

# Clasificación de movimientos de CrossFit: una aplicación con MoViNets

Agustín Piqueres

13/09/2022



## 1.1 Motivación

Meter explicación de CrossFit (AQUÍ Y EN LA MEMORIA) y alguna imagen.

- Es un deporte con un gran número de participantes, y el número de competiciones sigue creciendo.
- Automatizando del proceso de corrección de videos:
  - Las personas pueden dejar una tarea repetitiva (se ahorra en personal).
  - Se limitaría la inconsistencia entre distintos jueces en las correcciones.

## 1.2 Objetivos

Antes de llegar a ese punto, vamos a intentar crear una aplicación que sea capaz de clasificar movimientos de CrossFit, para lo cuál será necesario:

- Crear un *Dataset* con movimientos de los distintos ejercicios.
- Un modelo capaz de identificar correctamente los distintos movimientos.
- Una aplicación en la que un usuario sea capaz de subir un video y obtenga el movimiento del que se trata.

## 1.3 Estructura del proyecto

- **2 Estado del arte**
  - 2.1 Deep Learning
  - 2.2 Cloud
  - 2.3 Trabajos relacionados
- **3 Desarrollo**
  - 3.1 Extracción y recolección de datos
    - 3.1.1 Proceso de extracción
    - 3.1.2 Datos obtenidos
  - 3.2 Experimentación con Deep Learning
    - 3.2.1 Preprocesado de los datos
    - 3.2.2 Experimentos realizados y resultados
    - 3.2.3 Evaluación de resultados
  - 3.3 Cloud y despliegue de la aplicación
    - 3.3.1 Arquitectura cloud
    - 3.3.2 Resultado y funcionamiento
- **4 Conclusiones**

## 2.1 Deep Learning

El campo del *Reconocimiento de Acciones* (la tarea de identificar personas realizando acciones en imágenes o videos) ha crecido en los años. Las acciones humanas pueden reconocerse con diferentes metodologías (como *Optical Flow* o representaciones del esqueleto), pero este trabajo se centra en la *Clasificación de Video*:

La tarea de producir una etiqueta relevante para un video dados sus *frames*.

### Listado de Papers:

- Primeros datasets: [HMDB51](#) o [UCF101](#).
- Primer dataset para entrenar modelos de deep learning sobre clasificación de acciones en humanos: [Kinetics 400](#) (Modelo I3D).
- Uno de los modelos con resultados más prometedores es [MoViNets](#) (actualmente ha habido grandes avances), una familia de modelos eficientes en el uso de memoria y computación, que permite operar con videos en streaming.

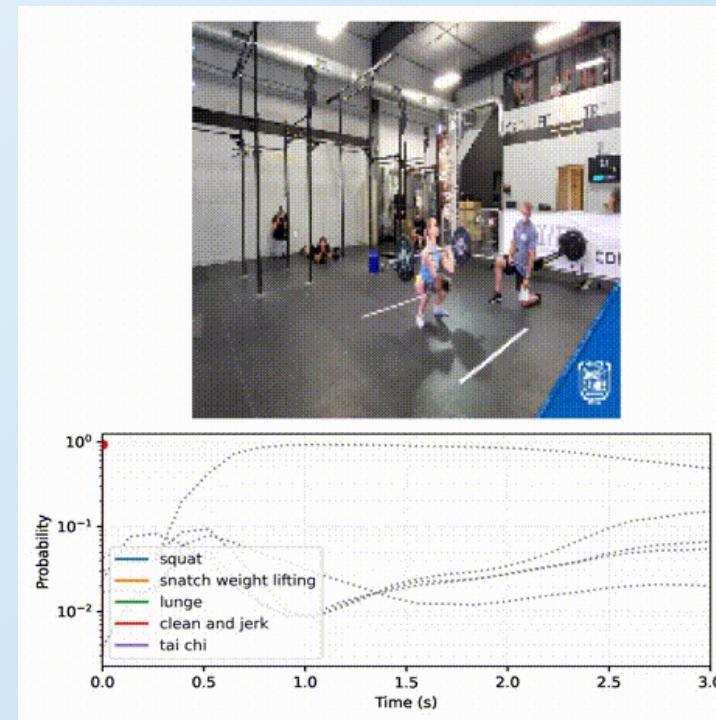
## 2.1 Deep Learning

### MoViNets

Esta familia de modelos se divide en 2 tipos diferentes de modelos: **base** y **stream**, según si procesan todo el video de golpe, o permiten procesar el contenido frame a frame, y en 5 arquitecturas distintas (desde *a0* hasta *a5*).

En este trabajo nos centramos en *MoViNet a2 base*, que es de entre los modelos más pequeños, el que tiene mejor capacidad predictiva y aún es capaz de ser utilizado en tiempo real (20 fps o más ([ref](#)))

- No nos podemos centrar en los modelos stream, hay errores al hacer fine-tuning con ellos: [issue 10730](#) o [issue 10463](#).
- Los autores obtienen en tan solo 3 epochs un buen accuracy en UCF101.



Ejemplo predicción en MoViNet Stream a2.

## 2.2 Cloud

Todo el despliegue se ha realizado utilizando los distintos servicios de AWS, sin recurrir a servicios como **AWS SageMaker**, **Azure ML** o **Google DataLab**, que ofrecen una solución completa al despliegue de modelos basados en deep learning.

## 2.3 Trabajos relacionados

Un par de trabajos relacionados han tratado la clasificación de acciones:

- Chen et al., 2022 hacen uso de Yolo4 para detectar y clasificar movimientos de fitness.
- En un artículo de [towardsdatascience](#) el autor hace uso de Optical Flow para contar repeticiones de unos pocos movimientos.

En otro [artículo de medium](#) el autor hace fine tuning sobre uno de los modelos de MoViNets stream al parecer, pero no se puede ver el código ni hay forma de encontrar al autor.

## 3.1 Extracción y Recolección de datos

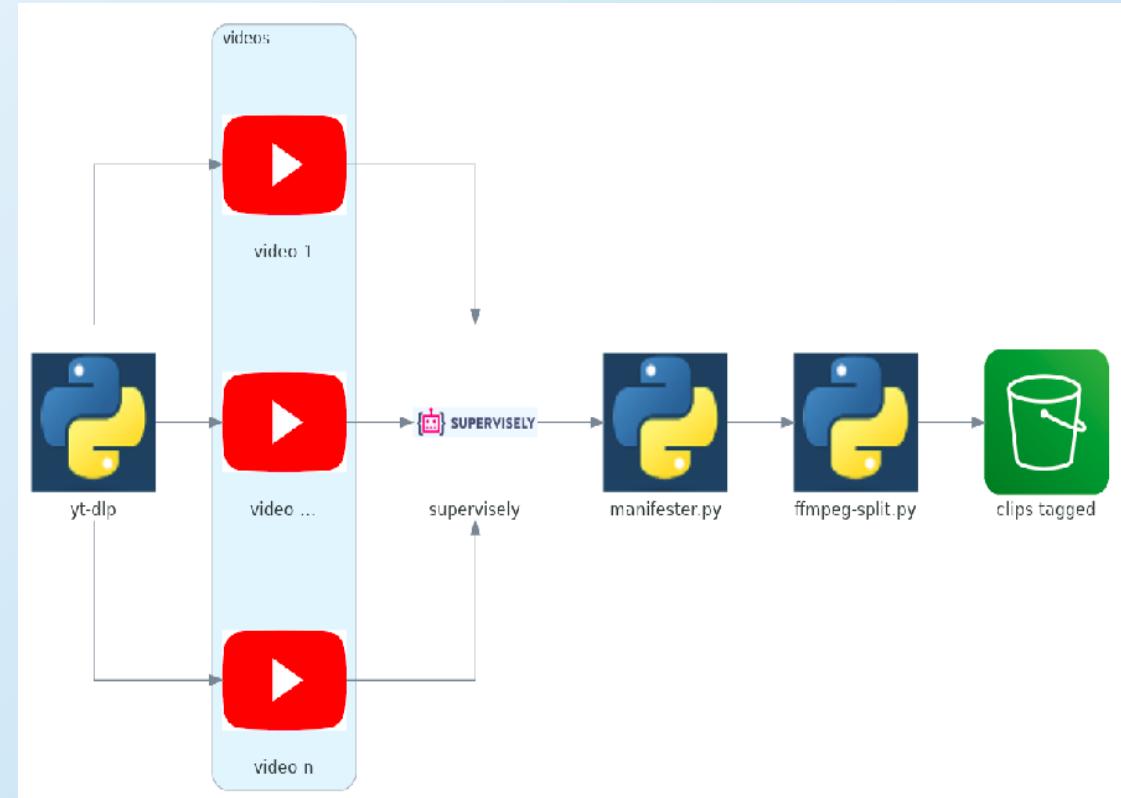
### 3.1.1 Introducción

- Creación de un dataset de movimientos de CrossFit.
  - Descarga de los videos de YouTube.
  - Etiquetado.
- Análisis del *dataset*.

## 3.1 Extracción y Recolección de datos

### 3.1.2 Proceso de extracción

- `download.py` : Descarga de los videos de YouTube utilizando `yt-dlp`.
- *Supervisely*: Software Open Source para *Computer Vision*, utilizado para el etiquetado de los vídeos.
- `manifester.py` : Script para la transformación del output de *Supervisely*, a un formato apto para el siguiente script: `manifest.json`.
- `ffmpeg-split.py` : Script para recortar los videos originales en mp4, para obtener los clips correspondientes a los frames etiquetados.



## 3.1 Extracción y Recolección de datos

### 3.1.3 Datos obtenidos

- 2700 videos, 300 ejemplos de cada movimiento en formato mp4, ~58 minutos de videos.

UCF101: 101 actividades, 131 muestras en media, alrededor de 13.000 clips, de unos 7 segundos de duración.

**Tabla 3.1:** Estadísticos descriptivos de los frames y duración de los clips

	Frames		Duración (s)	
	Media	Desviación típica	Media	Desviación típica
<b>thruster</b>	27.7396	4.5060	0.9419	0.1274
<b>chest-to-bar</b>	21.0478	7.4233	0.7196	0.2569
<b>double-unders</b>	14.5880	1.2740	0.4841	0.0439
<b>ghd</b>	58.9732	5.5611	1.9654	0.1860
<b>power clean</b>	67.3567	13.0092	2.2448	0.4337
<b>deadlift</b>	24.1229	7.1950	0.8048	0.2395
<b>shspu</b>	43.2100	15.1247	1.4412	0.5055
<b>ohs</b>	31.9767	5.4303	1.0664	0.1799
<b>bar-facing burpee</b>	62.0233	10.0239	2.0688	0.3340
<b>TOTAL</b>	39.0042	7.7275	1.3041	0.2563

## Ejemplo de los movimientos

thruster



chest-to-bar



double-unders



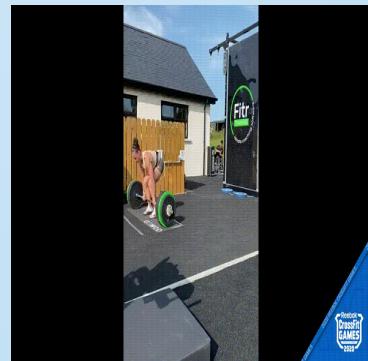
ghd



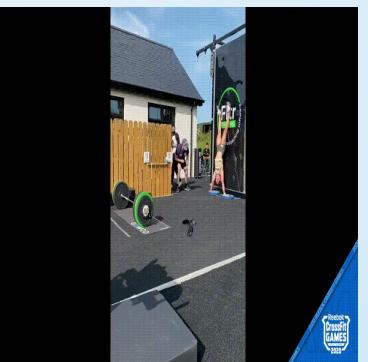
power clean



deadlift



shspu



ohs



bar-facing burpee



## 3.2 Experimentación con Deep Learning

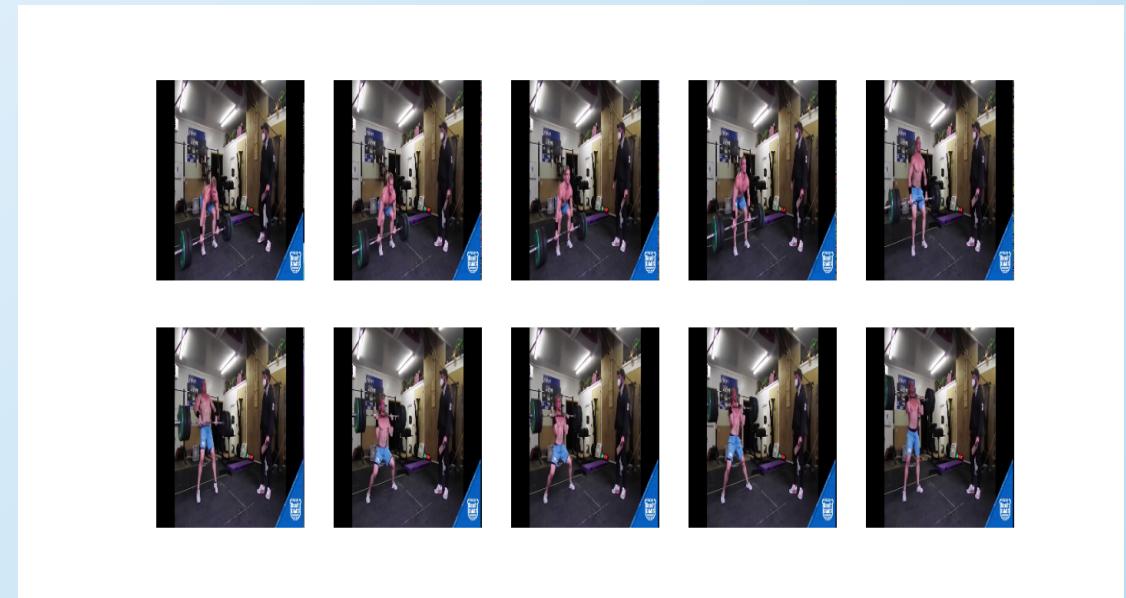
### 3.2.1 Introducción

- Selección del modelo
- Preprocesamiento de los datos
- Entrenamiento
- Resultados

## 3.2 Experimentación con Deep Learning

### 3.2.2 Preprocesado de los datos

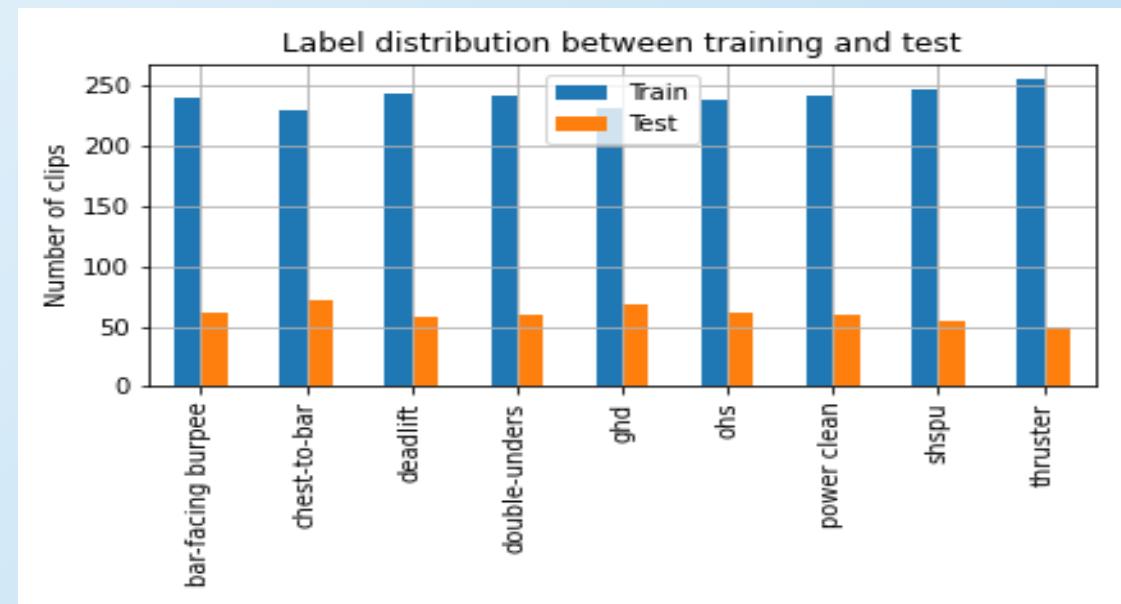
- Librería para el preprocesamiento y fine-tuning:  
[movinets\\_helper](#)
  - How To Guide
- Transformamos los datos originales a un nuevo [TFRecordDataset](#).
- Se reescalan los videos a una resolución de  $224 * 224p$  (arquitectura a2 base), reescalamiento RGB  $[0, 1]$ .
- Muestra de  $10\ frames$  equiespaciados de cada video.
- El dataset pasa de  $\sim 400\text{Mb} \rightarrow \sim 10\text{Gb(gzip)}$ .



## 3.2 Experimentación con Deep Learning

### 3.2.3 Experimentos realizados y resultados

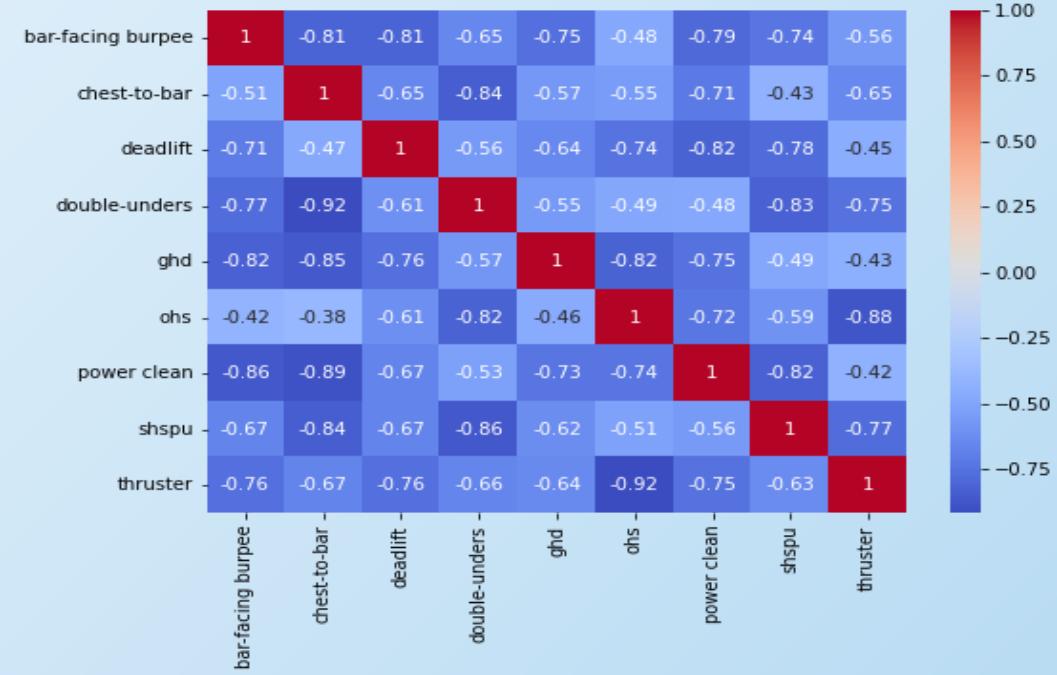
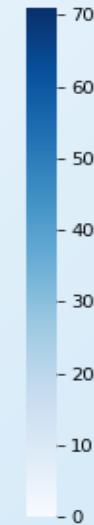
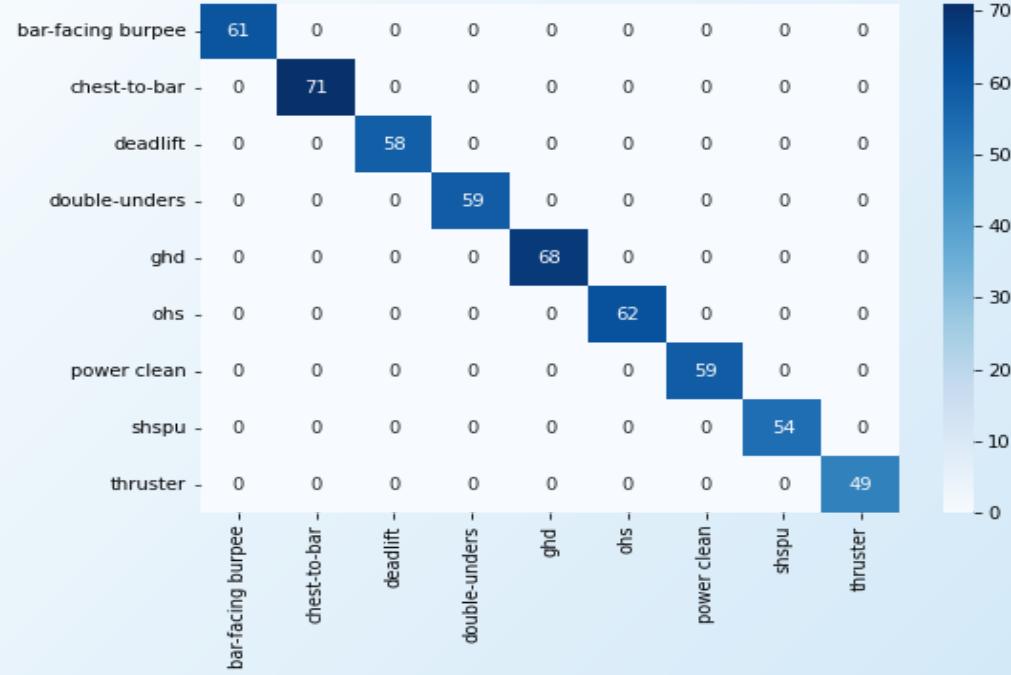
- Entrenamiento en Google Colab con GPU.
- Mismos hiperparámetros que en el paper (tamaño batch 8).
- 80% training (2164 clips), 20% test (541).



- Resultados del entrenamiento: [Tensorboard.dev](#)

## 3.2 Experimentación con Deep Learning

### 3.2.4 Evaluación de los resultados

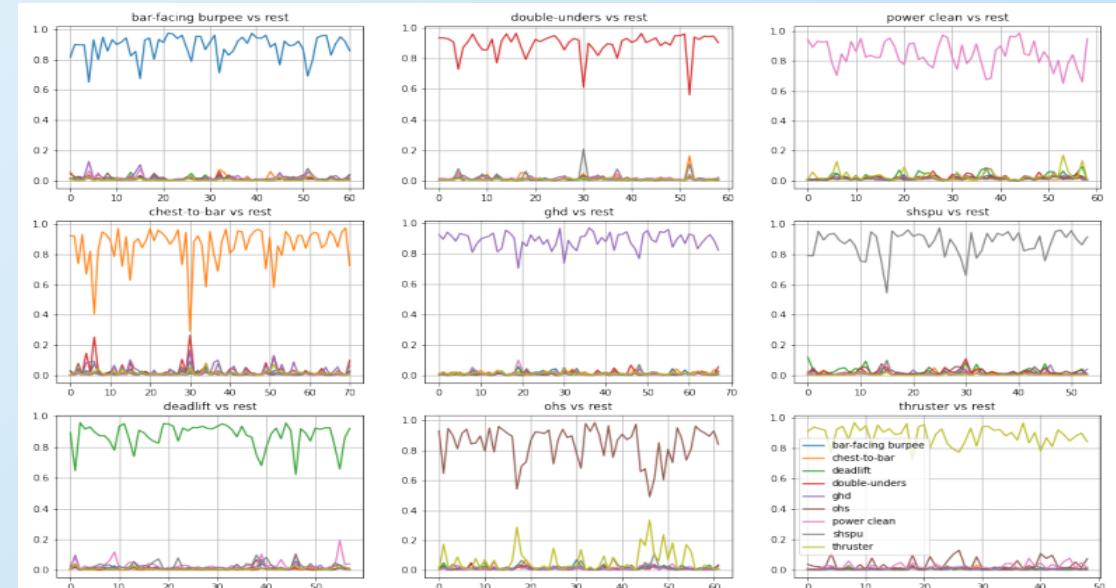


## 3.2 Experimentación con Deep Learning

### 3.2.4 Evaluación de los resultados

**Tabla 3.2:** Estadísticos principales de las probabilidades predichas para cada movimiento observado

	Media	Desviación típica
double-unders	0.8921	0.0748
ghd	0.8904	0.0533
bar-facing burpee	0.8876	0.0756
thruster	0.8838	0.0590
deadlift	0.8746	0.0796
shspu	0.8731	0.0843
chest-to-bar	0.8552	0.1282
power clean	0.8443	0.0851
ohs	0.8438	0.1163



**Figura A.1:** Probabilidades estimadas en la muestra de test. Para cada movimiento observado, cada gráfica representa las probabilidades obtenidas por el modelo.

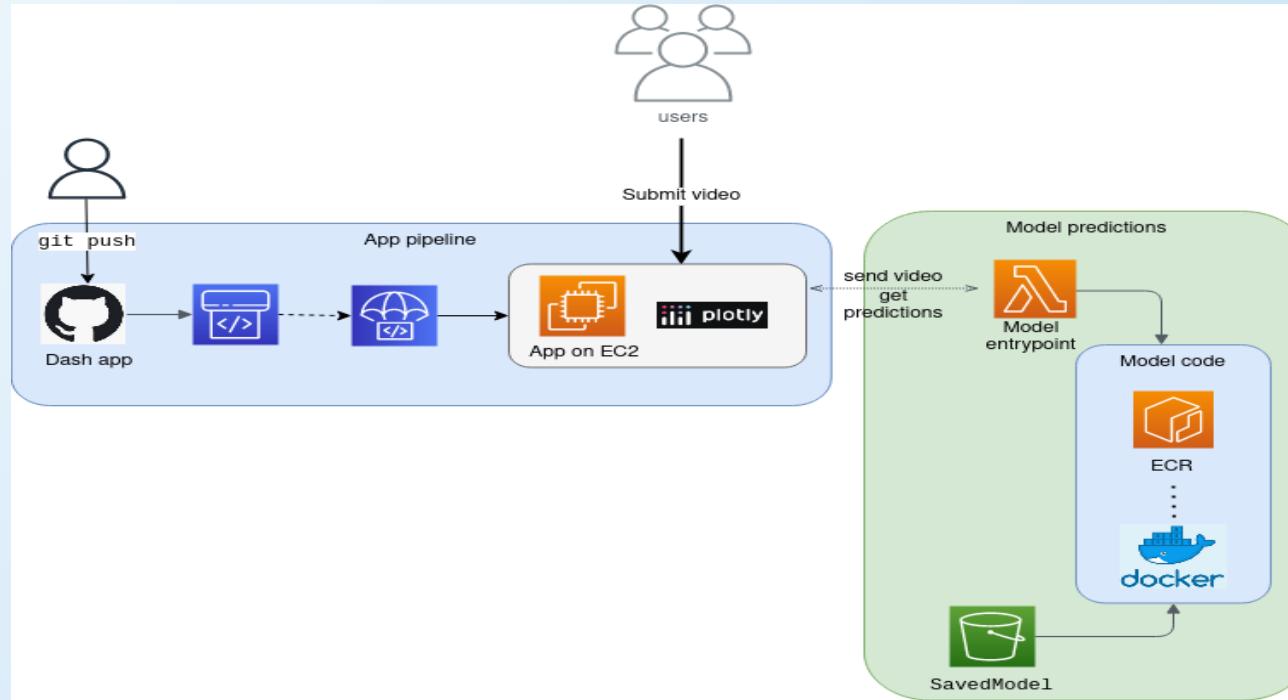
## 3.3 Cloud y despliegue de la aplicación

### 3.3.1 Introducción

- Arquitectura de la aplicación.
- Funcionamiento

## 3.3 Cloud y despliegue de la aplicación

### 3.3.2 Arquitectura cloud



- `movinets_dash_app`
- `lambda_aws`.

## 3.3 Cloud y despliegue de la aplicación

### 3.3.3 Resultado y funcionamiento

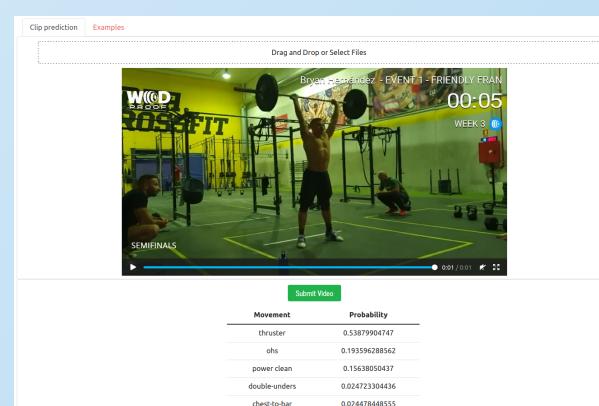
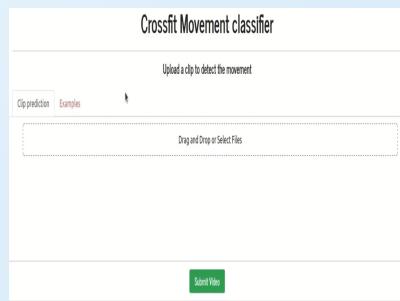
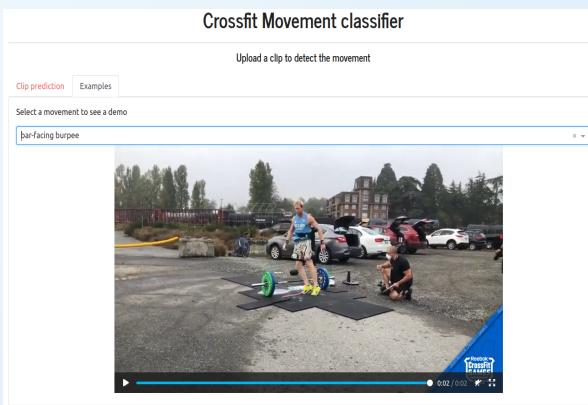
app -> movinet-crossfit-cidaen

Mejoras:

- *post-training quantization.*
- Muestrear los videos como en la pipeline de entrenamiento.
- Utilizar un modelo más sencillo

## 3.3 Cloud y despliegue de la aplicación

### 3.3.3 Resultado y funcionamiento



## 4. Conclusiones

- Aplicación de *MoViNets* para la clasificación de movimientos de CrossFit.
- Creación de un nuevo dataset de clips de ejercicios de CrossFit etiquetados.
- Desarrollo de una aplicación en AWS para clasificar clips.

Siguientes pasos:

- Entrenamiento de arquitecturas *stream*.
- Metodologías alternativas como *Optical Flow*.
- Desplegar el modelo por medio de *Tensorflow Lite*.