRACT

The *Deep Learning* and *Pose Detection* branches are rapidly evolving, but despite this progress, their utilization is quite complex. To facilitate their use , an application has been developed to enable the detection and analysis of the technique used during the practice of different physical activities. By the creation of a dataset and evaluating the system over this data, satisfactory results have been obtained, demonstrating the potential of pose detection in the field of sports.

Key words: Deep Learning, PoseDetection, keypoints

1. Introduction

During the last years, the branch of Deep Learning within the field of Machine Learning has risen in popularity considerably due to the continuous development of technologies and the emergence of new techniques and research. Deep Learning is defined as a set of techniques that are based on algorithms that aim to replicate the structure and functioning of the human brain with the creation of Neural Networks. Various of the most popular applications are autonomous driving, language modeling, Natural Language Processing (NLP) and image processing and classification.

The advancement in image processing techniques has led to a need for event classification, such as the movements or poses performed by an individual. The branch responsible of performing this kind of detection is referred as Pose Detection, applying techniques where computers attempt to replicate how humans assimilate information through their eyes. Pose Detection focuses on the creation and design of models that can locate the keypoints of the entity that is performing the action (e.g., a human). These keypoints refer to the body parts detected by the pose detection model, which returns the spatial coordinates of each of these points at a specific moment in time.

Once the coordinates from the model have been obtained, other systems can use these results and combine them with other types of information, such as the context or the type of activity that is being performed to obtain a deeper understanding of the action of the user.

Despite the existence of application that already incorporate pose detection techniques such as the detection of possible mistakes on during yoga practice[2] or analyzing a golfer's swing [3], these applications are often limited to simple analyses that provide limited information to the user. In addition, these applications are only focusing a narrow range of specific actions.

2. Project definition

This project defines the creation of a system that allows bringing to the user multiple functionalities and tools related to the detection and analysis of poses in the field of sports and physical exercise. Therefore, an application for the detection and analysis of physical activities will be designed, in which the user will be provided with the information related to his movements and the details to carry out the necessary corrections in case he is not executing them correctly according to the type of exercise he has performed.

This project is mainly focused on the detection of physical exercises that a user could perform either at home or in the gym and with the possibility of using different objects such as weights or bars to carry out their activity.

3. System description

The system created seeks to reduce the level of understanding necessary to be able to make use of the benefits of pose detection by creating a highly usable tool that can effectively and accurately detect events that occur during physical activity. In order to carry out this function, the system is divided into three fundamental sections, which are represented in figure 1.

- **Web application:** the web interface created in the system allows the user to interact with the system, letting the user upload videos for processing, analysis and representation of results.
- **Pose processing and detection**section of the system in charge of ingesting the video introduced by the user, carrying out the detection of the user's pose in each of the images (frames) that compose the video. This detection will be performed based on the MoveNet [4] open-source Pose Detection model.
- **Event analysis and retrieval**: Once all the images of the processed video with all the poses detected are obtained, the analysis of each of these poses is performed according to the type of physical exercise that the user has performed. Each type of exercise implies that a series of specific rules must be applied to achieve the target pose that is considered correct. This is done from the grouping of a series of general rules, which encompass the movement and position of a set of particular body parts such as legs, arms, or torso..

As a result, there is an adaptable system that allows the incorporation of new physical activities without the need to generate specific rules depending on the activity to be incorporated into the application. After the analysis, the results are summarized in unique events that provide useful information to the user where the actions that have been performed in the video are defined.

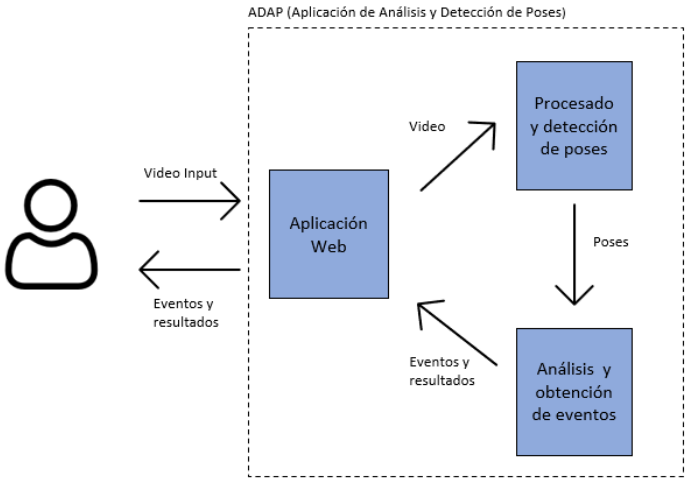

Figure 1 Principal sections of the system

4. Results

After the implementation, it is important to evaluate the behavior of the system, determining whether the predictions made by the analysis section are being carried out correctly.

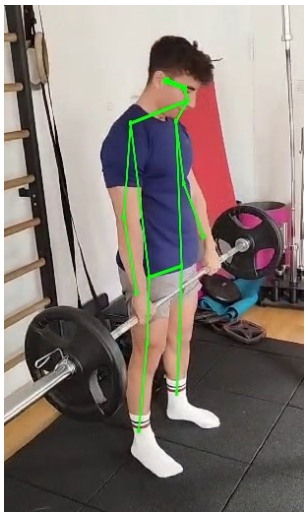

Figure 2 Objective position has been reached

Figure 3 Objective postion has not been reached

Figures 2 and 3 show examples of the images resulting from the process of analysis and detection of a video. These representations serve as indicators to the user about the performance obtained in the activity. The different parts of the body are highlighted in green or magenta, depending on whether they are within or outside the target range of the pose to be achieved based on the activity.

By creating a dataset where multiple different physical activities have been labeled, it has been possible to measure how accurate the system is in detecting the different physical activities..

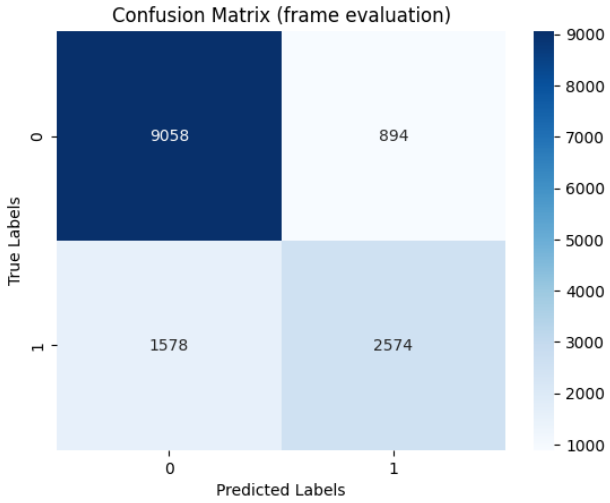

Figure 4 Confusion matrix of prediction results

Métrica	Valor
Accuracy	0.82
Precision	0.74
Recall	0.61
F1 Score	0.67

Table 1 Evaluation metrics obtained

Figure 4 shows the confusion matrix resulting from the comparison between all the labeled images and the images predicted by the system. The accuracy results show that more than 80% of the labeled frames are correctly predicted, other metrics such as precision indicate that 74% of the times an error-free pose is detected, it was also rated as correct.

On the other hand, recall exceeds 60%, which reflects that the system generally detects correctly the frames with a correct pose, but there are cases that the system does not detect. In view of these results, it can be stated that the system's performance is satisfactory, but there is room for improvement in its predictions.

5. Conclusions

Thanks to the creation of this project, it has been possible to present to the user the utilities offered by the detection of poses through sport, opening a new door to an innovative way of exercising.

The results obtained corroborate the good performance of the system, but it is of great importance to mention that the amount of data and the variability between them is a determining factor when carrying out an evaluation. The more data and situations you have, the more information you can obtain from the evaluation of the predicted data, obtaining better conclusions about the performance of the system. Therefore, it is recommended to take the established dataset as a useful starting point to determine the current accuracy of the system.

Regarding possible future work, it would be interesting to compose a specialized dataset within the framework of the project, which would present a much larger extension and concentrate numerous physical activities, performed by many people and from a greater number of different perspectives than those presented by the current dataset.

On the other hand, it is recommended to investigate Pose Detection methods that incorporate multiple cameras when recording the activity. Although MoveNet is a model that provides good results with the use of a single camera, a three-dimensional analysis of the action could improve the detection of the user's key points and therefore obtain greater accuracy in the analysis of physical activities.

6. References

[1] ABDULLAH, Malak; MADAIN, Alia; JARARWEH, Yaser. ChatGPT: Fundamentals, applications and social impacts. En *2022 Ninth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE, 2022. p. 1-8.

[2] Ajay Chaudhari et al. YOG-GURU: REAL-TIME YOGA POSE CORRECTION SYSTEM USING DEEP LEARNING METHODS.

URL:https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9509937.

[3] Marc Marais and Dane Brown. Golf Swing Sequencing Using Computer Vision. URL: https://link.springer.com/chapter/10.1007/978-3-031-04881-4_28

[4] Google. MoveNet: modelo de detección de pose ultrarrápido y preciso. URL https://www.tensorflow.org/hub/tutorials/movenet.

A mi familia, por apoyarme siempre de forma incondicional.

A mis amigos, por confiar en mí y siempre estar ahí.

A Alba por haber tenido que aguantarme y que nunca ha dudado de mí.

A BigML por concederme esta oportunidad y sobre todo a Álvaro Clemente , por ser un excelente tutor y que gracias a su guía este trabajo ha podido realizarse.

Índice

Índice de figuras

Índice de tablas

1 Introducción

En la actualidad el ser humano dispone de gran variedad de medios para acceder a la información que este desea. A partir del auge de las tecnologías, la sociedad contemporánea se encuentra inmersa en un mundo donde la accesibilidad a la información nunca había sido tan sencilla. Mediante las ciencias de los datos, también conocidas por el término *Data Science*, se da el análisis de dichos datos para la detección de posible patrones, realizar predicciones y poder llevar a cabo la toma de decisiones respecto a estas. Debido a la gran cantidad de dispositivos que se encuentran conectados cada día, se genera un nivel ingente de datos, el cual es difícil de comprender para los modelos convencionales de tratamiento de esta información. A partir de la necesidad de nuevas técnicas de procesado de la información surgió el nacimiento del *Big Data* y el *Machine Learning*.

Big Data corresponde al conjunto de tecnologías encargadas de almacenar, gestionar y procesar estos grandes volúmenes de datos. Por otro lado, el término *Machine Learning* hace referencia a la disciplina de la *Inteligencia Artificial* que mediante el uso de algoritmos, es capaz de dotar a los ordenadores la capacidad de procesar dichos datos, identificar patrones en estos y elaborar predicciones. A raíz de la evolución las técnicas empleadas por la rama del Machine Learning surgió la rama conocida como *Deep Learning*, esta está centrada en los algoritmos que tratan de replicar la estructura y funcionamiento del cerebro generando lo que se conoce por *Redes Neuronales*. A diferencia de otras técnicas empleadas en *Data Science*, las técnicas de *Deep Learning* son las que presentan un mayor rendimiento a medida que la cantidad de los datos aumenta, es decir, presenta una mayor escalabilidad.

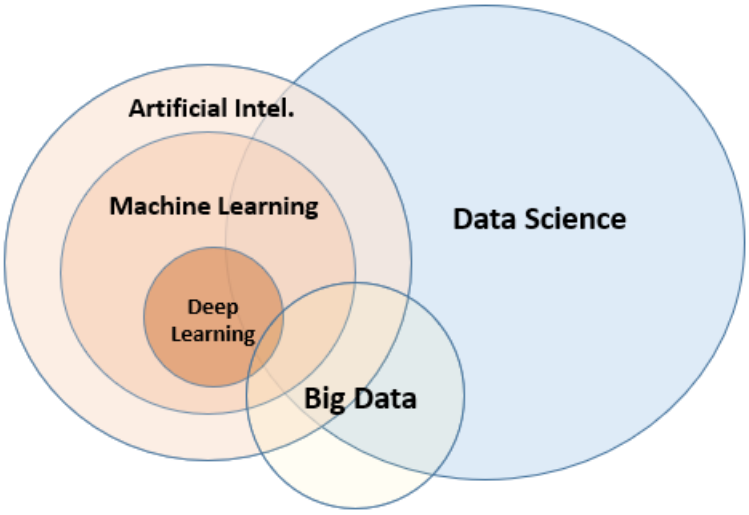

Figure 1.1: Contexto al que pertenece el Deep Learning

En la Figura 1.1 se puede apreciar el contexto previamente mencionado acerca de la relación entre el *Big Data*, las Ciencias de la Información y la Inteligencia Artificial con sus respectivas partes.

Entre las aplicaciones más utilizadas en el *Deep Learning* se encuentra su uso para el desarrollo de automóviles que circulen de forma autónoma, la creación de modelos de lenguaje y de Procesamiento de Lenguaje Natural (*NLP*) empleadas por herramientas tales como *ChatGPT* o al empleo de técnicas *Deep Learning* para el procesamiento y clasificación de imágenes. Del avance en técnicas de procesamiento de imágenes, surge la necesidad de la clasificación de eventos, tales como movimientos o poses realizados por un individuo.La rama perteneciente al *Deep Learning* que trata de llevar a cabo estas detecciones recibe el nombre de *Pose Detection*. Dicha detección utiliza los ordenadores empleando técnicas que tratan de asimilar la información tal como lo realiza el ser humano a través de los ojos. Estas técnicas pertenecen al ámbito de la visión computacional o *Computer Vision*.

Pose Detection se centra en el diseño y entrenamiento de modelos que sean capaces de detectar los puntos clave o *Key Points* de la entidad que está llevando a cabo una acción (por ejemplo, un humano). Tras esto, el modelo generado devuelve las coordenadas en el espacio de estos puntos en un instante de tiempo determinado.La obtención de estas coordenadas se da gracias al uso de *Redes Neuronales Convolucionales (CNNs)* , las cuales son abordadas en la sección **3**. Una vez se obtienen dichas coordenadas del modelo, otros sistemas pueden utilizar estos resultados y combinarlos con otros tipos de información, tal como el contexto o actividad que se está realizando, para obtener una visión más enriquecida de la acción empleada por el usuario.

Este análisis de movimientos puede ser aplicado a actividades físicas tales como el golf, tenis, yoga o halterofilia, donde la técnica es fundamental para obtener buenos resultados. Por otro lado, puede ser aplicado para entornos de realidad virtual donde se lleva a cabo la estimación de los movimientos y poses realizados por el jugador para que estos puedan trasladarse al mundo virtual. La detección y análisis de poses presenta un gran potencial debido a las numerosas posibles aplicaciones que presenta y al avance que se ha dado en la capacidad de procesamiento de las tecnologías actuales.

Este proyecto surge debido la necesidad de facilitar las técnicas y utilidades del *Computer Vision* y *Pose Detection* para hacer más accesible su uso a cualquier usuario que desee analizar la forma en la que realiza una determinada actividad física y este pueda beneficiarse reduciendo los conocimientos necesarios para hacer uso de dichas herramientas de forma efectiva. En este documento se va a exponer tanto la motivación para llevar a cabo la realización del proyecto como el diseño, implementación y análisis de resultados obtenidos del entorno de análisis creado.

Este trabajo se ha realizado en colaboración con la empresa BigML,Inc, cuya misión consisten en democratizar el *Machine Learning* , acercándolo a todo tipo de perfiles gracias a su herramienta en línea y fomentando el aprendizaje a partir de la impartición de cursos y entrega de certificaciones. Por lo tanto, este proyecto se ha conformado siguiendo dicho objetivo, contribuyendo al avance y difusión de las herramientas necesarias para la detección de poses, para que estas puedan ser empleadas por el usuario de una forma sencilla y útil.

2 Descripción de las tecnologías

En esta sección se van a describir las tecnologías utilizadas para el desarrollo de este proyecto. Esta selección determina el planteamiento y la forma en la que se va a abordar el diseño y la implementación del trabajo. Los objetivos de este proyecto, que se definen en el apartado 4.2, se centran en la creación de un entorno donde el usuario pueda registrar su actividad física (formato vídeo) y el entorno analice dicha actividad y devuelva al usuario recomendaciones/consejos de cómo debe realizar ésta de forma correcta. Para llevar a cabo la selección de las tecnologías, se han tenido en cuenta estos objetivos.

Las tecnologías principales utilizadas son las siguientes:

- Lenguaje de programación.
- Herramientas numéricas y de representación gráfica.
- Herramientas de *Machine y Deep Learning*.

2.1 Lenguaje de Programación

La elección del lenguaje de programación es una de las decisiones más determinantes a la hora de implementar a cabo el proyecto. Esto se debe a que todas las demás herramientas utilizadas girarán en torno a este lenguaje y por lo tanto se deberá elegir el lenguaje que concuerde en mejor medida con la solución que se quiere realizar.

El lenguaje que se ha decido emplear es *Python*, esto se debe a que es el lenguaje más popular en lo que se refiere al análisis, procesamiento de datos y a la aplicación de técnicas relativas al Machine Learning. A pesar de que hay lenguajes los cuales se han creado con el fin principal del procesado de datos tal como *R*, Python resulta un lenguaje más flexible y multifunción ya que su uso permite no solo realizar los apartados de análisis y procesamiento de la información, sino que gracias a librerías como *Dash* o *Streamlit* se pueden crear aplicaciones interactivas de forma sencilla sin la necesidad de la utilización de lenguajes tales como *Java, JavaScript, HTML o CSS*.

Tal como se ha mencionado anteriormente, *Streamlit* es una librería de Python que facilita la creación y despliegue de aplicaciones web de forma sencilla. Este marco permite la creación de interfaces interactivas a partir de una serie de componentes (*widgets*) ya integrados que facilitan la implementación sencilla de la interfaz de la aplicación. También proporciona herramientas para el procesamiento y visualización de información y *Machine Learning*. Además este entorno permite a los usuarios la creación y subida de múltiples componentes en forma de *plugins* o *widgets*, lo cual permite extender las funcionalidades básicas de la librería. Esto forma una comunidad activa de usuarios que

comparten sus componentes y colaboran respondiendo las posibles preguntas que puedan surgir respecto al uso e implementación de estos.

El poder llevar a cabo el proyecto en el mismo lenguaje de programación favorece que todos los apartados y funcionalidades del proyecto cohesionen correctamente fomentando la eficiencia en el desarrollo del proyecto.

2.2 Herramientas científicas y de representación gráfica

Tras la elección del lenguaje de programación, se han seleccionado las herramientas que se van a encargar de realizar el procesamiento y representación de los datos. Gracias al uso de Python, existe un gran entorno de librerías de libre acceso que permiten implantar las funcionalidades necesarias. Para el procesamiento numérico y de tratamiento de los datos destacan las librerías *NumPy*, *Math* y *Pandas*.

- *NumPy*[1]: librería enfocada en el cálculo numérico y de procesamiento de datos. Incluye estructuras de datos conocidas como *arrays*, estas estructuras permiten realizar colecciones de datos de múltiples dimensiones y su principal ventaja es que su velocidad de procesamiento es muy superior a las listas de elementos convencionales. La velocidad de procesamiento de esta librería se debe a que el núcleo de la librería se encuentra muy optimizado debido a que está compilado en lenguaje *C*.

  Presenta un mayor nivel de eficiencia en el uso de recursos debido a que es un lenguaje de más bajo nivel donde se da una gestión manual de la memoria, reduciendo los tiempos de procesado y ejecución. Debido a esto, NumPy es una herramienta muy popular a la hora de realizar operaciones con matrices y vectores de múltiples dimensiones. Siendo además la base de la computación numérica en *Python*, incluyendo librerías que emplean *dataframes*(*Pandas*), librerías de *Machine Learning* (*Sk-learn*) o librerías de computación científica.

- *Math*[2]: módulo intrínseco de Python, también se encuentra definido en lenguaje C y permite la utilización de funciones matemáticas, como cálculos aritméticos, exponenciales y logarítmicos.

- *Pandas*[3]: librería que aporta útiles estructuras de datos tales como las *Series* y *Dataframes*. Esta librería extiende las capacidades de la librería *Numpy* a la hora de llevar a cabo las operaciones sobre datos de forma tabular con un gran rendimiento. En este proyecto la estructura que se va a utilizar en mayor medida es el *Dataframe*, el cual tiene un aspecto tabular, donde las filas y columnas vienen marcadas por una serie de etiquetas o *labels*. Esta librería permite una manipulación de estas estructuras de forma flexible, tal como se realiza en programas de gestión de hojas de

cálculo o de bases de datos relacionales. Esta librería es una herramienta muy popular entre analistas y científicos de datos, también conocidos como *Data Scientists*.

[6]:

	time	nose_x	nose_y	left_ear_x	left_ear_y	left_eye_x	left_eye_y	left_hip_x	left_hip_y	left_knee_x
0	40.0	0.476735	0.842890	0.482119	0.842627	0.479408	0.845363	0.487508	0.777428	0.487423
1	80.0	0.477147	0.843426	0.482852	0.843172	0.479748	0.845785	0.488022	0.777431	0.487731
2	120.0	0.477015	0.842805	0.482724	0.842785	0.479743	0.845200	0.487666	0.777160	0.487478
3	160.0	0.476858	0.842729	0.482741	0.842638	0.479591	0.845134	0.487669	0.777235	0.487521
4	200.0	0.476024	0.842351	0.483962	0.844527	0.478886	0.844731	0.487853	0.777394	0.488137

Figure 2.2: Representación de un DataFrame creado en *Pandas*

Para llevar a cabo la representación gráfica las librerías principales utilizadas son las siguientes: *Matplotlib* y *Plotly*.

- *Matplotlib*[4]: una de las las librerías de visualización de datos más utilizadas en *Python*. Está inspirada en la forma de representación propia del software *Matlab*[5], por lo que la creación de figuras se pude aplicar de forma rápida para los que son conocedores del lenguaje utilizado en este programa. Esta librería presenta un gran grado de personalización, lo cual resulta muy funcional pero a su vez pude resultar compleja para los casos en los que se busque una método de representación más sencillo de implementar.

- *Plotly*[6]:librería orientada a la generación de gráficos interactivos y animados. Estos gráficos permiten ser modificados por los usuarios en tiempo real, resultando ser muy atractivos a la hora de exponer información a través de estos. Además, presenta una plataforma en línea colaborativa que permite a los usuarios compartir gráficos y visualizaciones de forma que el número de posibles representaciones gráficas e implementaciones aumenta a lo largo del tiempo. Esta librería presenta versiones para lenguajes tales como *Python*, *R* y *JavaScript* entre otros.

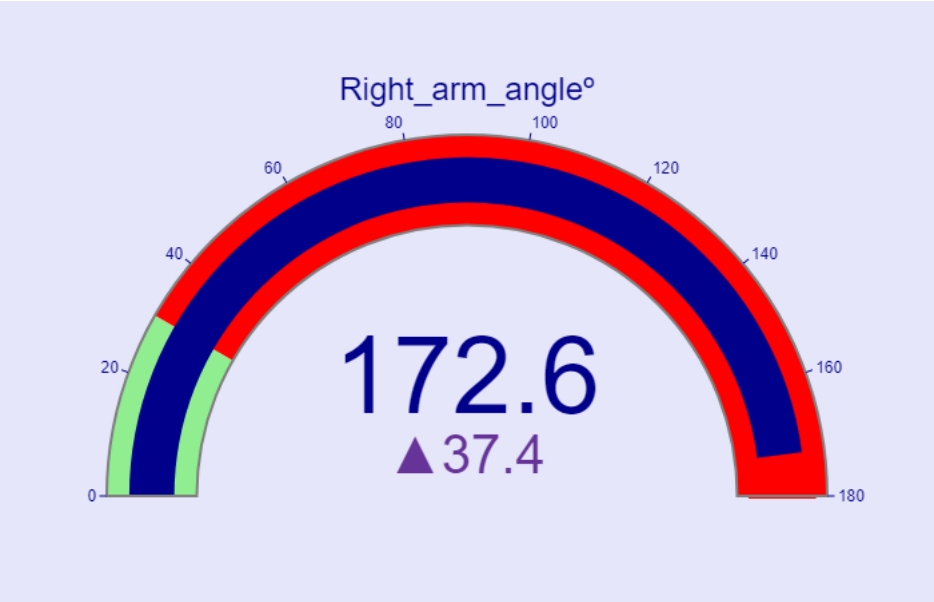

Figure 2.3: Ejemplo de gráfica obtenida con *Plotly*

2.3 Herramientas de *Machine, Deep Learning y Computer Vision*

Por último, restan por definir las herramientas orientadas al modo en el que se va a obtener la información de procedente del usuario para llevar a cabo la detección y análisis de sus movimientos. Para ello se han utilizado las herramientas de *OpenCV*, *MoveNet* y *Scikit-Learn*.

- *OpenCV*[7]: librería de código abierto de *Computer Vision*, utilizada para el procesamiento de imágenes y llevar a cabo detección de objetos, bordes, rasgos faciales y eventos entre otras opciones. En un primer momento se implementó en *C++*, pero también tiene integraciones para lenguajes como *Java* y *Python*.

- *MoveNet*[8]: modelo de detección de pose desarrollado por Google y basada en el propio *framework* de código abierto para *Deep Learning* : *TensorFlow 2.0*. Este modelo aporta las coordenadas en dos dimensiones de 17 puntos clave (*keypoints*) del cuerpo humano. A partir de estas coordenadas, se llevará a cabo todo el análisis relativo a la detección de errores en los vídeos aportados por el usuario. Lo que destaca de este modelo es que permite realizar detecciones en tiempo real a más de 30 imágenes por segundo tanto para la detección de múltiples entidades (hasta 6) en su versión *lightning* y también para la detección, que presenta un mayor grado de precisión de una sola entidad en la versión *thunder*.

- *Scikit-Learn*[9]: librería enfocada al aprendizaje automático, tal como las anteriores librerías, es de código abierto e implementada para su uso mediante lenguaje Python. Presenta diferentes algoritmos para la regresión, análisis y clasificación de grupos mediante métodos tales como *K-Means*, *Random Forest* o *Nearest Neighbours* entre muchos otros. Herramienta muy útil tanto para la aplicación de dichos algoritmos, como para llevar a cabo el análisis de las predicciones obtenidas por un modelo y realizar la evaluación acerca del rendimiento de este. Ante su sencilla implementación, permite a los usuarios no especialistas hacer uso de dichas técnicas y obtener sus beneficios.

3 Estado de la cuestión

A lo largo de este capítulo se va a exponer el estado del arte del *Pose Detection* y sus múltiples aplicaciones. Para comenzar se va a tratar del contexto en el que se encuentra dicha rama de la visión computacional respecto a su usos e implementaciones. Seguidamente, se tratará acerca de los modelos encargados de llevar a cabo la de detección de poses y su funcionamiento. Por último, se abordarán las limitaciones actuales de las implementaciones y se propondrá una nueva solución.

3.1 Pose Detection, origen e implementaciones

La detección de poses surge a partir de las investigaciones que se dieron en los campos del *Machine Learning* y *Computer Vision*. Los primeros intentos de llevar a cabo el reconocimiento de características relativas al ser humano surgen a mediados de los 90 con los primeros avances en detección y reconocimiento de rostros humanos [10]. Por otro lado, también surgieron los primeros intentos de detección de pose dinámica[11] en una transmisión de vídeo. A partir de estos hallazgos, numerosos han sido los avances y algoritmos diseñados en el *Pose Detection*. La gran mayoría de estos se basan en la aplicación de técnicas basadas en *Deep Learning* que presentan una gran precisión al trabajar con una gran cantidad de datos.

Es importante hacer la distinción entre una detección de pose basada en el análisis de imágenes a partir de técnicas *Deep Learning* y la detección de poses obtenida a partir de hardware específico como cámaras especializadas y sensores que rastrean físicamente los movimientos que realiza el usuario. Este enfoque comenzó a aplicarse en la industria del cine en 1984[12] y su uso actual está centrado en la generación de contenido enfocado al entretenimiento empleándose en películas, series y videojuegos. En este proyecto nos centramos en el primero, ya que no requiere una inversión en hardware específico, siendo un método más accesible y versátil.

Uno de los primeros modelos que emplea *Deep Learning* y que lograron la detección de poses en tiempo real sobre un vídeo con bajo retardo , fue realizado en 2016 por la Universidad Carnegie Mellon (Texas, USA) y recibió el nombre de Open Pose[13]. Otro modelo a destacar fue desarrollado por Google, el cual se mencionó en el apartado 2.3, nació en 2020 y sigue implementando modificaciones para llevar a cabo mejoras en su detección y rendimiento. Ante esto se puede afirmar que la rama del *Pose Detection*, es relativamente nueva y se encuentra en continuo progreso y presenta un gran potencial para llevar a cabo a numerosas implementaciones. La detección automática del lenguaje de signos [14], el reconocimiento automático y detección de posible errores en la realización de posturas de yoga [15] o el análisis del swing empleado por un golfista[16] son algunas de las numerosas aplicaciones que se pueden realizar con los datos obtenidos a partir de un modelo de detección de poses.

3.2 Composición y funcionamiento de los modelos de *Pose Detection*

La técnica más popularmente utilizada para llevar a cabo la creación de modelos focalizados en la detección de poses es el uso de Redes Neuronales Convolucionales, también conocidas como *CNNs*. Estas redes son capaces de entender la relación espacial entre los píxeles de una imagen, lo que les permite captar relaciones visuales complejas que otras técnicas no son capaces de detectar. Una de las primeras aplicaciones de las *CNNs* fue su utilización para la clasificación de patrones complejos tal como la detección de caracteres escritos a mano[17].

El funcionamiento de una Red Neuronal Convolucional consta de cuatro capas o niveles principales:

- *Capa de convolución*: esta capa se encarga de la obtención de *feature maps*, es decir, surgen sub-imágenes que contienen los rasgos más característicos de la imagen recibida. Dichos rasgos se obtienen a partir del aplicado de filtros que se encargan de determinar las características más importantes dependiendo del propósito para el que se esté aplicando dicha red. Por ejemplo, para un modelo de detección de pose, los filtros estarán enfocados de forma automática a la obtención de las diferentes partes del cuerpo como articulaciones y extremidades.

- *Capa de activación*: capa que introduce una función no lineal en la red neuronal. Esta permite a la red detectar relaciones y patrones no lineales entre los datos de entrada. Esto permite que la red neuronal sea más potente y capaz de aportar soluciones a problemas de mayor complejidad. Las funciones más destacadas relativas a la capa de activación son la función tangente hiperbólica, la función sigmoide y la función ReLU (*Rectified Linear Unit*)[18]. Esta última función es actualmente la función mayormente utilizada y popular en la construcción de *CNNs*.

- *Capa de agrupación*: nivel encargado de llevar a cabo un resumen de los rasgos más característicos de cada sub-imagen obtenida a partir del nivel anterior. Las sub-imágenes se pueden resumir mediante el uso de múltiples funciones. La función más popular se basa en la obtención del máximo valor entre los píxeles adyacentes de una sección de la imagen. Este resumen permite la reducción de la dimensión de la imagen aun conservando sus rasgos más significativos.

- *Capa completamente conectada*: capa en la que se lleva a cabo la interconexión entre todas datos obtenidos a partir de las capas anteriores. En esta última fase se lleva a cabo la tarea final de la red, por lo que en un modelo de detección de pose el objetivo final se basa en la obtención de las coordenadas procedentes de las diferentes partes del cuerpo referentes a la pose humana que se encuentre en una

imagen.

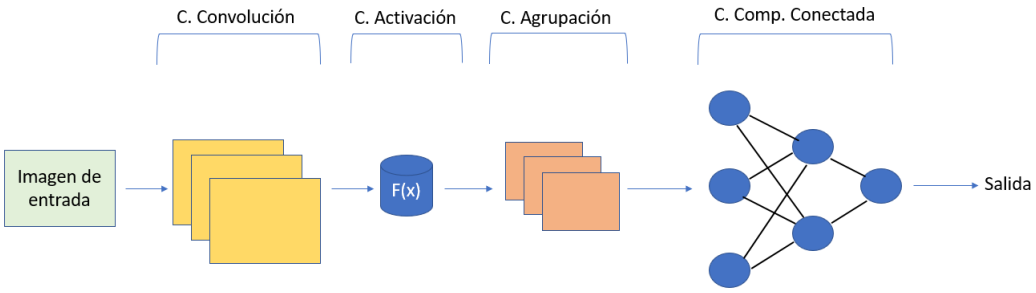

Figure 3.4: Esquema de las capas principales que conforman una Red Neuronal Convolucional

3.3 Principales limitaciones actuales del *Pose Detection*

El campo del *Pose Detection* ha experimentado grandes avances el progreso y desarrollo de las nuevas tecnologías, mejorando considerablemente la precisión de los resultados y velocidad de procesado . A pesar de esto, existen limitaciones a la hora de aplicar dichas técnicas. A continuación, se muestran las principales limitaciones actuales que presenta el campo de la detección de poses.

1. **Tasa de frames reducida:** los modelos entrenados que actualmente se encuentran disponibles, permiten el procesado de flujos de vídeo a velocidades que se encuentren entre los 30 y 60 frames por segundo. Dichas limitaciones son fundamentalmente a causa del hardware que se esté utilizando, donde el modelo no es capaz de procesar las imágenes con la misma rapidez a la que la cámara utilizada las está obteniendo.

2. **Ajuste fino limitado:** en el contexto del *Machine Learning* el*fine tuning* consiste en aplicar ligeras modificaciones sobre un modelo para establecer mejoras en su rendimiento. Aplicar estas modificaciones sobre los modelos de detección de pose presenta un gran grado de dificultad, esto principalmente se debe al gran tamaño y complejidad del modelo y por lo tanto, no resultaría ser una tarea sencilla y supondría una gran inversión de tiempo y recursos llevar a cabo el entrenamiento del modelo.

3. **Sensibilidad a la imagen de entrada:** otra de las posibles limitaciones que puede darse en el ámbito de la detección de poses reside en la calidad, iluminación y posición del usuario en la imagen recibida. Estos son factores determinantes a la hora de llevar a cabo la detección de la pose humana y obtener los resultados que se esperaban por el modelo. Además, la mayoría de los modelos actuales se basan en realizar una detección en dos dimensiones, por lo que ciertos movimientos

complejos que se capten en vídeo no serán procesados correctamente por el modelo. A lo largo de estos últimos años han surgido avances en la creación de modelos que detecten la pose humana en tres dimensiones a partir del uso de varias cámaras, que implementan modelos de detección 2D, cuyos resultados se combinan conformando una pose que presenta longitud, altura y profundidad[19].

Las dificultades que presenta el análisis del *Pose Detection* hace que muchas de las aplicaciones existentes enfocadas a la detección de poses, se limiten a análisis sencillos que aportan una información limitada. Facilitar el análisis de estos datos permitirá que los desarrolladores puedan realizar observaciones más complejas, aportando información más relevante y de mayor utilidad a los usuarios

4 Definición del trabajo

4.1 Motivación

La detección de errores en la realización de determinadas actividades es fundamental a la hora de analizar los resultados obtenidos en dicha actividad. Aplicada dicha detección sobre actividades físicas puede proporcionar una gran información al usuario que la realiza, un buen empleo de la técnica es de gran importancia para obtener buenos resultados y además para no perjudicar su salud. Muchas actividades realizadas de forma incorrecta pueden poner en peligro a los que la practican y a los que le rodean. Por ejemplo, en el ámbito industrial, la detección de poses puede emplearse para advertir a los trabajadores acerca de posturas que les puedan causar perjuicios físicos o ser la causa de un accidente laboral, favoreciendo la creación de entornos de trabajo más seguros y suponiendo un ahorro en indemnizaciones y reparaciones para las empresas.

La utilización del *Deep Learning* en el ámbito de la visión computacional cada vez es más popular. Los últimos avances en el desarrollo de las tecnologías y en la velocidad de procesamiento permiten que la estimación de poses presente un mayor número de aplicaciones y los métodos utilizados puedan ser investigados por un mayor número de personas. A pesar de el gran número de artículos, proyectos e información que se enfoca en esta rama de la *Inteligencia Artificial*, la detección y clasificación de movimientos sigue siendo una tarea compleja que requiere un alto nivel de comprensión e investigación para ser llevada a cabo. Ante esto, el proyecto busca promover que dichos modelos y herramientas sean más accesibles al usuario, reduciendo el nivel de comprensión necesario para poder emplear dichas técnicas y favorecerse de sus resultados.

Proyectos como el mencionado en el capítulo anterior[15], proporcionan escasa información al usuario acerca de los errores de técnica que se están cometiendo al realizar dicha actividad. Por otro lado, el proyecto centrado en análisis y corrección de movimientos de baile [20], es capaz de aportar una mayor información acerca de los pasos que se deben de tomar para llevar a cabo el movimiento de forma correcta. Este proyecto aplica el concepto de *CPM (Convolutional Pose Machine)*, el cual consiste en el uso agrupaciones de las *CNNs* mencionadas en la sección anterior. Esta agrupación obtiene un mayor rendimiento a costa del aumento de complejidad del sistema y por ende que realizar el *fine tuning* de este presente una mayor dificultad. Además al ser un proceso más complejo, el tiempo de procesado es mayor y es menos eficaz para realizar análisis de poses en tiempo real.

En resumen, un tiempo se procesado alto, un sistema difícil de configurar o aportar información de poca utilidad al usuario son varios de los posibles problemas que se pueden dar a la hora de hacer una aplicación de detección de pose. Ante esto, este proyecto plantea establecer una aplicación de detección y análisis de actividades físicas, en la cual se le

facilite al usuario la información relativa de sus movimientos y los posibles detalles necesarios para llevar a cabo la corrección de sus movimientos en el caso de que no los esté ejecutando de manera correcta. Por otro lado, este proyecto presenta otros objetivos como la eficiencia en la obtención de los resultados como que este sea fácilmente configurable. Estos objetivos serán abordados en la siguiente sección,

Este proyecto se focalizará en la detección de ejercicios físicos que un usuario podría realizar ya sea en su hogar o en el gimnasio y con la posibilidad de que pueda utilizar múltiples objetos tales como pesas o barras para llevar a cabo su actividad.

4.2 Objetivos del proyecto

El objetivo principal del proyecto se basa en el desarrollo de técnicas de procesado de datos obtenidos a partir de modelos *Deep Learning* de detección automática de poses para la extracción de información accionable a partir de la cual se le pueden asignar consejos y recomendaciones al usuario acerca de cómo debe realizar la actividad física deseada de forma correcta.

La aplicación a desarrollar debe de guiarse siguiendo los siguientes objetivos principales:

- **Precisión** : el sistema debe ser capaz de detectar los movimientos respectivos a una actividad física, analizarlos según el tipo, empleando un sistema de reglas determinado y detectar los posibles errores que se den de forma precisa.

- **Adaptabilidad** : el sistema debe de estar planteada de forma que se puedan llevar a cabo mejoras o nuevas implementaciones de una forma sencilla. El sistema debe permitir añadir el análisis de nuevas actividades físicas de forma sencilla.

- **Usabilidad**: la aplicación debe de ser implementada de forma que su utilización sea sencilla y no haga falta un conocimiento extenso en visión computacional.

- **Eficiencia** : el procesado de los movimientos y la detección de errores debe llevarse a cabo de forma eficiente para que la obtención de los resultados sea rápida.

Estos objetivos serán los que marquen la dirección a tomar relativa a las implementaciones, reglas, requisitos y decisiones que se vayan a aplicar a lo largo del desarrollo del proyecto.

4.3 Metodología de Trabajo

El proceso de desarrollo del proyecto va a seguir una metodología ágil, la cual a partir de los resultados que se obtengan irán generando cambios en el planteamiento, configuración o diseño del sistema planteado. El proyecto se divide en cuatro bloques principales, en los cuales se realizarán múltiples tareas siguiendo el orden de la enumeración. A su vez, las tareas dentro de un bloque también son iterativas y por ejemplo, el resultado del análisis de un vídeo puede suponer que se necesite recoger nuevos datos para comprobar el comportamiento del sistema ante un tipo de ejercicio específico.

1. **Obtención y procesamiento de la información de la actividad física:**

    1.1. Obtención de información procedente de las imágenes de vídeo.

    1.2. Tratamiento y limpieza de los datos obtenidos(filtrado,eliminación de ruido...).

    1.3. Procesado de la información a partir de modelos de detección de pose.

2. **Análisis de resultados y detección de errores :**

    2.1. Extracción de características (features) procedentes de los datos obtenidos por el modelo.

    2.2. Creación del sistema de reglas que serán aplicadas al modelo de detección de errores.

    2.3. Modificación de reglas generales en función de la categoría de la actividad física a analizar.

    2.4. Análisis y obtención de resultados.

3. **Creación de una aplicación WEB :**

    3.1. Diseño de una herramienta que permita la carga de vídeos por parte del usuario.

    3.2. Implementación de las secciones de procesado de imagen y de análisis de resultados.

    3.3. Representación de los resultados a partir de tablas,gráficas y vídeo.

4. **Evaluación del sistema :**

   4.1. Creación de una herramienta que facilite el etiquetado de vídeos.

   4.2. Grabación y etiquetado de vídeos que conformen un dataset.

   4.3. Evaluación de la precisión del sistema mediante la comparación entre los resultados originales y los obtenidos a partir del sistema de reglas.

   4.4. Evaluación de rendimiento del sistema.

4.4 Planificación

A continuación, se muestra la organización y el tiempo que han ocupado las principales tareas realizadas relativas al proyecto a partir de un digrama de Gantt. Este diagrama, figura 4.5, muestra una aproximación a la planificación real que se ha llevado. Se debe a que su creación estaba orientaba a la planificación de proyectos que empleaban planificaciones en cascada. Por lo tanto, se ha tratado de mostrar la metodología *Agile*, ya mencionada en el anterior apartado, a partir de la superposición de tareas durante el mismo periodo de tiempo, haciendo referencia a un proceso iterativo en el que se modifica el sistema a partir de los resultados que se van obteniendo.

Este enfoque se opone a los modelos de planificación más tradicionales, como el modelo en cascada, el cual es muy rígido y resulta muy costoso el llevar a cabo cambios en el proyecto debido a la poca adaptabilidad ante el cambio que presenta este modelo. Ante esto, se ha buscado una implementación que permita llevar a cabo modificaciones a lo largo de todo el proyecto de forma sencilla.

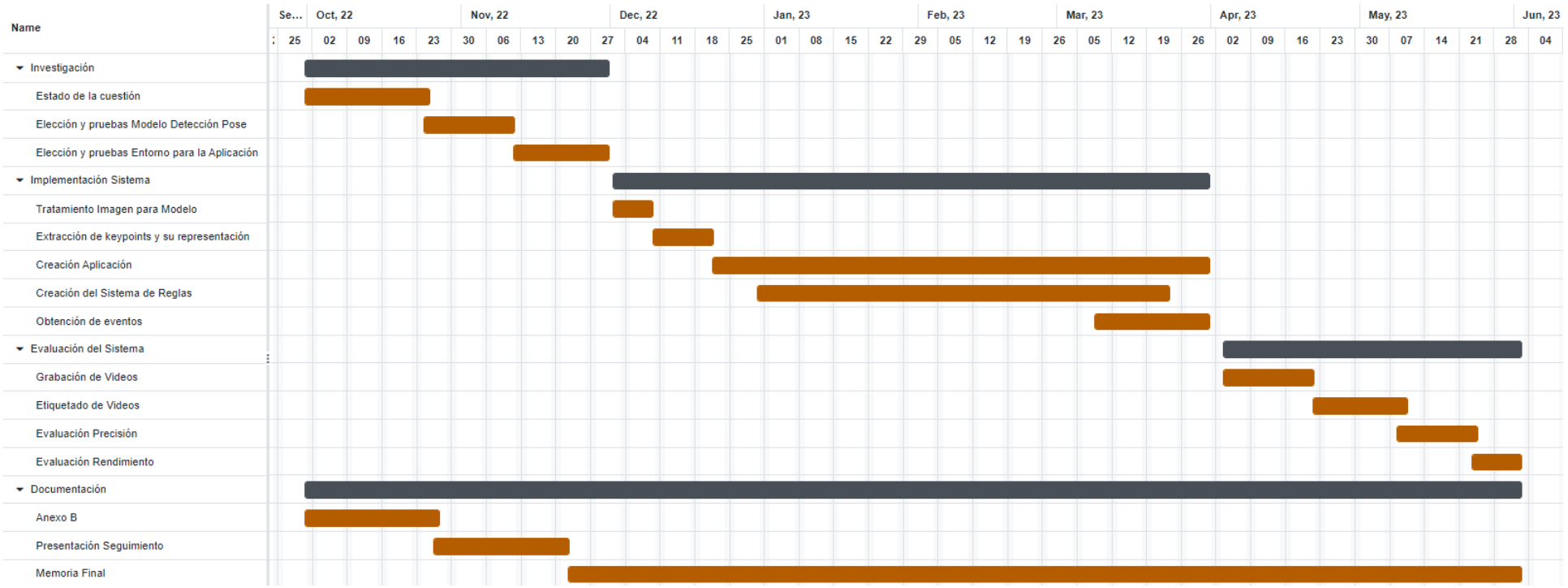

Figure 4.5: Diagrama de Gantt del proyecto

5 Sistema desarrollado

Esta sección está orientada a la descripción del sistema implementado. Se llevará a cabo el análisis del sistema, donde se tratará la estructura y arquitectura de este. Posteriormente, se explicará en mayor detalle el diseño de cada una de las partes que componen el proyecto y se finalizará con la explicación del proceso de implementación de este.

5.1 Análisis del sistema

Desde la visión más general del proyecto, como se muestra en la figura 5.6, se ha puesto en práctica la creación de un entorno que permita al usuario recibir información acerca de cómo está realizando una determinada actividad física gracias a partir de un sistema que analiza el desempeño aplicado en dicho ejercicio o movimiento para que el usuario pueda llevar a cabo la corrección de la práctica y mejorar la técnica empleada en la actividad. De ahora en adelante el sistema creado se le va a referenciar como *ADAP*, haciendo referencia a Aplicación de Detección y Análisis de Poses.

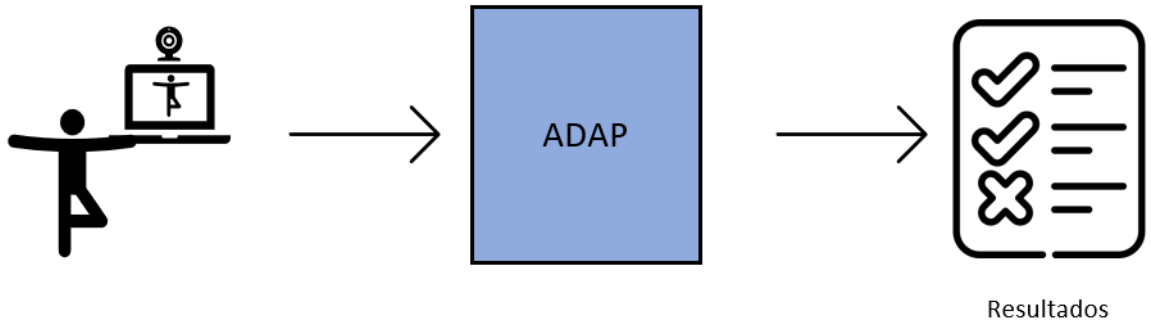

Figure 5.6: Visión conceptual del proyecto

El *ADAP* está compuesto por tres niveles/secciones fundamentales:

- **Procesado de imagen y detección de pose**: nivel enfocado al tratamiento de los frames procedentes del vídeo a analizar para su posterior procesado por el modelo de *Pose Detection* seleccionado. El resultado obtenido de dicho modelo permite el disponer de los puntos clave (*keypoints*) relativos al cuerpo del individuo que está llevando a cabo la actividad. A partir de la posición absoluta (coordenadas) de los puntos y la posición relativa entre estos, se logra obtener la pose realizada por el usuario en cada frame del vídeo.

- **Análisis de pose y obtención de eventos**: nivel encargado de la obtención de eventos y detección de errores propios del vídeo procesado por el modelo. Esto se realiza a partir de la agrupación de reglas generales, dichas reglas engloban el movimiento y posición de un conjunto de ciertas partes del cuerpo como las piernas, brazos o el torso. Se busca que estas reglas sean sencillas de definir permitiendo la implementación de reglas nuevas relativas a actividades físicas sin tener que ir generando nuevo código en función de que se de un ejercicio nuevo a analizar.

- **Aplicación Web**: interfaz Web que permite al usuario interaccionar con el *ADAP*, permitiendo la carga de vídeos para su respectivo análisis y obtención de resultados.

Las secciones mencionadas serán detalladas en los siguientes apartados pertenecientes a esta sección. A continuación, se muestra un esquema simplificado de como interactúan estos niveles entre sí y con el usuario.

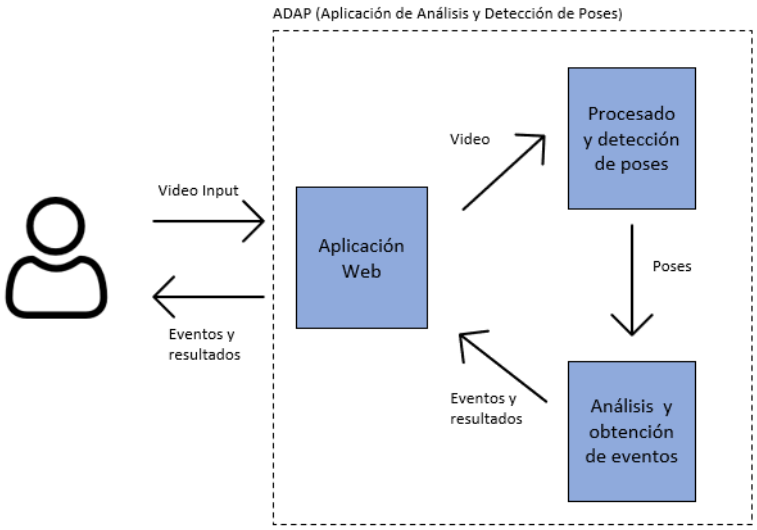

Figure 5.7: Esquema de las secciones del ADAP

5.2 Requisitos no funcionales del sistema

En esta sección se definen los requisitos necesarios que se deben cumplir por el sistema. Siguiendo los objetivos de la sección 4.2, aquellos relativos a la precisión y a la eficiencia se pueden cuantificar de forma sencilla para que al llevar a cabo la evaluación del sistema se pueda determinar si se han logrado o no. Otros objetivos tales como la adaptabilidad o la usabilidad son objetivos cualitativos y se valorarán siguiendo otros criterios en la sección 6.

- Precisión:
  - El porcentaje de frames detectados correctamente debe superar el 60%.
  - El 60% de los eventos que se den en un vídeo deben de detectarse correctamente.
  - El vídeo resultante con la pose detectada superpuesta sobre el original debe presentar el mismo *frame rate* que el vídeo de entrada.
- Eficiencia:
  - El tiempo de procesado de un vídeo no puede superar el 3.5 de la duración del mismo.
  - La carga del modelo de *Pose Detection* y de la aplicación no puede superar el medio minuto.
  - La descarga de resultados no puede superar el minuto.

Otros objetivos como la adaptabilidad o la accesibilidad no se pueden medir de manera numérica, pero cabe destacar que estos criterios junto a los mencionados anteriormente van a guiar el proceso del diseño

5.3 Diseño

En este apartado se explican los detalles del diseño de alto nivel del sistema y como responden a los requisitos y objetivos expuestos en el capítulo anterior. Posteriormente se va a ahondar en los subsistemas que componen y estructuran el *ADAP* y la funcionalidad concreta que cada uno de estos componentes implementan.

5.3.1 Funcionalidades y comportamiento del sistema

Este apartado hace referencia al diseño del sistema respecto a las funcionalidades que debe implementar la aplicación como a la secuencia de acciones que deben realizar los componentes principales del sistema ante el uso normal que le daría un usuario al sistema creado.

Funcionalidades del sistema

A partir de este diagrama de caso de uso se va a mostrar las funcionalidades que un actor, en este caso el usuario puede realizar en el sistema.

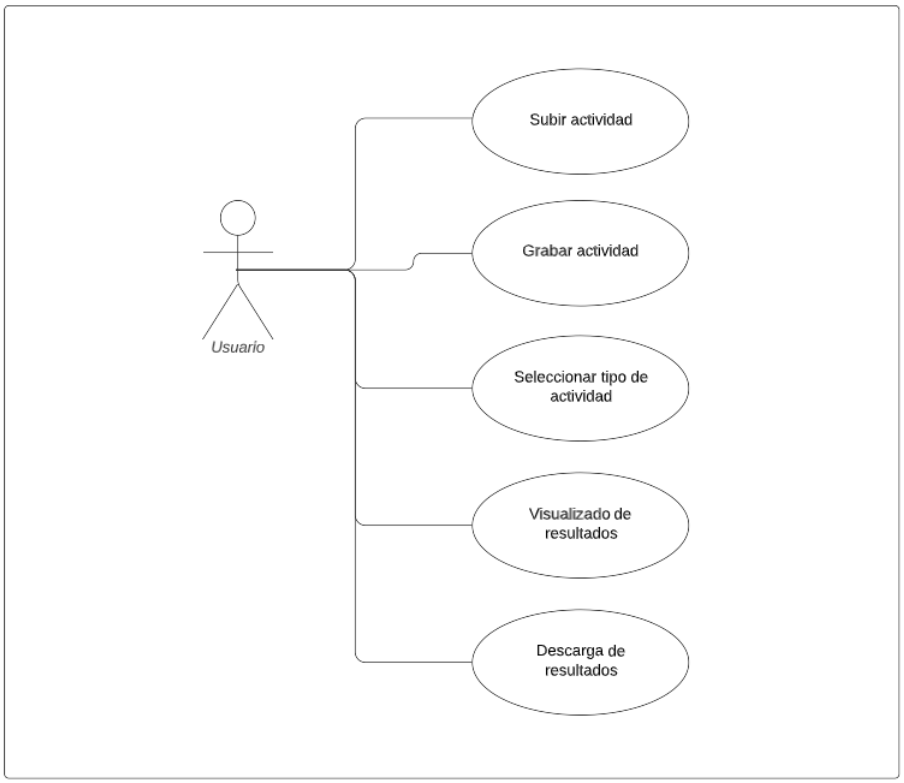

Figure 5.8: Diagrama de casos de uso del usuario

El usuario de la aplicación tiene a su disponibilidad el poder llevar a cabo las siguientes actividades:

* Cargar a la aplicación el vídeo deseado que se pueda procesar.

* Realizar la grabación de la actividad que desee en la propia aplicación.

* Seleccionar la actividad/categoría a la que pertenece el vídeo seleccionado para su procesado por el sistema.

* Visualizar los resultados de dicha actividad ya procesada por el sistema, seleccionando entre diferentes formas de representación de estos.

* Tener la posibilidad de descargar dichos resultados.

Comportamiento principal del sistema

A continuación mediante el siguiente diagrama, se va a mostrar cómo sería el orden de actuación de los principales componentes del sistema ante el caso de que un usuario quiera realizar la subida de un vídeo, para su procesamiento, visualizado de los resultados y la descarga de estos últimos.

1. En primer lugar, el usuario carga el vídeo a la aplicación.

2. Tras realizar la carga del vídeo, el usuario indica la categoría a la que el vídeo pertenece para que se lleve a cabo el procesado del vídeo y análisis correspondiente.

3. A continuación el vídeo es procesado y se obtienen los frames con las poses realizadas a lo largo de este.

4. Tras el procesado del vídeo,estas poses son analizadas y se obtienen los eventos propios a la actividad, tanto aquellos que han sido realizados correctamente como los que presentaban errores. Dichos resultados se devuelven para que estos puedan ser mostrados por la interfaz al usuario.

5. Por último, el usuario descarga los resultados referentes a la actividad realizada.

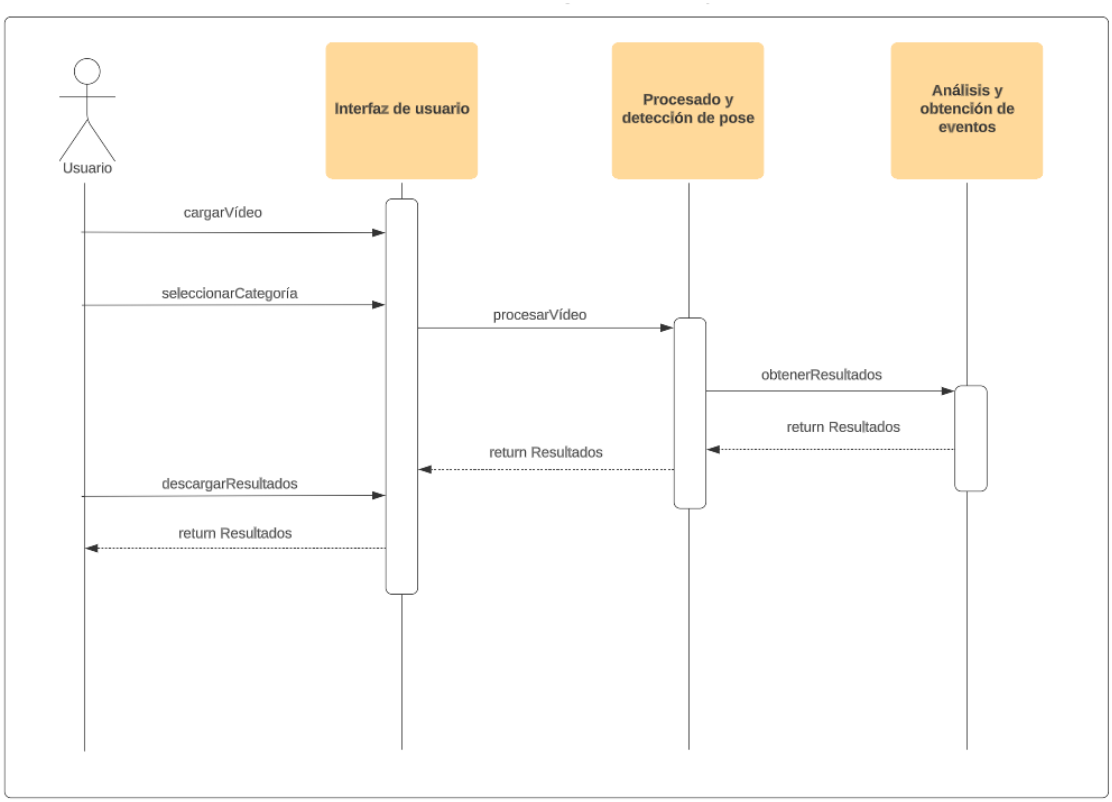

Figure 5.9: Diagrama de secuencia del sistema

5.3.2 Diseño de las secciones del sistema

En este apartado se definen las tres secciones principales que conforman el *ADAP*, mostrado en la figura 5.7 , detallando los subsistemas que componen cada una de estas secciones.

Procesado y detección de poses

Sección encargada de procesar el flujo de vídeo procedente del usuario, preprocesarlo para que se le pueda alimentar al modelo de *Pose Detection*, obtener los resultados de dicho modelo y convertirlos en información útil la cual pueda ser utilizada por el nivel de análisis. Movenet ha sido el modelo seleccionado y aunque ya se ha mencionado en el capítulo 2, es importante ahondar en su funcionamiento para una buena comprensión acerca de sus capacidades, requerimientos, configuración y modos de empleo.

Movenet utiliza la herramienta de aprendizaje computacional *TensorFlow*, a partir de esta herramienta se ha constituido el modelo de *Pose Detection* utilizando la arquitectura empleada en redes neuronales convolucionales (*CNNs*) como la que se muestra de ejemplo en la figura 3.4. Este modelo es de libre acceso y presenta dos posibles variaciones:

- **MoveNet Thunder:** versión enfocada a aplicaciones en las que la precisión de las predicciones tiene una mayor importancia que la velocidad de procesamiento. Solamente permite la detección de una entidad, en el caso de que se den varias personas dentro del marco del vídeo, el modelo seleccionará al individuo cuyas predicciones presenten un mayor grado de confianza.

- **MoveNet Lightning:**versión enfocada en velocidades de procesamiento menores que la versión *Thunder*, presenta una menor precisión pero permite la detección de múltiples entidades.

Cabe destacar que ambas versiones permiten la detección en tiempo real superando los 30 FPS. Para la realización del *ADAP* se ha seleccionado la versión *Thunder* debido a que la prioridad del entorno es detectar de forma correcta los eventos y errores que realice el usuario durante el transcurso de su actividad. El proyecto está enfocado en la detección y análisis de actividades físicas individuales por lo que utilizar una versión que permita una detección un mayor número de entidades no aportaría utilidad a la aplicación en el momento en el que se encuentra actualmente.

El modelo de detección requiere que se le suministren imágenes de una resolución específica, esto implica que los frames de vídeo a procesar deben ser convertidos de tal forma que cumplan con la resolución indicada sin que se de una pérdida de información sustancial que afecte a la precisión del modelo. Tras esto el modelo ingiere el frame de vídeo y se obtienen una estructura que contiene los 17 puntos del cuerpo humano detectados por el modelo. Por cada parte del cuerpo se obtiene su posición longitudinal y vertical, además de la el valor de confianza que presenta dicha detección. Este valor permitirá descartar aquellas mediciones que no cumplan un determinado mínimo, desechando una predicción que no aporte utilidad al sistema.

Una vez ya se han obtenido los resultados del frame procesado, a partir de la interconexión de estos puntos en el espacio, se puede trazar una imagen en dos dimensiones de la postura realizada por el usuario tal como se muestra en la figura 5.10.

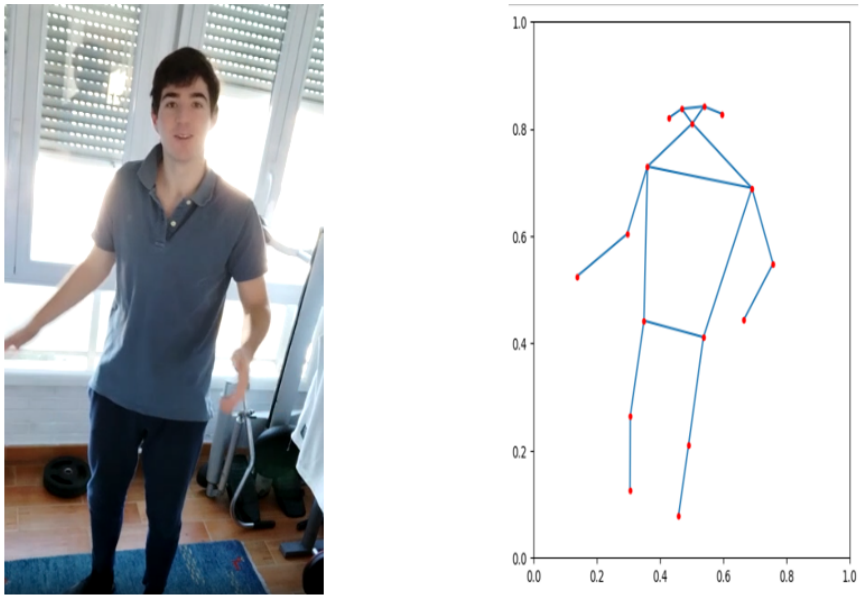

Figure 5.10: Imagen original y postura formada a partir de los resultados de MoveNet

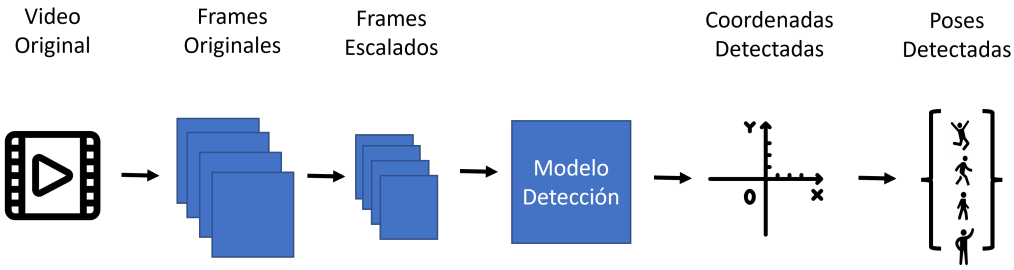

Figure 5.11: Proceso de obtención de las poses en un vídeo

En la siguiente figura (5.11) se puede observar el esquema del proceso de detección de pose comentado.

En este punto, estos resultados ya pueden ser analizados para la búsqueda de relaciones, reglas y criterios que permitan la detección de eventos asociados a diferentes actividades físicas. Toda esta labor se va a tratar en el siguiente apartado.

Análisis y obtención de eventos

Para llevar a cabo el análisis de las poses procedentes de los resultados obtenidos a partir del modelo es necesaria la creación de un sistema de reglas que permita transformar estos datos en eventos que definan las acciones que se están realizando en el vídeo. Se han implementado una serie de reglas que se consideran genéricas y son validas para muchos tipos de actividades. Para ello, se consideran agrupaciones semánticas de *keypoints* (por ejemplo un brazo) y se analizan en conjunto. Las reglas diseñadas abarcan los movimientos relativos a los brazos, piernas e inclinación del cuerpo.

Estas reglas generales que abarcan cada parte del cuerpo pueden agruparse de forma sencilla para crear un conjunto de reglas específicas enfocadas una actividad física determinada.

Análisis del brazo

Esta regla permite controlar si el ángulo de apertura del brazo indicado se encuentra dentro de un rango indicado. En primer lugar, se obtiene el producto escalar entre los vectores del hombro al codo y del codo a la muñeca. Posteriormente se calcula el coseno del ángulo entre ellos a partir del producto escalar y las magnitudes de dichos vectores. A partir de su inversa ya se obtiene el ángulo del brazo en radianes que posteriormente se transformará a grados para un uso más sencillo de la función.

A continuación se muestran las fórmulas utilizas para llevar a cabo la obtención del ángulo a partir de dos vectores. En este caso se u y v harían referencia a los vectores hombro-codo y codo muñeca respectivamente. Sobre estas fórmulas se rige el diseño de las reglas utilizadas en el sistema.

1. Producto escalar entre dos vectores

$$\vec{u} \cdot \vec{v} = u_1 v_1 + u_2 v_2$$

2. Obtención de el coseno a partir del producto escalar de los vectores y el producto de su módulo

$$\cos \alpha = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}|\,|\vec{v}|}$$

3. Obtención del ángulo a partir del uso del arco-coseno

$$angle = \arccos\left(\cos \alpha\right)$$

Por ejemplo, el valor teórico del ángulo en el caso de que en el frame se de un brazo recto y completamente estirado provocará que los vectores hombro-codo y codo-muñeca estén alineados, lo que significa que el ángulo entre ellos es cercano al nulo. Como resultado, el producto escalar de los dos vectores será máximo y su magnitud alcanzará su valor más alto. 5.3.2

A medida que el brazo se dobla, el ángulo entre el hombro, codo y muñeca variará. La magnitud del ángulo dependerá de la flexión del codo. En el caso de que el brazo se doble aproximadamente a la mitad, el ángulo entre el hombro, el codo y la muñeca será de aproximadamente 90 grados. A medida que el brazo continúa su flexión, el ángulo aumentará. Cuando el brazo está completamente doblado, el ángulo formado será de aproximadamente 180 grados.

Cabe destacar que en la realidad es improbable que se den detecciones de ángulos tan cercanos a los obtenidos de forma teórica debido a las limitaciones de flexión y extensión que presentan las articulaciones del cuerpo humano. Por lo tanto, es importante que dependiendo de la actividad, se defina un rango de ángulos adecuado para realizar una correcta detección, a la par que también se deben tener en cuenta ciertos márgenes de error a causa de posibles variaciones en la propia detección. Además del ángulo de estiramiento del brazo, a la función también se le puede asignar una altura objetivo a la que debe encontrarse el brazo, debido a que en muchos movimientos no solo basta con la flexión del brazo para considerarse correcto sino que también la posición de este es clave para realizar el movimiento de forma adecuada.

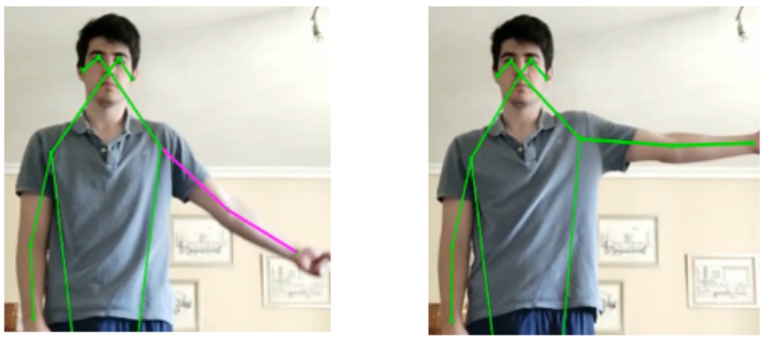

Figure 5.12: Comparativa pose incorrecta/correcta

A continuación, en la figura 5.12 se muestra un ejemplo simple de detección en el brazo izquierdo del usuario, donde se ha establecido que la postura es considerada como correcta cuando el brazo se encuentra extendido y además la muñeca debe permanecer a la altura del hombro.Se ha optado como opción visual actualizar el color de la pose en función de la detección, asignando el color magenta a aquella pose que no este bien realizada y verde a la que cumpla con los criterios específicos de la actividad.

Análisis de la pierna

El funcionamiento de esta regla tiene un comportamiento muy similar a la detección realizada en los brazos. La única variación que se da en el diseño de esta corresponde a la obtención de los vectores necesarios para hallar el ángulo de flexión de la pierna. En este caso a partir de los vectores cadera-rodilla y rodilla-tobillo se obtiene dicho ángulo.

Análisis del ángulo de inclinación

Esta detección difiere de las dos anteriores ya que el objetivo de esta se centra en detectar la inclinación que presenta el cuerpo al realizar la actividad. La inclinación se va a medir a partir del siguiente procedimiento.

1. En primer lugar se calcula el punto central de los hombros y el de las caderas en el frame.

2. A partir de estos se genera el vector que une dichos centros.

3. Calcular un nuevo vector que una un punto externo de la cadera, ambos extremos de la cadera serían válidos, en este caso se ha optado por el izquierdo.

4. Una vez se tienen los vectores se procede a calcular el ángulo de inclinación a partir de las fórmulas ya mencionadas anteriormente (Fórmulas 5.3.2)

5. Por último, conocido el ángulo se debe definir para qué rango de ángulos se considera que el usuario se encuentra con una postura recta o inclinada de alguna forma.

Si el usuario se encuentra de perfil ante la cámara el sistema podrá detectar si el usuario presenta una postura recta o realizando una inclinación hacia delante o hacia atrás calculando automáticamente hacia donde está mirando el usuario, en función de la posición relativa de la nariz con el hombro y oreja que estén mirando hacia la cámara. Por otro lado, en caso de que el usuario se encuentre de frente a la cámara, el sistema será capaz de detectar también que se encuentra recto o inclinándose a la izquierda o derecha.

Un detalle de gran importancia en esta detección es la limitación que presenta el uso de un modelo de *Pose Detection* en dos dimensiones tal como Movenet. El uso de una sola cámara para grabar la actividad impide obtener una pose con profundidad, es decir, tres dimensiones. Esto limita la funcionalidad de la detección y se debe conocer la posición que presenta el usuario ante la cámara. Por lo tanto, movimientos que incorporen inclinaciones en profundidad no podrán ser catalogadas completamente. A pesar de esto, se ha planteado que el *ADAP* sea un entorno sencillo en el que el usuario pueda subir sus actividades físicas al sistema, grabando con la cámara de su teléfono u ordenador

sin la necesidad de tener que llevar a cabo la instalación de varias cámaras que graben simultáneamente.

A continuación, se muestra ejemplos de la detección de inclinación por parte del usuario en dos vídeos, uno mirando hacia la cámara y el segundo de perfil. Se ha establecido el permanecer erguido como acción correcta en estos casos.

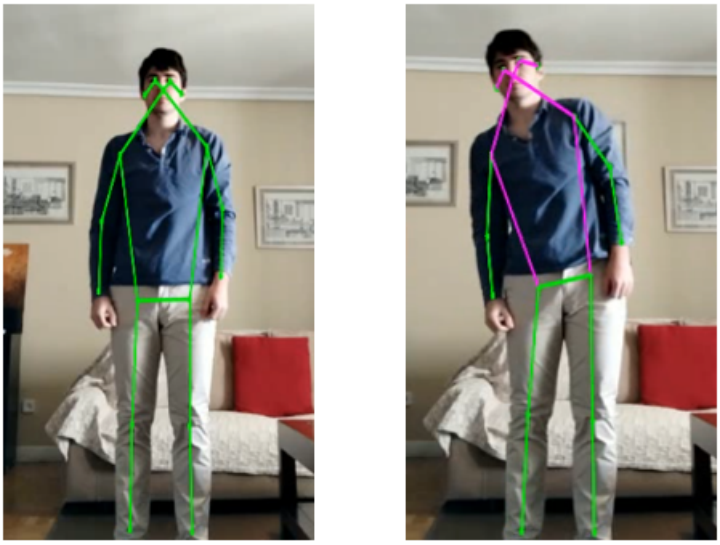

Figure 5.13: Comparativa inclinación (directa) correcta/incorrecta

Figure 5.14: Comparativa inclinación (perfil) correcta/incorrecta

Creación de reglas personalizadas

El diseño de las reglas generales anteriormente mencionadas presenta una serie de beneficios tales como su modularidad y facilidad de implementación. Gracias a que las reglas están enfocadas en funciones independientes, estas realizan tareas diferentes unas de otras, lo que facilita la comprensión del sistema además de que permiten la modificación individual de dichas funciones sin alterar el funcionamiento de las restantes.

Una vez se dispone de reglas generales que abarcan los diferentes campos del cuerpo, ya se puede llevar a cabo el uso simultáneo de estas reglas en una misma imagen para realizar la elaboración de detecciones más complejas que se adecúen a la actividad física que se quiere analizar. Es de importancia destacar que para la creación de reglas de análisis de pose efectivas requiere el estudio previo del movimiento a analizar, tanto de como se realiza correctamente como los posibles errores comunes que se pueden dar durante la ejecución de dicha actividad.

Un ejemplo de la creación y configuración del conjunto de reglas necesario para llevar a cabo un ejercicio se llevará a cabo en el capítulo siguiente.

Otra limitación que se da es el número reducido de actividades analizadas debido a la necesidad de la creación de las reglas personalizadas y su debido proceso de evaluación, el cual se abarcará en el capítulo 6.

La implementación de la creación de un conjunto de reglas específicas para una actividad se detalla en el siguiente capítulo, en la sección 5.4.4

Obtención de eventos

Una vez se obtienen las detecciones frame a frame se debe llevar a cabo el proceso de obtención de eventos. Es decir, se deben obtener aquellos momentos en la actividad que sean determinantes y que aporten gran utilidad al usuario. Esto es de gran importancia, ya que la información obtenida por cada frame puede ser repetitiva.Esto se debe a la naturaleza de las actividades que se están analizando, donde cabe esperar que en un espacio de un frame las poses no presenten grandes variaciones,por lo que un gran numero de detecciones redundantes.

Además de esto, el modelo de detección de pose y puede presentar fallos a la hora de detectar los puntos clave del cuerpo del usuario. Fallos de enfoque o disminución de la calidad de las imágenes de entrada,movimientos muy rápidos o la oclusión del cuerpo del usuario por objetos u otras entidades son ejemplos comunes que pueden provocar errores de predicción por parte del modelo.

En la siguiente imagen 5.15 se muestra un ejemplo donde la cámara al no tener visión directa con la cara del usuario, predice de forma incorrecta los puntos clave de este, generando una pose muy dispar a la original.

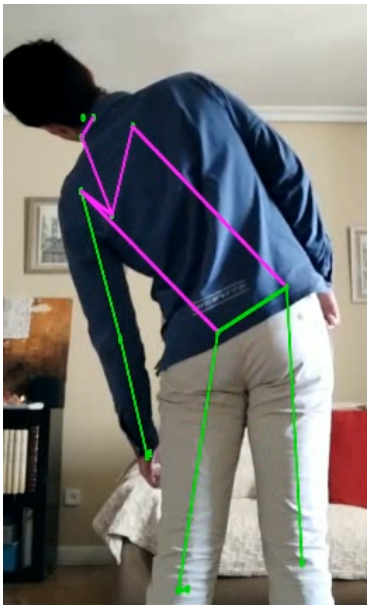

Figure 5.15: Ejemplo de fallo de predicción del modelo

Como solución al exceso de información y a los posibles errores momentáneos en la predicción del modelo, se ha optado por facilitar el consumo de la información, llevando a cabo la eliminación de eventos duplicados y posibles errores, permaneciendo unos eventos únicos que que representan errores u aciertos concretos en el desarrollo de la actividad.

En primer lugar, el conjunto de resultados de los frames se irá agrupando en eventos principales en función de si se dan al menos cinco frames con el mismo resultado. Además de almacenar el resultado común, el evento tiene asignado el intervalo de frames en el que se da dicho evento. A partir de este intervalo, posteriormente se podrá realizar la conversión al intervalo en segundos donde se detecta dicho evento en el vídeo para que aporte una mayor utilidad al usuario. Aunque esta agrupación puede resultar ya útil, se pueden dar eventos contiguos que presenten el mismo resultado en dos intervalos diferentes pero próximos. Esto se debe a que el proceso trata de detectar e ignorar los frames en los que se den fallos posibles en la detección. Ante esto, para finalizar la agrupación, se fusionan estos eventos contiguos e iguales conformando uno solo y actualizando su intervalo de aparición.

Figure 5.16: Proceso de obtención de eventos

Aplicación Web

La aplicación permite la interconexión de todas las secciones comentadas anteriormente, permitiendo al usuario de forma sencilla acceder a todas las funcionalidades del *ADAP* a través de su interfaz. Tal como se ha mostrado en la sección 4.2, el diseño de la aplicación debe fomentar la usabilidad a la herramienta, por lo que en la creación de las diferentes páginas que componen la interfaz web se ha buscado un enfoque simple, limpio y que no sobrecargue de información al usuario. En la figura 5.17 se muestra el diagrama de navegabilidad de la apreciación, donde cada página presenta sus funcionalidades específicas.

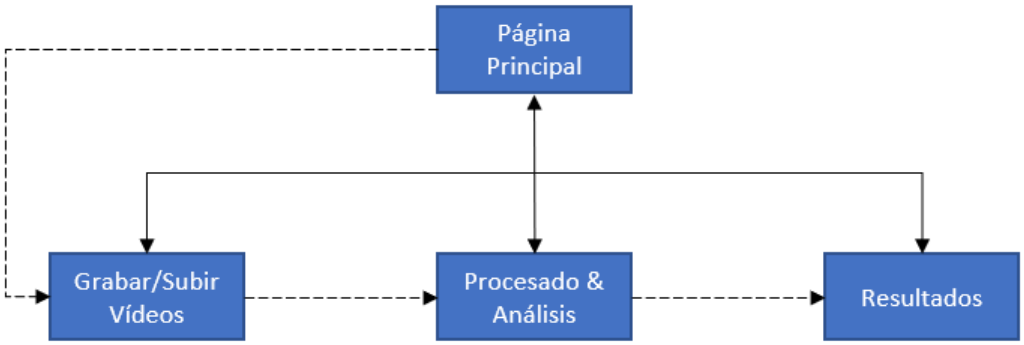

Figure 5.17: Diseño de navegabilidad de la interfaz web

La aplicación presenta la página principal, que introduce el objetivo principal de la aplicación y sus funciones. Las líneas discontinuas marcan el recorrido guiado que se indica en la propia interfaz para que el usuario pueda hacer uso de todas las funcionalidades que presenta la aplicación. Desde la carga o subida de un vídeo por parte del usuario, al análisis y procesado de la actividad física y finalizando con la representación de los resultados obtenidos en forma de tablas y gráficas. Por otro lado, un usuario que presente ya conocimiento acerca del funcionamiento de la aplicación debe de poder navegar por la aplicación tal como desee, ante este caso todas las páginas se encuentran interconectadas entre sí por lo que el acceso a estas no se encuentra limitado.

Este diseño de la interfaz fomenta un uso rápido de la aplicación permitiendo al usuario acceder a los resultados que ya presenta procesados sin la necesidad de repetir la carga y procesamiento de la actividad en cuestión. Otra posibilidad,por ejemplo, se puede dar en que el usuario quiera procesar múltiples vídeos de forma seguida antes de observar sus resultados. En resumen, la sencillez y flexibilidad de la aplicación permiten un uso que se acomode a los intereses del usuario en cada momento.

5.4 Implementación del sistema

En esta sección del capítulo se va a detallar los aspectos más importantes acerca de la implementación del diseño expuesto en el apartado anterior. Se va a abordar desde una visión más detallada, el proceso desde que un usuario carga su actividad física realizada hasta que recibe los resultados propios del procesado y análisis.

5.4.1 Carga y/o subida de un vídeo a la aplicación:

Es fundamental para el correcto funcionamiento del sistema que el vídeo introducido por el usuario se cargue de forma correcta en la aplicación. Para ello el usuario tiene la posibilidad de subir una actividad ya grabada o de realizar una grabación desde la propia aplicación.

Grabación desde la aplicación

Gracias al uso de *Streamlit*, se pueden emplear una gran cantidad de módulos creados por la propia comunidad. El módulo que permite realizar la grabación desde la interfaz web, recibe el nombre de *streamlit-webrtc*[21], mediante este módulo se permite capturar la entrada de vídeo en tiempo real procedente del dispositivo que esté ejecutando la aplicación y es el usuario el que a partir una lógica sencilla de botones el que puede indicar el comienzo y finalización de la grabación. Tras finalizar, el vídeo se guarda en el sistema con el nombre seleccionado por el propio usuario.

Figure 5.18: Sección de grabación (interfaz)

Carga de vídeo previamente grabado

Esta opción también hace uso de un módulo *file uploader* propio de *Streamlit*, este no requiere de configuración adicional y permite al usuario seleccionar el archivo propio que desee, este en primer lugar se almacenará en la memoria caché de la aplicación y posteriormente se va a redirigir a la carpeta asociada a las grabaciones relativas al usuario.

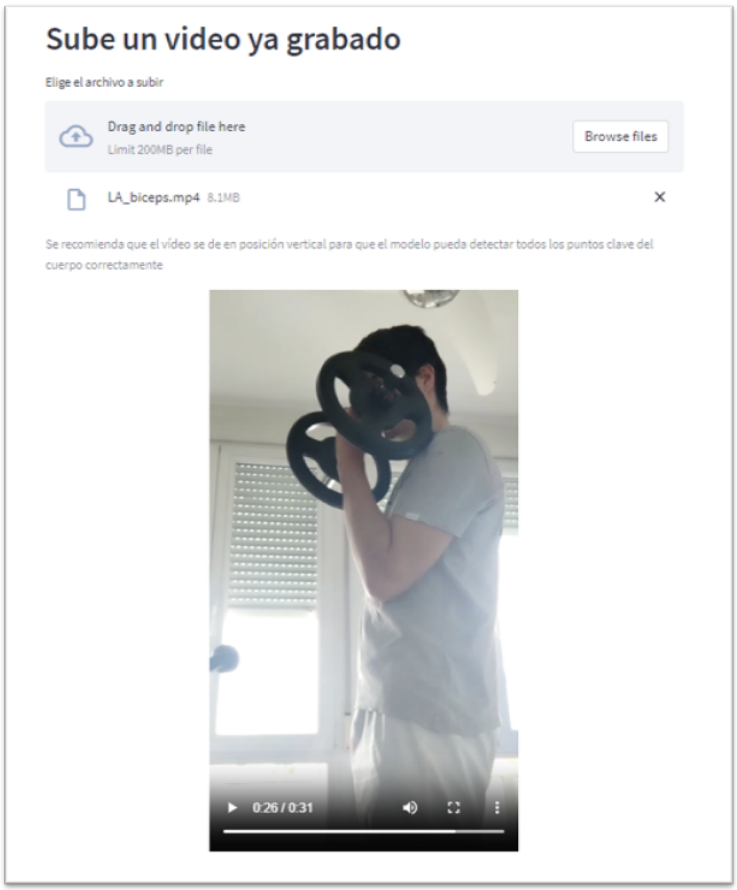

Figure 5.19: Sección para la carga de vídeos (interfaz)

5.4.2 Selección de categoría para su procesado y análisis:

Tras la carga del vídeo deseado el usuario puede acceder a todos los vídeos que han sido previamente cargados y seleccionar el tipo de ejercicio/categoría a la que pertenece la actividad física que ha realizado. Esta categoría determinará las reglas de análisis que se emplearan tras las detecciones realizadas por el modelo de *Pose Detection*.

Además, el usuario debe indicar cómo se ha realizado el vídeo grabado, indicando si se ha realizado con una cámara que presente modo espejo, como por ejemplo las lentes frontales de los móviles, o si se ha realizado con una lente que no invierta la imagen de salida. Es de gran importancia tener constancia de la lente que se utiliza para en el caso de que sea necesario, invertir de nuevo la imagen para que las detecciones sean realizadas correctamente. Se ha planteado que toda imagen que sea recibida por el sistema y se haya declarado como modo espejo, sera invertida de nuevo para que sea tratada tal como si de una lente sin inversión se tratara.

```
1     if (orientation==1): # 1-> Modo espejo; 0-> Modo normal
2         frame = cv2.flip(frame,1) # Inv. horizontal
```

Listing 1: Rotación de imagen

A la hora de llevar a cabo la grabación, hay que tener en cuenta una serie de consideraciones respecto al entorno en el que se lleva a cabo. Ciertos casos pueden producir errores en la detección en el estado actual en el que se encuentra el modelo. Ante esto, es importante que las grabaciones se den lugares en los que no se de una gran cantidad de objetos, imágenes o fotos de figuras humanas o personas físicas que puedan desviar la atención del modelo.También se recomienda que las grabaciones se realicen en formato vertical, englobando toda la figura que participa en la actividad, para que el sistema pueda detectar correctamente los puntos clave del cuerpo y las relaciones que se den entre estos.

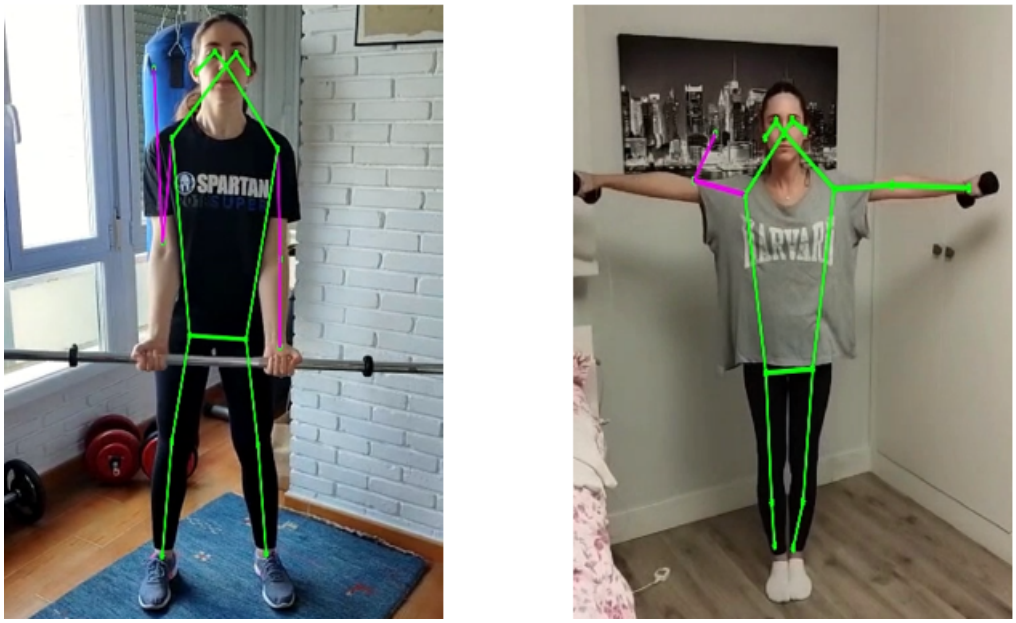

Figure 5.20: Ejemplos de errores producidos por el entorno

5.4.3 Preprocesado de vídeo y paso por el modelo de detección de pose:

Una vez se ha llevado a cabo la obtención de la actividad física es necesario adaptar el formato de vídeo a uno que el modelo de *Pose Detection* pueda procesar correctamente. Tal como se comentó en la sección de 5.3.2, el modelo *Movenet Thunder* solo permite la ingesta de frames que presenten una resolución específica, en esta versión solo admita imágenes cuadradas de resolución 256x256 píxeles. La aplicación permita la introducción de vídeos con diferentes tamaños por lo que en un primer lugar se estandarizan todos a un

tamaño cuadrado de 1280x1280, este tamaño es el suficiente para la conservación de los detalles y de la calidad de las imágenes de entrada.

Para realizar el redimensionado de la imagen existen diferentes métodos de interpolación [22]. Para este proyecto se ha escogido el método de interpolación bilineal, el cual permite el cálculo de nuevos píxeles a partir del promedio ponderado de los píxeles circundantes. A pesar de que el nivel de detalle de la imagen escalada disminuye respecto a otros métodos tales como la interpolación de *Lanczos* o la bicúbica, se ha optado por el método bilineal debido a que es una opción que permite el dimensionado de imágenes con la suficiente calidad y un tiempo de procesado menor que favorece que la duración total del tratamiento de la imagen no sea muy elevado.

En la figura 5.22 se puede observar como la calidad de las imágenes resultantes a partir de los métodos de interpolación es muy similar por lo que el método a seleccionar seleccionado será aquel que procese dichos frames de la forma más rápida. Tal como muestra la gráfica 5.21, el método más eficaz es la interpolación bilineal y por ende el método elegido. Cabe mencionar que la interpolación bicúbica presenta tiempos de procesado muy similares, por lo que podría ser considerado un método válido para cumplir los objetivos del proyecto.

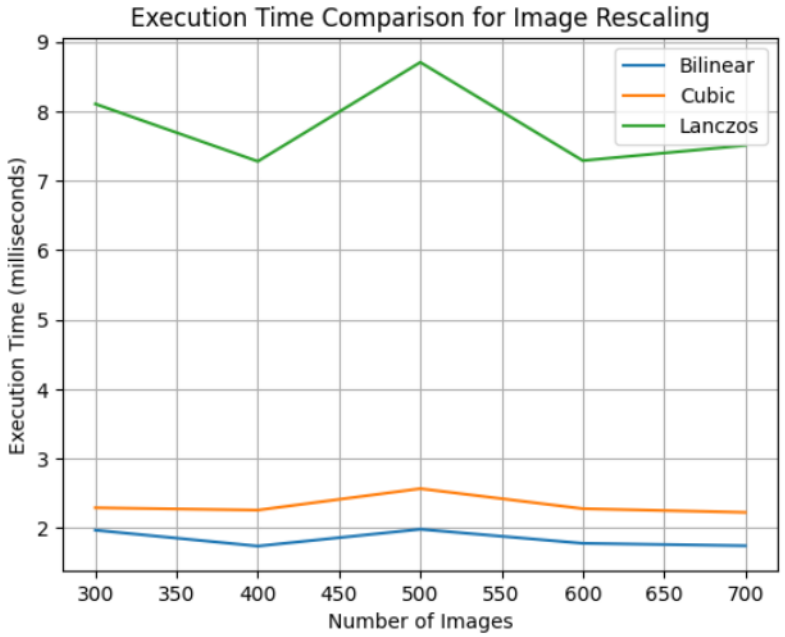

Figure 5.21: Tiempos de ejecución de los métodos de interpolación

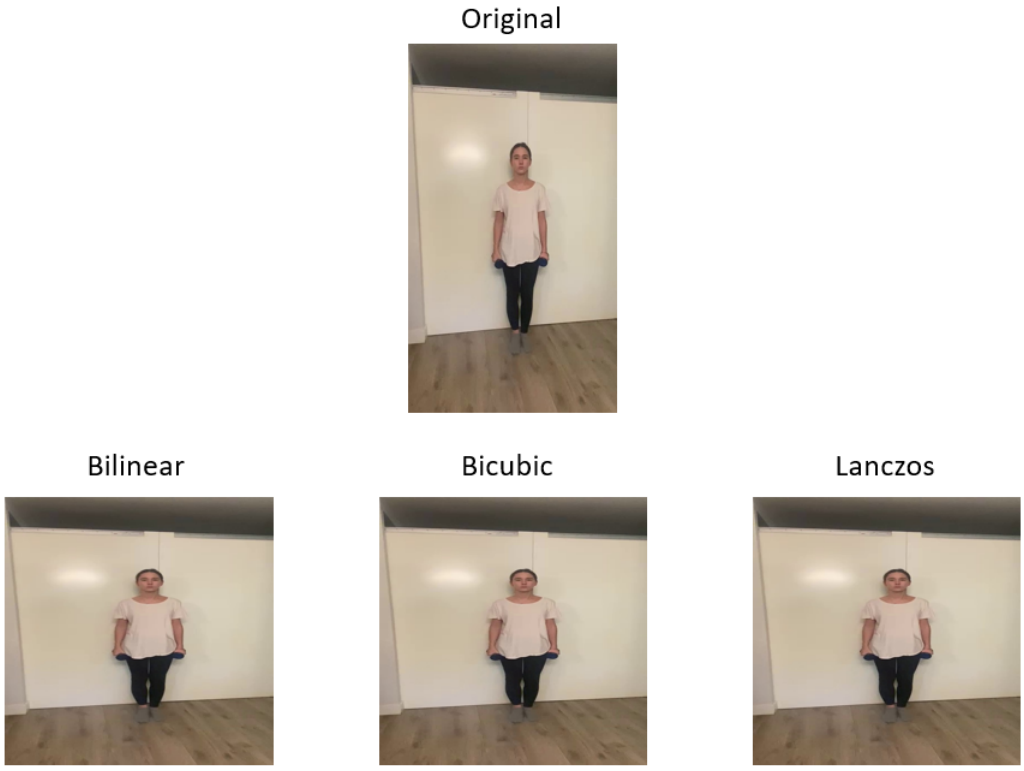

Figure 5.22: Comparación de las imágenes dimensionadas a 1280x1280 con la original

Tras llevar a cabo la reestructuración de la imagen esta se convierte a un tensor, este corresponde a la estructura fundamental con la que se representan arrays multidimensionales en modelos implementados en *TensorFlow*. Un tensor es una estructura inmutable, por lo que las operaciones que se realizan sobre esta herramienta de *Deep Learning* generan como salida un nuevo tensor a modo de respuesta al tensor de entrada. Una vez realizada la conversión de la imagen, el modelo de detección de pose calcula el nuevo tensor con los resultados de la detección. Se accederá a dichos resultados y se realizará un *mapping* que permita emplear estos de una forma más sencilla. Este consiste en obtener un dataframe de *pandas* donde cada columna represente la coordenada horizontal o vertical de cada punto clave detectado del usuario.

nose_y	nose_x	left_eye_y	left_eye_x	right_eye_y	right_eye_x	left_ear_y	left_ear_x	right_ear_y	...
0.363752	0.552718	0.422784	0.616731	0.410237	0.518583	0.393841	0.724504	0.355892	...
0.364339	0.552933	0.422821	0.616403	0.410439	0.520101	0.392510	0.723426	0.357344	...
0.366598	0.554964	0.423877	0.618517	0.413031	0.523071	0.393235	0.725337	0.357779	...
0.366880	0.555805	0.423435	0.619193	0.412186	0.522894	0.392172	0.729117	0.356766	...
0.366618	0.555844	0.423681	0.619196	0.411857	0.522769	0.393822	0.729518	0.355551	...

Figure 5.23: Dataframe (Pandas) obtenido tras el mapping del tensor de salida

5.4.4 Análisis de la actividad en función de la categoría:

Una vez ya procesado el vídeo y se disponen los frames del usuario con las detecciones ya realizadas, es el momento de que el sistema de reglas reciba esta información junto a la categoría del ejercicio seleccionada. Esta categoría permite la asignación de la combinación personalizada de reglas generales específica para el ejercicio en cuestión.

A continuación, se va a detallar a modo de ejemplo, el proceso de configuración del conjunto de reglas necesario para llevar a cabo un ejercicio. Este consiste en realizar una dominada isométrica (*isometric chin-up*) en la que el agarre de la barra se realiza con las palmas de las manos mirando hacia el cuerpo. Este ejercicio requiere que el usuario tire de su cuerpo aplicando fuerza de forma controlada en sus brazos y abdomen hasta que sus hombros alcancen la altura de la barra. Por lo tanto, para llevar a cabo el análisis de la actividad se debe aplicar las reglas generales de los brazos e inclinación y llevar a cabo su configuración. La posición y flexión de las piernas no influye en el resultado de esta actividad.

- **Análisis de los brazos:** se deben de aplicar una regla por cada brazo, para considerar el movimiento de los brazos como correcto, cada brazo debe permanecer doblado y la altura de las muñecas debe aproximarse a la altura de los hombros.

- **Análisis de inclinación:** para considerar la postura del cuerpo como correcta, el cuerpo debe de estar recto sin permitir que se den inclinaciones.

En l figura 5.24 se pueden observar dos imágenes, en la primera el usuario presenta el torso recto pero los brazo a pesar de que se encuentran doblados, los hombros no alcanzan la altura de las muñecas, mientras que, en la siguiente imagen el usuario se encuentra realizando la pose correctamente manteniendo el torso recto y los brazos doblados y a la altura correspondiente. Tal como se ha mencionado anteriormente, la posibilidad de creación de nuevas reglas tiene un gran potencial gracias a la modularidad de las reglas generales.

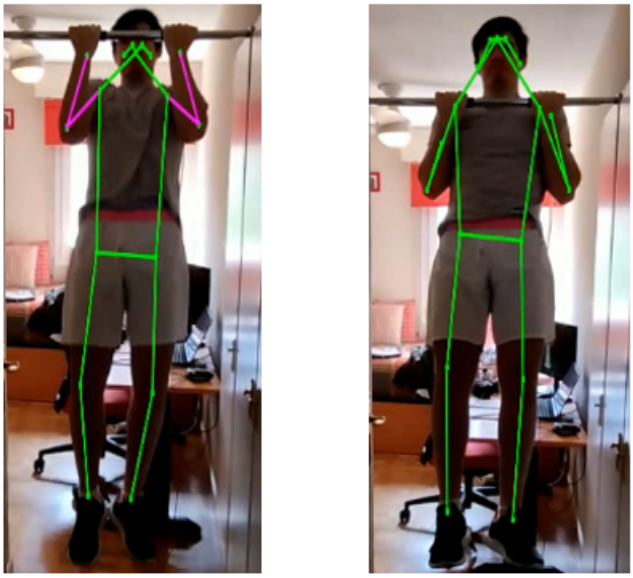

Figure 5.24: Comparativa dominada incorrecta/correcta

Cada ejercicio físico tiene asignada una serie de reglas, pero a pesar de esto, todos presentan una misma estructura y funcionamiento. Esto permite la reutilización de código además de posibilitar el almacenamiento de resultados utilizando los mismos procedimientos. Cada tipo de ejercicio devuelve una serie de diccionarios, cuatro en total y cada uno enfocado a una funcionalidad diferente.

1. **Diccionario de detecciones:** por cada frame que recibe el sistema se devuelve un diccionario con todas las detecciones en función de la parte del cuerpo utilizada y analizada por las reglas. Este diccionario es le que se emplea para llevar a cabo la obtención de eventos ya comentada.

2. **Diccionario booleano:** se elabora un diccionario que reúne en forma booleana si las reglas asignadas a dicho ejercicio se han cumplido de forma individual en el frame procesado. Este diccionario se empleará para la creación de un nuevo vídeo donde se superponga la pose detectada sobre el cuerpo del usuario presentado en el vídeo original.

3. **Diccionario de ángulos:** se genera un diccionario que almacena los ángulos calculados en el frame por las reglas generales. Una vez se obtengan todos los diccionarios relativos a cada frame de vídeo, se generarán gráficas para mostrará dichos ángulos al usuario.

4. **Diccionario de evaluación:** se establece un diccionario sencillo que contiene los resultados necesarios para considerar que la detección se ha realizado de forma correcta. Este diccionario está generado para llevar a cabo la evaluación de la precisión de las reglas establecidas. El funcionamiento y proceso de la evaluación será definido en el capítulo 6.

5.4.5 Representación de resultados y gráficas

Una vez se han almacenado toda la información referente a la actividad física, esta se le devuelve al usuario en múltiples formatos.

Detección de pose en vídeo

Tal como se ha mencionado en el apartado 5.4.3, el sistema genera un vídeo con la pose detectada superpuesta sobre el vídeo original. Refiriendo a lo que se comentó en el diseño perteneciente a la sección 5.3.2, a partir de la interconexión entre los puntos clave detectados por el modelo se puede representar en el espacio la pose realizada por el usuario en cada frame. A partir de esta pose y el diccionario booleano asignado al mismo frame, el cual contiene el estado de las secciones al realizar el movimiento seleccionado, se puede elaborar un vídeo que resulta atractivo y claro para el usuario sobre qué partes del cuerpo están actuando y sobre si la pose realizada en ese instante se encuentra en la posición objetivo o no.

Se mencionó en la sección 5.3.2 que las partes del cuerpo que realicen la acción y se encuentren dentro del objetivo determinado por las reglas asignadas a dicho ejercicio, serán representadas en color verde. Por otro lado, aquellas partes del cuerpo que no alcancen los requisitos de la actividad, mostrarán un color magenta y los eventos generados indicarán qué es lo que está provocando que dicho movimiento no sea considerado como correcto. Es decir, la representación solo se encontrará en verde al completo cuando todas las reglas individuales indiquen que la posición objetivo ha sido alcanzada.

Las figuras 5.12, 5.14 y 5.24 reflejan muestras de los vídeos resultantes. La siguiente figura muestra de forma esquemática el proceso que se ha implementado para generar el vídeo con la detección incorporada.

Para realizar la creación del vídeo es muy importante tener en cuenta el códec que se utiliza para formar el vídeo. Un códec de vídeo [23] está compuesto por una serie de algoritmos que permiten codificar y decodificar archivos digitales de vídeo para que su almacenamiento y transmisión sea más eficiente. H264, es el seleccionado para la realización de este proyecto debido a que presenta una alta tasa de compresión, reduciendo el tamaño del archivo original considerablemente a la par que mantiene una alta calidad de la imagen. También, es altamente compatible con la mayoría de plataformas y navegadores

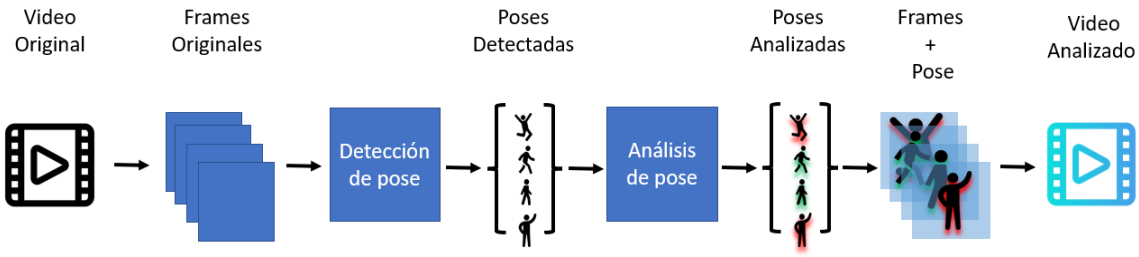

Figure 5.25: Proceso de creación del vídeo analizado

web, siendo un estándar aceptado y ampliamente utilizado para realizar transmisiones a través de Internet. Por lo tanto, la selección de este códec es una medida adecuada para la carga de vídeos por aplicaciones web tal como se realiza mediante *Streamlit*.

Representación de resultados y gráficas

Mediante el uso de los *dataframes* con los puntos clave y la agrupación de los diccionarios que almacenan los eventos y ángulos realizados a lo largo de la actividad, se elaboran métricas y resultados que permiten al usuario comprender en mayor medida cómo ha actuado y conocer cómo deberá modificar sus movimientos para corregir los posibles errores que haya cometido.

A partir de la representación de los eventos principales, el usuario puede reconocer los momentos claves de la actividad ejecutada donde se determinan los errores y aciertos cometidos. Por otro lado, a partir de representaciones gráficas, el usuario puede conocer el estado y posición de cada parte de su cuerpo en cada instante del vídeo procesado y aportando gran información acerca de la ejecución de su movimiento. Representaciones tales como la posición de los puntos clave del cuerpo a lo largo del ejercicio permite observar si se dan picos o si se mantienen estables las partes del cuerpo encargadas de realizar dicho movimiento. La siguiente gráfica representa la posición de la nariz durante el ejercicio de realizar una dominada isométrica.

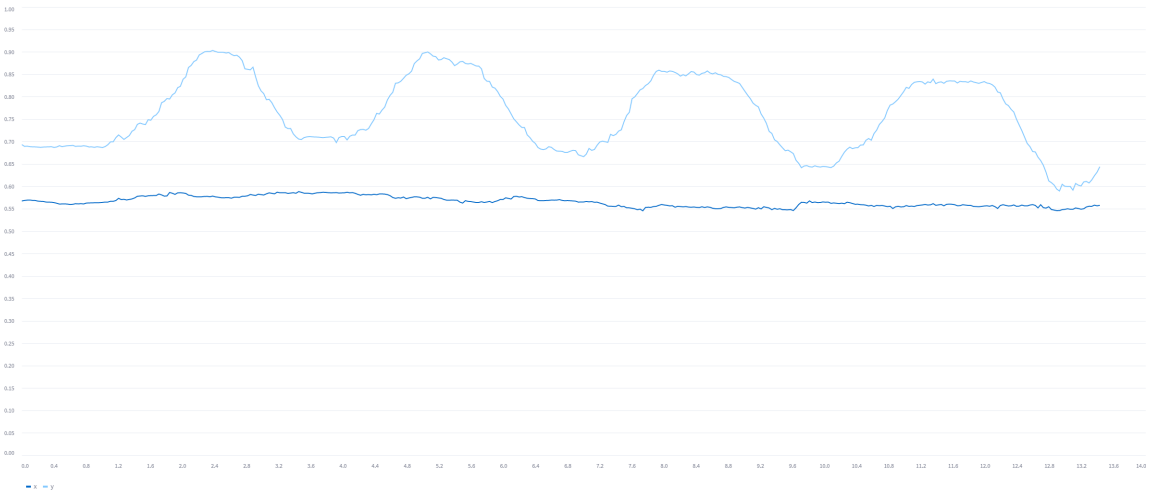

Figure 5.26: Variación de las coordenadas de la nariz durante toda la acción

La señal azul clara muestra la coordenada y ,mientras que la señal más oscura representa la coordenada x. En este ejemplo se puede ver como el usuario mantiene la cabeza firme y esta solamente asciende y desciende en los momentos que este realiza una dominada. Además se puede observar el factor que hace el cansancio sobre el usuario, ya que los picos de las oscilaciones,los cuales representan los máximos que alcanza la nariz, cada vez son menores a medida que aumentan las repeticiones.

Por otro lado, se dan representaciones de los ángulos calculados para cada parte del cuerpo que ha intervenido en el movimiento. En estas gráficas se delimita un rango de valores en los que el ángulo realizado debe darse para que se considere que dicha parte del cuerpo esté actuando correctamente respecto al movimiento. La siguientes figuras 5.28 y 5.27 aluden de nuevo al ejercicio de la dominada isométrica realizada correctamente en la imagen derecha de la figura comparativa 5.24. Las gráficas muestran el ángulo actual en el instante en el que se encuentra dicho frame respecto al brazo derecho y la inclinación del cuerpo. El rango permitido para considerarse como correcto está representado en verde y los ángulos fuera de rango están representados en rojo. Además, como información adicional se muestra la diferencia relativa en el ángulo con el anterior frame, indicando si el ángulo en dicho instante del vídeo esta creciendo o disminuyendo.

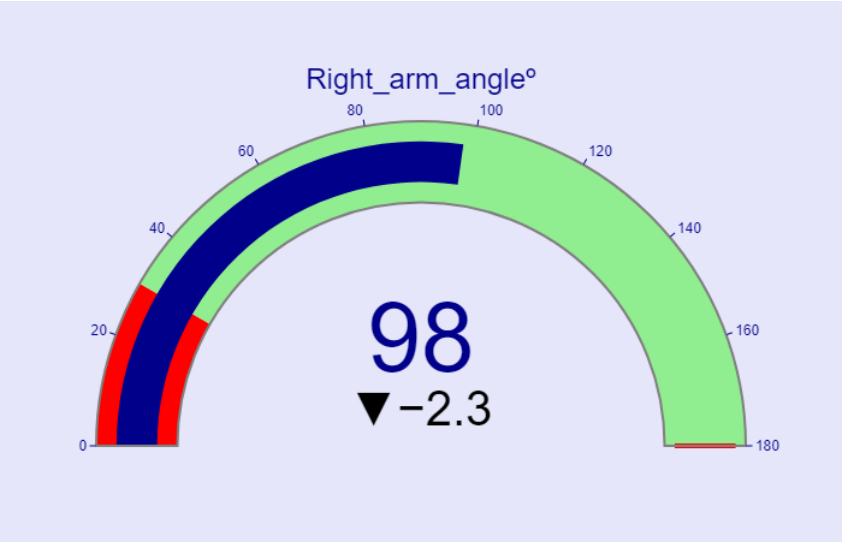

Figure 5.27: Representación del ángulo del brazo derecho obtenido en un instante del ejercicio

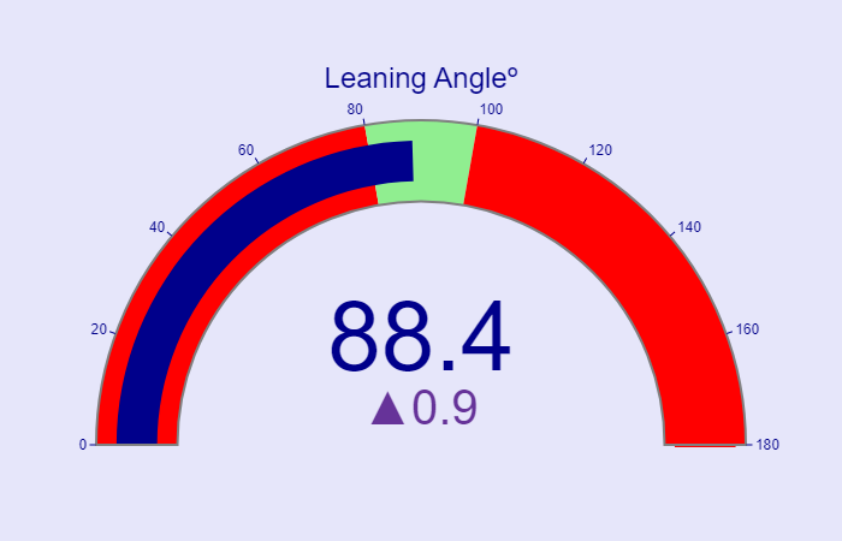

Figure 5.28: Representación del ángulo de inclinación obtenido en un instante del ejercicio

6 Análisis de resultados

En este capítulo se van a presentar los resultados obtenidos durante la realización de este proyecto y de los respectivos elementos, componentes implementados en este. A lo largo de esta sección se va a analizar y evaluar el funcionamiento, rendimiento y precisión del sistema creado. Una correcta evaluación del estado actual del sistema permite identificar los puntos fuertes del entorno e identificar aquellos aspectos que puedan ser mejorados en futuros desarrollos.

6.1 Precisión del sistema

En este apartado se va a evaluar la precisión del sistema a la hora de llevar a cabo la detección y análisis de las poses realizadas por el usuario. Dicha evaluación se centra en identificar el actual funcionamiento de las reglas generales establecidas en el sistema y comprobar los resultados que se obtienen de estas. Uno de los objetivos de este proyecto (capítulo 4.2) se basa en poder analizar diferentes actividades físicas a partir de una serie de reglas, esto produce que una de las principales limitaciones que se da en el análisis de este proyecto,consiste en la falta de conjuntos de datos públicos referentes a las poses humanas y coordenadas de sus puntos clave en las que se practiquen diferentes actividades físicas. Existen conjuntos de datos que reúnen diferentes actividades físicas, tales como *Yoga82*[24], *Human 3.6M*[25] o *MPII Human Pose*[26], pero estos no se da el etiquetado de movimientos acertados o erróneos a la hora de realizar la actividad, sino que están orientados para el entrenamiento de modelos de detección de poses y no del análisis de la realización de actividades a más alto nivel.

Ante esta limitación es necesario llevar a cabo la creación de una nueva recopilación de datos. De esta forma se ha creado un dataset en el que se han clasificado diferentes actividades físicas, catalogando las respectivas imágenes de las actividades como correctas o incorrectas en función de la pose objetivo que se pretende alcanzar en cada actividad.

6.1.1 Creación del *dataset*

El llevar a cabo la creación de un *dataset* desde cero supone un desafío debido a que es necesario recopilar, analizar y etiquetar las diferentes actividades físicas. Esto provoca que por cada actividad física diferente, se deba analizar a priori para obtener una comprensión profunda de esta para tener la capacidad de reconocer la pose objetivo que se considere correcta y poder etiquetar con precisión aquellos frames en los que se esté realizando de manera satisfactoria y aquellos en los que no. Idealmente el etiquetado del *dataset* debería de ser realizado por expertos deportistas que analizaran los frames de la actividad de la que disponen conocimiento, dotando al conjunto de datos de una mayor precisión en el etiquetado.

Realizar este *dataset* queda fuera de los objetivos de este proyecto, aunque se recomienda como un desarrollo a futuro para promover el progreso en este campo. Esto puede causar que los datos que se han recopilado puedan tener cierto grado de subjetividad y desconocimiento técnico debido a la necesidad del aprendizaje del funcionamiento y técnica de las diferentes actividades que componen dicho *dataset*.

A pesar de estas las limitaciones que puede presentar el *dataset*, esta agrupación de datos es un punto de partida útil que podrá ser sometido a mejoras. Estableciendo una base sólida para la detección y análisis de poses en actividades físicas. Además, este va a permitir obtener las primeras mediciones específicas y concretizar la precisión actual que presenta el sistema.

Etiquetado del dataset

Para realizar el etiquetado de los frames de cada actividad física, se ha creado una nueva herramienta en *Streamlit* que permita etiquetar los vídeos subidos para su evaluación de forma sencilla, teniendo la posibilidad de etiquetar varios frames mediante el uso de intervalos, permitiendo una mayor rapidez a la hora de añadir nuevas actividades al dataset.

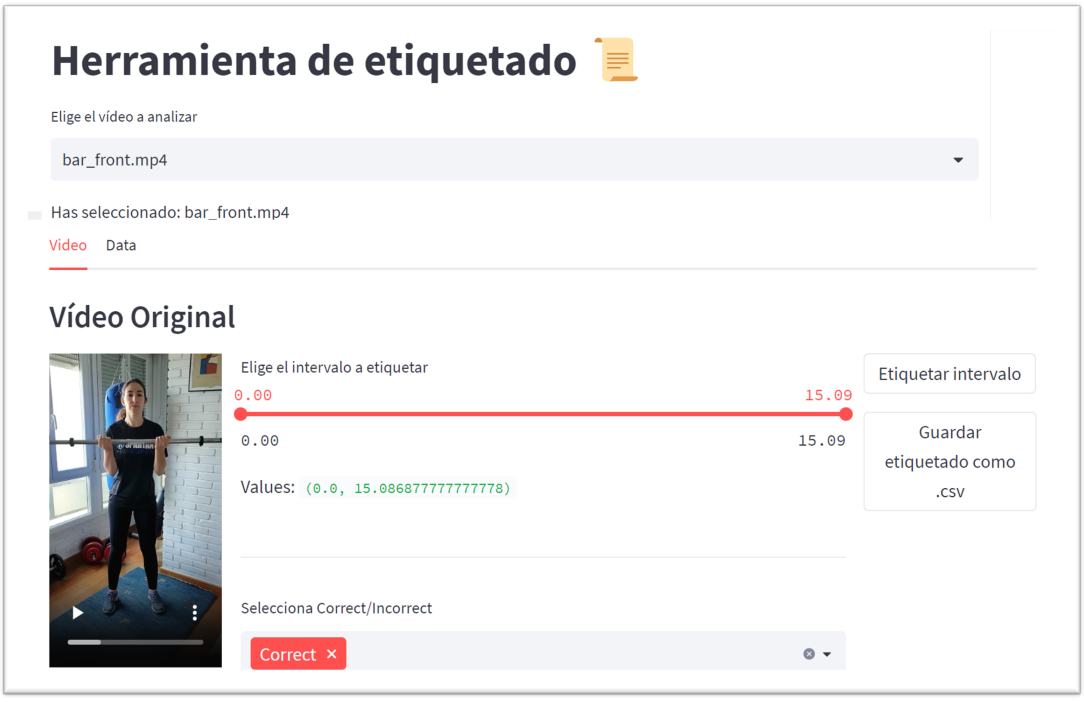

Figure 6.29: Herramienta de etiquetado de vídeos

	Time(s)	Error_List
4	0.1338	Correct;
5	0.1673	Correct;
6	0.2007	Correct;
7	0.2342	Correct;
8	0.2676	Correct;
9	0.3011	Correct;
10	0.3345	Incorrect;
11	0.368	Incorrect;
12	0.4014	Incorrect;

Figure 6.30: Muestra de un dataframe etiquetado

La figuras anteriores representan la interfaz (figura 6.29) de la herramienta de etiquetado y el dataframe (figura 6.30), que se muestra también en la interfaz, el cual contiene los resultados del etiquetado. Una vez se haya concluido con la documentación del vídeo, el usuario de la herramienta puede pulsar el botón de guardado para que los resultados se almacenen correctamente en su correspondiente fichero en formato .csv, que es un formato estándar que facilita su consumo por otras herramientas.

Composición del dataset

Este apartado está destinado a detallar los vídeos que componen dicho dataset. Para lograr una evaluación más precisa es necesario que se de la mayor cantidad de vídeos posible realizando múltiples actividades físicas con diferentes grados de complejidad. Además se ha tratado de que los vídeos presenten cierta variabilidad respecto al usuario que realice la actividad, el entorno en el que se encuentra, la cámara utilizada para realizar la grabación y diferentes configuraciones de iluminación con el objetivo de observar el comportamiento y actuación del sistema en escenarios diferentes.

En total se han recogido un total de 26 vídeos, con un total de 14104 frames referentes a las actividades físicas, lo cual supone una media de unos 542 frames por vídeo, es decir, vídeos próximos a los 18 segundos de media (empleando cámaras que trabajen a 30 imágenes por segundo).

Respecto a la proporción de frames resultantes del etiquetado, se obtiene un total de 4152 frames calificados como poses que se han realizados correctamente y los 9952 imágenes restantes consideradas como incorrectas. Estos valores, muestran un desequilibrio del dataset, esto se debe a que en la realización de una actividad hay un rango disminuido de imágenes en la postura realizada se califica como correcta, dando lugar a que en una

grabación se de una cantidad mayor de *frames* etiquetados como una postura incorrecta.

6.1.2 Evaluación del ADAP (a nivel de frame)

El sistema de análisis de poses produce detecciones por cada frame que entra al sistema. Por lo tanto, para evaluar la precisión de dichas detecciones a nivel de frame, se va a proceder a realizar la comparación entre estas detecciones y las imágenes etiquetadas pertenecientes al *dataset* creado.

A continuación, se va a generar la matriz de confusión con los resultados obtenidos y sus respectivas métricas. Estas nos proporcionan información detallada acerca del rendimiento del sistema y se obtienen a partir de las siguientes fórmulas:

1. *Accuracy*

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}}$$

2. *Precision*

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

3. *Recall*

$$Recall = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN}}$$

4. *F1 Score*

$$F1Score = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Las variables representan la siguiente información:

Variable	Función
P (Positives)	Representa el número total de detecciones positivas, tanto si se se han detectado correctamente o no.
N (Negatives)	Representa el número total de detecciones negativas, tanto si se se han detectado correctamente o no.
TP (True Positive)	Detección positiva que coincide con el valor real positivo
TN (True Negative)	Detección negativa que coincide con el valor real negativo
FP (False Positive)	Detección positiva errónea cuyo valor real es negativo
FN (False Negative)	Detección negativa errónea cuyo valor real es positivo

Table 6.1: Tabla de variables para la obtención de las métricas

Cabe destacar que en las evaluaciones realizadas, se considera como positiva a la imagen etiquetada real en la que la posición objetivo se ha alcanzado y negativa a aquella pose que haya sido etiquetada como incorrecta.

Resultados de la evaluación:

La siguientes figura (6.31) y tabla (6.2) muestran el resultado de computar la matriz de confusión entre todos los frames que componen el *dataset* y sus repectivas métricas.

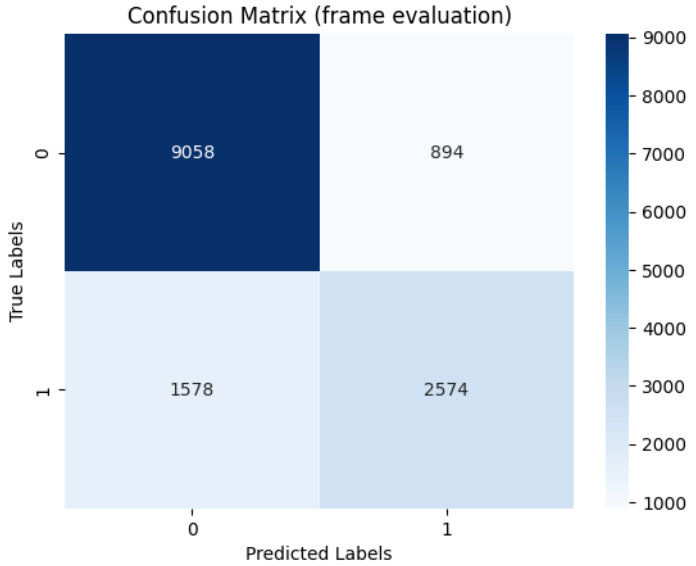

Figure 6.31: Matriz de confusión resultado de la evaluación frame a frame

Métrica	Valor
Accuracy	0.82
Precision	0.74
Recall	0.61
F1 Score	0.67

Table 6.2: Métricas obtenidas (a nivel de frame)

Siguiendo las fórmulas previamente mencionadas, se obtiene una *accuracy* superior al 80%, esto implica que el sistema detecta de forma correcta tanto posiciones positivas como negativas de forma satisfactoria. La *precision* indica la proporción de *TPs* entre todas las detecciones clasificadas como positivas y en este caso, se ha clasificado correctamente como positivos el 74% de las detecciones. Indicando que el sistema es capaz de detectar la mayor parte de las poses correctas que se dan en cada *frame* de un vídeo. Por otro lado, el *recall* supera el 60% , lo cual refleja que el sistema por lo general detecta correctamente los frames positivos pero existen casos en los que el sistema introduce casos falsamente negativos *FNs*.

Ante estos resultados, se puede afirmar que se cumple el requisito establecido referente al porcentaje de frames total que deben ser correctamente detectados (*60%*). Dicho objetivo se encuentra en el apartado 5.2.

Además se va a proceder a analizar la proporción de frames detectados correctamente, es decir, la *accuracy* obtenida por vídeo para observar el comportamiento específico del sistema dependiendo del vídeo procesado.

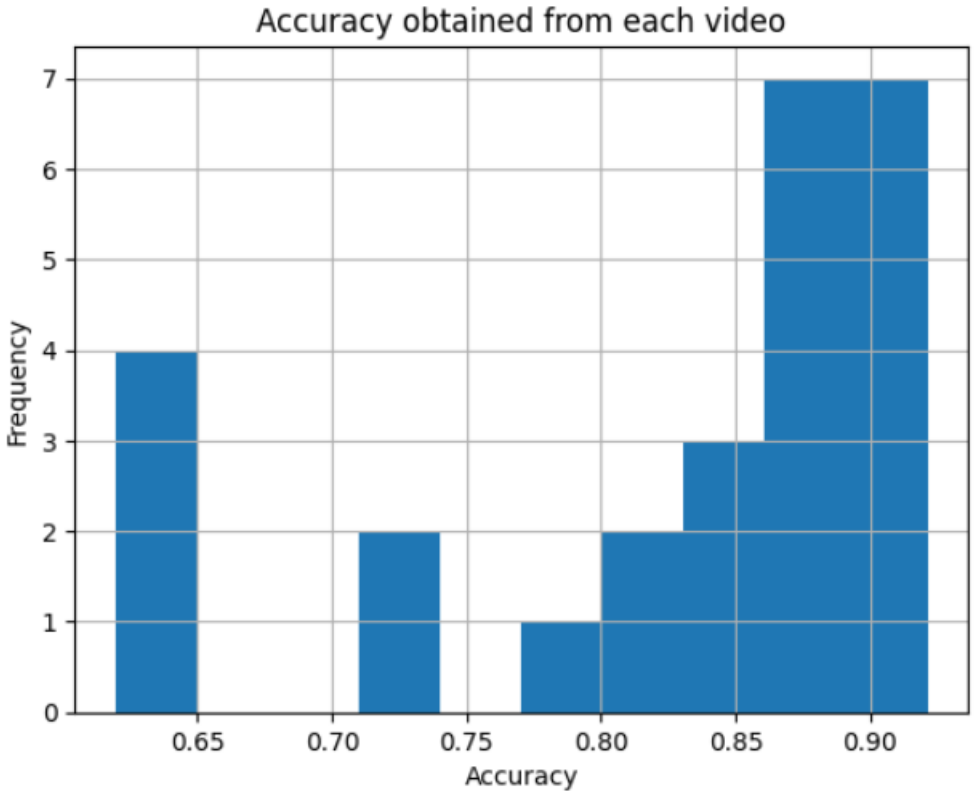

Figure 6.32: Histograma de la *accuracy* obtenida por cada vídeo

En la figura 6.32 se puede contemplar que cada columna del histograma supone un incremento del *accuracy* en 0.05, comenzando en 0.6 y finalizando el la unidad. Ante esto, se puede de nuevo confirmar que para todos los vídeos analizados ,se supera el 60% de los frames detectados correctamente por el sistema. A pesar de esto, es importante analizar los casos en los que la métrica es más baja para determinar cuál ha sido la posible causa de los errores durante la detección.

Atendiendo al vídeo que muestra el resultado menor, este es del 61.93% y hace referencia a un vídeo del *dataset* en el que se llevan a cabo pruebas del ángulo de inclinación establecido, donde el objetivo es detectar una postura recta del usuario. Estos resultados d reflejan lo importante que es para el modelo de detección de pose el poder visualizar todos los puntos clave del cuerpo sobre la imagen. Como se muestra en la figura 6.33,la detección del usuario, aún estando recta su postura, no es correcta debido a que el sistema no es capaz de calcula con precisión la postura verdadera de sus hombros y por lo tanto el valor que predice no es correcto,generando unas coordenadas que no son ciertas y produciendo fallos en el análisis de la pose generada.

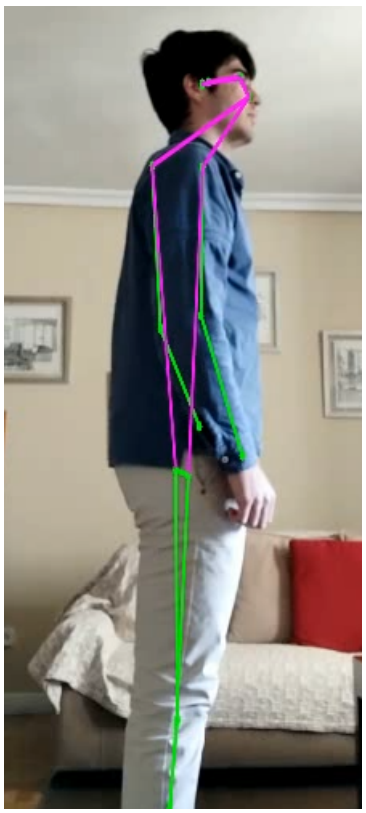

Figure 6.33: Fallo en la detección de una postura recta

Es de gran importancia la variabilidad en los vídeos que componen el *dataset* de análisis para explorar las limitaciones y fortalezas del sistema. Otro posible problema del modelo de detección es la aparición de múltiples individuos en la grabación, dependiendo del movimiento que se esté realizando puede provocar que el modelo falle en la predicción, detectando momentáneamente a esta nueva persona. A continuación se muestran dos casos en los que el modelo se somete ante la misma situación. En la primera imagen (6.34) el modelo detecta a la persona que no está realizando el ejercicio , mientras que, en la segunda imagen(6.35) no se desvía la atención del modelo.

Figure 6.34: Ejercicio con pérdida de la atención por el modelo

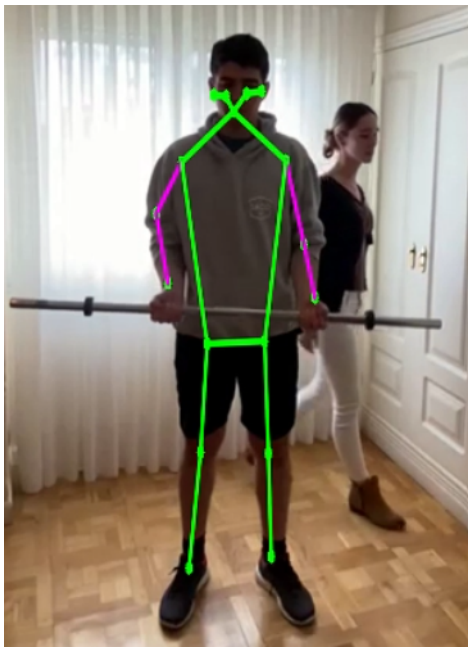

Figure 6.35: Ejercicio sin pérdida de la atención por el modelo

6.1.3 Evaluación del ADAP (a nivel de evento)

A continuación, se va a detallar otro método para definir la precisión del sistema. El método utilizado está muy influenciado por una métrica conocida por *Intersection over Union* o *IoU*,esta es muy utilizada en *Object Detection* para la evaluación de las predicciones que definen cuadrados delimitadores en los objetos, personas o animales que se deseen analizar. Supóngase que se dispone de una imagen y A representa un cuadro delimitador o *bounding box* real, el cual ha sido etiquetado de forma manual y representa los límites reales del objeto que se da en la imagen. Por otro lado, B sería el cuadro delimitador obtenido a partir de un modelo de *Object Detection*. *IoU* sería la métrica resultado del cálculo de la división del área de la intersección entre las dos regiones partida del área de unión formada por estas. En la figura 6.36 se muestra el proceso de cálculo del IoU entre dos regiones delimitadoras. Cuanto más próximo es el resultado a la unidad, más solapadas son las áreas y más parecido presenta la detección con el resultado real.

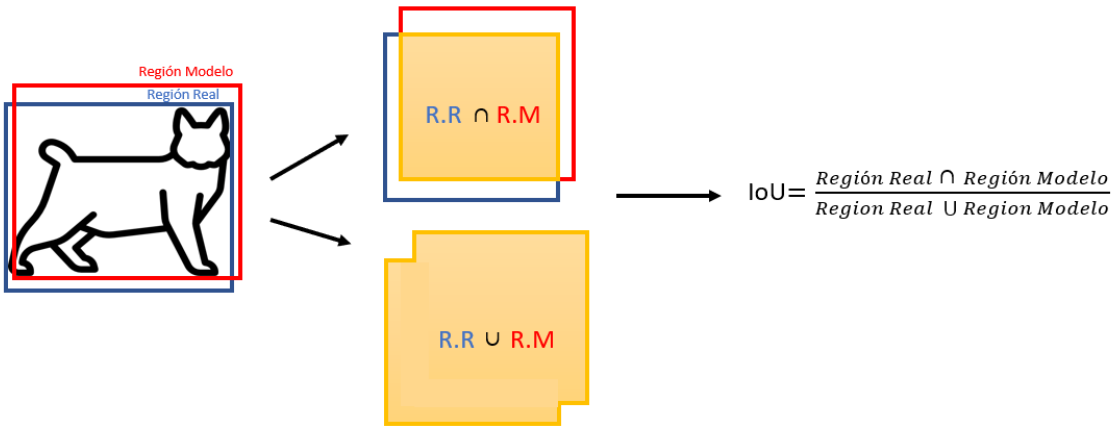

Figure 6.36: Cálculo del IoU

En el caso de este proyecto, el análisis realizado no se da a partir de las imágenes sino a partir de los eventos que se dan en las actividades físicas. Durante el vídeo el usuario ejecuta una serie de movimientos que se clasifican como eventos en un intervalo de tiempo determinado del vídeo. El método de análisis consiste en comparar los eventos reales con los predichos. Este proceso consiste en iterar por los conjuntos de eventos, en el caso de que el resultado del evento sea el mismo, se calculará los intervalos de intersección y unión para posteriormente obtener el *IoU* del evento correspondiente. En este caso el *IoU* representa la proporción de solape que se da entre ambos intervalos,los cuales solo presentan una dimensión. Al solo medir eventos en la dimensión temporal, el área de unión entre eventos no es necesaria y simplemente teniendo en cuenta la proporción de solape, se puede establecer un umbral similar al del *IoU*. Una vez se ha calculado

dicha proporción, es necesario definir un valor que determine cuánto porcentaje de intersección entre eventos es necesario para poder determinar si el intervalo calculado por el sistema es lo suficientemente elevado como para considerarse que el evento predicho se ha definido correctamente. Un evento predicho será evaluado como correcto siempre que la proporción de solape supere el umbral establecido y las etiquetas de ambos eventos coincidan.

A medida que este umbral aumenta se requerirá un mayor solapamiento entre los intervalos de los eventos dando lugar a un análisis más preciso y estricto, posibilitando que se detecten menos eventos de forma correcta. La relación entre las métricas de *precision* y *recall* (6.1.2), reflejan cuánto de estricto es un sistema a la hora de llevar a cabo la clasificación. Un sistema que esté enfocado en la precisión como métrica fundamental tiende a presentar un umbral elevado, esto aumenta que los requisitos para establecer una detección como correcta, por lo que se reduce el número de falsos positivos del sistema. Por otro lado, a mayor valor de *recall*, se minimiza el número de falsos negativos en las detecciones, indicando que todas las instancias positivas sean identificadas correctamente. Dependiendo de la aplicación que presente el sistema clasificador se puede buscar priorizar una métrica u otra.

En el marco de desarrollo del proyecto, se busca que el sistema sea capaz de detectar tanto los errores como aciertos durante la actividad física. Por lo que se ha obtado por un balance entre *precision* y *recall* para poseer un sistema equilibrado que no presente un sesgo muy elevado en sus detecciones. Se ha optado por un umbral del 60% considerándolo como un buen punto de partida que permitirá obtener resultados del sistema siendo lo suficientemente estricto y riguroso, ya q que se requiere que al menos más de la mitad del intervalo original se haya dado en la predicción para determinar que el evento se ha detectado correctamente.

Resultados de la evaluación:

A continuación, se muestra la matriz de confusión 6.37 y las métricas6.3 obtenidas resultado de evaluar los eventos únicos detectados en cada vídeo una vez realizada la agrupación de los frames detectados. Tanto los frames etiquetados como los frames detectados se han sometido al mismo proceso de agrupación. Este análisis proporcionará la evaluación de si esta síntesis de información en eventos actúa de la forma esperada.

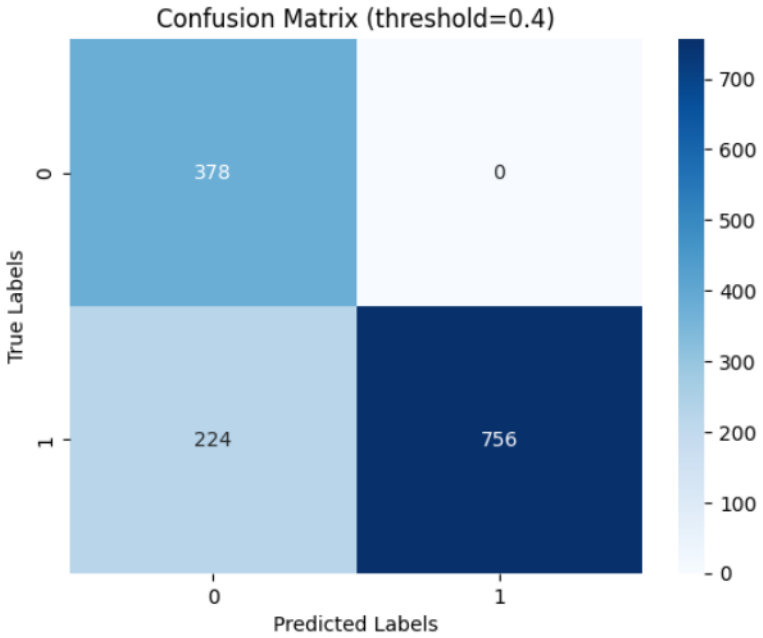

Figure 6.37: Matriz de confusión resultado de la evaluación a nivel de evento

Métrica	Valor
Accuracy	0.82
Precision	0.69
Recall	0.89
F1 Score	0.77

Table 6.3: Métricas obtenidas (a nivel de evento)

Los resultados obtenidos muestran que el sistema de obtención de eventos predice de forma correcta el 80% de los eventos generados. La diferencia principal de esta evaluación y la realizada a nivel de *frame* reside en las métricas relativas a la *precision* y *recall*. La precisión se reduce ligeramente a costa de aumentar el *recall*, provocando que el número de falsos negativos se reduce manteniendo prácticamente la misma proporción de eventos positivos correctamente predichos. Esto produce un mayor equilibrio en las detecciones ya que al agrupar los frames en eventos únicos se ignoran aquellos frames momentáneo en los que las detecciones el modelo no son precisas, como ya se comentó en el apartado 5.3.2, produciendo unos resultados que sintetizan la información obtenida por el análisis frame a frame de forma correcta y que dicha información sea de mayor utilidad para el usuario.

6.2 Rendimiento del sistema

A la hora de realizar una aplicación es muy importante tener en cuenta la velocidad y tiempos que ocupan los procesos que la componen. Estos tiempos deben ser lo suficientemente bajos para que el uso de dicha aplicación no sea molesto para el usuario. Aquellos que hagan uso de la aplicación esperan que la aplicación responda de forma rápida y eficiente conforme a las acciones que se realizan en ella. Respecto al cálculo del tiempo que conlleva una operación o proceso, esta duración puede calcularse forma sencilla a través de la librería *time* y aplicándola de la siguiente forma:

```
import time #Importar libreria

start_time=time.time() #Tiempo de inicio del proceso
#Ejecutar el proceso a medir...
end_time=time.time()  #Tiempo final del proceso

tiempo_transcurrido=end_time-start_time # Tiempo total
```

Listing 2: Medición de tiempo de ejecucion mediente la librería time

Tal como se ha expuesto en el apartado 5.2, la aplicación debe de cumplir una serie de tiempos para garantizar la eficiencia y velocidad del sistema.

6.2.1 Tiempo de carga de la aplicación y del modelo:

El proceso de inicialización de la aplicación y del modelo es simultáneo, a continuación se han calculado los tiempos de inicialización de la aplicación múltiples veces para observar si la duración de la carga es constante.

Nº de ejecución	Tiempo de carga necesitado (segundos)
1	15.299
2	16.087
3	16.1949
4	16.167
5	16.269
Tiempo medio	**16.003**

Table 6.4: Tabla de los tiempos de inicialización de la aplicación y modelo de detección

Los resultados de la tabla 6.4, muestran que el tiempo de carga de la aplicación y modelo es inferior al tiempo máximo de 30 segundos establecido como requisito. El modelo se debe descargar desde el *Hub* de *Tensorflow*, lo que supone un ligero retardo a la hora de arrancar la aplicación.

6.2.2 Tiempo de procesamiento y análisis del vídeo

La duración del procesamiento y análisis del vídeo de la actividad física depende fundamentalmente del número de frames que contiene dicho vídeo.

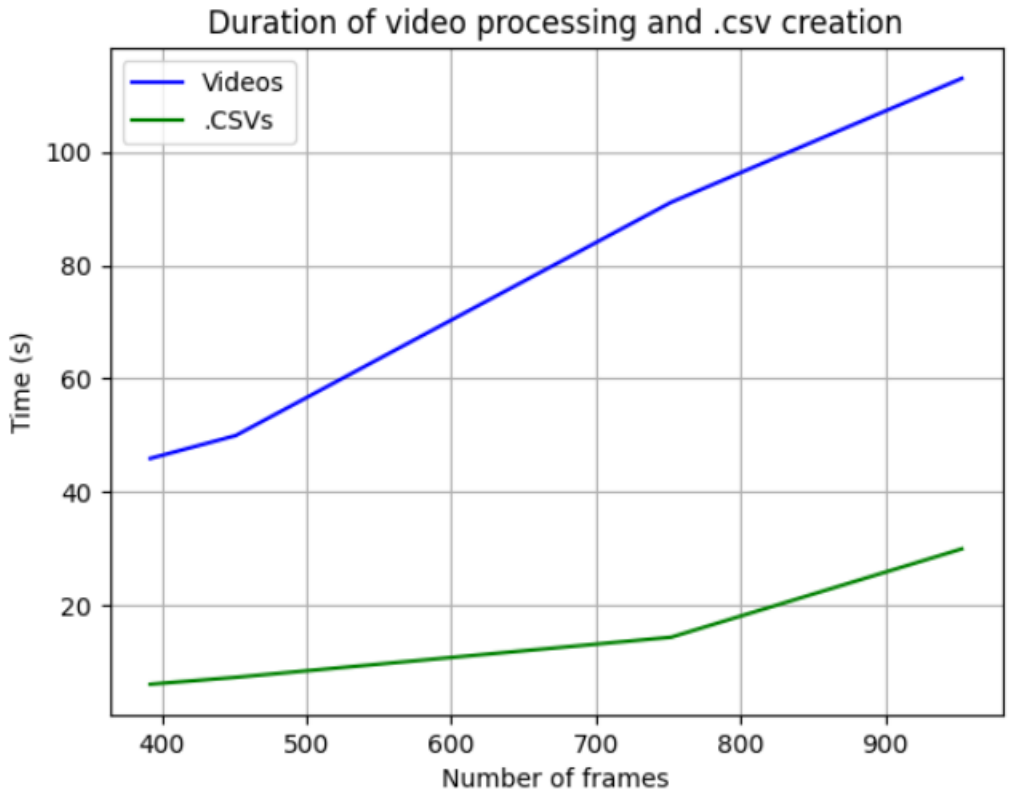

Figure 6.38: Tiempos de procesamiento de vídeo y creación de ficheros .csv

En la anterior figura, se muestra una gráfica de los tiempos que ocupa dicho procesamiento de la actividad en función de los frames que contiene, además de los tiempos de creación de todos los ficheros necesarios para la muestra de resultados. La gráfica muestra que a medida que la cantidad de la información aumenta, el tiempo de procesado aumenta de forma proporcional a esta, asemejándose a una relación lineal (O(n)) . El tiempo de procesamiento de vídeos se encuentra al límite de los requisitos establecidos. Por ejemplo, un vídeo de 392 frames se ha procesado con un tiempo de 45.29 segundos, el vídeo original al estar grabado a 30 imágenes por segundo presenta una duración es de 13.06 segundos. Esto implica que el procesamiento del vídeo supone que este tiempo es de 3.467 veces la duración del original.

Estos resultados entran del marco establecido por los requisitos pero se encuentran muy próximos al límite deseado, como trabajo futuro podría idearse la forma de reducir estos

tiempos de procesado.

6.2.3 Descarga de resultados:

Gracias a la implementación de la aplicación a través de *Streamlit*, se pueden descargar las tablas, vídeos y gráficas generadas de forma rápida y sencilla. El tiempo de descarga será limitado simplemente por la conexión a internet que posea el usuario. En el estado actual del sistema, toda la aplicación se rige localmente por lo que la descarga de los resultados se produce en un tiempo muy reducido.

6.3 Adaptabilidad del sistema

Uno de los objetivos principales del proyecto estaba centrado en la adaptabilidad del sistema. Esto hace referencia a la capacidad de modificación de la aplicación a la hora de introducir modificaciones o nuevas combinaciones de reglas para llevar a cabo el análisis de nuevas actividades. Actualmente en la aplicación, el usuario puede introducir al sistema para su procesado y análisis actividades que puedan emplear o no herramientas. Se han implementado reglas para el análisis de sentadillas, dominadas y para levantamientos de pesas(*curls* de bíceps, elevaciones de hombro y levantamiento estilo peso muerto). A pesar de la diferente técnica que se requiere para realizar todas estas actividades, los resultados mostrados en el apartado 6.1 indican que la implementación de las reglas que se está realizando, está actuando de forma correcta y concorde a los objetivos esperados por la aplicación.

Por lo tanto, se puede afirmar que la adaptabilidad del sistema es la esperada y permite añadir actividades de forma sencilla a la par que los resultados son los esperados. Es importante mencionar de nuevo la necesidad de tener grandes conocimientos en la actividad que se desea añadir para alcanzar la pose objetivo que requiere el ejercicio. Por ende, el número de actividades implementadas en este proyecto es limitado.

6.4 Usabilidad del sistema

En este apartado se lleva a cabo la evaluación de la aplicación respecto a la facilidad de uso y comodidad. En primer lugar, el diseño de la interfaz, como se muestra en la figura 5.17, es un diseño limpio en el que cada ventana presenta una funcionalidad del sistema específica, lo cual es beneficioso para no confundir al usuario . Además, este usuario es guiado durante todo el proceso, desde la carga de la actividad física a la muestra de resultados, facilitando la comprensión de las funcionalidades del sistema y del uso de los componentes de la aplicación en gran medida.

Por otro lado, el realizar la aplicación empleando *Streamlit*, produce una interfaz sencilla con componentes visuales que permiten representar los datos de forma atractiva y comprensible para el usuario. permitiendo ser un gran entorno web en el que mostrar resultados y gráficos debido a la gran interactividad que presentan estos componentes.

A continuación, se muestran a modo de ejemplo varias secciones de las ventanas propias de la aplicación

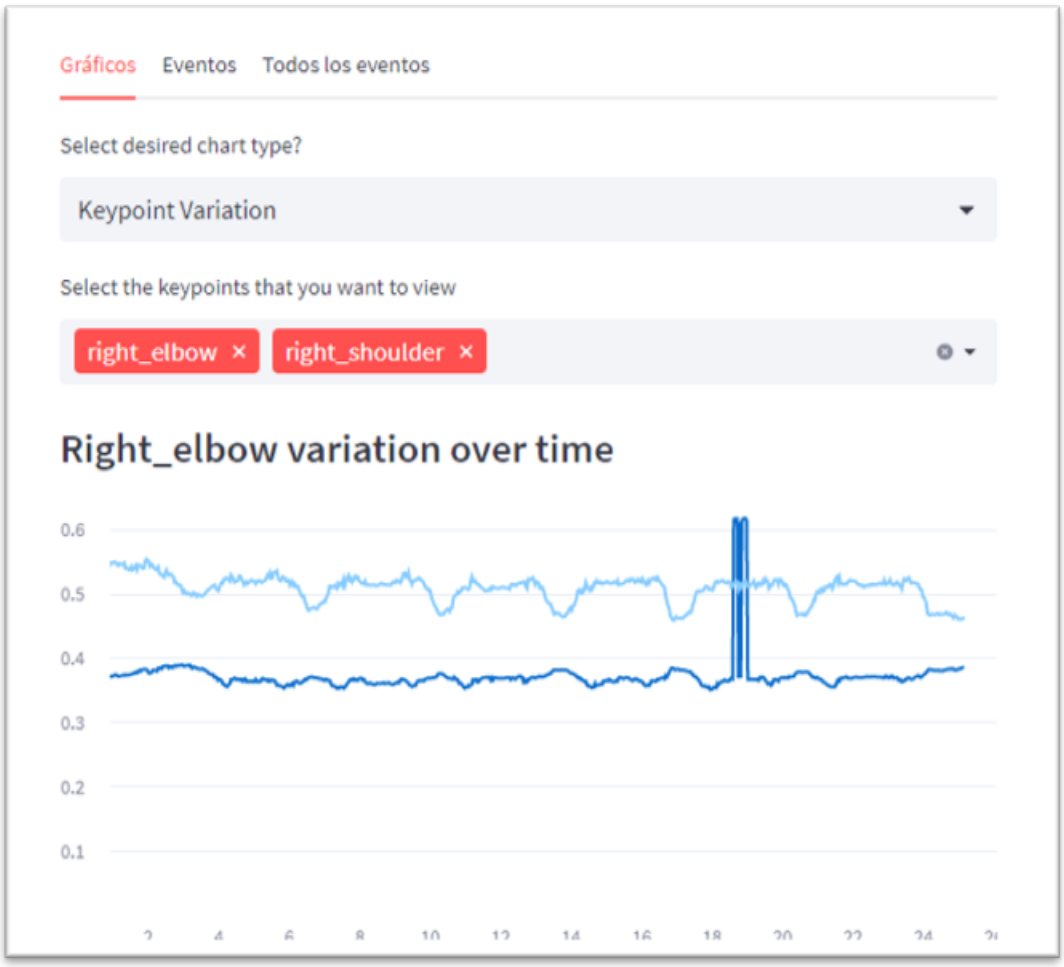

Figure 6.39: Pestaña de gráficas mostrando la variación de los keypoints a lo largo de la actividad

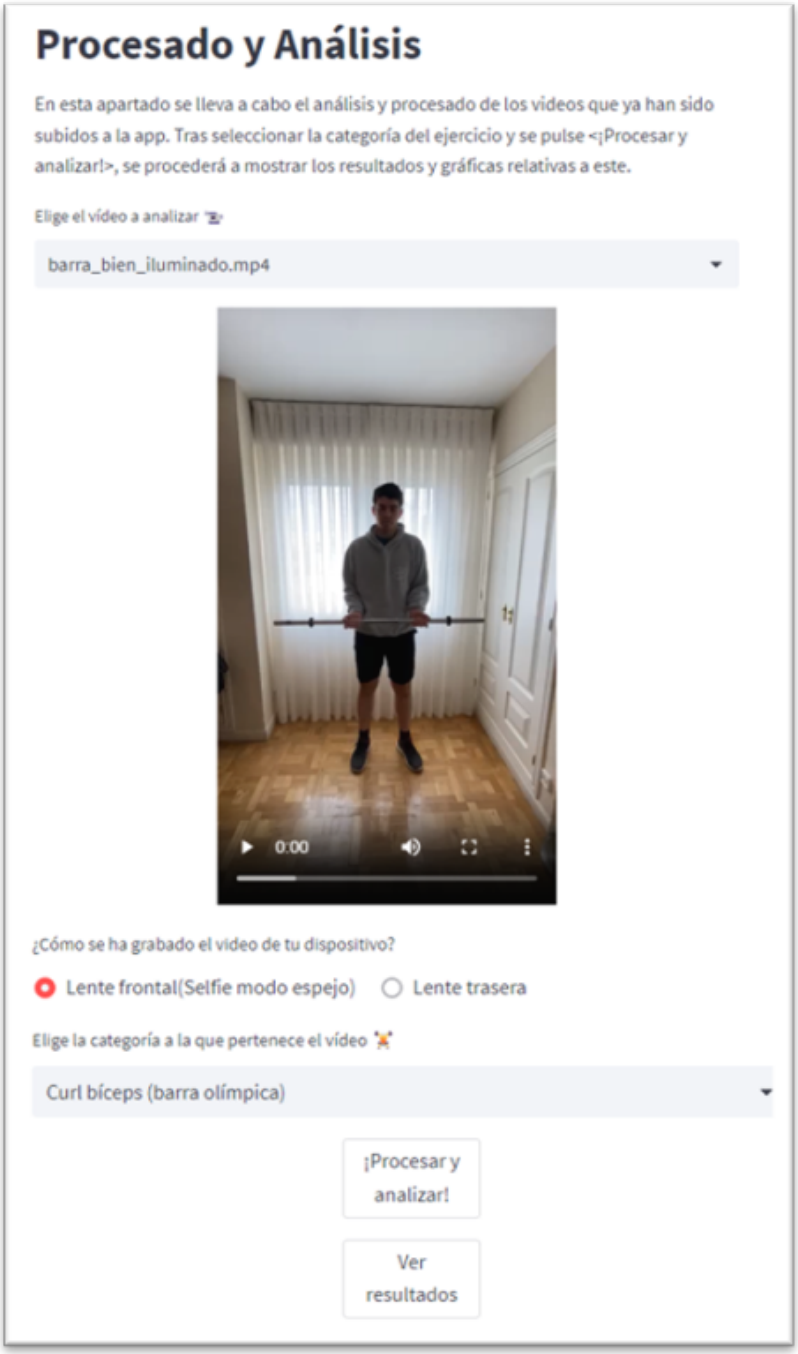

Figure 6.40: Ventana de la aplicación enfocada al procesado y análisis de la actividad física

7 Conclusiones y trabajos futuros

7.1 Conclusiones

El interés y atracción por las ramas del *Deep Learning* y *Pose Detection* se encuentra en aumento desde los últimos años, con ello surgen múltiples proyectos que tratan de aportar nuevas aplicaciones a estos ámbitos, incorporando nuevas ideas y mejoras gracias al avance de la tecnología.

La realización de este proyecto tenía como foco principal facilitar al usuario las herramientas *Deep Learning* necesarias para que este pudiera beneficiarse de la utilidad de la detección y análisis de poses en el ámbito del deporte. Gracias al *Pose Detection*, a partir de la grabación de una actividad física, se puede obtener una gran cantidad de información útil acerca del desempeño realizado por el usuario en dicha actividad.

Mediante la creación de una aplicación se ha posibilitado al usuario subir una actividad física específica y que esta en función de su tipo o categoría pueda ser analizada detectando la posición y postura del usuario y anotando en qué momentos la actividad se está realizando de forma correcta e incorrecta para posteriormente proporcionar *feedback* al usuario relativo a la técnica empleada por este. A partir del estudio de múltiples métodos y técnicas pertenecientes al ámbito de la visión computacional y *Pose Detection*, se ha optado por emplear aquellas que permitieran crear un sistema rápido y eficiente manteniendo la precisión necesaria para detectar los posibles eventos que se den en una actividad de forma correcta y consistente.

Por otro lado, se ha buscado realizar un sistema que permitiera de forma sencilla la adición de nuevas reglas para el análisis de las actividades físicas que se deseen incorporar a la aplicación. Gracias a la creación de reglas genéricas que agrupan de forma semántica las partes del cuerpo, se ha posibilitado realizar este proceso de una forma sencilla, creando reglas personalizadas para una actividad específica sin la necesidad de generar nuevas funciones especializadas. Para la configuración de las reglas personalizadas, es importante poseer los conocimientos necesarios referentes a la técnica del ejercicio físico que se quiere añadir. Esto permitirá determinar cuánto de estricto debe de ser la pose objetivo que debe ser alcanzada por el usuario, siempre teniendo en cuenta los posibles márgenes de error que se pueden dar a la hora de utilizar resultados provenientes de la predicción de un modelo.

Durante el procesamiento de actividades se ha observado cómo las predicciones del sistema están sujetas de manera considerable a la forma en la que se lleva a cabo la grabación de la actividad física. Aquellas grabaciones en las que se den varios individuos, objetos de gran tamaño o donde el usuario no muestre toda su figura a la cámara, puede suponer errores en la detección y por lo tanto los resultados del análisis se desviarán de los resultados reales.

Evaluar la precisión de los resultados ha supuesto un reto debido a la carencia de conjuntos de datos públicos relativos a detecciones de poses en actividades físicas etiquetadas según si se están realizando correctamente o no. Para solucionar este problema se ha diseñado un *dataset* con múltiples actividades físicas, el cual ha permitido analizar el comportamiento del sistema ante diferentes ejercicios y permitiendo observar en que circunstancias el sistema presenta sus fortalezas o deja entrever las limitaciones actuales de los modelos de predicción.

Estos son varias de las lecciones aprendidas durante la creación del *dataset*:

- La cantidad de datos que se poseen y la variabilidad entre estos es determinante a la hora de llevar a cabo una evaluación. A mayor número de datos y situaciones que se poseen, más información se puede obtener de la evaluación de los datos predichos, obteniendo mejores conclusiones acerca del funcionamiento del sistema.

- El proceso de grabación de actividades y del etiquetado de imágenes supone una ardua tarea la cual requiere una inversión considerable de tiempo y debe ser realizada con precisión para poder tener un conjunto de datos fiable.

- Realizar una herramienta que permita el etiquetado de forma rápida es muy recomendable. A pesar de que también supone una inversión de tiempo, una vez creada, la incorporación de datos al *dataset* va a ser mucho más eficaz.

Relativo al análisis se han cumplido con los requisitos establecidos en el marco de desarrollo del proyecto, las detecciones obtenidas correctamente como el rendimiento del sistema, supera los objetivos marcados. A pesar de estos resultados, sería de gran relevancia realizar de nuevo esta evaluación sobre un *dataset* que presente una mayor cantidad de datos y de casos posibles, permitiendo profundizar en mayor medida en el comportamiento del sistema y su eficiencia.

Gracias a la creación de este proyecto, se ha podido presentar al usuario las utilidades que presenta la detección de poses a través del deporte, produciendo que se abra una nueva puerta a una innovadora forma de hacer ejercicio.

7.2 Trabajos futuros

Este proyecto ha explorado un ámbito que se encuentra en continuo desarrollo, este presenta un gran potencial y aspira a ser de gran utilidad para cualquier usuario que realice actividades físicas. A pesar de esto, aún hay labores pendientes por realizar.

El sistema se puede adaptar para incluir requisitos y buenas prácticas asociadas a entornos de producción como puede ser: la preparación del sistema para su despliegue en entornos cloud como *Google Cloud Platform (GCP)[27]* o *Amazon Web Services (AWS)[28]* , junto a establecer servicios de autenticación, control de acceso y seguridad para la prevención de posibles ataques maliciosos. Además, la aplicación deberá de ser sometida a mejoras y actualizaciones, basándose en los ejercicios que se deseen añadir, los posibles errores o *bugs* que puedan aparecer durante su uso o a partir de la retroalimentación que aporten los usuarios. Para mejorar el rendimiento y reducir el tiempo de procesado y análisis de las actividades, el uso de técnicas como la paralelización de procesos o el almacenamiento de información sobre la memoria caché pueden contribuir a este objetivo.

A pesar de que *MoveNet* es un modelo que aporta buenos resultados con el uso de una sola cámara. Se recomienda explorar la implementación de este modelo con múltiples entradas de vídeo para la grabación de la actividad. Este método puede ser fundamental para identificar movimientos que presenten profundidad, los cuales no pueden ser detectados correctamente debido a que sólo se dispone de una imagen en dos dimensiones. Este método supondría una mejora en las predicciones a costa de requerir más dispositivos, sacrificando la facilidad del uso de la aplicación mediante una cámara individual.

Por otro lado, sería interesante la composición de un dataset especializado en el marco de trabajo del proyecto, el cual presente una extensión mucho mayor y concentre numerosas actividades físicas, realizadas por una gran cantidad de personas y desde perspectivas distintas. Esto permitiría estudiar diferentes posibilidades de actuación, abriendo la puerta a la creación de diferentes modelos clasificadores, los cuales necesitarán de grandes cantidades de información para que puedan ser entrenados y evaluados para su correcto funcionamiento.

8 Bibliografía

[1] Travis E. Oliphant. *Guide to NumPy*. USA: Trelgol Publishing, 2006.

[2] Python Software Foundation. *Math-Mathematical functions*. URL: https://docs.python.org/3/library/math.html?highlight=math#module-math.

[3] Wes McKinney. *Python for Data Analysis*. California, USA: O'Reilly, 2013.

[4] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science amp; Engineering* (2007).

[5] Desmond J Higham and Nicholas J Higham. *MATLAB guide*. SIAM, 2016.

[6] Alex Johnson et al. *Plotly Graphing Libraries*. URL: https://plotly.com/graphing-libraries/.

[7] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[8] Google. *MoveNet: modelo de detección de pose ultrarrápido y preciso*. URL: https://www.tensorflow.org/hub/tutorials/movenet.

[9] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[10] Ara V Nefian and Monson H Hayes. "Face detection and recognition using hidden Markov models". In: *Proceedings 1998 international conference on image processing. icip98 (cat. no. 98cb36269)*. Vol. 1. IEEE. 1998, pp. 141–145.

[11] Michael Isard and Andrew Blake. "CONDENSATION–conditional density propagation for visual tracking". In: *International journal of computer vision* 29.1 (1998), p. 5.

[12] Alberto Menache. *Understanding motion capture for computer animation and video games*. Morgan kaufmann, 2000.

[13] Carnegie Mellon University. *OpenPose: Whole-Body Pose Estimation*. URL: https://www.ri.cmu.edu/publications/openpose-whole-body-pose-estimation/.

[14] Mihir Garimella. *Sign Language Recognition with Advanced Computer Vision*. URL: https://towardsdatascience.com/sign-language-recognition-with-advanced-computer-vision-7b74f20f3442.

[15] Ajay Chaudhari et al. *YOG-GURU: REAL-TIME YOGA POSE CORRECTION SYSTEM USING DEEP LEARNING METHODS*. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9509937.

[16] Marc Marais and Dane Brown. *Golf Swing Sequencing Using Computer Vision*. URL: https://link.springer.com/chapter/10.1007/978-3-031-04881-4_28.

[17] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[18] John Pomerat, Aviv Segev, and Rituparna Datta. "On Neural Network Activation Functions and Optimizers in Relation to Polynomial Regression". In: *2019 IEEE International Conference on Big Data (Big Data)*. IEEE. 2019, pp. 6183–6185.

[19] Angjoo Kanazawa et al. "End-to-end recovery of human shape and pose". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7122–7131.

[20] Yuxin Hou et al. *Dancing like a superstar: Action guidance based on pose estimation and conditional pose alignment*. URL: https://ieeexplore.ieee.org/document/8296494.

[21] Yuichiro Tachibana. *streamlit-webrtc*. https://github.com/whitphx/streamlit-webrtc. Version 0.45.0. 2022.

[22] Dianyuan Han. "Comparison of commonly used image interpolation methods". In: *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*. Atlantis Press. 2013, pp. 1556–1559.

[23] S. Pasqualini et al. "Comparison of H.264/AVC, H.264 with AIF, and AVS based on different video quality metrics". In: *2009 International Conference on Telecommunications*. 2009, pp. 190–195. DOI: 10.1109/ICTEL.2009.5158642.

[24] Manisha Verma et al. "Yoga-82: a new dataset for fine-grained classification of human poses". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 1038–1039.

[25] Catalin Ionescu et al. "Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments". In: *IEEE transactions on pattern analysis and machine intelligence* 36.7 (2013), pp. 1325–1339.

[26] Mykhaylo Andriluka et al. "2d human pose estimation: New benchmark and state of the art analysis". In: *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*. 2014, pp. 3686–3693.

[27] Google. *Google Cloud: Cloud Computing Services*. URL: https://cloud.google.com/.

[28] Amazon. *Cloud Computing - Servicios de informática en la nube - AWS*. URL: https://aws.amazon.com/es/.

[29] Organización de las Naciones Unidas (ONU). *Objetivos de Desarrollo Sostenible*. URL: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

ANEXO A: Alineación del proyecto con los ODS

Este proyecto tiene gran relación con dos de los Objetivos de Desarrollo Sostenibles (ODS)[29] propuestos por los líderes mundiales para el año 2030 a pesar del nivel técnico y de investigación que este proyecto presenta.

El objetivo que más destaca por su relación con el proyecto correspondería al noveno objetivo: **"Industria, Innovación e Infraestructuras"**. A continuación, se muestran los puntos en los que coincide en mayor medida el proyecto realizado:

- 9.b Apoyar el desarrollo de tecnologías, la investigación y la innovación nacionales en los países en desarrollo, incluso garantizando un entorno normativo propicio a la diversificación industrial y la adición de valor a los productos básicos, entre otras cosas

- 9.5 Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países, en particular los países en desarrollo, entre otras cosas fomentando la innovación y aumentando considerablemente, de aquí a 2030, el número de personas que trabajan en investigación y desarrollo por millón de habitantes y los gastos de los sectores público y privado en investigación y desarrollo

Durante la realización del proyecto se ha llevado a cabo la investigación acerca de las técnicas más innovadoras empleadas en el ámbito de la visión computacional con el objetivo de obtener un modelo óptimo para el reconocimiento de actividades físicas.

Por otro lado, se encuentra el tercer objetivo referido a la **Salud y Bienestar**. Este objetivo se asocia con el proyecto en el sentido de que este está enfocado a que la realización de la actividad física escogida se lleve a cabo a través de una buena aplicación de la técnica necesaria para ejecutar dicha actividad. En el caso de numerosos deportes, una realización incorrecta de un movimiento puede llegar a causar perjuicios sobre la salud de la persona que lo practica, ya sea por medio de esguinces, roturas, distensiones u otras lesiones. Por lo tanto, se puede afirmar que este proyecto contribuye a la realización correcta de actividades físicas y por ende a contribuir con un desarrollo del deporte de manera segura.

ANEXO B: Detalles de implementación y código

B.1 Implementación módulo de grabación

A continuación, se muestra el código con la configuración e implementación requerida para la creación de una sencilla herramienta que permita al usuario llevar a cabo la grabación de la actividad física en .MP4 desde la propia aplicación en *Streamlit*. Esta herramienta aparece mencionada en la sección 5.4.1.

```
from streamlit_webrtc import webrtc_streamer, WebRtcMode,
   RTCConfiguration,VideoProcessorBase
class VideoRecorder(VideoProcessorBase):
    def __init__(self, out_file):
        self.recording = False   # variable que indica si se ha
           pulsado grabar
        self.out_file = out_file # nombre del archivo de salida
        self.frames = [] #array que va a almacenar los frames de
            la grabación

    def recv(self, frame): #En el caso de que se grabe se
       almacenará cada frame
        if self.recording:
            img = frame.to_ndarray(format="rgb48")
            self.frames.append(img)
        return frame
    def start_recording(self):
        self.recording = True #Comenzar la grabación
        self.frames = [] #Reinicializar el listado de frames

    def stop_recording(self):#Función que crea el vídeo una vez
       se ha pausado la grabación
        self.recording = False
        if self.out_file is not None:
            if self.frames:
                imageio.mimsave(self.out_file ,self.frames, fps
                   =30,format='mp4')
                st.success("Grabación guardada!")
            else:
                st.warning("No se ha comenzado la grabación aún.
                   ")
        self.frames = []
```

Listing 3: Clase VideoRecorder

```python
def app():

    st.header("Herramienta de grabación")

    file = st.text_input("Nombre de la grabación: (Tiene que
       terminar en .mp4) ", "output.mp4")

    webrtc_ctx = webrtc_streamer(
        key="pose-detection",
        mode=WebRtcMode.SENDRECV, # Se permite al componente
           enviar y recibir
        rtc_configuration={"iceServers": [{"urls": ["stun:stun.l
           .google.com:19302"]}]},
        video_processor_factory=lambda: VideoRecorder(out_file=(
           recordings_folder_path+file)),
        media_stream_constraints={"video": True, "audio": False
           }, #El audio es descartado
    )

    if st.button("Empezar Grabación"):
        recorder = webrtc_ctx.video_processor
        recorder.start_recording()
        st.session_state.recorder = recorder
        st.info("Grabación comenzada")

    if st.button("Finalizar Grabación"):
        recorder = st.session_state.recorder
        recorder.stop_recording()
        st.info("Grabación finalizada")
```

Listing 4: Implementación de la herramienta de grabación

Lo más destacable del código se da en la implementación del módulo *webrtc-streamer* en el cual se configuran los siguientes parámetros:

- **key**: identificador único de la transmisión.
- **mode**: determina el modo de la transmisión, en este caso se define como *WebRtcMode.SENDRECV* para poder enviar y recibir los frames por la aplicación
- **rtc-configuration**: parámetro que permite determinar lo que se realiza con el stream de vídeo recibido, en este caso se instancia la clase VideoRecorder, donde se especifica el procedimiento para grabar los frames de entrada. Además este permite determinar qué tipos de media se aceptan en la retransmisión, en el marco de desar-

rollo del proyecto, el audio es innecesario y por lo tanto se decide no incluirlo en las grabaciones para no ocupar más espacio del necesario.

B.2 Implementación de las reglas generales

En esta sección se muestra el código requerido para el establecimiento de las reglas generales por las que se rige el sistema de análisis. Se va a mostrar a modo de ejemplo, la regla general obtenida para la detección relativa al brazo, cuyo diseño y funcionamiento ya se detalló en el capítulo 5.3.2

```
1
2  def is_arm_correct(arm,keypoints_map,angle_range,height_desired=
       None):
3      #Selección de keypoints en función del brazo escogido
4      if arm == "RA":
5          shoulder_x = keypoints_map["right_shoulder_x"]
6          shoulder_y = keypoints_map["right_shoulder_y"]
7          elbow_x = keypoints_map["right_elbow_x"]
8          elbow_y = keypoints_map["right_elbow_y"]
9          wrist_x = keypoints_map["right_wrist_x"]
10         wrist_y = keypoints_map["right_wrist_y"]
11     elif arm == "LA":
12         shoulder_x = keypoints_map["left_shoulder_x"]
13         shoulder_y = keypoints_map["left_shoulder_y"]
14         elbow_x = keypoints_map["left_elbow_x"]
15         elbow_y = keypoints_map["left_elbow_y"]
16         wrist_x = keypoints_map["left_wrist_x"]
17         wrist_y = keypoints_map["left_wrist_y"]
18
19     # Cálculo de los vectores
20     shoulder_elbow_vec = (elbow_x - shoulder_x, elbow_y -
           shoulder_y)
21     elbow_wrist_vec = (wrist_x - elbow_x, wrist_y - elbow_y)
22
23     # Cálculo odel producto escalar y de las magnitudes de los
           vectores
24     dot_product = shoulder_elbow_vec[0] * elbow_wrist_vec[0] +
           shoulder_elbow_vec[1] * elbow_wrist_vec[1]
25     shoulder_elbow_mag = math.sqrt(shoulder_elbow_vec[0] ** 2 +
           shoulder_elbow_vec[1] ** 2)
26     elbow_wrist_mag = math.sqrt(elbow_wrist_vec[0] ** 2 +
           elbow_wrist_vec[1] ** 2)
27
28     # Cálculo del ángulo
29     cos_angle = dot_product / (shoulder_elbow_mag *
```

```
        elbow_wrist_mag)
30      angle = math.degrees(math.acos(cos_angle))
31
32      #Se tiene en cuenta si se requiere una altura específica
           para la regla
33      if height_desired!=None:
34          height_diff=(wrist_y-height_desired)
35      else:
36          height_diff=0
37
38      #Determinar la detección en función de si se cumplen los
           requerimientos de la actividad
39      (rango de angulos y altura del brazo deseada)
40      if angle_range[0]<=angle<=angle_range[1]  and abs(
           height_diff)<0.1:
41          return arm+ ",Correct;",True,angle
42      elif angle_range[0]<=angle<=angle_range[1] and height_diff
           <-0.1:
43          return arm+ ",Too Low;",False,angle
44      elif angle_range[0]<=angle<=angle_range[1]  and height_diff
           >0.1:
45          return arm+ ",Too High;",False,angle
46      elif angle>angle_range[1]:
47          return arm + ",Angle too wide(folded);",False,angle
48      elif angle<angle_range[0]:
49          return arm + ",Angle too narrow(straight);",False,angle
```

Listing 5: Implementación de regla genérica relativa al análisis del brazo

B.3 Agrupación de reglas para la creación de reglas personalizadas

Una vez se disponen de reglas genéricas, es necesario agruparlas para poder analizar ejercicios físicos específicos. El código siguiente refleja la aplicación práctica de la explicación detallada ,ya comentada en la sección 5.4.4, del ejemplo en la creación de unas reglas personalizadas para una actividad con la respectiva.

```
1 def get_rule_parameters(category,keypoints_map): #se
2     if category=="Dominada bíceps":
3             right_arm_text,is_right_correct,ra_angle=
                 is_arm_correct("RA",keypoints_map,[30,180],
                 height_desired=keypoints_map["right_shoulder_y"])
4             left_arm_text,is_left_correct,la_angle=
                 is_arm_correct("LA",keypoints_map,[30,180],
```

```
                height_desired=keypoints_map["left_shoulder_y"])
            body_angle_text,is_alignment_correct,body_angle=
                is_person_leaning(keypoints_map,desired_position=
                "Straight;",facing="Front")

            #Diccionario de ángulos para la representacion de
                resultados
            angle_dict={}
            angle_dict["RA_angle"]=ra_angle
            angle_dict["LA_angle"]=la_angle
            angle_dict["Body_angle"]=body_angle

            #Diccionario de detecciones para la representación
                gráfica de la pose
            boolean_dict={}
            boolean_dict["left_leg"]=True
            boolean_dict["right_leg"]=True
            boolean_dict["left_arm"]=is_left_correct
            boolean_dict["right_arm"]=is_right_correct
            boolean_dict["body_angle"]=is_alignment_correct

            @#Diccionario d edetecciones
            body_detections_dict={}# The detection that are
                gonna be analysed
            body_detections_dict["RA_dect"]=right_arm_text
            body_detections_dict["LA_dect"]=left_arm_text
            body_detections_dict["Leaning_dect"]=body_angle_text

            #Diccionario de evaluación
            required_detections_dict={}
            required_detections_dict["Leaning_dect"]="Straight;"
            required_detections_dict["RA_dect"]="RA,Correct;"
            required_detections_dict["LA_dect"]="LA,Correct;"
        elif category =...
```

Listing 6: Creación de regla personalizada

La anterior función muestra la composición de reglas necesaria para el análisis correcto de un vídeo en el que ese estén realizando dominadas. La función *get-rule-parameters* tiene como objetivo recopilar todas las reglas incorporadas al sistema de análisis.

B.4 Implementación del método de evaluación a nivel de evento

Este anexo hace referencia a la implementación del método de evaluación del sistema en el que se comparan los eventos predichos de los eventos etiquetados una vez se han formado por el sistema. La explicación detallada acerca del diseño del método de esta evaluación se da en el la sección 6.1.3. El siguiente código muestra el proceso de obtención de la matriz de confusión y sus respectivas métricas.

```
threshold=0.6 #Umbral definido

#Valores de la matriz de confusión
true_positive_events = 0
true_negative_events = 0
false_positive_events = 0
false_negative_events = 0
total_true_events = 0

confusion_matrix = np.zeros((2, 2), dtype=int)

for video_pred_events,video_true_events in zip(total_pred_events
   ,total_labeled_events):#Iterar cada vídeo etiquetado y
   procesado
    for pred_event in video_pred_events: #Iterar por cada evento
        de la predicción

        # Inicializar las variables de duración del evento
           predicho
        pred_event_duration = pred_event[2] - pred_event[1]
        true_detect = 0

        for true_event in video_true_events:
            # Verificar si hay superposición entre los eventos
               predicho y verdadero
            # Verificar si la etiqueta entre eventos real y
               predicho coincide
            if pred_event[0] == true_event[0] and pred_event[1]
               <= true_event[2] and pred_event[2] >= true_event
               [1]:
                intersection_start = max(pred_event[1],
                   true_event[1])
                intersection_end = min(pred_event[2], true_event
                   [2])
                intersecting_frames = max(0, intersection_end -
                   intersection_start)
```

```
                true_detect += intersecting_frames

        # Calcular la proporción de solape
        pred_ratio_detect = true_detect / pred_event_duration

        # Evaluar el resultado de la detección
        if pred_ratio_detect>=threshold:
            if pred_event[0]=="Correct;":
                true_positive_events+=1
            else:
                true_negative_events += 1
        else:
            if pred_event[0]=="Correct;":
                false_positive_events+=1
            else:
                false_negative_events+=1


    confusion_matrix[0][0] = true_negative_events
    confusion_matrix[0][1] = false_positive_events
    confusion_matrix[1][0] = false_negative_events
    confusion_matrix[1][1] = true_positive_events

#Calcular las métricas de la matriz de confusión
total_samples = sum(sum(row) for row in confusion_matrix)

true_positive = confusion_matrix[1][1]
false_negative = confusion_matrix[1][0]
false_positive = confusion_matrix[0][1]
true_negative = confusion_matrix[0][0]

accuracy = (true_positive + true_negative) / total_samples
precision = true_positive / (true_positive + false_positive)
recall = true_positive / (true_positive + false_negative)
f1_score = 2 * (precision * recall) / (precision + recall)

print("Threshold:",threshold)
# Muestra de las métricas
print(confusion_matrix)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-score:", f1_score)
```

Listing 7: Implementación de la evaluación a nivel de evento