# Human Action Recognition

Jose Antonio Picazos Carrillo

**Resumen**— Este proyecto se centra en el proceso de reconocimiento y clasificación de las acciones que realiza una persona en un momento concreto, también conocido como *Human Activity Recognition*. Para realizar el reconocimiento de movimiento y pose se ha hecho uso de un algoritmo de *pose estimation*, el cual proporciona el esqueleto de la persona y es de ayuda para clasificar la acción que se está realizando. Para entrenar el modelo encargado de realizar la clasificación se ha hecho uso de una red neuronal recurrente como es LSTM y también se ha hecho uso de AutoML. Estos métodos se han puesto a prueba con datos reales para comprobar su validez.

**Palabras clave**— Visión por computador, Human acitivity recognition, clasificador, AutoML, pose detection.

**Abstract**— This project focuses on the process of recognition and classification of the actions being performed by a person, also known as Human Activity Recognition. To perform the recognition, we have made use of a pose estimation algorithm, which provides the skeleton of the person and helps to classify the action being performed. To train the model in charge of the classification, a recurrent neural network such as LSTM has been used and AutoML has also been used. All these methods have been tested with real data to check their validity.

**Index Terms**— *Computer vision, human activity recognition, classifier, AutoML, pose detection*.

## 1 Introducción

El deporte tradicional ha ido evolucionando en los últimos años en lo que respecta a la tecnología utilizada para transmitir información y datos: herramientas para el *fair play*, captación de las mejores jugadas, ayuda para las decisiones de los árbitros, herramientas de *coaching* y mucho más. Muchas de ellas se han creado con la ayuda de sistemas de *Machine Learning* y de visión por computador que, gracias a los avances de los últimos años, se han convertido en herramientas tecnológicas muy potentes.

Un sistema que los fanáticos del fútbol seguramente conozcan es la asistencia al árbitro por video, más conocido como VAR [1], un sistema implementado por la FIFA. El VAR consiste en una serie de cámaras situadas en el campo con la finalidad de dar información para evitar los errores de arbitraje y, así, conseguir un juego más limpio.

También se conocen algunos proyectos de deportistas muy conocidos como el de Xavi Hernández, con el proyecto de Kognia[2], o también el de Andrés Iniesta con First V1sion[3]. El proyecto de First V1sion consiste en implementar una cámara en las camisetas de los jugadores para poder ver, desde el punto de vista del jugador, sus acciones durante el partido.


- *E-mail de contacte: joseantonio.picazos@e-campus.uab.cat*
- *Menció realitzada: Computació*
- *Treball tutoritzat per: Coen Antens (CVC)*
- *Curs 2020/21*


Esto es realmente útil para aumentar la calidad de las retransmisiones, ya que se añade como una nueva experiencia para el espectador. El proyecto de Kognia se basa en un sistema automático para analizar las tácticas de los equipos en los partidos de fútbol en tiempo real, o bien, después del partido. Para ello, Kognia recolecta datos de los partidos y las sesiones de entrenamiento y, con ellos, mediante la inteligencia artificial, se entrena un modelo de datos. Los datos obtenidos son revisados y complementados por los conocimientos de los entrenadores.

En el ámbito del *coaching* no existe variedad de herramientas de este tipo, una de las pocas conocidas se implementa en el mundo del básquet, el HomeCourt[4], una aplicación que cuenta con el soporte de la NBA. Esta es una herramienta muy interactiva que ayuda a mejorar las habilidades relacionadas con el básquet, ya que realiza la función de entrenador personal desde un dispositivo móvil. Realiza una detección de la persona y, también, una detección de la pose del jugador para ver su posición a la hora de lanzar el balón o de realizar un ejercicio de habilidad. La información la obtiene a raíz de capturas de imágenes de los ejercicios propuestos y el posterior análisis de estas, a la vez, ofrece unos consejos guiados para mejorar el rendimiento del atleta.

Por lo tanto, teniendo en cuenta lo comentado en los párrafos anteriores, se puede llegar a la conclusión que, en el mundo del deporte, sea cual sea la disciplina, la posición del

cuerpo del atleta a la hora de realizar el movimiento es muy importante, ya que, en función de esta, se realizará mejor o peor su movimiento o acción final y, por lo tanto, sus resultados. Por ejemplo, en la gimnasia rítmica, en la que la posición del cuerpo de los gimnastas es muy importante para la ejecución de la rutina y su puntuación final.

El atletismo es otro deporte en el cual la pose del deportista es muy importante. Este deporte recoge diferentes disciplinas: velocidad, salto, lanzamiento, carrera de resistencia y marcha. En todas ellas se necesita una técnica de movimiento corporal específica para que el atleta ejecute bien el movimiento haciendo que su rendimiento y sus marcas sean buenas. Por el contrario, una mala posición de este hará que la ejecución del movimiento sea incorrecta y, consecuentemente, que sus marcas no sean las deseadas, que su rendimiento empeore o que, incluso, pueda llegar a sufrir alguna lesión. Para mejorar la técnica de movimiento interviene e influye el papel del entrenador, ya que es el que ve la acción desde otra perspectiva, es decir, desde fuera. Desde su punto de vista verá como el atleta realiza, por ejemplo, el salto, e indicará las correcciones necesarias para que no cometa los mismos errores en los siguientes saltos.

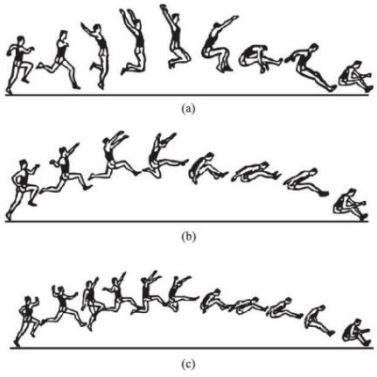


Figura 1: *Fases salto longitud*

Gracias al análisis de la pose, como se ha descrito anteriormente, se puede analizar la acción realizada por una persona, pero, como se describe a continuación, también se puede llegar a predecir qué movimiento o acción está ejecutando alguien, a esto último se le denomina *Human Activity Recognition* (HAR).

El reconocimiento de actividad que examina HAR teniendo en cuenta la pose, permite deducir la acción que está realizando una persona en un momento exacto. La pose concreta de cada acción se obtiene y clasifica recogiendo datos de distintas personas, aunque cada persona es diferente, existen similitudes en los gestos.

El reconocimiento de actividad y de pose es utilizado en muchas aplicaciones informáticas y tecnológicas (móvil, Tablet, ordenador…) que se basan en entrenadores personales virtuales. Estas aplicaciones que funcionan como entrenadores virtuales, graban el ejercicio que está realizando la persona y, gracias a la detección de la pose del cuerpo y un clasificador de pose, deducen si se la acción se ejecuta de forma correcta y cuenta el número de repeticiones hechas en caso necesario. Un ejemplo de aplicación es FitnessAlly[20], de la empresa *Twenty Billion Neurons*. Esta realiza la función de asistente para los entrenamientos de fitness que se realizan en casa de forma autónoma, ofreciendo, a su vez, una gran posibilidad de entrenamientos y posibilidades.

## 2 OBJETIVOS

Observando todo el interés que hay relacionado con nuevas iniciativas tecnológicas en el mundo del deporte relacionadas con *pose detection* y *Machine Learning*, son muchos los deportes en los que, todavía, no se ha intentado implementar esta tecnología y, por eso, el objetivo de este proyecto es analizar la pose y del movimiento del atleta en atletismo.

Para ello me centraré en la gran importancia del entrenador y la corrección de la pose del atleta, la finalidad teórica de este proyecto es, mediante un detector de pose, analizar la posición del cuerpo del atleta durante su movimiento y poder mejorar su acción.

Con este trabajo se puede crear una herramienta de soporte que permita crear una base de prototipo para ejecutar todo el entorno gracias al software *StreamLit* [5]. Esta herramienta dirigida a los entrenadores les permitiría obtener posibles mejoras en la pose del atleta en base a la posición analizada.

Lo desarrollado en este proyecto sirve para poder crear un prototipo de aplicación móvil en la que se pueda implementar el *pose detection* y AutoML [6], ya que es un dispositivo que llevamos todos en nuestro bolsillo y facilita mucho su uso al no depender de un ordenador.

Las tecnologías y los algoritmos de *Machine learning* y *Deep Learning* han ido evolucionando en el paso del tiempo dando a conocer nuevos métodos para la optimización del proceso de aprendizaje y para poder obtener modelos entrenados de una forma más eficiente y rápida. Una de las últimas tecnologías conocidas es el método de autoaprendizaje llamado AutoML. Esta nueva tecnología permite entrenar un modelo de datos y obtener el mejor modelo gracias a la optimización automática de los hiperparametros. Por ello uno de los objetivos de este proyecto es poner a prueba este novedoso algoritmo de autoaprendizaje. Se comprobará si realmente este método proporciona la mejor opción para el modelo de datos. Este método se utilizará para entrenar el modelo de datos y, así, poder ahorrar una gran cantidad de tiempo en generar el modelo de datos, ya que esta técnica, al ser automática, proporciona un modelo optimizado. El modelo obtenido se compara con otros métodos de aprendizaje, y otros manuales a la hora de optimizar para verificar si, realmente, sería una mejora a la hora de entrenar modelos de datos.

## 3 METODOLOGÍA

Para llevar a cabo el proyecto y gestionar de forma efi-

ciente todo el trabajo se ha aplicado la metodología *Agile*, esta permite llevar un control de todas las tareas y saber en el punto que se encontraba el proyecto en cada momento.

La metodología *Agile* recoge varias metodologías, pero la utilizada para desarrollar este proyecto ha sido la metodología *Scrum*. Esta trabaja con periodos de tiempo fijos que recogen las diferentes tareas que se deben realizar en cada uno de ellos, también conocidos como *sprints*. Los *sprints* tendrán una duración de entre una y dos semanas dependiendo de la fase del proyecto. Al finalizar cada *sprint* se espera obtener un avance significativo del proyecto aportándole valor de forma progresiva.

Siguiendo la metodología descrita en los párrafos anteriores, se han realizado reuniones semanales con el tutor que ha seguido el proyecto. En ellas se ha observado y valorado el avance y se han ido planificando los *sprints* posteriores. También se han ido definiendo qué tareas eran prioritarias y cuáles no para que el avance del proyecto fuese el correcto en cada momento. Una vez definida la metodología aplicada cabe mencionar, también, la herramienta utilizada, en este caso ha sido *ClickUp*[7], ya que proporciona el entorno necesario para crear el proyecto y las tareas a realizar durante cada *sprint*. Dichas reuniones se han desarrollado a través de la aplicación de *Microsoft Teams*, a su vez, esta también ha servido como *SandBox*, ya que toda la documentación que se ha aportado durante el proyecto queda almacenada en el equipo de *Teams*.

## 4 Planificación

El trabajo realizado para desarrollar el proyecto se ha dividido en distintas fases, cada una de ellas corresponde a un objetivo (descritos en el punto 2).

Fase 1: análisis de AutoML.

En esta primera fase, se ha realizado un análisis de dos de los algoritmos de AutoML y una comparativa con métodos más clásicos como RandomForest, KNN, DecisionTree, etc. Al final de esta fase se ha definido cuál de los dos algoritmos se ha utilizado durante el proyecto para la clasificación de poses.

Fase 2: clasificación de poses.

En esta segunda fase se han aplicado diferentes técnicas de *Human Activity Recognition* para poder clasificar el movimiento que está realizando una persona en un momento concreto como puede ser caminar, saltar, sentarse en la silla, etc.

Dentro de *Human Activity Recognition* se han seleccionado dos técnicas, una básica de reconocimiento de actividad mediante un clasificador de imágenes, y otra que utiliza un algoritmo de *pose estimation*. Esta última proyecta los puntos del esqueleto, esto es de utilidad para la fase en la que clasifican las poses.

Fase 3: reconocimiento de actividades.

En esta tercera fase se ha creado un modelo de datos para la clasificación de las distintas poses.

Para la creación del modelo de datos se han realizado distintos entrenamientos de diferentes modelos a partir de un mismo algoritmo, AutoML (que ha sido seleccionado en la fase de análisis).

A parte de AutoML, también se ha hecho uso de un segundo clasificador de poses que permite realizar el análisis del movimiento partiendo de una secuencia de imágenes. Posiblemente uno de los algoritmos adecuados para este segundo clasificador sea una red convolucional.

## 5 Desarrollo

A continuación, se describen los diferentes métodos aplicados en cada una de las fases del proyecto.

### 5.1 Análisis del algoritmo AutoML

El algoritmo AutoML es un proceso que realiza un aprendizaje automático e inteligente de forma rápida sobre los datos que se le envían para poder resolver problemas sin la necesidad de la intervención de un especialista, es decir, facilita información al usuario sin que este tenga nociones de *machine learning*. El algoritmo AutoML recoge el conjunto de datos que el usuario le aporta y proporciona un modelo de datos de forma automática para poder ser analizados, por ejemplo, para recoger datos ciudadanos que puedan ser cargados por un especialista, para analizar a clientes, analizar datos de marketing, etc.

Para comprobar cómo funcionan y familiarizarse con los algoritmos de AutoML se ha utilizado un dataset que resuelve tareas de clasificación. Para implementarlo se ha desarrollado un primer caso básico utilizando una base de datos pública, la *Breast Cancer Wisconsin (Diagnostic)* [14]. Este *dataset* dispone de un total de 569 filas y 32 atributos, entre los atributos se encuentra el de diagnóstico, que es el objetivo de una posible clasificación y que puede tener como resultado dos posibles valores M (*Malignant*) o B (*Benign*).

Entre todos los algoritmos de AutoML existentes, para el desarrollo del proyecto, se han escogido dos: *Auto-Sklearn*[13] y *AutoKeras*[15]. El algoritmo *Auto-Sklearn* se basa en funciones de *Scikit-learn*, que corresponde a la famosa librería de *Machine Learning*. El algoritmo AutoKeras se basa en funciones de Keras, de la misma compañía. Tanto Keras como AutoKeras hacen uso de los algoritmos de *Deep Learning* y de la librería *Tensorflow* de Google que ayuda a realizar el entrenamiento del modelo.

A parte de los dos algoritmos de AutoML mencionados, también se han implementado otros algoritmos de clasificación clásicos: *RandomForest*, *KNN*, *DecisionTree*, *Multi Layer Perceptron* y *Gaussian Process*. Se han utilizado para realizar una comparativa entre estos y los dos de AutoML y observar si realmente los de AutoML obtienen y ofrecen mejores resultados.

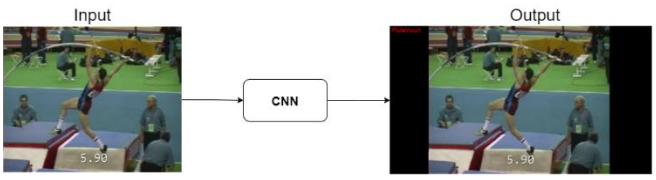


Figura 2: *Diagrama Single-Frame CNN*

## 5.2 *Human Activity Recognition*

*Human Activity Recognition* es un campo de investigación que recoge diferentes técnicas para poder clasificar las distintas acciones y movimientos que está realizando una persona en un momento concreto, por ejemplo, al levantar un brazo.

Para poder realizar el reconocimiento de una acción o movimiento, se debe seguir un largo proceso, por eso, para el desarrollo del proyecto, se ha desglosado en diferentes fases. Estas fases son:

- Fase de clasificación de poses: en esta fase se realiza la detección de la persona y se aplica un algoritmo de pose *estimation* para la detección y extracción de las características de la pose. En base a estas características extraídas, posteriormente, se realiza la clasificación de acción.
- Fase de reconocimiento de actividades: en esta fase se catalogan los puntos de la pose para obtener la clasificación de acción resultante mediante un modelo de clasificación.

### *5.2.1 Fase clasificación de poses*

Esta primera fase del proceso del algoritmo *Human Activity Recognition* consiste en realizar el reconocimiento de la persona, para ello se han aplicado distintos métodos: *Single-Frame CNN*, red convolucional y *Pose Detection*. En la aplicación de los dos primeros métodos no se han tenido en cuenta las características pose de la persona, solo se ha tenido en cuenta el *frame input*. A continuación, se explican más detalles de cada uno de los tres métodos mencionados.

- *Single-Frame CNN*

El primer método aplicado para el desarrollo del proyecto es una forma sencilla de poder clasificar las acciones, se trata de una red convolucional de clasificación de imágenes. Este tipo de redes son muy efectivas para la clasificación y segmentación de imágenes. Hay que tener en cuenta que este método determina la acción que se observa en la imagen utilizando únicamente un *frame*, no una secuencia de *frames*.

Para entrenar el modelo de clasificación de imágenes de este método se han utilizado seis clases del *dataset UCF-101*: *HighJump, PoleVault, JavelinThrow, LongJump, ThrowDiscus* y *Biking*. De estas clases se han extraído los *frames* de distintos vídeos para realizar dicho entrenamiento. La configuración para el entrenamiento del modelo de clasificación de imágenes consta de dos capas convolucionales.

Una vez entrenado el modelo este se ha aplicado, en la figura 2 el *input* del método es el *frame*, a continuación, la imagen es procesada por CNN, que clasificará la imagen y proporcionara un *output* con la imagen clasificada, es decir con el valor de la acción o movimiento.

Para llevar a cabo el método Single-Frame, se han extraído imagenes de un vídeo, es decir, se han separado sus *frames* y se ha realizado la clasificación *frame* a *frame*. Esto sirve para que el modelo determine la acción que se observa en cada una de los *frames*. Esto solo funciona si la acción no depende de los *frames* anteriores.

Por ejemplo, en la detección de gestos para controlar un *Automated Guided Vehicle* (AGV), solo es necesaria una imagen para que el AGV responda al gesto, por ejemplo, una mano derecha alzada indicará al AGV que debe girar hacia a la derecha de forma autónoma. El proyecto T-RREX[21] se basa en el desarrollo de un robot móvil (AGV) que dispone de un brazo robótico y que, gracias a la movilidad del robot, este puede interactuar con distintas máquinas en la factoría. Este AGV debe convivir con el personal de la factoría, y por ello hace uso de un sistema de reconocimientos de gestos haciendo que se mueva detectando y siguiendo los gestos físicos que hace una persona.

- *ResNet*

En el segundo método aplicado para el desarrollo del proyecto se hace uso de una red convolucional ya entrenada, es decir, al contrario del método anterior, se hace uso de un modelo ya creado, *ResNet*. Esta red convolucional ha sido entrenada con un *dataset* público llamado *DeepMind Kinetics human action video dataset*[19]. Este *dataset* contiene 400 clases de acciones de las cuales se han obtenido 400 clips de vídeo de cada una de estas clases para poder realizar el entrenamiento de la red.

Para aplicar este método no se ha utilizado la clasificación *frame* a *frame* como en el caso anterior, en este caso se ha determinado la acción observada en la imagen mediante una secuencia de *frames*.

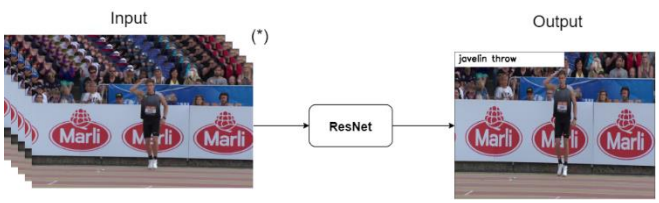


*Figura 3: Diagrama ResNet*

Esto conduce a darse cuenta de que con un único *frame* es complicado determinar la acción que se está realizando en la imagen, ya que no tiene en cuenta el contexto de la acción ni que la acción es realizada en una secuencia de pasos.

- *Pose Detection (OpenPose)*

En los métodos anteriores se ha pretendido determinar la acción observada en una imagen sin tener en cuenta la pose de la persona.

Para reforzar la clasificación de *frames* que determine la acción se ha utilizado un algoritmo de *pose estimation*. Se han investigado distintas opciones de algoritmos de *pose estimation*: DensePose, PoseNet, AlphaPose… pero para el desarrollo del proyecto se ha escogido utilizar el algoritmo *OpenPose*. *OpenPose* detecta la pose en tiempo real permitiendo extraer los puntos (*keypoints*) del cuerpo detectado.

*OpenPose* permite configurar dos modelos de detección diferentes, COCO y BODY_25. El modelo de detección COCO detecta un total de 18 puntos en el cuerpo observado, mientras que BODY_25 detecta un total de 25 puntos. Por ello, para las pruebas realizadas para el desarrollo del proyecto se ha decidido utilizar BODY_25, ya que proporciona una mayor cantidad de puntos sobre el esqueleto y será más eficaz a la hora de reconocer la acción realizada.

*Figura 5: Ejemplo OpenPose Body_25*

Inicialmente se ha decidido realizar un caso simple utilizando un *dataset* propio y creado a partir de 14 vídeos, también, propios, en los cuales únicamente aparece una persona realizando dos acciones, saltar y caminar. Luego se ha hecho un procesado previo de ellos en el cual se ha extraído *frame* a *frame* la imagen. Posteriormente, mediante *OpenPose* se han obtenido los *keypoint* de cada uno de los *frames* procesados.

Tras desarrollar el caso descrito en el párrafo anterior con un *dataset* propio, se ha decidido realizar nuevamente el mismo proceso, pero en este caso utilizando el *dataset* público UCF-101. De las 101 clases de este *dataset* se ha utilizado únicamente una, la acción de salto de longitud (*Long-Jump*).

### 5.2.3 Fase reconocimiento de actividades

En esta segunda fase se hace uso del tercer método, *Pose Detection*, descrito en el apartado anterior. Se utiliza este método ya que refleja el esqueleto de la persona y los *keypoints* permitiendo realizar una clasificación de la pose del cuerpo.

Para poder reconocer la acción que se observa en la imagen se han aplicado dos algoritmos diferentes, un algoritmo de autoaprendizaje (AutoML) y el algoritmo de la red convolucional *Long short-term memory* (LSTM).

Por lo tanto, el método Pose Detection permite obtener las características de las imágenes, en este caso de las poses, y los algoritmos mencionados permitirán crear un modelo de datos que permita reconocer la acción que se está realizando en la imagen.

Para realizar el entrenamiento de AutoML y de LSTM se ha hecho uso del *dataset* público UCF-101. De este *dataset* se han escogido 8 clases (*archery, basketball, bowling, golfswing, highjump, longjump, playingguitar* y *shotput*) y de cada una de ellas se han seleccionado 12 vídeos.

- AutoML

Tras realizar el análisis del proceso de autoaprendizaje AutoML se ha decidió aplicarlo para obtener un modelo de reconocimiento de las actividades en base a los *keypoints*.

Inicialmente se ha desarrollado un primer caso haciendo uso de la base de datos T-RREX que se utiliza para controlar por gestos un AGV. En este *dataset* hay acciones simples que pueden ser reconocidas en un único *frame*.

Tras desarrollar el caso mediante la base de datos T-RREX, se ha optado por realizar el mismo proceso, pero esta vez utilizando el *dataset* UCF-101 que contiene acciones más complejas, ya que las acciones más complejas como saltar, andar… no pueden ser reconocidas por un único *frame* sino que necesitan ser reconocidas por una secuencia de *frames*.

- LSTM

Una red convolucional LSTM, es un tipo de red que "recuerda" estados previos, es decir tiene en cuenta lo que ha sucedido anteriormente, y utiliza esta información para predecir una acción. Para desarrollar este el proyecto la red LSTM es de gran importancia, ya que permite reconocer una acción a partir de una secuencia de *frames*.

La técnica *Sentiment Analysis* ha servido para crear el modelo de red LSTM. Es una técnica utilizada para realizar el análisis y la clasificación de textos en positivo, neutral o negativo. Por ejemplo, se utiliza para analizar la información que publican las personas en las redes sociales y extraer información valiosa a la hora de saber cómo se sienten los clientes con los productos, las marcas, los servicios…

En el caso de este proyecto, de forma semejante, el *Sentiment Analysis*, se utiliza para analizar la información que dan los *keypoints* (que en el ejemplo de las redes sociales serían los textos), y para, posteriormente, determinar la acción que se observa en la secuencia de imágenes (en el ejemplo de las redes sociales los textos se clasificarían en positivo, negativo o neutral).

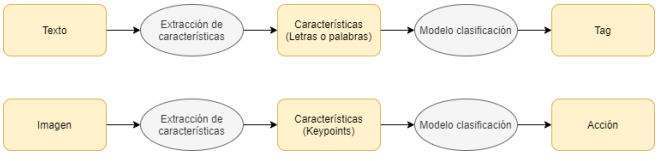


*Figura 6: Procedimiento Sentiment Analysis*

Un ejemplo de red convolucional ya creada es *ResNet*, ya mencionada con anterioridad. Esta evalúa una secuencia de *frames* para predecir una acción. Pero para desarrollar este proyecto se ha creado una red convolucional LSTM propia aplicando la técnica del *Sentiment Analysis*.

Para determinar la acción que se realiza en un vídeo se ha analizado una secuencia de diez *frames* que se ha procesado mediante *OpenPose* para obtener los correspondientes *keypoints* de cada uno de los *frames* de la secuencia.

Los *keypoints* son procesados por la red convolucional LSTM dando como resultado la probabilidad de que en el vídeo se pueda determinar las diferentes acciones.

Tras obtener los datos probabilísticos se selecciona la acción con mayor probabilidad para cada uno de los *frames*, y entre todas las acciones seleccionadas de cada uno de los *frames* se escoge la que más se repite.

### 5.3 Entorno Prototipo

Durante el desarrollo del proyecto se ha realizado una base prototipo para el uso y demostración de los algoritmos implementados.

Todo ello se ha desarrollado con la ayuda del novedoso *framework StreamLit*. Este es un *framework open-source* desarrollado en Python, que ofrece la posibilidad de desarrollar de forma sencilla una aplicación web (web app) a partir de los *scripts* del proyecto, esto es muy útil para desarrollar proyectos de *Machine Learning* y *Data Science*. Esta aplicación es accesible desde cualquier dispositivo y sin tener la necesidad de invertir una gran cantidad de tiempo.

Actualmente, para realizar demostraciones relacionadas con *Machine Learning* o *Data Science*, es necesario invertir una gran cantidad de tiempo en desarrollar un prototipo en un lenguaje distinto a Python o tener que contratar a desarrolladores para realizarlo. También existe *Jupyter Notebook* o *Google Colab*, ambos son documentos que permiten compartir y mostrar el código desarrollado y también visualizar los resultados del código. Esta es una forma de mostrar ejemplos de implementación, ya que permite a los usuarios poder modificar el código y ejecutarlo para ver los resultados.

En el caso de querer mostrar un prototipo funcional y en el que únicamente se pretende realizar inferencia y que el usuario no modifique el código, *StreamLit* es la mejor opción.

## 6 Resultados

### 6.1 *Datasets*

Para realizar el desarrollo y obtener los resultados de los métodos aplicados en este proyecto se han utilizado distintos *datasets*:

- *UCF101-Action Recognition*[16], se trata de un data set publico comúnmente usado para el reconocimiento de acciones, está compuesto por un total de 101 clases de acciones formadas por vídeos extraídos de YouTube. Algunas de las acciones de las que dispone este *dataset* son: *TaiChi, HighJump, Biking, Skate Boarding, etc.*
- T-RREX, se trata un *dataset* compuesto de imágenes de gestos para controlar una AGV.
- Un tercer *dataset* de creación propia para el cual se han grabado diferentes vídeos realizando la acción de saltar y caminar para la clasificación de poses.


*Figura 7: Muestra dataset UCF-101*

### 6.2 Análisis del algoritmo AutoML

Para analizar los algoritmos de AutoML se han utilizado *Auto-Sklearn* y *AutoKeras*.

Para comprobar el funcionamiento de *Auto-Sklearn* se ha entrenado un modelo de clasificación de datos en el cual se han obtenido los mejores valores de hiperparametros y el mejor clasificador para el conjunto de datos. Con este modelo se ha realizado un proceso de validación y se ha obtenido un *accuracy* de un 95.9%. Analizando a fondo los valores del modelo se observa que el mejor clasificador de datos es *Multi Layer Perceptron (MLP)*, una red neuronal formada por múltiples capas y con la capacidad de resolver problemas que no son linealmente separables.

Para comprobar el rendimiento de *Auto-Sklearn* se ha realizado el mismo entrenamiento con diferentes clasificadores de datos: *Decision Tree, Random Forest, KNN, MLP y Gaussian Process*. Todos han sido utilizados con los valores por defecto y, posteriormente, de forma manual, se han ajustado los hiperparametros de cada uno de ellos hasta obtener los mejores resultados.

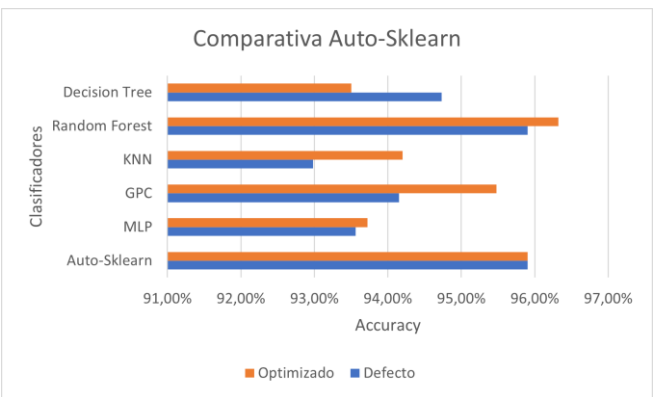


*Figura 8: Comparativa Auto-Sklearn*

Tras analizar *Auto-Sklearn* con los diferentes algoritmos de clasificación se ha decidido compararlo con otro algorit-

mo de *AutoML*, *AutoKeras*.

Tras realizar el mismo entrenamiento con los dos algoritmos de AutoML se puede observar que *AutoKeras* tiene un mayor *accuracy* y rendimiento respecto a *Auto-Sklearn*, aproximadamente un 2% más. Esta mejora de rendimiento se debe al aplicar algoritmos del *Deep Learning*, haciendo que el modelo obtenga un mayor rendimiento y mejor optimización.

## 6.3 Human Activity Recognition

### *6.3.1 Clasificación de poses*

Para realizar la clasificación de poses se han probado distintos métodos obteniendo los siguientes resultados:

- *Single-Frame CNN*

Tras haber entrenado el modelo, se han realizado distintas validaciones utilizando vídeos extraídos de *YouTube*. Para realizar la validación se ha procesado el video *frame* a *frame* prediciendo la acción que se está realizando en un único *frame*.

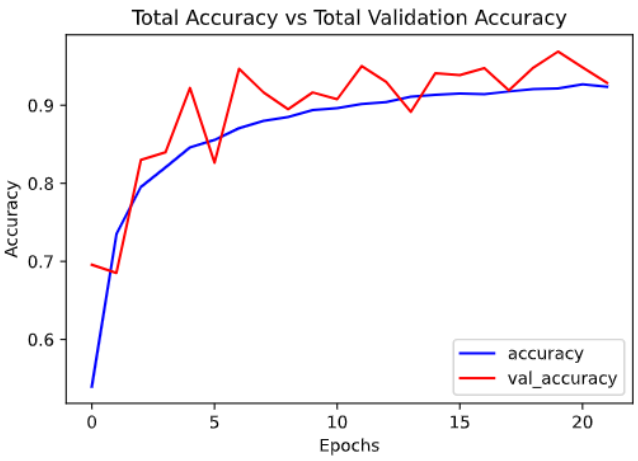


*Figura 9: Comparativa Auto-Sklearn*

Analizando los resultados obtenidos tras procesar diferentes vídeos, se puede observar que en algunos casos se ha reconocido correctamente la acción realizada, pero en la mayoría de los casos el reconocimiento es muy inestable. Esta inestabilidad se genera por el hecho de procesar el video *frame* a *frame* y no hacer uso de una secuencia de *frames*.


*Figura 10: Resultados Single-Frame*

- *ResNet*

Analizando los resultados obtenidos se ha incrementado el número de aciertos en el reconocimiento de la acción y de una forma más estable que el método anterior. Se ha podido observar que en algunos si generaba algo de confusión y no reconocía correctamente la imagen, esto puede ser debido a la gran cantidad de clases que contiene la red que pueda encontrar muchas similitudes entre distintas acciones.


*Figura 11: Procedimiento Sentiment Analysis*

- Pose Detection (OpenPose)

Inicialmente se han analizado los resultados obtenidos con el *dataset* propio. En la siguiente tabla, se detalla el número de vídeos y el total de *frames* que sean extraído de cada uno.

| Acción | N.º Vídeos | N.º Frames |
|---|---|---|
| Caminar | 8 | 1034 |
| Saltar | 6 | 464 |

*Tabla 1: Tabla de datos sobre dataset propio*

Tras a ver extraído todos los *frames* de los vídeos, se ha procesado cada una de las imágenes con el algoritmo de *OpenPose* para extraer las características de la pose, es decir los *keypoints*. Estos *keypoints* sirven para crear un *dataset* reuniendo todas las poses detectadas y con ello entrenar el modelo de clasificación de pose propio.

Para realizar la clasificación de la pose se ha utilizado un clasificador clásico, *RandomForest* y posteriormente se ha hecho uso de un algoritmo de AutoML, *AutoKeras*. Para los dos clasificadores, los entrenamientos han sido realizados

con el mismo porcentaje de entrenamiento y de prueba, en este caso un 30% y 70% respectivamente.

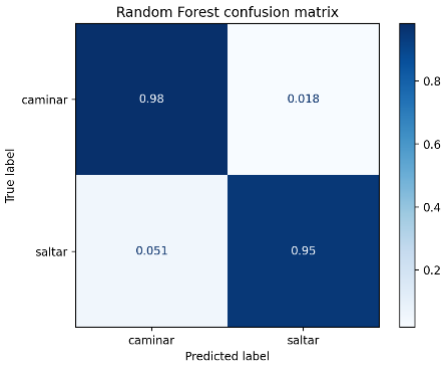


*Figura 12: Matriz Confusión Random Forest*

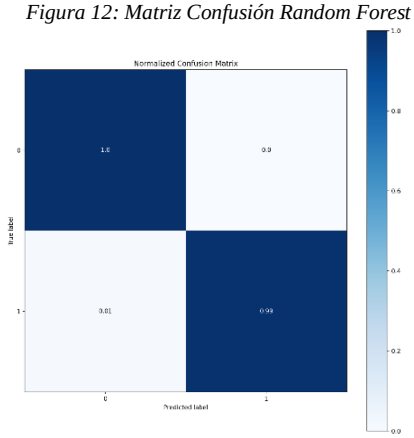


*Figura 13: Matriz Confusión AutoKeras*

Analizando las matrices de confusión que se han obtenido tras realizar el entrenamiento se puede observar que al tener una clasificación binaria y al tener más datos de la acción caminar respecto a la acción saltar hace que se obtenga un modelo que se adapta mucho a los datos, es decir se está produciendo *overfitting*. Se necesitarían más acciones o datos para obtener un modelo más eficiente.

Otro punto que destacar es que sería necesario grabar más vídeos desde diferentes puntos de vista para tener una mayor variedad de ángulos.

Tras haberse analizado los resultados obtenidos con el *dataset* propio, se han vuelto a analizar con el *dataset* público UCF-101. Los resultados han sido los siguientes:

Al tratarse de un *dataset* compuesto por vídeos de YouTube, es muy difícil encontrar gran cantidad de vídeos en los cuales únicamente aparezca una persona. Esto ha sido un problema, ya que *OpenPose* extrae la información de todas las poses detectadas de todos los cuerpos que aparecen en la imagen y es difícil averiguar qué conjunto de puntos corresponde a la persona que está realizando la acción.

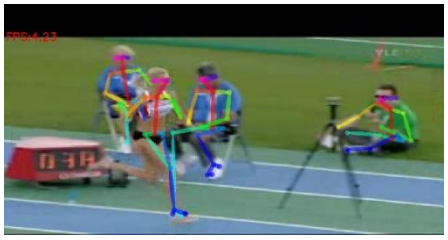

*Figura 14: OpenPose sin tracking*

Para solucionar este problema se ha aplicado un algoritmo de *tracking*, así se ha podido realizar un seguimiento de la persona que realiza la acción a través de los diferentes *frames* del vídeo. El algoritmo aplicado para realizar el *tracking* de la persona ha sido *DeepSort*. Este algoritmo facilta la extracción de las características de la pose únicamente de la persona que realiza la acción y no de todas aquellas que se encuentran en el *frame* (Figura 9).

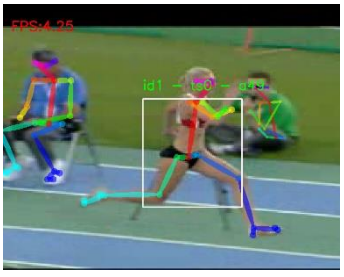

*Figura 15: OpenPose con DeepSort*

### 6.3.2 Reconocimiento de actividades

- AutoML

Para realizar el entrenamiento del clasificador de imágenes, se ha escogido el algoritmo de *AutoKeras*, ya que ha sido el que mayor rendimiento y mejores resultados ha ofrecido. Para realizar el proceso de entrenamiento se ha seguido un proceso similar a la fase de reconocimiento y se ha entrenado el modelo en base a las características de la pose detectada.

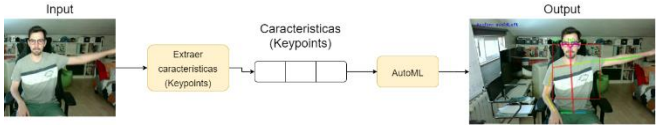


*Figura 16: Diagrama AutoML*

Para este clasificador se han realizado pruebas de validación clasificando la pose detectada *frame* a *frame*. En el caso de analizar secuencias de video este método de validación no es válido ya que una acción no sucede en un *frame*, si no que transcurre durante varios frames. Por esta razón, se ha implementado una red neuronal LSTM.

- LSTM

Para realizar el reconocimiento de actividad a través de la red LSTM se han seguido un conjunto de pasos (Figura 17). En esta red se han analizado secuencias de imágenes y no una única imagen como en *AutoKeras*.

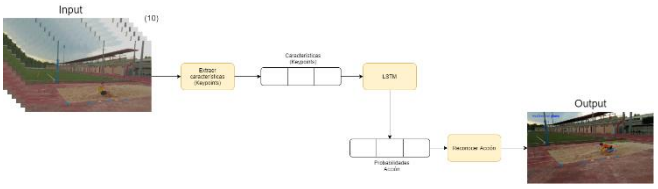


*Figura 17: Diagrama LSTM*

Analizando los resultados obtenidos tras realizar diferentes análisis de vídeos, se ha podido comprobar que este tipo de red proporciona un gran rendimiento para el analizar secuencias de video. Esto ha hecho que se obtengas resultados mucho más estables y con mayor precisión al tener en cuenta una secuencia de vídeos y no un único *frame*.

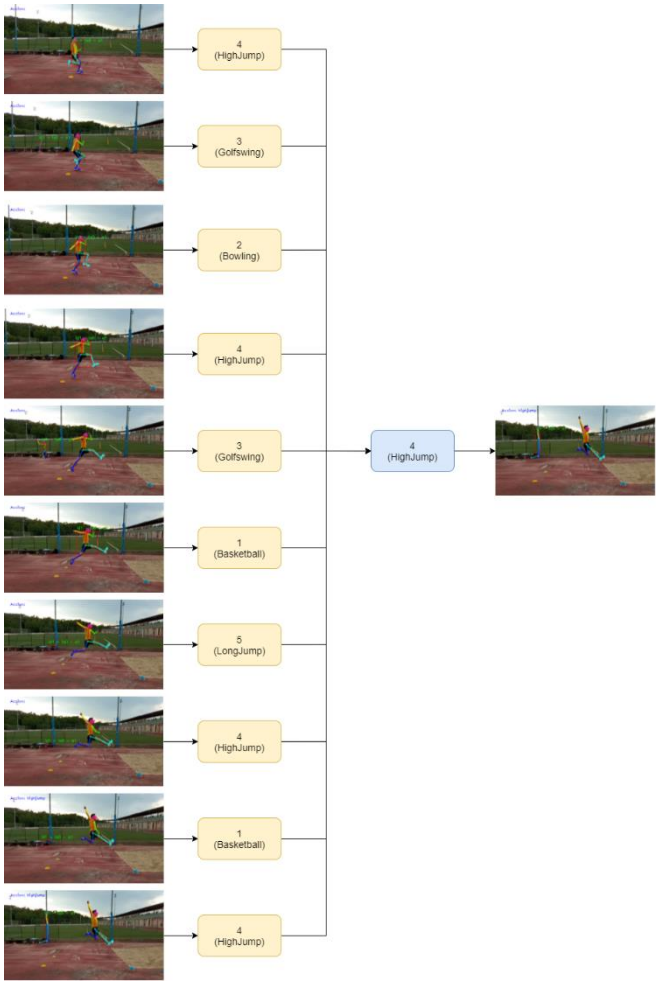


*Figura 18: Resultado LSTM*

### 6.4 Entorno prototipo

Para realizar el prototipo de este proyecto, se ha hecho uso del *framework StreamLit*. Actualmente, este *framework* está en auge, ya que permite generar aplicaciones web de forma sencilla a partir de código Python.

*StreamLit* está muy enfocado a poder realizar demostraciones de *Data Science* y *Machine Learning* de forma sencilla, esto hace que el programador o investigador evite el tener que realizar un programa de demostración online sin tener conocimiento de programación web. Una vez se ha obtenido el conjunto de funciones que muestra el uso de la red neuronal para hacer inferencia, se han incluido distintas llamadas al *framework* de *StreamLit*. Esto hace que se obtenga una *webApp* accesible desde el navegador y accesible desde cualquier sitio.

## 7 CONCLUSIONES

Uno de los objetivos es hacer un análisis de los algoritmos de AutoML, para ello se ha hecho una comparativa con diferentes algoritmos de clasificación y comprobar su funcionamiento. Tras realizar una larga serie de pruebas y probar diferentes métodos, se ha podido comprobar que los algoritmos de autoaprendizaje como *AutoKeras* y *AutoSklearn*, proporcionan un buen método de entrenamiento para obtener modelos de forma rápida y sin tener que invertir mucho tiempo en el proceso de optimización de los hiperparametros. Comparado con los algoritmos clásicos de clasificación como: *RandomForest*, *KNN*, *DecisionTree*, *Multi Layer Perceptron* y *Gaussian Process*, se ha observado que es necesario invertir tiempo para obtener un modelo de datos bien optimizado.

Por lo que respecta al reconocimiento de actividades, se han implementado distintos métodos. El primer método se basa en la clasificación de imágenes para poder extraer la acción y el segundo método se basa en la pose de la persona que realiza la acción. En el caso del reconocimiento en base a la pose, se ha podido ver que tras hacer pruebas *frame* a *frame*, en muchos casos al no saber el contexto de la acción que se está realizando no clasifica correctamente la acción. Para solucionar esta falta de contexto, es necesario aplicar redes neuronales que tengan en cuenta el tiempo para poder reconocer la acción. Se ha aplicado una red neuronal LSTM, la cual ha permitido que se pueda realizar el reconocimiento de actividades de secuencias de video haciendo que se obtengan unos buenos resultados.

Por lo que respecta al prototipo, se ha realizado una primera base haciendo uso del *framework StreamLit*. Ha sido una herramienta muy ágil y sencilla para poder obtener un primer prototipo sencillo con este *framework*. Es un *framework* muy interesante, permite llegar a crear una aplicación web funcional y ágil para crear demostraciones de aplicaciones de *Machine Learning* y *Data Science*.

Todo el código desarrollado durante el proyecto se ha incluido en un repositorio GitHub. ( https://github.com/japicazosuni/TFG_HAR )

## AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi tutor del proyecto, Coen Antens (CVC), el apoyo mostrado en el proyecto y por ayudarme a guiar el trabajo semana tras semana, dándome consejos, herramientas y aspectos a mejorar durante el desarrollo. También darle las gracias a mi familia, amigos y pareja por el apoyo que me han dado y mostrar interés

en el proyecto desarrollado.

## BIBLIOGRAFIA


[1] FIFA [Online]. Disponible: https://football-technology.fifa.com/es/media-tiles/video-assistant-referee-var

[2] Kognia [Online]. Disponible: https://kogniasports.com

[3] Bnk To The Future [Online]. Disponible: https://app.bnktothefuture.com/pitches/2079/_first-v1sion-the-sports-broadcasting-revolution-with-andres-iniesta-and-serge-ibaka

[4] HomeCourt [Online]. Disponible: https://www.homecourt.ai

[5] Streamlit [Online]. Disponible: https://Streamlit.io

[6] AutoML [Online]. Disponible: https://www.automl.org

[7] ClickUp [Online]. Disponible: https://clickup.com

[8] 2018. Unite the People [Online]. Disponible: https://www.simonwenkel.com/2018/12/09/Datasets-for-human-pose-estimation.html

[9] 2017. A History of Machine Learning and Deep Learning [Online]. Disponible: https://www.import.io/post/history-of-deep-learning/

[10] 2020. Creating a Human Pose Estimation Application with NVIDIA DeepStream [Online]. Disponible: https://developer.nvidia.com/blog/creating-a-human-pose-estimation-application-with-deepstream-sdk/

[11] MPII Human Pose Dataset [Online]. Disponible: http://human-pose.mpi-inf.mpg.de

[12] Hawk-Eye [Online]. Disponible: https://www.hawkeyeinnovations.com/

[13] 2015. Efficient and Robust Automated Machine Learning, Feurer et al., Advances in Neural Information Processing Systems 28 (NIPS 2015) [Online]. Disponible: https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

[14] Breast Cancer Wisconsin (Diagnostic) Data Set [Online]. Disponible: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

[15] 2019.Haifeng Jin, Qingquan Song, and Xia Hu. "Auto-keras: An efficient neural architecture search system." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining [Online]. Disponible: https://autokeras.com

[16] 2012. University of Central Florida [Online]. Disponible: https://www.crcv.ucf.edu/data/UCF50.php

[17] 2021. Taha Anwar. Introduction to Video Classification an Human Activity Recognition[Online]. Disponible: https://learnopencv.com/introduction-to-video-classification-and-human-activity-recognition/

[18] 2019. Adrian Rosebrock. Human Activity Recognition with OpenCV and Deep Learning [Online]. Disponible:https://www.pyimagesearch.com/2019/11/25/human-activity-recognition-with-opencv-and-deep-learning/

[19] 2017. The Kinetics Human Action Video Dataset [Online]. Disponible: https://arxiv.org/abs/1705.06950

[20] FitnessAlly. Disponible: https://fitnessallyapp.com/

[21] 2019. Proyecto Trrex [Online]. Disponible: https://grupopromaut.com/proyecto-trrex-cdti-2019-2021/