

**TITULO DEL PROYECTO**

**Corrector de ejercicios en gimnasios (peso muerto y sentadilla libre)**

Universitario (a): Cristian Alejandro Duran Ignacio

Carrera: Ing. Ciencias De La Computación

Docente: Ing. Carlos Walter Pacheco Lora

Sucre 28 de mayo de 2025

Tabla de contenido

[1. Antecedentes 4](#_Toc199331978)

[2. Problema principal 4](#_Toc199331979)

[3. Abordaje de la solución 4](#_Toc199331980)

[4. Objetivo General 4](#_Toc199331981)

[5. Objetivos Específicos 5](#_Toc199331982)

[6. Fundamentos teóricos considerados en el trabajo 5](#_Toc199331983)

[a. Ámbito de la inteligencia artificial, técnicas, algoritmos, modelos base, entre otros 5](#_Toc199331984)

[b. Ámbito al que se aplicó la inteligencia artificial 6](#_Toc199331985)

[7. Metodología 6](#_Toc199331986)

[a. Sustento metodológico 6](#_Toc199331987)

[b. Técnicas de recolección de datos 7](#_Toc199331988)

[c. Materiales y herramientas 8](#_Toc199331989)

[I. Arquitectura Software Desarrollado 8](#_Toc199331990)

[II. Esquema y descripción de componentes del software 9](#_Toc199331991)

[III. Esquema y descripción de componentes de hardware. 10](#_Toc199331992)

[IV. Esquema y descripción de modelos o componentes inteligentes (esquemas y/o graficas). 11](#_Toc199331993)

[V. Valores de parámetros e hiper parámetros aplicados. 13](#_Toc199331994)

[VI. Especificaciones técnicas. 14](#_Toc199331995)

[VII. Lenguajes de programación, frameworks, entre otros. 15](#_Toc199331996)

[d. Plan de trabajo 16](#_Toc199331997)

[e. Cronograma 16](#_Toc199331998)

[i. Definición de tareas 16](#_Toc199331999)

[ii. Diagrama de Pert 17](#_Toc199332000)

[iii. Diagrama de Gantt 17](#_Toc199332001)

[8. Resultados 17](#_Toc199332002)

[a. Dataset 17](#_Toc199332003)

[I. Descripción y preprocesamiento realizado (Evidencia del antes y después) 17](#_Toc199332004)

[II. Conjunto de entrenamiento, evaluación y validación 20](#_Toc199332005)

[III. Técnicas, criterio y/o métodos aplicados para la conformación de los conjuntos datos de entrenamiento prueba y validación 20](#_Toc199332006)

[IV. Información adicional que considere importante incluir 21](#_Toc199332007)

[b. Resultados de entrenamiento y prueba (Métricas de rendimiento) 21](#_Toc199332008)

[I. Matriz de Confusión 21](#_Toc199332009)

[II. Exactitud (Accuracy) 22](#_Toc199332010)

[III. Precisión 23](#_Toc199332011)

[IV. Sensibilidad (Recall) 23](#_Toc199332012)

[V. Métrica de reporte 23](#_Toc199332013)

[VI. Curvas ROC 24](#_Toc199332014)

[VII. F-Score 25](#_Toc199332015)

[c. Resultados de aplicación y utilización 25](#_Toc199332016)

[d. Coincidencia de los resultados con los objetivos planteados. 25](#_Toc199332017)

[e. Conclusiones 25](#_Toc199332018)

[f. Recomendaciones 26](#_Toc199332019)

[9. Bibliografía 26](#_Toc199332020)

# Antecedentes

En el ámbito del entrenamiento físico, la ejecución incorrecta de los ejercicios sigue siendo un problema significativo que puede llevar a lesiones graves y obtención de resultados no óptimos. Tradicionalmente, la corrección de ejercicios en gimnasios ha dependido de entrenadores quienes tienen que estar presencialmente para poder corregir los ejercicios.

Con el avance de las tecnologías de visión por computadora y el aprendizaje automático, han surgido herramientas que automatizan el proceso de corrección de ejercicios. Por ejemplo, proyectos como “**Human Posture Detection and Correction Using MediaPipe and OpenCV**”, desarrollado con MediaPipe han utilizado algoritmos de estimación de poses usando heurística de ángulos de cada punto para analizar ejercicios como flexiones y planchas, logrando identificar y rastrear movimientos corporales con precisión en tiempo real.

# Problema principal

Riesgo de realizar ejercicios de forma incorrecta y sufrir lesiones por la mala ejecución del ejercicio, sin supervisión y poco o nulo conocimiento de la técnica.

# Abordaje de la solución

Para abordar la corrección de errores en los ejercicios de peso muerto y sentadilla libre, se empleará un enfoque basado en tecnologías de visión por computadora y redes neuronales. Se utilizará una cámara de celular para capturar video en tiempo real del usuario durante la ejecución del ejercicio, permitiendo corregir errores de forma inmediata, ya que el presente proyecto será una aplicación móvil, realizado con Flutter.

Este proyecto se apoyará en Yolo Pose para extraer puntos clave del cuerpo, analizando la biomecánica del movimiento. La detección de errores y la posterior corrección se basarán en un modelo entrenado con redes LSTM, que evaluará si la técnica es correcta o no, utilizando datos previamente etiquetados como correctos e incorrectos (errores como la postura de la espalda y las rodillas).

El conjunto de datos se construirá mediante la captura de videos de ejercicios ejecutados de manera correcta e incorrecta. Estos videos se convertirán en fotogramas que luego serán procesados usando librerías como OpenCV.

# Objetivo General

Evitar ejecuciones erróneas en dos tipos de ejercicios (peso muerto y sentadilla libre) durante las sesiones de entrenamiento con un porcentaje de precisión superior al 90%. Al lograr este objetivo, se busca mejorar la calidad del entrenamiento al proporcionar a los usuarios información en tiempo real sobre la técnica de los ejercicios mediante el constante monitoreo en el ejercicio desde el principio hasta el fin de la ejecución.

# Objetivos Específicos

* Identificar los elementos teóricos del estado del arte en cuanto a la técnica y la biomecánica del movimiento humano en ejercicios como el peso muerto y la sentadilla libre, analizando los fundamentos científicos que explican la ejecución correcta de estos ejercicios y los riesgos asociados a una mala técnica.
* Realizar un análisis y verificación del estado del arte en proyectos similares que hayan sido implementados y aplicados en el área de inteligencia artificial.
* Desarrollar un modelo que corrija los ejercicios de peso muerto y sentadilla libre utilizando características visuales y del movimiento de los usuarios, y fundamentado en técnicas de visión por computadora y aprendizaje profundo.
* Validar y evaluar el modelo comparando los resultados de corrección con un conjunto de datos de prueba, garantizando la corrección de errores.

# Fundamentos teóricos considerados en el trabajo

## Ámbito de la inteligencia artificial, técnicas, algoritmos, modelos base, entre otros

Este proyecto implementa un conjunto de técnicas y algoritmos de visión por computadora para la corrección de postura en tiempo real. El modelo base seleccionado es YOLO-Pose 11n, elegido por su capacidad de detectar 17 puntos clave articulares con alta precisión y rapidez en dispositivos móviles.

**Técnicas utilizadas:**

* **Preprocesamiento de imágenes:** Se emplea OpenCV para aplicar filtros básicos y normalización de tamaño a 640×480 píxeles para estandarizar la resolución. Cada fotograma se guarda y organiza en carpetas de resultados mediante cv2.imwrite, permitiendo visualizar las anotaciones generadas posteriormente.
* **Extracción de características:** YOLO-Pose 11n divide la imagen en cuadrículas, predice cajas delimitadoras y genera mapas de calor para ubicar articulaciones (rodillas, caderas, hombros) con baja latencia.
* **Análisis temporal:** Una red LSTM procesa secuencias de vectores normalizados para modelar la dinámica de los movimientos y diferenciar ejecuciones correctas e incorrectas.

**Algoritmos aplicados:**

* **YOLO-Pose 11n:** Aplicación del algoritmo de detección única que divide la imagen en cuadrículas, predice simultáneamente cajas delimitadoras y mapas de calor para puntos clave, utiliza regresión de bounding boxes y no-maximum suppression (NMS) para refinar resultados.
* **LSTM (Long Short‑Term Memory):** Variante de redes neuronales recurrentes que emplea puertas de entrada, olvido y salida para gestionar el flujo de información en secuencias temporales, capaz de ajustar pesos mediante backpropagation through time.

## Ámbito al que se aplicó la inteligencia artificial

La Inteligencia Artificial se aplicó en el ámbito del deporte y fitness desde el año 2010 en deportes profesionales como fútbol y baloncesto que empleaban visión por computadora para rastrear estadísticas. Se aplicaron también para la corrección de técnicas en golf y tenis entre 2013-2015, analizando swings y golpes mediante sensores y cámaras. En 2016 se implementó para el reconocimiento de posturas en yoga usando MediaPipe y redes neuronales. Para 2017 ya se usaban sistemas LSTM + OpenPose para analizar secuencias de movimiento en ejercicios como flexiones.

En el presente proyecto se está aplicando en el ámbito de ejercicios en gimnasios y se usa visión por computadora y aprendizaje profundo, utilizando YOLO-Pose 11n, la aplicación extrae en tiempo real 17 puntos clave corporales desde la cámara de un teléfono inteligente, mientras que una red LSTM procesa cada secuencia de puntos normalizados para identificar desviaciones de la técnica correcta directamente en el entorno de gimnasios para la corrección de postura durante el levantamiento de peso muerto y la sentadilla libre.

# Metodología

## Sustento metodológico

La metodología empleada es un **diseño experimental cuantitativo**, en el que se evalúa la eficacia de un modelo de corrección postural. Se contempla cinco fases que son:

* **Recopilación de datos:**

Se realizará una recolección de videos que incluyan la ejecución correcta e incorrecta de los ejercicios de peso muerto y sentadilla empleando un dispositivo móvil a 1080px y 30 fps. Estos videos se convertirán en fotogramas que representen diversas posturas y movimientos.

* **Preprocesamiento de datos:**

Los videos recopilados serán sometidos a un proceso de conversión en fotogramas para facilitar su análisis. Luego, se procederá a eliminar los fotogramas repetidos, ya que la presencia de fotogramas duplicados puede afectar negativamente la calidad de los datos y la eficiencia del modelo. También se extraerán 17 puntos clave normalizados como por ejemplo (rodillas, caderas, hombros) como vectores de entrada para la LSTM.

* **Entrenamiento del modelo LSTM:**

Los datos preprocesados alimentarán un modelo de redes neuronales LSTM el cual será entrenado para identificar secuencias de movimientos correctas e incorrectas. Durante esta fase, el modelo aprenderá de manera autónoma a identificar los movimientos correctos e incorrectos del ejercicio.

* **Validación y ajustes finales:**

Después de finalizar el entrenamiento, se llevará a cabo una validación del modelo utilizando un conjunto de datos independiente. A partir de los resultados, se realizaron ajustes en hiperparámetros para optimizar el equilibrio entre precisión y tiempo de respuesta.

* **Desarrollo de la aplicación móvil:**

Se desarrollará una interfaz móvil que permita a los usuarios capturar los ejercicios en tiempo real mediante la cámara del dispositivo móvil, procesar los datos, y recibir retroalimentación inmediata sobre su técnica.

## Técnicas de recolección de datos

Para garantizar la calidad y representatividad del dataset, se emplearon las siguientes técnicas:

* **Grabación en campo controlado:**

Se recopilarán videos de ejercicios de peso muerto y sentadilla realizados en entornos controlados, como gimnasios o centros de entrenamiento, asegurando que los movimientos sean claros y las condiciones de iluminación sean óptimas. Las grabaciones se hicieron a 1080px y 30 fps desde un dispositivo móvil fijo a dos metros del sujeto usando un trípode.

* **Estrategias de organización:**

Los videos se dividieron en secuencias de 30 fotogramas para preservar la continuidad temporal. Cada secuencia se organizó en carpetas según los tipos de errores que se contemplaron para este proyecto (sentadilla: rodillas correcto, rodillas incorrecto, cadera correcta, cadera incorrecta), (peso muerto: columna correcta, columna incorrecta, extensión correcta, extensión incorrecta)

* **Preprocesamiento automatizado:**

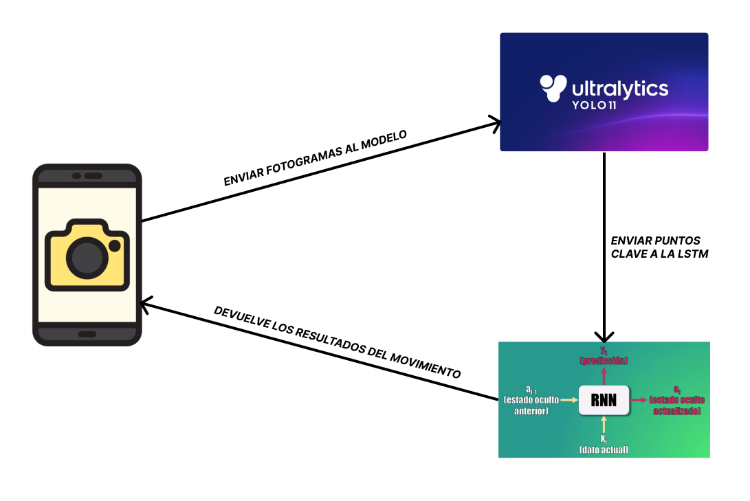
Con OpenCV, se cargaron (cv2.imread), filtraron y normalizaron los fotogramas a 640 x 480 px. Se generaron estructuras de directorios de resultados donde cv2.imwrite almacenó los fotogramas anotados tras la extracción de puntos clave.

* **Revisión de protocolos de grabación:**

Se consultaron estudios previos sobre corrección de postura y estimación de pose para definir protocolos de grabación, criterios de calidad de imagen y esquemas de etiquetado [2][3].

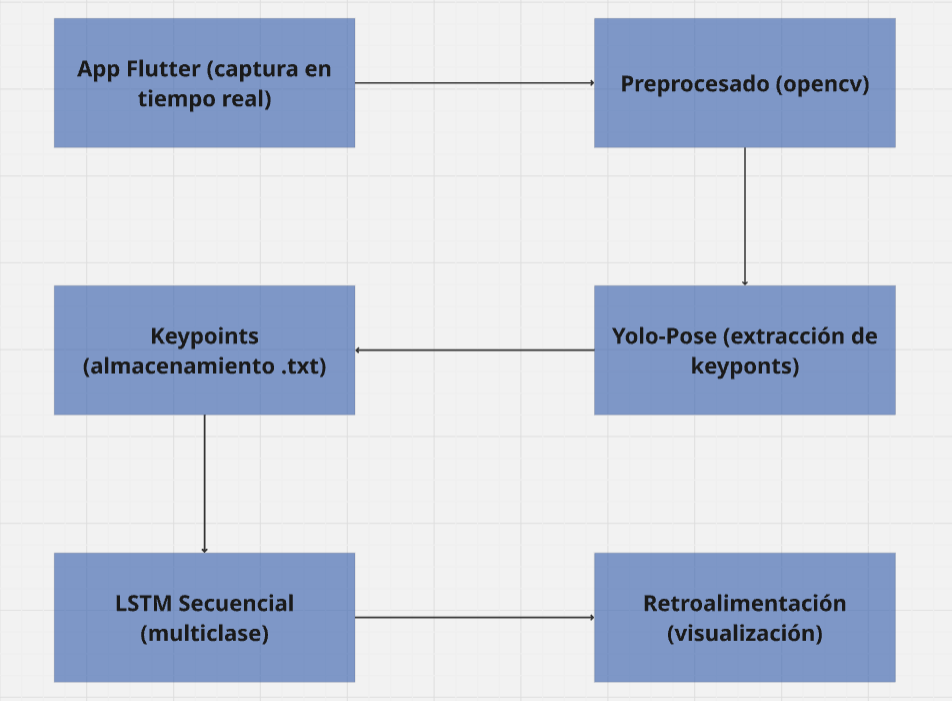
## Materiales y herramientas

### Arquitectura Software Desarrollado



* El componente de la aplicación móvil es el software desarrollado con Flutter, un framework de código abierto para crear aplicaciones nativas para iOS y Android. Esta aplicación recibe las imágenes desde la cámara, las procesa y las envía al modelo Yolo Pose para la extracción de puntos clave del cuerpo.
* El modelo Yolo Pose recibe los fotogramas procesados desde la aplicación y devuelve las coordenadas de los puntos clave del cuerpo (rodillas, caderas, hombros) en tiempo real.
* La red LSTM se encarga de analizar las secuencias temporales de los movimientos del usuario. Utiliza los datos de los puntos clave proporcionados por Yolo Pose para detectar errores en los ejercicios. Este modelo está entrenado para distinguir entre movimientos correctos e incorrectos, proporcionando retroalimentación inmediata de la corrección.

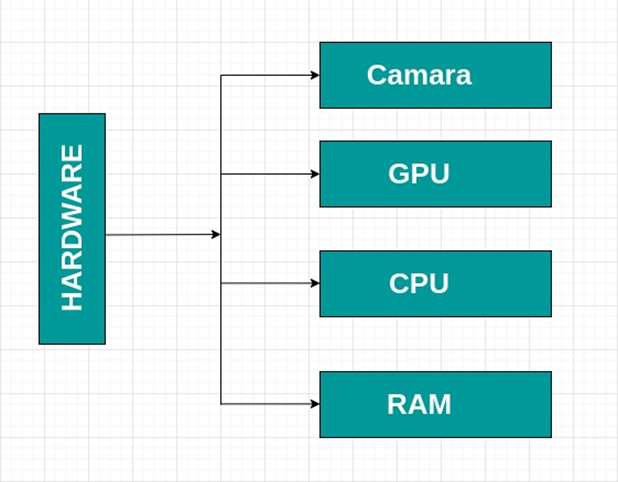
### Esquema y descripción de componentes del software



* **App Flutter (captura y UI):** Captura el stream de video desde la cámara nativa y envía fotogramas preprocesados y recibe resultados para mostrarlos en tiempo real y por último renderiza la interfaz de usuario con indicadores de corrección mediante widgets de Flutter.
* **Preprocesado de imágenes (OpenCV):** Aplica conversión de formato y redimensionado (640x480 px).
* **Yolo-Pose (extracción de keyponts):** Procesa cada fotograma para detectar 17 puntos clave corporales, devuelve coordenadas normalizadas en formato el formato requerido a la aplicación.
* **Keypoints (almacenamiento temporal):** Almacena secuencias de vectores de 34 coordenadas (x, y) para cada ventana de fotogramas.
* **LSTM Secuencial:** Recibe secuencias de puntos clave y aplica la red LSTM, emite una etiqueta en base al tipo de error y un score de confianza.
* **Retroalimentación (visualización):** Traducir la salida de la LSTM en alertas visuales o auditivas.

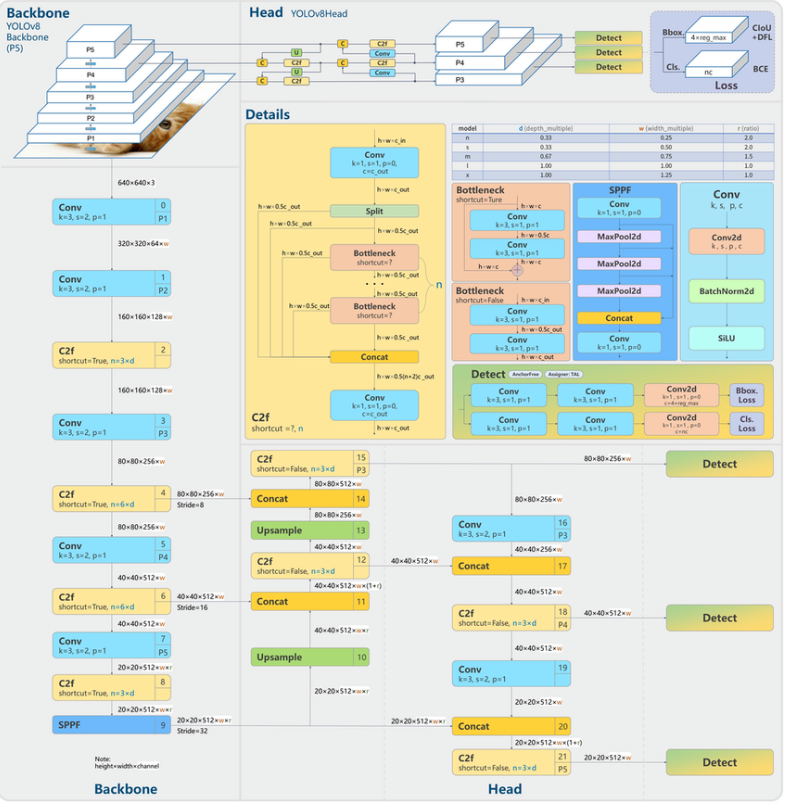
### Esquema y descripción de componentes de hardware.

### 



* + **Cámara (Cam):** La cámara es el componente de entrada que captura el video en tiempo real. La aplicación utiliza la cámara para adquirir los datos visuales de los movimientos de los ejercicios que se corregirán.
  + **CPU (Unidad Central de Procesamiento):** La CPU, el cerebro de nuestro dispositivo, es responsable de procesar los datos capturados por la cámara. Realiza tareas como la conversión de imágenes en datos digitales, el control de flujo de la aplicación y el procesamiento inicial de las imágenes.
  + **GPU (Unidad de Procesamiento Gráfico):** La GPU es un componente fundamental para acelerar el procesamiento en tiempo real de YOLO Pose y redes LSTM. Su capacidad de procesamiento paralelo permite ejecutar eficientemente las operaciones intensivas como procesar múltiples fotogramas simultáneamente, ejecutar en paralelo las capas de las redes neuronales, manejar las operaciones matriciales de las LSTM, mantener baja latencia en la inferencia del modelo.
  + **RAM (Memoria de Acceso Aleatorio):** La RAM proporciona un espacio de memoria temporal para almacenar datos críticos durante el procesamiento. Los datos de imágenes procesados, los resultados de la detección y otra información relevante se almacenan en la RAM para su acceso rápido. Esto es esencial para la eficiencia de la aplicación en tiempo real.

### Esquema y descripción de modelos o componentes inteligentes (esquemas y/o graficas).

**Yolo Pose:**

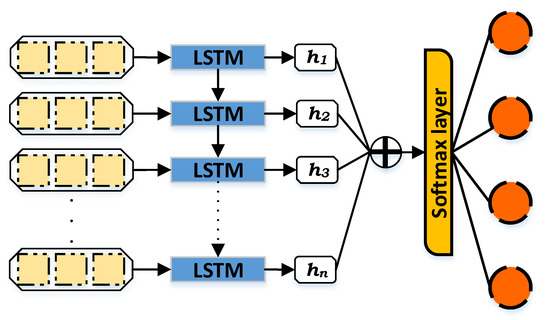
1. **Espina dorsal (Backbone):**

* El backbone en Yolo-pose es responsable de extraer características de la entrada (generalmente una imagen)
* Utiliza una versión modificada de la arquitectura CSPDarknet53 como su espina dorsal.
* La espina dorsal procesa la entrada y aprende características a diferentes niveles de abstracción
* Los modelos preentrenados, como ResNet, Xception o MobileNet, a menudo se utilizan como backbones en YOLOv8-pose para aprovechar su capacidad de extracción de características.

1. **Cabeza (Head):**

* La cabeza sigue al backbone y se encarga de tareas específicas, como la detección de objetos o la segmentación.
* En Yolo-pose, la cabeza incluye capas adicionales que procesan las características extraídas por el backbone.
* La configuración de la cabeza puede variar según la tarea. Por ejemplo, en la detección de objetos, la cabeza genera cajas delimitadoras y puntajes de confianza para cada objeto detectado.

**LSTM:**



1. **Espina dorsal (Backbone):**

* El backbone en una LSTM es responsable de extraer características de la entrada (generalmente una secuencia de datos, como una serie temporal o texto). A diferencia de las RNN estándar, que a menudo utilizan capas recurrentes simples, las LSTM emplean una estructura más compleja.
* El backbone procesa la entrada y aprende características a diferentes niveles de abstracción.

1. **Célula LSTM:**

* La unidad fundamental es la célula LSTM. Cada célula tiene tres componentes principales:
* **Celda:** Almacena información a lo largo del tiempo y permite que la LSTM mantenga dependencias a largo plazo.
* **Puerta de entrada (Input Gate):** Decide qué nueva información se debe agregar a la celda en función de la entrada actual y la información anterior.
* **Puerta de olvido (Forget Gate):** Controla qué información antigua se debe descartar de la celda.
* **Puerta de salida (Output Gate**): Determina qué información de la celda se debe utilizar como salida.

### Valores de parámetros e hiper parámetros aplicados.

**Parámetros Sentadilla Libre:**

* **Units en capas LSTM:** 64 número de neuronas por capa recurrente (tres capas LSTM), controla la capacidad de modelar dependencias temporales.
* **Units en capa LSTM intermedia:** 128 neuronas en la segunda capa recurrente.
* **Units en capa BatchNormalization:** normaliza activaciones tras LSTM para acelerar convergencia.
* **Units en capa Dense intermedia**: 64 neuronas en la primera capa densa con activación ReLU, seguido de dropout 0,3.
* **Units en segunda capa Dense:** 32 neuronas con activación ReLU, seguido de dropout 0,3.
* **Units en capa de salida:** 4 neuronas correspondientes a las cuatro clases (caderas correctas/incorrectas, rodillas correctas/incorrectas) con activación Softmax.

**Hiper parámetros Sentadilla Libre:**

* **Tamaño de lote (Batch Size):** 32 indica el número de ejemplos de entrenamiento que se utilizan en cada iteración durante el entrenamiento del modelo.
* **Número de épocas (Epochs):** 70 indica la cantidad de veces que el modelo pasará por todo el conjunto de datos de entrenamiento durante el entrenamiento.
* **Función de pérdida (Loss):** sparse\_categorical\_crossentropy especifica la función de pérdida que se utilizará durante el entrenamiento para evaluar la precisión del modelo.
* **Optimizador (Optimizer):** optimizer="adam" especifica el algoritmo de optimización que se utilizará para ajustar los parámetros del modelo durante el entrenamiento.
* **learning\_rate:**  por defecto de Adam (~0,001) → controla el tamaño del paso en el ajuste de pesos.
* EarlyStopping: patience=8 detiene si no mejora la validación.
* ReduceLROnPlateau: factor=0.3, patience=5, min\_lr=1e-6 reduce la tasa de aprendizaje al estancarse la validación.

**Parámetros Peso Muerto:**

* **Units en capas LSTM:** 64 número de neuronas por capa recurrente (tres capas LSTM), controla la capacidad de modelar dependencias temporales.
* **Units en capa LSTM intermedia:** 128 neuronas en la segunda capa recurrente.
* **Units en capa BatchNormalization:** normaliza activaciones tras LSTM para acelerar convergencia.
* **Units en capa Dense intermedia**: 64 neuronas en la primera capa densa con activación ReLU, seguido de dropout 0,5.
* **Units en segunda capa Dense:** 32 neuronas con activación ReLU, seguido de dropout 0,5.
* **Units en capa de salida:** 4 neuronas correspondientes a las cuatro clases (columna correcta/incorrecta, extensión correcta/incorrecta) con activación Softmax.

**Hiper parámetros Peso Muerto:**

* **Tamaño de lote (Batch Size):** 32 indica el número de ejemplos de entrenamiento que se utilizan en cada iteración durante el entrenamiento del modelo.
* **Número de épocas (Epochs):** 120 indica la cantidad de veces que el modelo pasará por todo el conjunto de datos de entrenamiento durante el entrenamiento.
* **Función de pérdida (Loss):** sparse\_categorical\_crossentropy especifica la función de pérdida que se utilizará durante el entrenamiento para evaluar la precisión del modelo.
* **Optimizador (Optimizer):** optimizer="adam" especifica el algoritmo de optimización que se utilizará para ajustar los parámetros del modelo durante el entrenamiento.
* **learning\_rate:**  por defecto de Adam (~0,001) → controla el tamaño del paso en el ajuste de pesos.
* EarlyStopping: patience=8 detiene si no mejora la validación.
* ReduceLROnPlateau: factor=0.5, patience=5, min\_lr=1e-6 reduce la tasa de aprendizaje al estancarse la validación.

### Especificaciones técnicas.

El desarrollo del proyecto se sustentó en librerías especializadas en visión por computadora y aprendizaje profundo:

**Librerías de preprocesamiento y visualización:**

* **OpenCV (cv2):** Módulo fundamental para la lectura (cv2.imread) y escritura (cv2.imwrite) de imágenes, segmentación de video en fotogramas, normalización de resolución y generación de anotaciones visuales.
* **Math:** Proporciona funciones matemáticas básicas como operaciones trigonométricas o cálculo de distancias euclidianas, útiles en el procesamiento de coordenadas de puntos clave.
* **NumPy:** Biblioteca para estructuras de datos de arrays multidimensionales y operaciones vectorizadas, esencial para manejar matrices de coordenadas y transformaciones rápidos en el pipeline de datos.
* **Pandas:** Facilita la construcción y manipulación de dataframes, permitiendo organizar resultados de detección en tablas, realizar estadísticas descriptivas y exportar datasets a CSV o TXT.
* **Matplotlib (pyplot):** Genera visualizaciones estáticas como curvas de aprendizaje, matrices de confusión y gráficos de métricas (accuracy, loss, ROC), soportando ajustes de estilo y anotaciones dinámicas.
* **Ultralytics / YOLO:** Incluye el paquete principal ultralytics y la clase YOLO para cargar el modelo Yolo-Pose-11n. El comando ultralytics.checks() valida la configuración del entorno y las dependencias requeridas antes de iniciar la inferencia.

**Frameworks de modelado y entrenamiento:**

* **typing (Sequence):** Se usa para anotar colecciones de datos de entrada y garantizar tipado estático en funciones de preprocesamiento.
* **TensorFlow/Keras (Sequential, LSTM, Dropout, Dense, BatchNormalization):** API de alto nivel para construir modelos secuenciales.
* **Sequential:** Contenedor lineal de capas.
* **LSTM:** Capa recurrente que retiene memoria de largo plazo.
* **Dropout:** Regularizador que desactiva neuronas aleatoriamente para evitar sobreajuste.
* **Dense:** Capa completamente conectada para clasificación binaria.
* **BatchNormalization:** Normaliza activaciones para acelerar y estabilizar el entrenamiento.
* **Keras Callbacks (EarlyStopping, ReduceLROnPlateau):**
* **EarlyStopping**: Detiene el entrenamiento si no hay mejoría en la métrica de validación después de un número de epochs.
* **ReduceLROnPlateau:** Reduce la tasa de aprendizaje cuando la métrica de validación se estanca, favoreciendo convergencia más fina.
* **Scikit-learn:**
* **datasets:** Provee utilidades para generar y cargar conjuntos de datos de prueba.
* **train\_test\_split:** Divide aleatoriamente el dataset en conjuntos de entrenamiento, validación y prueba.
* **Precision, Recall:** Métricas de Keras para monitorear balance entre verdaderos positivos y negativos durante el entrenamiento.
* **confusion\_matrix, precision\_score, recall\_score:** Funciones para evaluar el rendimiento final del modelo en el conjunto de prueba.

### Lenguajes de programación, frameworks, entre otros.

**Lenguaje de Programación y Marcos de Trabajo:**

* Python es el lenguaje principal utilizado en el desarrollo del proyecto.
* Flutter es un marco de desarrollo de código abierto de Google que utiliza el lenguaje de programación Dart para crear aplicaciones móviles con interfaces de usuario atractivas y fluidas.

**Entorno de Desarrollo y ejecución:**

* **IDE:** Visual Studio Code.
* **Entorno virtual:** Se creo uno especifico para trabajar con librerías y dependencias de visión por computadora **IA\_env (Python 3.12.9)**
* **Hardware de entrenamiento:** estación con GPU NVIDIA para acelerar entrenamiento de LSTM.
* **Dispositivo de inferencia:** smartphone Android de gama media/alta S21 Ultra.

## Plan de trabajo

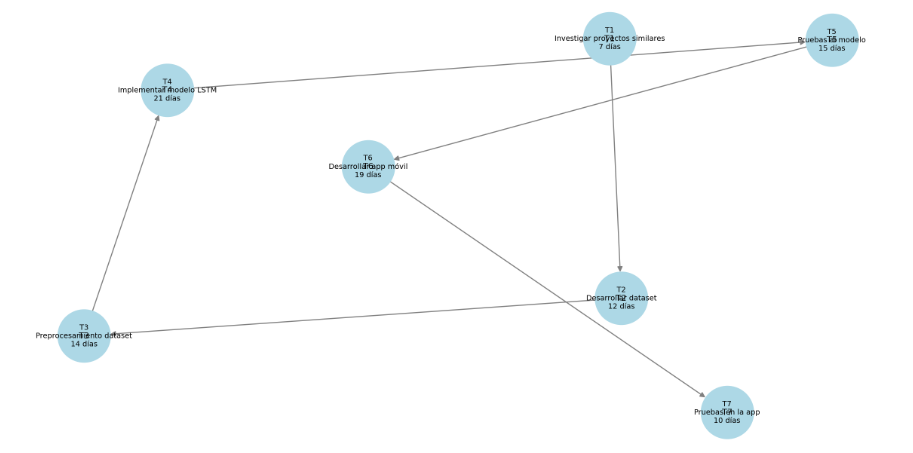
* **Desarrollar el dataset de ejercicios:** Grabar videos y organizar por carpetas dependiendo el ejercicio de peso muerto (con sus respectivos errores tanto de columna y extensión) y sentadilla (con sus respectivos errores tanto de rodillas y cadera) ejecutados de manera correcta e incorrecta.
* **Preparación de datos:** Utilizar Free Video to JPG Converter para convertir a fotogramas los videos recolectados, eliminar fotogramas repetidos manualmente y luego sacar puntos clave de la secuencia de fotogramas con el modelo Yolo-pose para guardar los puntos clave en archivos .txt con los cuales se entrenará el modelo LSTM.
* **Entrenamiento del modelo LSTM:** Configurar un modelo secuencial preparando al modelo para aprender a identificar los movimientos correctos o incorrectos de los errores que se tomaron en cuenta para este proyecto.
* **Realización de pruebas al modelo entrenado:** Evaluar el modelo entrenado mediante pruebas exhaustivas, verificando su capacidad para identificar con precisión los errores en la ejecución de los ejercicios de peso muerto y sentadilla.
* **Desarrollo de la aplicación móvil para Android:** Desarrollar una interfaz de usuario en el entorno de Visual Studio Code (VSCode) que sea compatible con dispositivos Android, permitiendo a los usuarios interactuar con la aplicación.
* **Realización de pruebas en la APP desarrollada**: Ejecutar pruebas en la aplicación móvil completa, asegurándose de que funcione correctamente al identificar con precisión los errores en la ejecución de los ejercicios de peso muerto y sentadilla.

## Cronograma

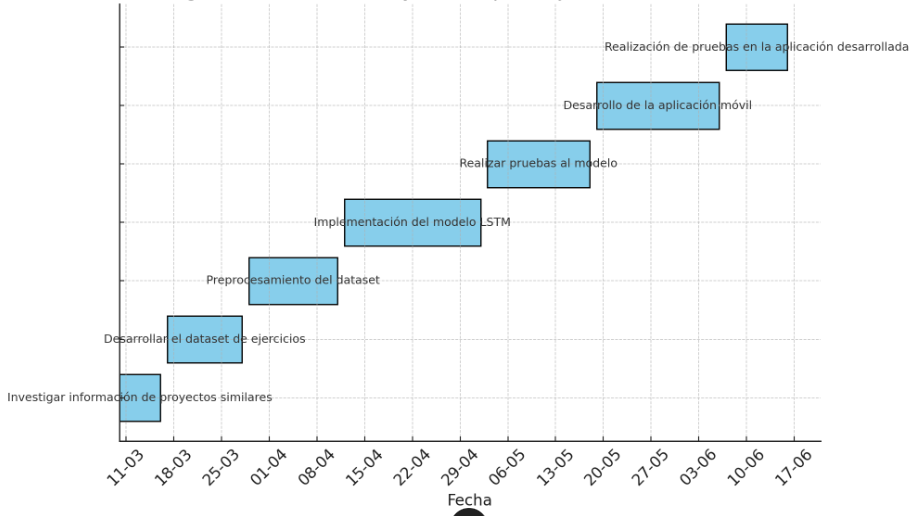
### Definición de tareas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Tareas** | **Duración** | **Fecha** | **Dependencia** |
| **T1** | Investigar información de proyectos similares | 1 semana | 10-03-2025 – 16-03-2025 | - |
| **T2** | Desarrollar el dataset de ejercicios | 2 semanas | 17-03-2025 – 28-03-2025 | T1 |
| **T3** | Preprocesamiento del dataset | 2 semanas | 29-03-2025 – 11-04-2025 | T2 |
| **T4** | Implementación del modelo LSTM | 3 semanas | 12-04-2025 – 02-05-2025 | T3 |
| **T5** | Realizar pruebas al modelo | 2 semanas | 03-05-2025 – 18-05-2025 | T4 |
| **T6** | Desarrollo de la aplicación móvil | 3 semanas | 19-05-2025 – 06-06-2025 | T5 |
| **T7** | Realización de pruebas en la aplicación desarrollada | 2 semanas | 07-06-2025 – 16-06-2025 | T6 |

### Diagrama de Pert



### Diagrama de Gantt

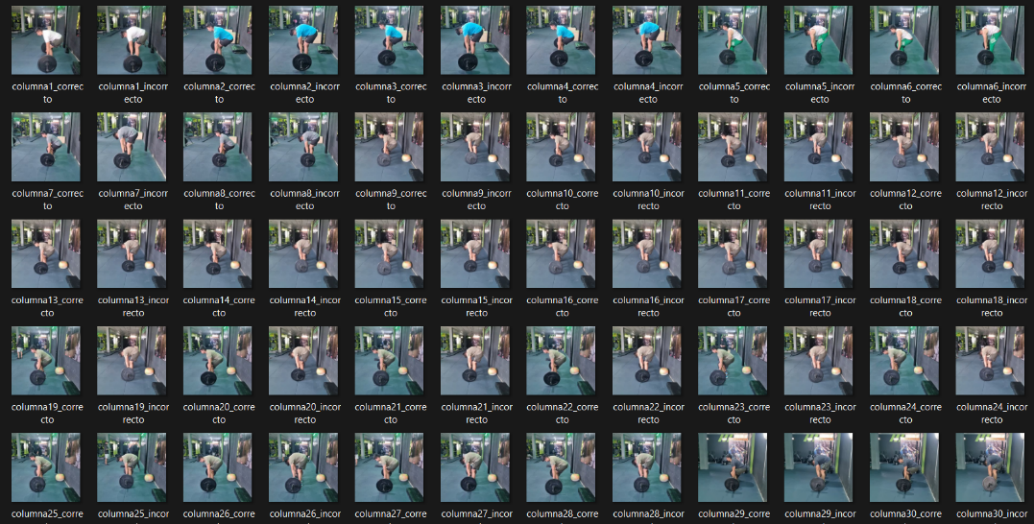


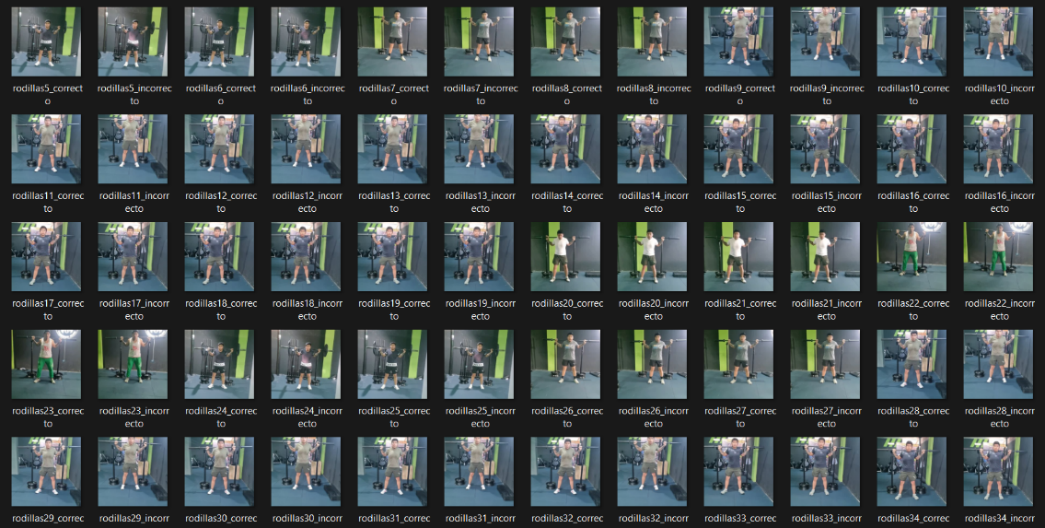
# Resultados

## Dataset

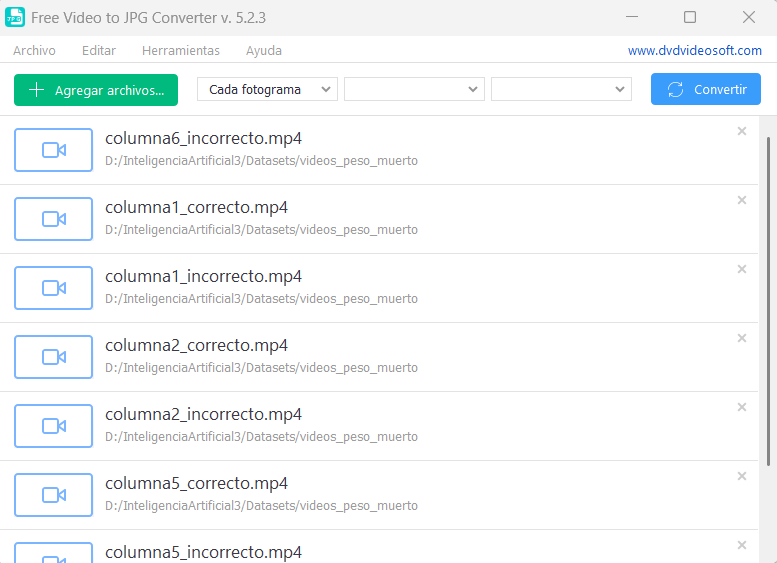
### Descripción y preprocesamiento realizado (Evidencia del antes y después)

El preprocesamiento de los datos se llevó a cabo en varias etapas. Se grabaron videos de ejercicios de peso muerto y sentadillas tanto correctos como incorrectos.

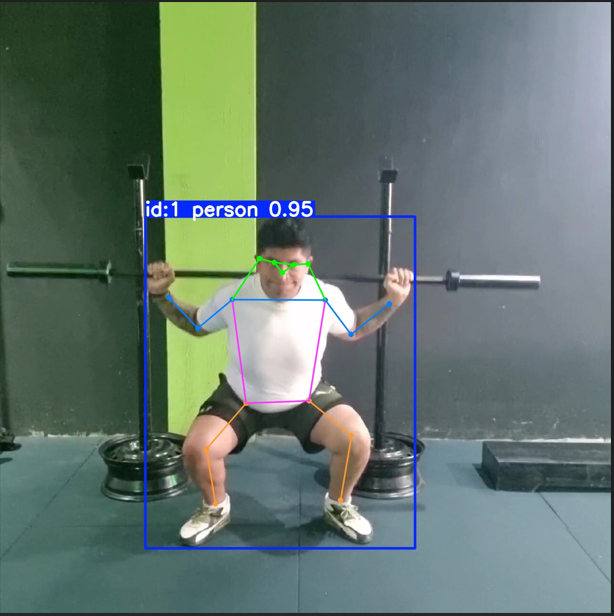




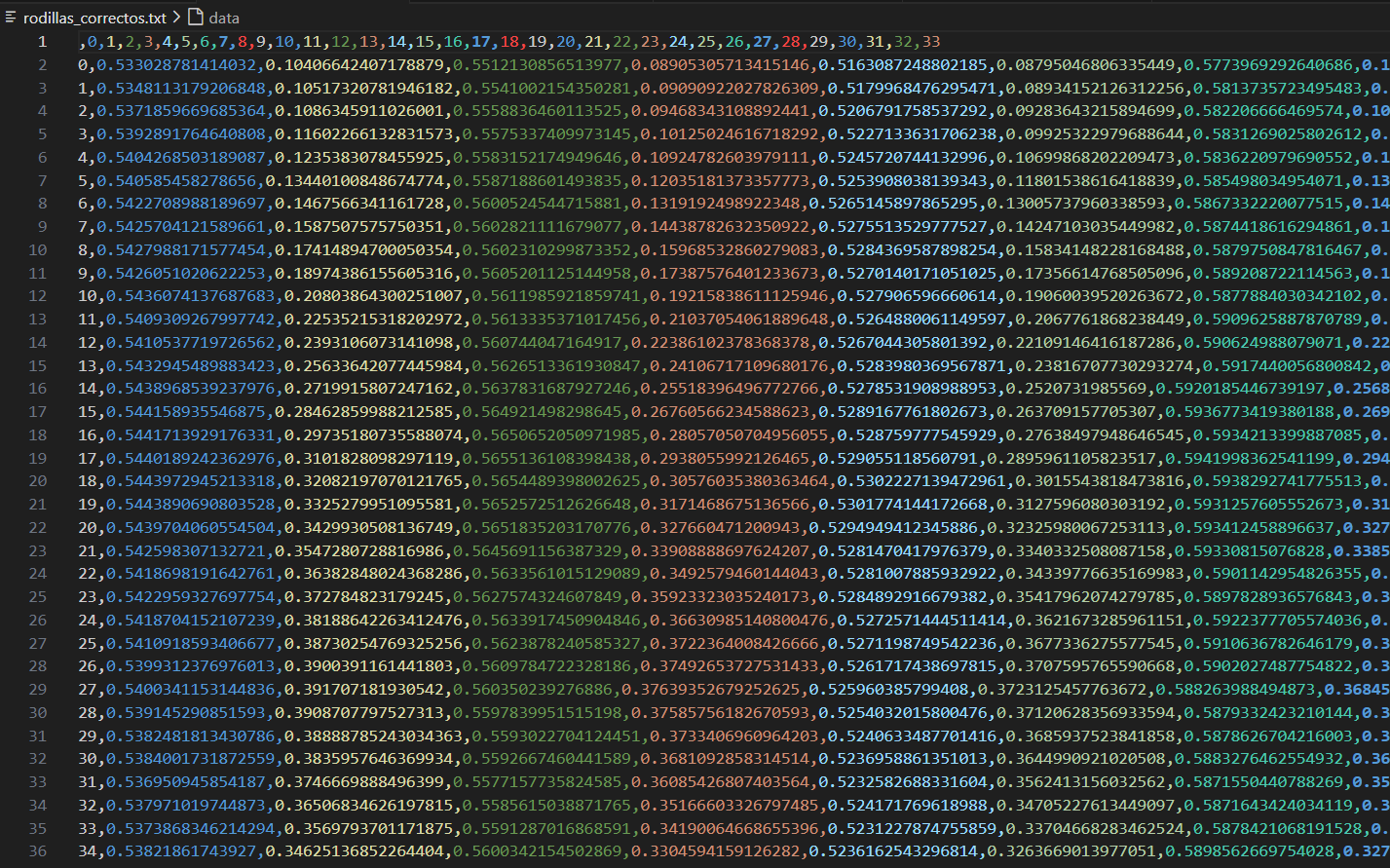
Luego se saca fotogramas de los videos recortados con Free Video to JPG Converter



Al final se utiliza el modelo Yolo-pose para obtener los puntos clave de persona al realizar los ejercicios, se recuperan esos puntos clave y se guardan en un archivo .txt resultado de los puntos clave en imágenes y el archivo.txt







### Conjunto de entrenamiento, evaluación y validación

**Sentadilla:**

Mi conjunto de datos de mi dataset de sentadilla son en total 45.108 el cual se divide en datos de entrenamiento, validación y prueba. Cada muestra corresponde a una ventana de 60 fotogramas x 34 coordenadas.

* **Entrenamiento:** 34.380 muestras
* **Validación:** 6.068 muestras
* **Prueba:** 4.495 muestras

**Peso muerto:**

Mi conjunto de datos de mi dataset de peso muerto son en total 33.165 el cual se divide en datos de entrenamiento, validación y prueba. Cada muestra corresponde a una ventana de 30 fotogramas x 34 coordenadas.

* **Entrenamiento:** 25.892 muestras
* **Validación:** 4.570 muestras
* **Prueba:** 3.385 muestras

### Técnicas, criterio y/o métodos aplicados para la conformación de los conjuntos datos de entrenamiento prueba y validación

Se utilizo train\_test\_split de scikit-learn. En este caso, se dividió el dataset en tres conjuntos distintos 76.5% para entrenamiento (X\_train y y\_train), 13.5% para validación (X\_val y y\_val) y 10% para pruebas (X\_test y y\_test). Esto se hizo para poder entrenar el modelo con el conjunto principal (76.5%), ajustar hiper parámetros y monitorear el rendimiento durante el entrenamiento con el conjunto de validación (13.5%) y evaluar el desempeño final con datos completamente independientes (10% de prueba).

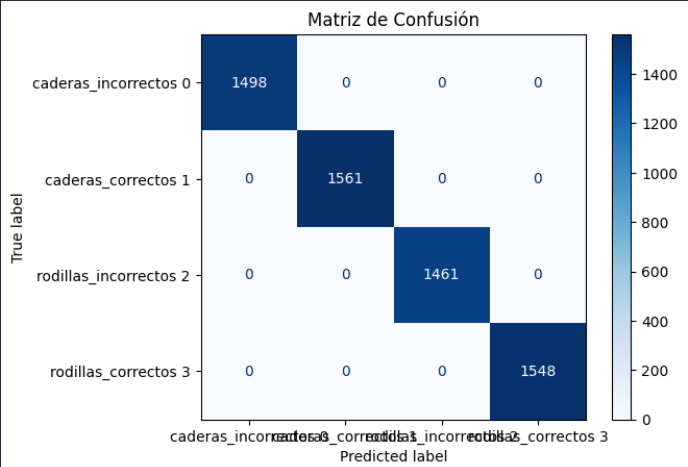
### Información adicional que considere importante incluir

Los videos de voluntarios se almacenan en carpetas seguras y se comparten únicamente para fines académicos, respetando normativas de protección de datos personales, además se sugiere incluir a varios sujetos (edad, género, complexión), también cabe recalcar que el modelo no ha sido probado en ejercicios combinados o multi angulares (por ejemplo, variaciones de sentadilla), limitando su aplicabilidad también es importante considerar la exploración de arquitecturas ligeras de transformers adaptados a secuencias de pose para comparar con LSTM en términos de precisión y latencia.

## Resultados de entrenamiento y prueba (Métricas de rendimiento)

### Matriz de Confusión

**Sentadilla Libre**

****

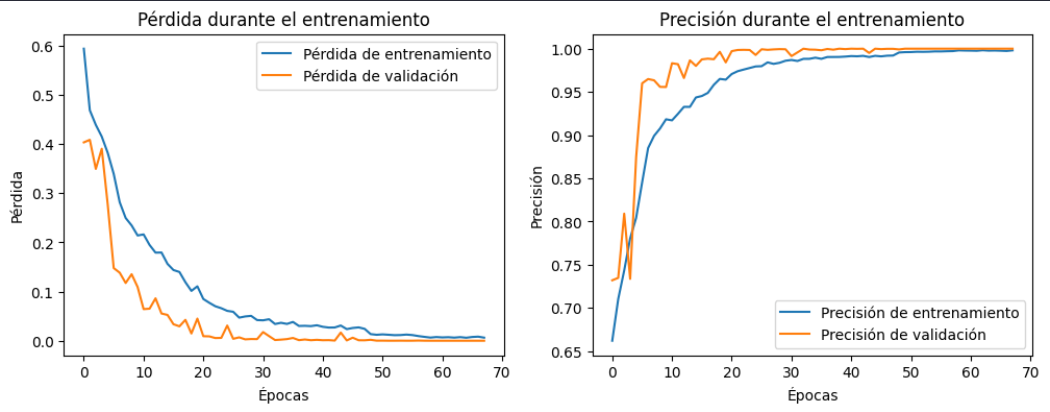
**Peso Muerto**

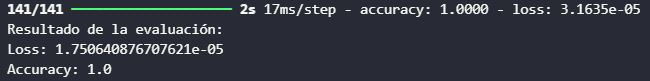


La matriz de confusión muestra cómo de bien clasifica el modelo las imágenes según las clases que tienen. Cada clase tiene una fila y una columna en la matriz. Los valores de la diagonal son los que el modelo acierta, y los valores fuera de la diagonal son los que el modelo se equivoca.

### Exactitud (Accuracy)

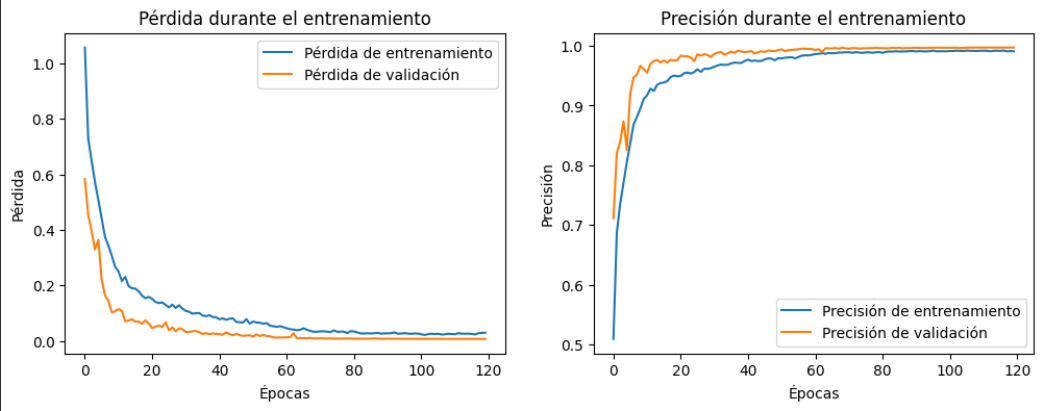
**Sentadilla Libre**

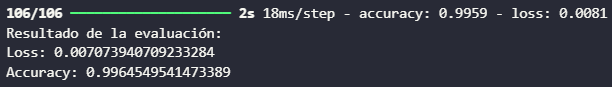
****

****

**Accuracy**: 1.0 La precisión o 'accuracy' del modelo es del 100%. Esto significa que el 100% de las veces, el modelo hizo una predicción correcta.

**Peso Muerto**





**Accuracy**: 0.99 La precisión o 'accuracy' del modelo es del 99%. Esto significa que el 99% de las veces, el modelo hizo una predicción correcta.

### Precisión

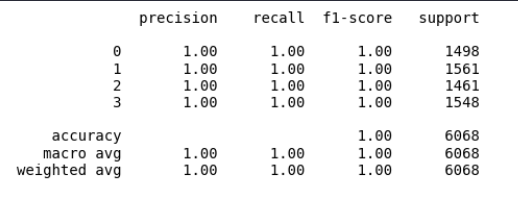
**Precisión score**: La precisión es la proporción de verdaderos positivos (TP) sobre la suma de verdaderos positivos y falsos positivos (FP). En otras palabras, representa la habilidad del modelo para identificar correctamente los casos positivos entre todos los casos que predijo como positivos

### Sensibilidad (Recall)

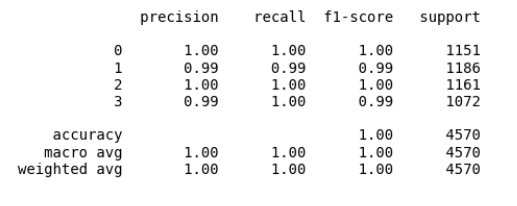
**El recall** es la proporción de verdaderos positivos (TP) sobre la suma de verdaderos positivos y falsos negativos (FN). Representa la habilidad del modelo para encontrar todos los casos positivos

### Métrica de reporte

**Sentadilla Libre**

****

**Peso Muerto**

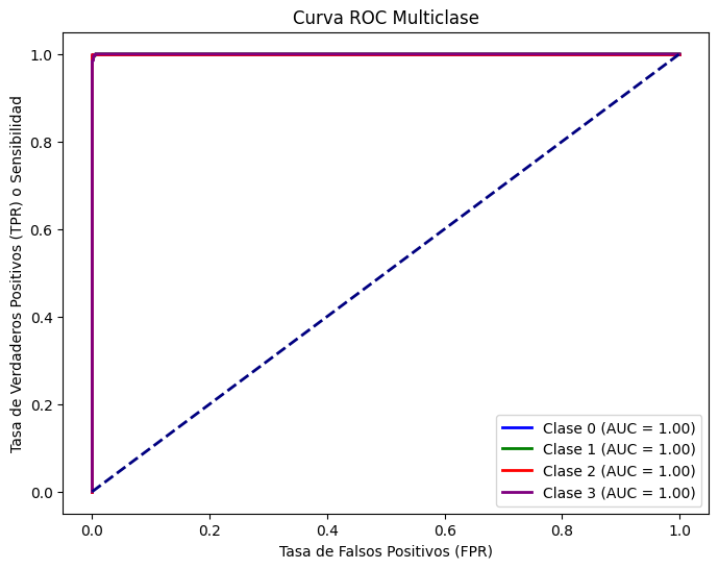


### Curvas ROC

**Sentadilla Libre**

****

**Peso Muerto**



**Curvas ROC**

El área bajo la curva ROC (AUC) cuantifica la capacidad del modelo para distinguir correctamente entre las diferentes clases a lo largo de todos los umbrales posibles. Su valor oscila entre 0 y 1, donde 0,5 indica un comportamiento equivalente al azar y 1,0 representa una separación perfecta entre clases. En la curva ROC multi clasificación presentada, todas las clases alcanzan un AUC de 1,00, lo que evidencia que el modelo clasifica sin errores las categorías de postura de sentadilla libre (cadera y rodillas correctas e incorrectas) y peso muerto (columna y extensión correctas e incorrectas) en el conjunto de prueba.

### F-Score

**El F1-score** es la media armónica de precisión y recall. Es útil cuando se desea tener en cuenta tanto la precisión como el recall en una sola métrica. Cuanto más alto sea el F1-score, mejor será el equilibrio entre precisión y recall.

## Resultados de aplicación y utilización

## Coincidencia de los resultados con los objetivos planteados.

## Conclusiones

El desarrollo de este proyecto se enfocó en alcanzar cuatro objetivos fundamentales:

1. **Investigación y Análisis Biomecánico:**

* Se identificaron y validaron científicamente los errores críticos en peso muerto y sentadilla, incluyendo:
* Peso muerto: Espalda arqueada y hiperextensión lumbar.
* Sentadilla: Rodillas valgas (hacia adentro) e inclinación frontal excesiva.
* Se establecieron parámetros biomecánicos de referencia para la ejecución correcta de ambos ejercicios.

1. **Realizar un análisis y verificación del estado del arte en proyectos similares:**

* Se optimizó el uso de YOLO Pose para detectar los 17 puntos clave corporales con precisión (>90% en pruebas).
* Las redes LSTM demostraron efectividad para analizar secuencias temporales

1. **Desarrollo y Validación del Modelo:**
   * La implementación y entrenamiento de las capas LSTM para la clasificación centrado en la detección precisa y la clasificación de movimientos incorrectos en los ejercicios de peso muerto y sentadilla es un paso significativo en el proyecto. El modelo ha demostrado realizar tareas precisas de clasificación de los movimientos correctos e incorrectos.
2. **Validar y evaluar el modelo:**

## Recomendaciones

* Ajuste Continuo y Entrenamiento del Modelo Implementar una estrategia de ajuste constante de hiper parámetros y entrenamiento del modelo utilizando Yolo Pose y LSTM para mejorar su precisión y eficiencia en la corrección de posturas y el conteo de repeticiones.
* Ampliación del Dataset con Variedad de Ejercicios y Escenarios Incrementar gradualmente el dataset, incorporando videos de diversos ejercicios y condiciones. Una mayor diversidad en el conjunto de datos puede mejorar la capacidad del modelo para adaptarse a diferentes situaciones y técnicas de ejercicios.
* Investigación Continua sobre la Integración con Flutter Explorar continuamente bibliotecas y actualizaciones para la integración de Yolo Pose y LSTM con Flutter, priorizando la mejora en la detección en tiempo real y optimización de recursos en los dispositivos Android.

## Bibliografía

1. Pawel Kapica. (29-08-2024). **Using pose estimation algorithms to build a simple gym training aid app*.***  
   [https://medium.com/@pawelkapica/using-pose-estimation-algorithms-to-build-a-simple-gym-training-aid-app-ef87b3d07f94](https://medium.com/@pawelkapica/using-pose-estimation-algorithms-to-build-a-simple-gym-training-aid-app-ef87b3d07f94" \t "_new)
2. Hitesh Kotte, Milos Kravcik, Nghia Duong-Trung (October-2023). **Real-time posture correction in gym exercises: A computer vision-based approach for performance analysis, error classification and feedback*.***[https://www.researchgate.net/publication/374659509\_Real-Time\_Posture\_Correction\_in\_Gym\_Exercises\_A\_Computer\_Vision-Based\_Approach\_for\_Performance\_Analysis\_Error\_Classification\_and\_Feedback](https://www.researchgate.net/publication/374659509_Real-Time_Posture_Correction_in_Gym_Exercises_A_Computer_Vision-Based_Approach_for_Performance_Analysis_Error_Classification_and_Feedback" \t "_new)
3. Esraa Samkari, Muhammad Arif, Manal Alghamdi, Mohammed A. Ghamdi (13-November-2023). **Human Pose Estimation Using Deep Learning: A Systematic Literature Review.**  
   [https://www.mdpi.com/2504-4990/5/4/81](https://www.mdpi.com/2504-4990/5/4/81" \t "_new)
4. Ayush Kumar, Sunil Maggu, Bhaskar Kapoor, Ajay Kumar Kaushik, Seema Kalonia (16-October-2022). **Human Posture Detection and Correction Using MediaPipe and OpenCV.**

<https://link.springer.com/chapter/10.1007/978-981-97-5862-3_11>

1. Bengtsson, V., et al. (17-07-2018). *Narrative review of injuries in powerlifting with special reference to their association to the squat, bench press and deadlift*. BMJ Open Sport & Exercise Medicine. [https://pmc.ncbi.nlm.nih.gov/articles/PMC6059276/](https://pmc.ncbi.nlm.nih.gov/articles/PMC6059276/" \t "_new)
2. Regueira Lorenzo, F. (24-07-2020). *Claves en lesiones de los powerlifters*. RV Strength.

<https://rvstrength.com/claves-lesiones-powerlifters/>

1. Mitchell Holistic Health. (09-09-2024). *Why Does My Back Hurt When I Squat and Deadlift?*

<https://mitchellholistichealth.com/back-pain-squats-deadlifts-causes-solutions/>