Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Telemática

Trabajo Fin de Grado

Diseño, implementación y evaluación de un sistema de segmentación y seguimiento de partes del cuerpo basado en "Deep Learning"

Autor: Cristina Rodríguez Larrén

Tutor: Javier Macías Guarasa

Cotutora: Leticia Monasterio Expósito

2022

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Telemática

Trabajo Fin de Grado

Diseño, implementación y evaluación de un sistema de segmentación y seguimiento de partes del cuerpo basado en "Deep Learning"

Autora: Cristina Rodríguez Larrén

Tutor: Javier Macías Guarasa

Cotutora: Leticia Monasterio Expósito

Tribunal:

Presidente: Manuel Mazo Quintas

Vocal 1º: Daniel Pizarro Pérez

Vocal 2º: Javier Macías Guarasa

Fecha de depósito: 12 de septiembre de 2022

A mi familia, amigos y seres queridos.

"El logro real no depende tanto del talento como de la capacidad de seguir adelante a pesar de los fracasos"
Daniel Goleman

Agradecimientos

La vida es un diez por ciento como la hacemos y un noventa por ciento como la tomamos.

Irving Berlín

A mis tutores, Javier y Leticia, les agradezco toda la ayuda y conocimientos que me han ofrecido y transmitido para poder llevar a cabo este proyecto.

Mención especial a todas aquellas personas que han estado conmigo en este largo camino de la universidad. Como Valeria, mi compañera en el comienzo de esta gran aventura y María por contribuir en este bonito final.

Mi eterno agradecimiento a toda mi familia, en especial a mi madre, padre, hermana y como no, mis segundas madres, mi tía Inés y mi tía Tere. Todos ellos siempre participes de cada paso y cada progreso.

Y, por supuesto, también se lo quería agradecer a esa familia que se elige, a todo mi grupo de amigas por no dudar nunca de mi, y a Gorka por ese apoyo incondicional.

Resumen

El objetivo de este trabajo es la investigación y desarrollo de un sistema de segmentación y seguimiento de partes del cuerpo basado en técnicas de "Deep Learning". El aprendizaje profundo utiliza redes neuronales artificiales con diferentes niveles o capas conectadas. Estas redes son utilizadas para hacer predicciones de unos datos de entrada tras una etapa de entrenamiento. Los datos de entrada orientados a este proyecto son vídeos RGB recopilados de diferentes bases de datos.

Tras una larga búsqueda, se recopilaron diferentes sistemas y bases de datos disponibles orientados a la detección de la postura humana en 3D. Se seleccionó uno de los sistemas y una base de datos para comenzar con el desarrollo. Se puso en marcha el sistema consiguiendo unos resultados óptimos. Y, después, se introdujeron dos propuestas. La primera propuesta orientada a comprobar la funcionalidad del sistema con una nueva base de datos. Y, la segunda propuesta, consistía en hacer diferentes experimentos con los distintos tipos de entrenamiento que nos encontramos, cambiando también ciertos valores (como el *batch size*) para detectar la combinación que mejores resultados ofrecía.

Palabras clave: Estimación de pose humana 3D, aprendizaje profundo.

Abstract

The objective of this work is the research and development of a body part segmentation and tracking system based on deep learning techniques. Deep learning uses artificial neural networks with different levels or layers connected. These networks are used to make predictions from input data after a training stage. The input data targeted for this project are RGB videos collected from different databases.

After a long search, different available systems and databases oriented to 3D human pose detection were collected. One of the systems and a database was selected to start with the development. The system was implemented and optimal results were achieved. Then, two proposals were introduced. The first proposal was to test the functionality of the system with a new database. And, the second proposal consisted of carrying out different experiments with the different types of training that we found, also changing certain values (such as the batch size) to detect the combination that offered the best results.

Keywords: 3D human pose estimation, deep learning.

Índice general

Índice de figuras

Índice de tablas

Lista de acrónimos

API	Interfaz de Programación de Aplicaciones.
BIRCH	Equilibrio Iterativo de Reducción y Agrupación mediante Jerarquías.
BNN	Redes Neuronales Biológicas.
CNN	Red Neuronal Convolucional.
COCO	Microsoft Common Objects in Context.
CPN	red piramidal en cascada.
DBSCAN	Agrupamiento Espacial Basado en Densidad de Aplicaciones con Ruido.
FC	Fully Connected.
FPN	Función de red piramidal para detección de objetos.
FPS	fotogramas por segundo.
GIT	Grado en Ingeniería Telemática.
GT	ground truth.
IA	Inteligencia Artificial.
IDE	entorno de desarrollo integrado.
IMC	índice de masa muscular.
LSP	Leeds Sports Pose Dataset.
MLP	perceptrón multicapa.
MPJPE	Error medio de posición por articulación.
OED	Observatorio Estatal de la Discapacidad.
OPTICS	Ordenar Puntos para Identificar la Estructura de Agrupamiento.
PCA	Análisis de Componentes Principales.
PSP	potencial post-sináptico.
ReLu	Rectifier Linear Unit.
RGB	red, green, blue.
RNA	Redes Neuronales Artificiales.

RNN	Redes Neuronales Recurrentes.
SGD	Stochastic Gradient Descent.
SOTA	state-of-the-art.
STE	Strided Transformer Encoder.
SVM	Support Vector Machine.
TFG	Trabajo Fin de Grado.
TI	Tecnologías de la Información.
VGG	Visual Graphics Group.
VTE	Vanilla Transformer Encoder.

Capítulo 1

Introducción

Vive como si fueras a morir mañana; aprende como si el mundo fuera a durar para siempre.

Mahatma Gandhi

1.1 Presentación

Este documento describe el Trabajo Fin de Grado (TFG) de la titulación correspondiente al Grado en Ingeniería Telemática (GIT) de la universidad de Alcalá. El trabajo se enfoca dentro del área de la tecnología aplicada en el entorno sanitario, en el problema del análisis de las restricciones cognitivas y de movilidad física que afectan a millones de personas en el mundo. Para poder solventar este problema o "hacer que sea más fácil convivir con él", se debe abordar un estudio las capacidades cognitivas y motoras de las personas y así conocer el grado de discapacidad que pueden tener. La solución a este problema se centra en sistemas basados en aprendizaje máquina capaces de analizar dichos aspectos. Estos sistemas se desarrollan dentro del contexto de la Inteligencia Artificial (IA) con tecnologías propias de *maching learning*, en concreto *deep learning* o aprendizaje profundo.

Las técnicas propias de las redes neuronales profundas son utilizadas para el desarrollo de algoritmos que permitan a través de unos datos de entrada poder llegar a conocer la capacidad funcional que tiene la persona en estudio. Los datos de entrada pueden verse como secuencias de imágenes o vídeo que captan los movimientos y gestos de uno o varios individuos. Esto se consigue con un entrenamiento previo de la arquitectura con una gran cantidad de datos de entrada para poder procesarlos y conseguir resultados fiables.

Por todo ello, este trabajo tiene como objetivo el estudio y desarrollo de estrategias y algoritmos que permitan la detección y localización de las diferentes articulaciones del cuerpo humano utilizando secuencias de imágenes captadas por cámaras de vídeo y profundidad $3D$ y RGB [20]. Para el desarrollo del trabajo los datos utilizados referidos a las secuencias de vídeo son extraídos de bases de datos del sitio web correspondiente. Y, para su posterior utilización, se llevan a cabo diferentes técnicas de procesamiento de vídeo.

Han sido motivo de estudio y evaluación aquellas arquitecturas y bases de datos halladas en la red capaces de operar con la cuestión de la estimación de pose humana en $3D$.

1.2 Motivación

Este proyecto ha sido impulsado por dos grandes campos: la necesidad de poder detectar y así ayudar a personas con funcionalidad limitada y, el interés de conocer e interactuar con máquinas inteligentes de procesamiento de datos.

Actualmente hay muchas personas con **movilidad física reducida** debido a diferentes causas como pueden ser una enfermedad, patología hereditaria o desarrollada con el tiempo o simplemente una dolencia, lesión o facturación de algún miembro del cuerpo. Esto indica que también se incrementa el número de personas con discapacidad o con dependencia hacia otra persona, animal o maquina. Esto último siempre que sea necesario para favorecer el nivel de vida de las personas afectadas y con él, el desarrollo de sus actividades cotidianas.

Para hablar de cifras reales, el Observatorio Estatal de la Discapacidad (OED) es una herramienta técnica de la Dirección General del Estado enfocada a la ciudadanía, las administraciones públicas, las universidades y el tercer sector con el fin de recopilar, sistematizar, actualizar, generar y difundir información relacionada con el ámbito de la discapacidad. Según un estudio realizado por el OED se conoce que el número total de personas que se han declarado con discapacidad en el año 2020 suma un total de $4,38$ millones de personas ($94,9$ por cada 1,000 habitantes). Es decir, este número aumenta sumando todas aquellas personas no declaradas. En la tabla de la figura 1.1 se muestra en más detalle el número de personas con discapacidad según el sexo y el tipo de discapacidad.

	Hombres	Mujeres
Total	**81,2**	**112,1**
Movilidad	38,9	68,5
Vida doméstica	31,8	57,8
Audición	24,2	31,0
Autocuidado	22,9	38,0
Comunicación	18,8	23,7
Visión	18,4	28,6
Interacciones y relaciones personales	13,5	13,8
Aprendizaje, aplicación del conocimiento y desarrollo de tareas	13,4	18,2

Tabla 1.1: Tabla de personas con incapacidad reconocida separadas por sexo y tipo de discapacidad en el año 2020 [19].

Si analizamos en profundidad estos valores podemos apreciar como el número de personas con cierta discapacidad es muy grande, aumentando esta cifra, por desgracia, cada año. Por ello, hoy en día se da mucha importancia a este tema y se están llevando a cabo muchas investigaciones y proyectos tanto para evaluar y calcular la incapacidad de un individuo como para más tarde tener conocimientos para ayudar a esas personas a tener una vida más cómoda.

Por otro lado, la IA está revolucionando la sociedad actual llevando el mundo entero a una era aún más motivada por la tecnología. Las máquinas inteligentes son cada vez más utilizadas en diferentes ámbitos, hasta el punto de que todo lo que nos rodea día a día está influido por ello. El principal objetivo de la aplicación de esta ciencia en el mundo de la medicina es analizar la relación entre los métodos de prevención, diagnóstico o tratamiento y sus resultados para los pacientes [21]. El *maching learning* o aprendizaje automático es uno de los tipos más conocidos de IA que aprovecha los avances en tecnología de las Tecnologías de la Información (TI) para el aprendizaje supervisado. Y, dentro de ella cabe destacar

la tecnología *deep learning* que consiste en el despliegue de redes neuronales profundas para la generación de sistemas predictivos que pretenden simular cómo el cerebro humano aprende una información.

Por todo ello, recopilando y mezclando ambos campos podemos encuadrar el trabajo dentro en el proyecto EYEFUL [22], que plantea explotar las capacidades de los sistemas multisensoriales para realizar una evaluación automática de las limitaciones funcionales de personas al realizar las actividades de la vida cotidiana. La forma de hacerlo es analizando la interacción entre personas y objetos en secuencias audiovisuales de dichas actividades.

Dicho esto, este trabajo fin de grado se integra en los objetivos del **proyecto EYEFUL** (ver logo del proyecto1.1) n lo referido al estudio de los elementos relacionados con la pose y el movimiento del cuerpo humano.

Figura 1.1: Logo del proyecto EYEFUL

La estimación de la postura humana en $2D$ y $3D$ [23] ha logrado alcanzar una precisión notable en los últimos tiempos ya que es uno de los campos más estudiados dentro del aprendizaje profundo, y la literatura al respecto es bastante extensa. Consta de un conjunto de técnicas de *deep learning* que, a partir de imágenes y vídeos hacen las labores de captar y determinar la forma del cuerpo humano y percibir los movimientos del cuerpo del individuo en estudio (ver imagen 1.2). Lo que se busca es hacer una segmentación de las articulaciones lo más real posible con ayuda de puntos clave para calcular la posición "real" dentro del espacio bidimensional o tridimensional.

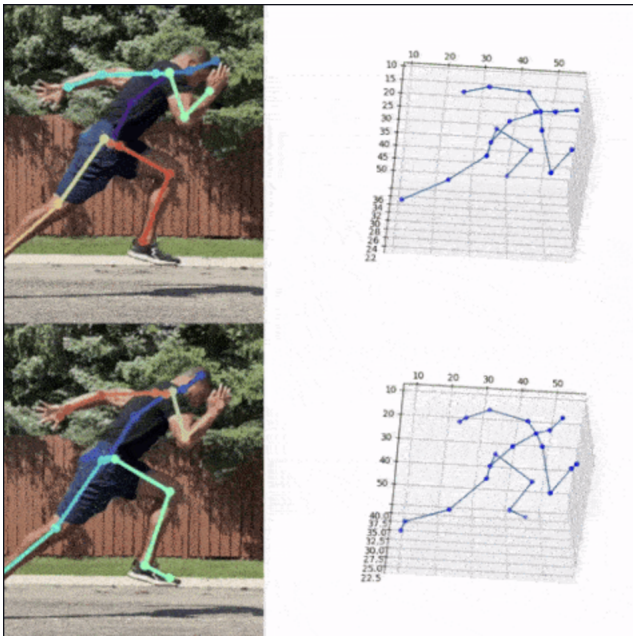


Figura 1.2: Estimación de pose sobre una imagen $2D$ vs la representa de la estimación de pose $3D$ sobre el eje de coordenadas [1].

Son numerosas las aplicaciones que tienen estos sistemas de aprendizaje profundo. No solo se desarrollan en el área de la salud, que es en el sector al que corresponde este proyecto, sino que también se experimenta con ellos en las áreas de seguridad, deporte, industria, etc. Todas estas áreas se encuentran aún en desarrollo. Pero el empleo de estas técnicas de aprendizaje profundo para construir un algoritmo de este calibre no es tarea fácil puesto que hay que enfrentarse a problemas complejos como son:

- Imagen rotada.
- Oclusión de alguna articulación.
- Necesidad de conseguir un gran número de imágenes anotadas por expertos humanos.
- Buena comprensión de las diferentes disciplinas de la IA.

1.3 Objetivos y desarrollo

La investigación que se lleva a cabo en este proyecto está enfocada a la identificación y posterior estudio del problema de movilidad que tienen algunas personas en la actualidad. Es decir, ser capaces de identificar el nivel de capacidad motora y cognitiva de un individuo de cualquier edad. Para reconocer e identificar las competencias físicas del individuo es necesario hacer un estudio detallado de ello. Dicho estudio requiere tratar con elementos relacionados con la pose de personas (posición y orientación), y el movimiento de la misma (velocidad, cadencia, continuidad, equilibrio, coordinación, etc.). Y, a través de la visualización virtual realizar un análisis de los cambios de desplazamiento del cuerpo y articulaciones para para poder reconocer cada una de ellas y apreciar los movimientos.

El objetivo fundamental de este proyecto es el diseño, implementación y evaluación de ciertas estrategias y algoritmos que permitan la identificación de las diferentes articulaciones del cuerpo. El trabajo implicará la utilización de técnicas de procesamiento de vídeo tanto RGB como de profundidad [24]. Estas se combinan con las de IA basándose en sistemas algorítmicos de estimación de pose con redes neuronales. Lo que se trata es de conseguir una estimación de pose $3D$ lo más cercana a la realidad, es decir, que los resultados obtenidos tengan una tasa de error lo más baja posible.

El entorno de desarrollo se apoyará en una plataforma GNU/Linux, sobre el lenguaje de programación Python, usando librerías avanzadas de procesamiento.

Los objetivos específicos de este proyecto expuestos en orden cronológico acorde a su tiempo de ejecución son los mostrados a continuación:

- Formación en tecnologías de aprendizaje máquina aplicadas a estimación de pose.
- Revisión del estado del arte para estimación de pose $3D$, incluyendo técnicas de *deep learning* basadas en Red Neuronal Convolucional (CNN). Búsqueda de información relacionada con ayuda de trabajos anteriores, así como conocer las librerías necesarias de Python.
- Selección de sistemas con criterios de funcionalidad y disponibilidad. Durante el desarrollo del trabajo se plantean y analizan diferentes arquitecturas y bases de datos seleccionando después la más adecuada para trabajar con ella.
- Puesta en marcha del sistema de estimación de pose. Descarga de las aplicaciones que se van a utilizar, así como todas las librerías Python necesarias para poder ejecutar el código.
- Modificación y evaluación de los sistemas seleccionados.

- Evaluación rigurosa. Analizar los resultados obtenidos y compararlos con los reales (*ground truth*).
- Documentación de dicho trabajo. Dejar escrito todos los procesos y evaluaciones realizadas, así como los resultados.

1.4 Organización de la memoria

Esta memoria se organiza en 6 grandes capítulos que recogen todo el proceso desde el inicio del estudio hasta su implantación final. El inicio parte de las primeras investigaciones de fuentes de datos, así como la formación en tecnologías de aprendizaje máquina aplicadas a estimación de pose. Y, el final, comprende los resultados obtenidos tras todo el estudio. A continuación se detalla la estructura:

- Capítulo 1. Contiene la presentación del TFG, las causas principales de la motivación del estudio de las técnicas de *deep learning*, los objetivos esenciales y complementarios del proyecto, así como el desarrollo hasta llegar a ellos. También se deja documentada la estructura del documento y los capítulos que lo componen.
- Capítulo 2. Comprende el estado del arte para estimación de pose $3D$. Se explica la teoría en tecnologías de aprendizaje de máquina aplicadas a estimación de pose que hay detrás del proyecto. Incluye explicaciones detalladas de las técnicas de *deep learning* basadas en CNN relatando los diversos tipos que hay y sus funciones básicas. También se cuenta la importancia del entrenamiento en redes, así como sus diferentes versiones. Incluye, además, información relacionada con el lenguaje de programación utilizado, así como las librerías necesarias de Python.
- Capítulo 3. En este capitulo se plantean y analizan una lista de diferentes sistemas disponibles así como de bases de datos distintas. Todas ellas puestas en estudio por el cumplimiento de algunas de las características que se pone como obligatoria para llegar a elegir la más adecuada para trabajar con ella. Estas características se basan en los criterios de funcionalidad y disponibilidad.
- Capítulo 4. Se documenta de forma detallada el completo desarrollo del proyecto. También describe la puesta en marcha del sistema de estimación de pose $3D$ y la descarga de las aplicaciones que se van a utilizar. Incluye las modificaciones realizadas al sistema.
- Capítulo 5. Apartado de recopilación de resultados. Se lleva a cabo una evaluación rigurosa analizando los resultados obtenidos y su posterior comparación con los estimados. Para ello se explican en detalle todos los experimentos realizados.
- Capítulo 6. Recoge las conclusiones finales del proyecto. Incluye un apartado de "líneas futuras" donde se sugieren posibles mejoras o avances que se pueden añadir en un futuro al trabajo.

Tras la bibliografía, que recoge todas las referencias empleadas para el desarrollo del proyecto, también aparece el apéndice A. Incluye un listado de todas las herramientas y recursos necesarios para la elaboración del proyecto. Dividido en elementos *hardware* y *software*.

Capítulo 2

Estudio teórico

Siempre se llega a alguna parte si se camina lo bastante.

Alicia en el país de las maravillas

2.1 Introducción

En este capítulo se aborda el estudio de las redes neuronales y su clasificación. Para poder entender mejor la estructura de una red neuronal, se amplía la información comparándolas con las neuronas del cerebro humano.

Por otro lado se especifica qué es la CNN y para qué se utiliza. También se explica con exactitud todo el conjunto de capas que compone una red neuronal, tanto la primera capa de partida, la capas ocultas y la capa final. En estas capas ocultas se encuentra la verdadera potencia de la red ya que depende de la configuración que se especifique para que realice un tipo de tareas u otro. La configuración de cada una de las diferentes capas define la arquitectura de la red neuronal.

Además, uno de los apartados de este capítulo explica la importancia del entrenamiento de las redes neuronales. Se desarrollan también de forma detallada, los diversos tipos de entrenamiento que existen (supervisado, no supervisado y semi-supervisado), ya que en el desarrollo de este proyecto se evalúan diferentes situaciones dependiendo de este factor.

Y, por último pero no menos importante, se pretende dar un espacio en este trabajo a definir uno de los lenguajes de programación con más éxito de nuestros tiempos como es el caso de Python y sus librerías.

2.2 Redes neuronales

El *deep learning* es un tipo de aprendizaje automático que utiliza algoritmos diseñados para funcionar de manera parecida al cerebro humano cuando averigua nueva información [25]. El equivalente al cerebro humano en lenguaje máquina son las llamadas redes neuronales. El cerebro humano es un órgano complejo capaz de regular y controlar la mayoría de funciones de nuestro cuerpo (movimientos y gestos), controlar las funciones vitales, inclusive dominar los pensamientos (emociones y conductas) que pasan por nuestra mente. Su similitud con la redes neuronales ha pasado a ser un foco de estudio para muchas personas y un auténtico reto tratar de reproducirlo con ayuda de las técnicas de TI y de IA.

Una red neuronal es un algoritmo que tiene como objetivo simular el comportamiento del cerebro biológico tratando de replicarlo con neuronas artificiales interconectadas almacenadas en filas llamadas capas [26].

2.2.1 Redes Neuronales Biológicas (BNN) vs Redes Neuronales Artificiales (RNA)

Primeramente es necesario conocer el funcionamiento del **cerebro biológico**. Se compone de neuronas que son células emisoras y receptoras especializadas en recibir, procesar y transmitir información a través de pequeños impulsos nerviosos. Los sentidos de la vista, el oído, el gusto, el tacto y el olfato proporcionan la información al cerebro. Luego procesa esta información, la interpreta y la integra utilizando las neuronas. Las partes principales de cada una de estas células son (ver figura 2.1):

- Soma o cuerpo celular (6) que corresponde a la parte principal de la neurona.
- Axones (3) que son extensiones que salen del soma.
- Botones terminales (2) en las partes finales de los axones. Puntos de conexión neuronal donde se produce la sinapsis (5), es decir el intercambio de información neuronal.
- Algunas neuronas, como las neuronas sensoriales y motoras, están envueltas en una capa de mielina (4). Esta capa ayuda a facilitar una transmisión más rápida de información entre las neuronas.
- Las dendritas (1) son ramificaciones en forma de árbol que salen del soma. Las dendritas son el componente de las neuronas que permiten la comunicación.

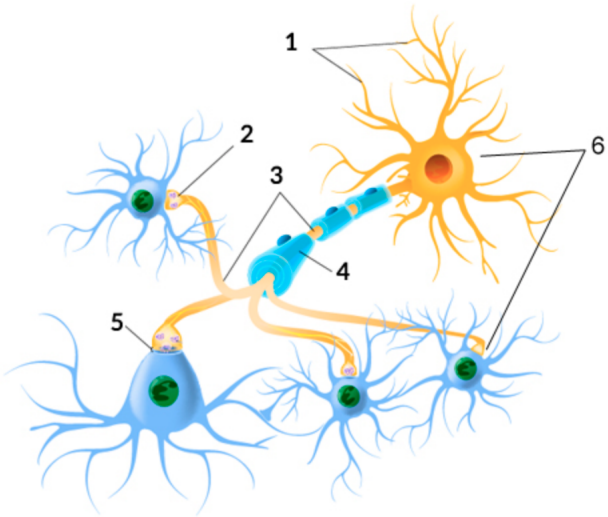


Figura 2.1: Esquema de neuronas conectadas. Los números pertenecen a las partes descritas en este apartado [2].

Ahora bien, como se ha dicho en ocasiones anteriores, las **RNA** tratan de asemejarse a la estructura de BNN. La forma de hacerlo es creando las redes por *software* pudiendo usar computadores muy potentes. Para mantener las características propias de las BNN, la neurona artificial o *perceptron* se define por tener una gran variación de entradas compuestas por datos originales o salidas de neuronas próximas. Cada entrada tiene dada por una conexión que tiene cierto peso los cuales corresponden con la eficacia sináptica de la neurona biológica. Cada neurona posee un valor umbral. Dentro de cada neurona se

suman las entradas, y se resta el valor umbral de la neurona. Todo ello genera la activación de la neurona o potencial post-sináptico (PSP) mediante una señal de activación que se procesa por una función de activación o función de transferencia, dando a la salida de la neurona.

En la figura 2.2 se puede ver de una manera más visual la relación que existe entre las partes de las neuronas biológicas y las neuronas artificiales.

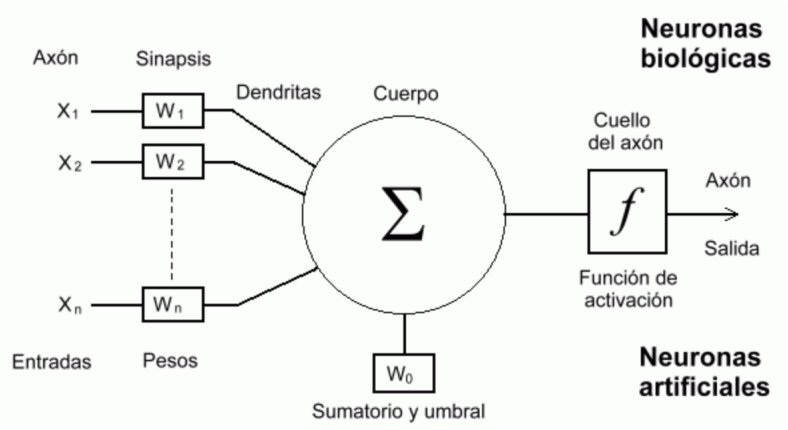


Figura 2.2: Esquema de equivalencia entre las partes de las neuronas biológicas y las neuronas artificiales [3].

Como dato curioso, cabe mencionar que los datos por el silicio se transportan mucho más rápido que en el cerebro biológico sin embargo, el cerebro tiene un mayor cantidad de conexiones como ventaja.

2.2.2 Elementos de las redes neuronales

Algunos de los elementos más importantes de las redes neuronales son los siguientes:

- Número de capas. Se diferencian las monocapa o perceptrón simple de las perceptrón multicapa (MLP) (ver figura 2.3). Las monocapa tienen la capa de entrada conectada a la capa. Por su lado, las multicapas tienen diversas capas de neuronas intermedias entre las conexiones de entrada y las de salida, denominadas capas ocultas conectadas o no entre ellas. Ejemplo: *red Adeline* (red monocapa) y *red Madaline* (red multicapa, surge de la combinación de módulos *Adeline*) [27].

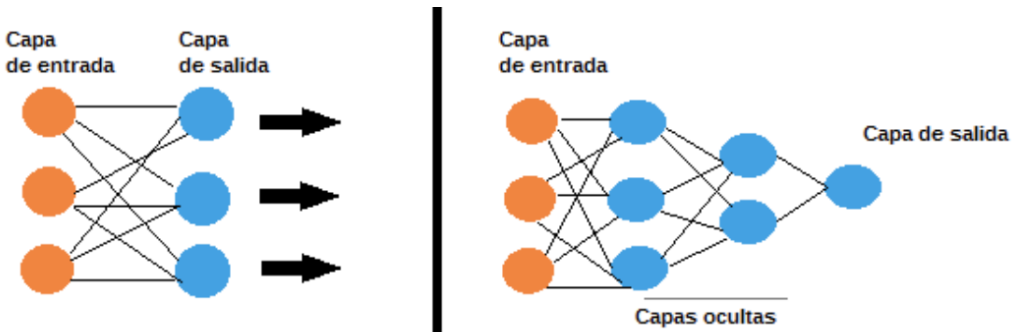


Figura 2.3: A la izquierda: esquema de red monocapa. A la derecha: MLP [4].

- Tipo de conexiones. La distribución de las neuronas forman capas y se pueden conectar con neuronas situadas justo después en la estructura sin conexiones hacia atrás, en este caso se trata de una configuración hacia adelante, llamada *feedforward* [28] o redes neuronales no recurrentes. Por

el contrario, hablaríamos de las Redes Neuronales Recurrentes (RNN). En las no recurrentes la información tiene un solo sentido de forma que siempre se conectan a capas siguientes y no posteriores. Carecen de realimentación y memoria, y no forman ciclos (no se utilizan mucho). En las redes neuronales recurrentes si hay conexiones en la misma capa o distintas de realimentación entre las neuronas, por ello forman ciclos y tienen memoria.

- Grado de conexiones. La entrada carece de propiedades neuronales, ya que las unidades que la forman solo introducen valores que más adelante serán procesados por la red. Las neuronas de las capas oculta y de salida están interconectadas al resto de neuronas de las capas precedentes. En general, en la mayoría de aplicaciones son mejores las redes completamente conectadas, aunque es posible el diseño de redes en las que se pueda mejorar el rendimiento con la eliminación de alguna conexión. Existen dos tipos: las redes neuronales parcial o totalmente conectadas. En las redes neuronales totalmente conectadas todas las neuronas están interconectadas. En las redes parcialmente conectadas esto no ocurre.

2.3 Red Neuronal Convolucional (CNN)

De los múltiples sistemas de RNA que existen, el que más nos interesa para tratar este proyecto son las CNN o redes neuronales convolucionales. Estas redes neuronales poseen uno de los algoritmos principales que contribuyen en el desarrollo y perfeccionamiento del aprendizaje máquina. Son especialmente útiles para la búsqueda de patrones en imágenes, clasificación de escenas y categorización de objetos, tomando las imágenes y escenas como *input* para posteriores predicciones.

Pero, para poder utilizar las CNN hace falta conocer las diferentes capas que componen su red así como qué se requiere para un entrenamiento. Realizar un entrenamiento no resulta tarea fácil ya que para llevarlo acabo es necesario una gran cantidad de muestras. Es decir, cuantas más muestras se empleen, con mayor exactitud captarán las neuronas de la red las características únicas de cada uno de los objetos. Por poner un ejemplo, en el caso de este trabajo, la red va a ser capaz de reconocer una articulación porque ya la ha visto anteriormente en muchas ocasiones. El sistema no solo buscará articulaciones semejantes sino que interferirá en imágenes que no conozca pero sí relacione por la existencia de similitudes. Esta es considerada la parte inteligente del conocimiento que obtiene una red neuronal.

2.3.1 Capas

Las CNN son una serie de redes que fueron creadas pensando en cómo funciona un cerebro humano. La corteza cerebral está formada por diferentes capas de tejido neuronal y cada una de estas capas posee una especialización funcional diferente. Llevando esto al campo de la IA, las redes neuronales son capaces de aprender en los diferentes niveles de abstracción hasta conseguir definir y clasificar con exactitud el objeto de la imagen inicial de entrada.

Para poder explicar la segmentación de capas de la arquitectura de las CNN, es necesario saber que, cuando se procesan imágenes, la red capta el contenido de la entrada como píxeles. Una imagen en escala de grises (1 color) necesita $28 \times 28 píxeles = 784$ neuronas. Sin embargo, la convolución con una **imagen RGB** (ver figura 2.4) en color necesita 3 canales de entrada (colores rojo, verde y azul) con $28 \times 28 \times 3 píxeles = 2352$ neuronas; el resultado en ambos casos se da en 1 solo canal. También hay que tener en cuenta que al alimentar la red, hay que normalizar los valores de entrada. Si los colores de los píxeles se codifican de 0 a 255; cuando se introducen los valores se transforman como "$valor/255$" quedando siempre un número entre 0 y 1.

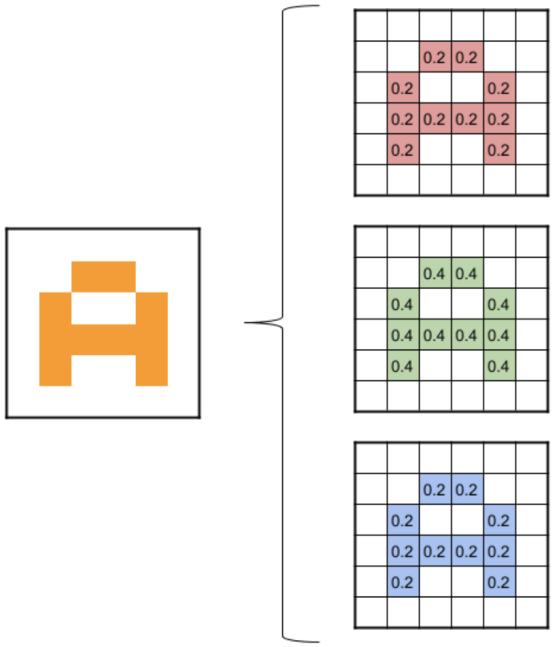


Figura 2.4: Ejemplo de imagen en color [5].

El concepto de arquitectura definida en el ámbito de las redes neuronales hace mención tanto al número de capas neuronales y de neuronas en cada una de ellas, como a las conexiones entre neuronas o capas (tipo de neuronas e incluso a la forma en la que son entrenadas). La arquitectura de las CNN puede ser muy variada y por ello existen varios tipos como AlexNet, ImageNet, Red Visual Graphics Group (VGG), GoogleNet, ResNet (y variaciones), SegNet, etc. A pesar de las variedades de arquitecturas neuronales que existen todas se componen de 3 capas principales. Estas capas varían en número según las funciones que quiera cumplir la red:

- Primera capa de partida o entrada. Alcanza a diferenciar formas simples, colores o bordes.
- Capa(s) intermedias ocultas o *hidden layers*. Toman ese nombre ya que las entradas y salidas de estas capas intermedias se encuentran enmascaradas por la función de activación y la implicación de una operación matemática que es la convolución (de ahí el nombre de CNN) (ver ejemplo en la figura 2.5). Pueden llegar a distinguir combinaciones de colores y bordes.
- Capa de salida. Es capaz de reconocer formas más complejas como un rostro o silueta [29]

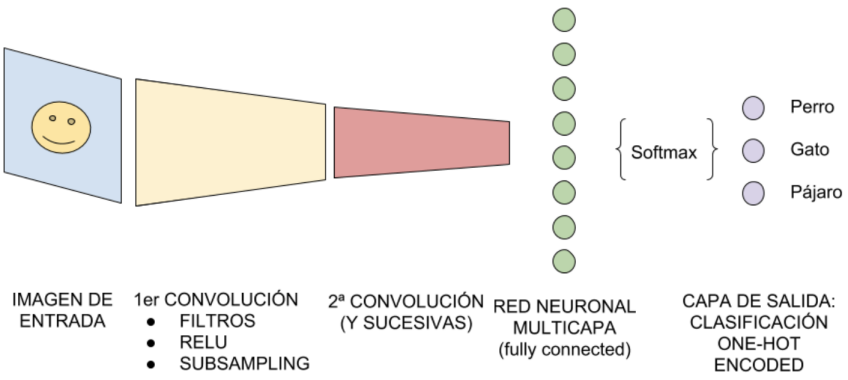


Figura 2.5: Ejemplo de arquitectura de una CNN [5].

Dentro de las capas intermedias o capas ocultas, algunas de las funciones más importantes que pueden desempeñar dichas capas son: convolucionales (incluye función de activación), *pooling* y *fully connected*. Estas operaciones se repiten consecutivamente en cientos de capas, de manera que cada capa aprende a identificar diversas características.

2.3.1.1 Convolucional

Aquí es donde comienzan el "procesado distintivo" de las CNN, es decir, se llevan a cabo las **convoluciones**. Consiste en tomar "grupos de píxeles adyacentes" de la imagen de entrada, convertida ya en matriz (tensor), y procesarlos matemáticamente haciendo un producto escalar contra pequeñas matrices llamadas ***kernel*** (o núcleo de convolución) como se puede ver en la figura 2.6. Un *kernel* es un componente importante de una red neuronal convolucional. Se considera un **filtro** que se puede aplicar a las imágenes para conseguir características o patrones de ella (bordes, enfoque, desenfoque, etc.). La aplicación no es de 1 *kernel*, si no que se hace con varios *kernel* (filtros).

Para poner un ejemplo, en una primera convolución utilizando 32 filtros, obtendríamos 32 matrices de salida (este conjunto se conoce como *feature mapping*). El tamaño de cada una sería de $28 \times 28 \times 1$ sumando 25,088 neuronas para la primera capa oculta de neuronas.

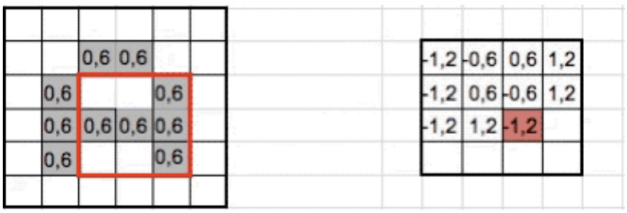


Figura 2.6: Ejemplo del funcionamiento del *kernel*. Realiza el producto escalar con la imagen de entrada. Desplazamiento o ***stride*** de 1 píxel dirección izquierda-derecha y de arriba-abajo mientras genera una matriz nueva [5].

A medida que se desplaza el *kernel* se obtiene imagen nueva filtrada. Continuando con el ejemplo anterior, para esta primera convolución se obtendrían 32 imágenes nuevas filtradas. Estas imágenes lo que dibujan son características de la original. Esto ayuda en un futuro a la distinción entre objetos.

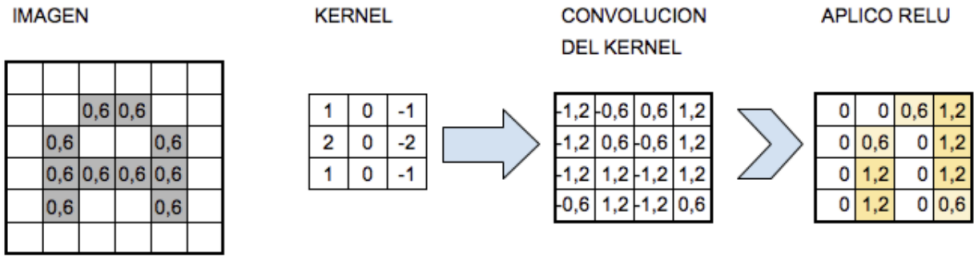


Figura 2.7: Ejemplo de convolución con un *kernel* aplicando la función de activación Rectifier Linear Unit (ReLu) [5].

Una vez hecho esto, para conseguir una predicción lo más próxima a la realidad posible, es necesario una correcta elección de la **función de activación**. Por tanto, debe ser elegida cuidadosamente para cada una de las redes de aprendizaje profundo alcancen la convergencia y conseguir así que la red aprenda. La función de activación actúa como **filtro, función limitadora o umbral**, para modificar el valor del resultado o imponer un límite que se debe sobrepasar para poder proseguir a otra neurona (ejemplo

de aplicación en figura 2.7). Transmite la información generada por la combinación lineal de los **pesos** y las entradas por las conexiones de salida. El producto escalar más utilizado para este tipo de redes neuronales suele ser el producto interno de Frobenius que consiste en una operación binaria que devuelve un escalar a partir de dos matrices. Por su parte, el interés por la función activación es generalmente, crear un **modelo no lineal** y por lo tanto, más complejo (explicación apartado 2.3.1.2).

La **normalización en lotes** (*Batch normalization*) consiste en añadir un paso extra (habitualmente entre las neuronas y la función de activación), con la idea de normalizar las activaciones de salida. Los tres hiperparámetros [30] de la normalización por lotes y que controlan el tamaño del volumen de salida son: la profundidad, la zancada y el relleno con ceros. La profundidad del volumen de salida corresponde a la cantidad de filtros que se plantea usar. La zancada con la que se desplaza el filtro. Y, el relleno a ceros o ***padding*** que consiste en agregar píxeles de valor 0 en torno a la imagen original. Destacan dos usos: uno es para que cuando se realiza la convolución la imagen resultante sea del mismo tamaño al de la imagen inicial; el segundo uso es para recopilar la información relevante más cerca del centro, ya que esta información puede estar en las esquinas de la imagen y, al realizar convolución, el filtro pase más por el centro que en las esquinas[29].

El tamaño de la salida de la capa de convolución se calcula con la siguiente fórmula 2.1 donde la variable "Entrada" corresponde al tamaño del filtro de entrada:

$$Salida = ((Entrada)/zancada) + 1. \tag{2.1}$$

A la imagen resultante de este proceso se le denomina **"mapa de características"** o de activación.

2.3.1.2 Función de activación

Las funciones de activación más utilizadas y conocidas son: función lineal o identidad (no modifica), función limitante, función escalón, función sigmoidal, función tangente hiperbólica, ReLu, funciones de Base Radial (Gausianas, multicuadráticas, multicuadráticas inversas...) [31]. Se pueden ver en las figuras 2.8 y 2.9. Existen otras algo más complejas como la función softmax y argmax.

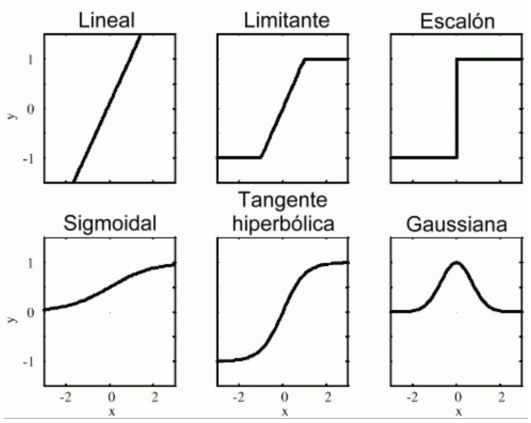


Figura 2.8: Ejemplos de funciones de activación [6].

La función de activación más conocida es la ReLu que tiene la función de sustituir los valores negativos que se reciben de la entrada por ceros (ver figura 2.9).

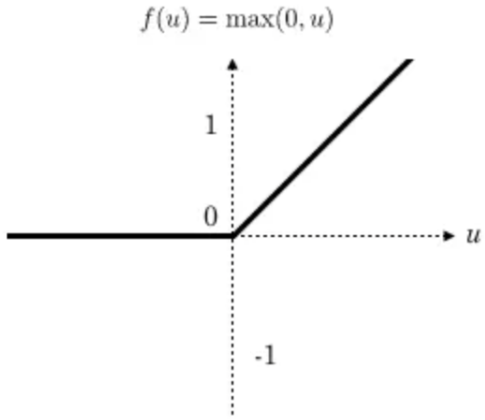


Figura 2.9: Función de activación ReLu [7].

La **función de activación softmax**[8] se utiliza en problemas de clasificación multiclase. Es una función matemática que calcula las probabilidades relativas y su función es la siguiente (ecuación 2.2):

$$Softmax(Z_i) = \frac{exp(Z_i)}{\sum_j exp(Z_j)}. \tag{2.2}$$

Usa los valores de las neuronas de la capa de salida "Z21", "Z22" y "Z23" de la figura 2.10 para determinar la probabilidad final. La exponencial hace de función no lineal. Después, estos valores se normalizan con la suma de las exponenciales para convertirlos en probabilidades que apuntan al dato de las respectivas clases.

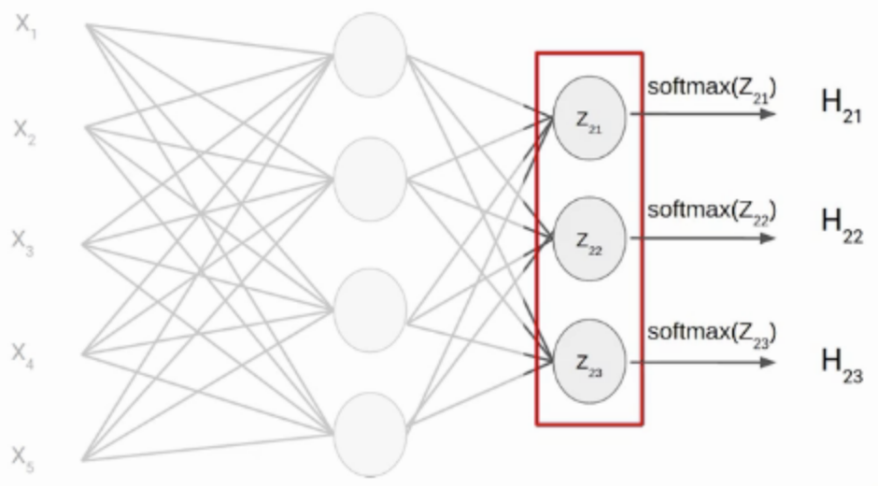


Figura 2.10: Función de activación softmax[8].

Por su lado, la **función argmax** emplea una función matemática que devuelve la lista que contiene el valor más grande.

2.3.1.3 *Pooling*

En estas **capas de reducción** (como también se les puede llamar) el objetivo es disminuir el número de neuronas antes de hacer una nueva convolución. Como hemos visto, en el inicio con la imagen blanco y negro de 28×28 píxeles tenemos la primera capa de entrada con 784 neuronas, más tarde de la primera

convolución tenemos una capa oculta con 25,088 neuronas (los 32 mapas de características de 28×28). Pues bien, si se hace una nueva convolución partiendo de esta capa, la cantidad de neuronas de la próxima capa sería muy elevado, necesitando un mayor procesamiento. Para que esto no ocurra y el tamaño de la próxima capa sea considerado, se lleva a cabo la técnica de muestreo ***subsampling***.

Esta técnica consiste en reducir el tamaño de las imágenes filtradas sin alterar las características importantes que se detectaron previamente (cada filtro). Es decir, se produce una agrupación local o global de las capas para reducir el dimensionamiento de los datos. La agrupación local actúa combinando grupos pequeños, los tamaños de 2×2 se utilizan comúnmente. Por otro lado, la agrupación global puede actuar sobre todas las neuronas del mapa de características. Se conocen dos tipos de agrupación: máximo y promedio. Como su propio nombre indica, la agrupación promedio coge el valor promedio, mientras que la agrupación máxima coge el valor correspondiente al máximo de cada grupo local de neuronas dentro del mapa de características. La técnica de *subsampling* más utilizada es ***max-pooling***.

Se denomina *pooling* a la operación de analizar el contenido de una imagen por bloques para extraer la información más relevante de ellas. El *max-pooling* reduce el número de datos entre una capa y otra, facilitando así el procesamiento de las imágenes y el entrenamiento de la red, preservando la información importante. Su funcionamiento consite en coger la imagen y dividirla en regiones de mismo tamaño, y para cada región se coge el valor correspondiente al máximo de un píxel en la imagen del resultado [32] (ver ejemplo en la figura 2.11).

Dicho numéricamente, esta técnica de muestreo de agrupación local tiene un tamaño de 2×2. Esto quiere decir que se recorre cada una de las 32 imágenes de características obtenidas anteriores de 28×28 píxeles de izquierda-derecha, arriba-abajo pero en vez de coger 1 píxel, se coge una matriz 2×2, preservando el valor más elevado de entre esos 4 píxeles. En este caso, usando 2×2, la imagen que queda es reducida a la mitad (14×14 píxeles). Finalmente de este proceso de submuestreo quedan 32 imágenes de 14×14, pasando de 25,088 neuronas a 6272.

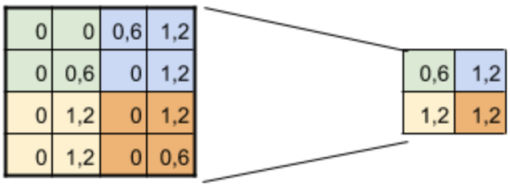


Figura 2.11: Ejemplo de *max-pooling* [5].

2.3.1.4 *Fully connected*

Estas capas se colocan al final de la arquitectura de las CNN y se conectan completamente a todas y cada una de las neuronas de salida (de ahí su nombre). Tras recibir el vector en la entrada, la capa Fully Connected (FC) aplica continuamente una combinación lineal junto con una función de activación con el fin de clasificar la imagen de inicio. Después en la salida devuelve el vector de tamaño igual al número de clases donde cada componente representa la probabilidad de que la imagen de entrada pertenezca a la clase [33].

2.3.2 Arquitectura

Después de entender qué son las redes neuronales y cómo funcionan, pasamos a analizar las arquitecturas más importantes. Como hemos dicho en ocasiones anteriores, esta arquitectura viene dada no solo por el

número de capas neuronales que se utilizan o al número de neuronas en cada una de ellas, sino también a la conexión entre neuronas o capas y a los tipos presentes. Además constituye un factor importante la forma en la que son entrenadas.

2.3.2.1 AlexNet

Se conoce que el primer trabajo que popularizó las redes convolucionales en visión artificial fue AlexNet y sus desarrolladores fueron Alex Krizhevsky, Ilya Sutskever y Geoff Hinton. Ganó el premio ImageNet en 2012.

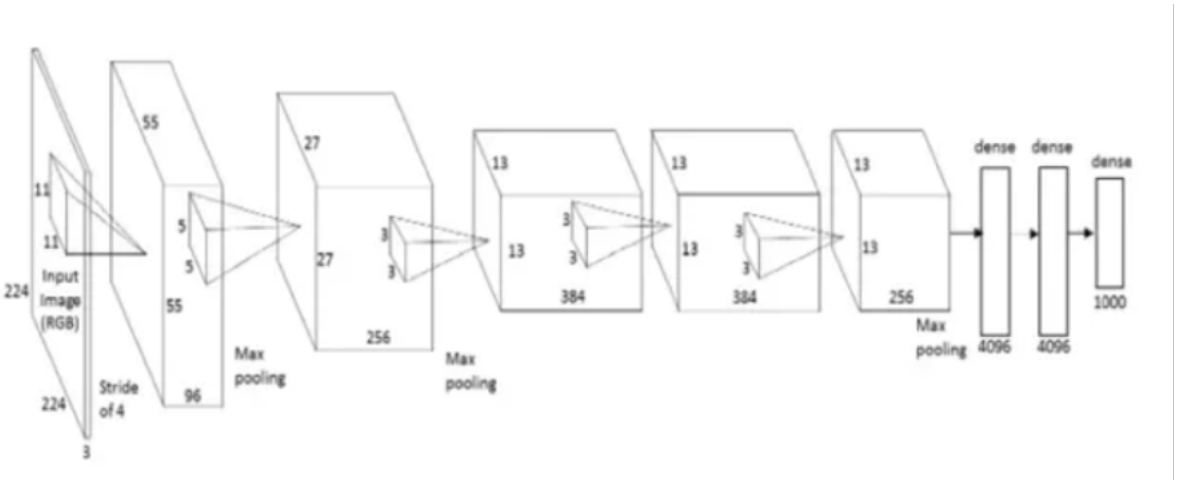


Figura 2.12: Ejemplo de arquitectura Alexnet [9].

Alexnet [34] se compone de una arquitectura profunda con 5 capas de convolución convinadas con capas de agrupación máxima como se puede ver en la figura 2.12. Sigue la siguiente estructura:

1. *Input*. Imágenes de entrada RGB de tamaño $227x227x3$.
2. Primera capa de convolución. Se compone de 96 filtros con tamaño $11x11$ y con stride valor 4. Se utiliza en esta capa la función de activación ReLu y el mapa de características de esta capa en la salida es $55x55x96$.
3. Primera capa de Maxpooling de tamaño $3x3$ y zancada 2. El mapa de características resultante es de $27x27x96$.
4. Segunda capa de convolución. Se compone de 256 filtros con tamaño $5x5$. Con zancada valor 1 y *padding* 2. Función de activación ReLu y tamaño de salida de esta capa es $27x27x256$.
5. Segunda capa de Maxpooling de tamaño $3x3$ y zancada 2. El mapa de características resultante es de $13x13x256$.
6. Tercera capa de convolución. Con 384 filtros de tamaño $3x3$. Con stride valor 1 y *padding* 1. Función de activación ReLu y tamaño de salida de esta capa es $13x13x384$.
7. Cuarta capa de convolución. Con 384 filtros de tamaño $3x3$. Con zancada valor 1 y *padding* 1. Función de activación ReLu y tamaño de salida sin cambios ($13x13x384$).
8. Quinta capa de convolución. Con 256 filtros de tamaño $3x3$. Con zancada valor 1 y *padding* 1. Función de activación ReLu y tamaño de salida $13x13x256$.

9. Tercera capa de Maxpooling de tamaño $3x3$ y zancada 2. El mapa de características resultante es de $6x6x256$.

10. Capa de abandono. Tasa de abandono de 0,5.

11. Primera capa *fully connected*. Con función de activación ReLu y tamaño de salida de 4096.

12. Capa de abandono. Tasa de abandono de 0,5.

13. Segunda capa *fully connected* igual que la anterior.

14. Tercera y última capa *fully connected* con 1000 neuronas y función de activación softmax.

AlexNet todavía está en uso como el punto de arranque para colocar redes neuronales profundas al trabajo de visión artificial o la identificación de voz.

2.3.2.2 Red Visual Graphics Group (VGG)

La red VGG fue desarrollada por expertos investigadores del Visual Graphics Group en Oxford (por ello su nombre). Estos expertos han lanzado un conjunto de modelos de CNN que comienzan con VGG, que se aplican al reconocimiento de rostros y la clasificación de imágenes, estando ahora disponibles VGG16 a VGG19 (el nombre con mayor número corresponde a la arquitectura con más avances y mejoras). El propósito original de la investigación de VGG consistía en demostrar que la profundidad de la red supone ser un componente crítico para un buen desempeño; es decir, cómo la profundidad de estas redes afecta a la precisión y la exactitud de la clasificación y el reconocimiento de imágenes a gran escala. Para profundizar el número de capas evitando demasiados parámetros, la red VGG utiliza un pequeño núcleo de convolución $3x3$ en todas ellas, y el tamaño del paso de la capa de convolución se establece en 1.

El modelo se caracteriza por tener un aspecto piramidal (como se puede ver en la imagen 2.13) que sólo requiere de un pre-procesamiento específico que consiste de restar a cada píxel el valor RGB medio que se ha calculado en el conjunto de entrenamiento.

En el entrenamiento del modelo, la entrada a la primera capa de convolución es una imagen RGB con tamaño $224x224$. El núcleo de convolución es de tamaño 3×3 para todas las capas de convolución debido a que es la dimensión más pequeña posible. Esta es una característica del modelo VGG16 ya que, muchos modelos anteriores estaban orientados a núcleos de convolución con mayores dimensiones (tamaño 11 o tamaño 5). El objetivo de estas capas es filtrar la imagen manteniendo sólo la información discriminante como pueden ser las formas geométricas atípicas.

Las capas de convolución van acompañadas de capas Max-Pooling de tamaño 2×2, con intención de reducir el tamaño de los filtros durante el entrenamiento.

En la salida de las capas de convolución y agrupación, tenemos 3 capas de neuronas completamente conectadas. Las 2 primeras compuestas por 4096 neuronas y la última por 1000 neuronas con una función de activación softmax para determinar la clase de imagen.

La arquitectura de la red VGG es clara y sencilla de entender, siendo también su punto fuerte.

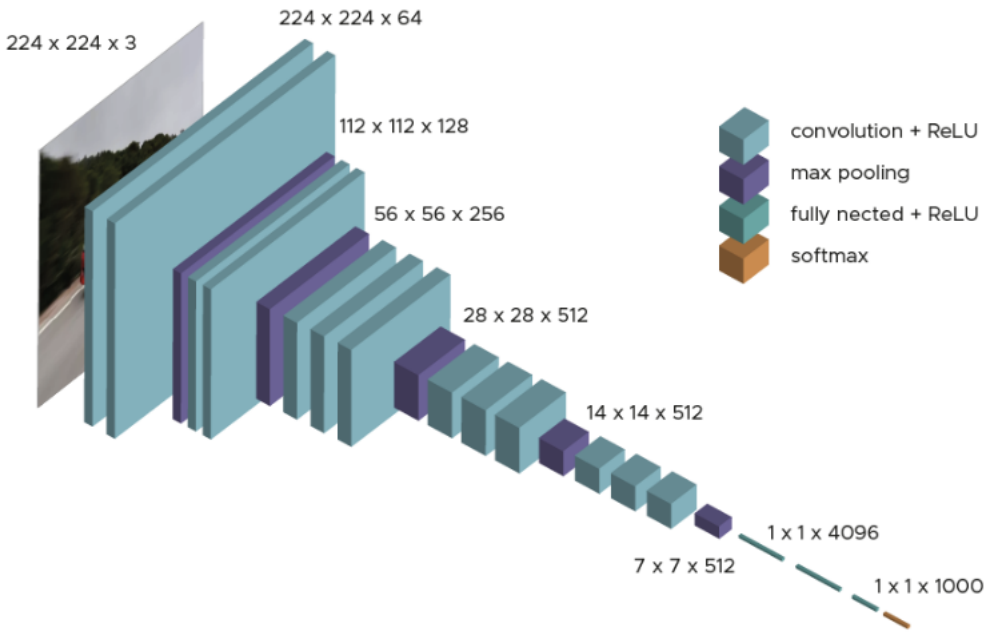


Figura 2.13: Ejemplo de arquitectura VGG16 [10].

2.3.2.3 GoogleNet

GoogleNet (o Inception Network) es un tipo de arquitectura creada por investigadores de Google. GoogleNet ganó el premio ImageNet en 2014, donde demostró ser uno de los modelos más potentes. Su principal contribución fue el desarrollo de un módulo de inicio que redujera drásticamente la cantidad de parámetros en una red (4M, en comparación con AlexNet que tiene 60M) teniendo en cuenta la eficiencia computacional. La idea es que se pueda ejecutar en dispositivos individuales incluso con pocos recursos computacionales.

Esta arquitectura abarca 22 capas de profundidad (como se aprecia en la imagen 2.14) y toma imágenes de tamaño $224x224$ con canales de color RGB. La innovación que ofrece GoogleNet es la capa de clasificador auxiliar Inception para inyectar directamente el gradiente de la función de pérdida a las capas más inferiores de la red.

La arquitectura consta de tres secciones diferenciadas [35]:

- La primera sección convolucional. Compuesta de dos capas convolucionales ordinarias seguidas por una capa de max-pooling. Utiliza la función ReLu.

- En la segunda sección se disponen sucesivamente 3 bloques de 2, 5 y 2 módulos de Inception respectivamente. Los tres bloques vienen separados entre sí a través de una capa de Maxpooling. La última capa de Incepción produce 1,024 mapas de dimensiones 7×7. Utiliza la función ReLu.

- En la sección de salida se suceden una capa de *pooling*, otra de apagado de un 40 por ciento de las neuronas y una última capa con la función de activación softmax, que es realiza la clasificación. El procedimiento de *pooling* calcula el valor medio de los 49 píxeles de los mapas producidos por el módulo anterior. Los canales quedan reducidos a dimensiones de 1×1.

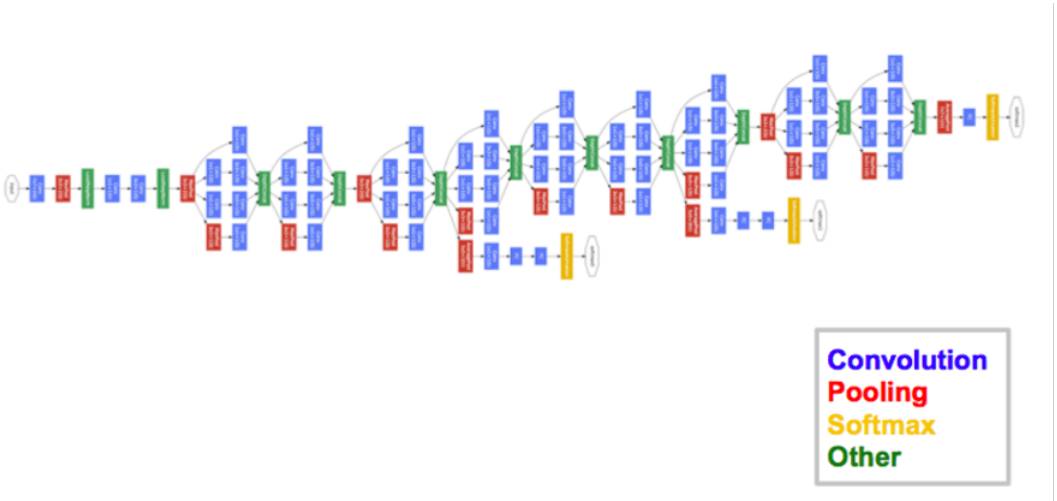


Figura 2.14: Ejemplo de arquitectura GoogleNet [9].

2.3.2.4 ResNet

ResNet define cómo de profunda puede ser una arquitectura de aprendizaje profundo. La red Residual (abreviado ResNet) tiene una serie de módulos residuales consecutivos, los cuales son la estrutura básica de la arquitectura, figura 2.15.

Para poder explicarlo en pocas palabras, el módulo restante tiene dos opciones, puede configurar la función de entrada o puede suprimir este paso totalmente. Los módulos restantes se juntan uno encima del otro formando una red directa completa. Cuenta con conexiones de salto especiales y un uso intensivo de la **normalización por lotes**.

Algunas de las nuevas tecnologías introducidas por ResNet incluyen:

- El uso del modelo Stochastic Gradient Descent (SGD) (algoritmo de optimización de redes neuronales).
- Cambios en el proceso de entrada. Se divide en primer lugar con un parche, y luego alimentando a la red.

La primordial ventaja de ResNet es que hay miles de estas capas residuales, las cuales se pueden utilizar para construir la red y luego entrenarla. Resulta ser relativamente distinto a una red secuencial regular, ya que en ella se verán mejoras de rendimiento que irán disminuyendo a la vez que se incrementa el número de capas. Actualmente, ResNet es un modelo de redes neuronales convolucionales de última generación y la mejor opción para usar ConvNets en la práctica.

Existen varias aplicaciones de Resnet dependiendo del número de capas que utiliza (101, 152 o 50): Resnet-101, Resnet-101V2, Resnet-50, Resnet-50V2, ResNet-152 y Resnet-152V2 (V2 indica que es una versión mejorada). Es por ello que para conjuntos de datos más simples, donde las clases se distinguen fácil o donde las características son más simples de identificar y clasificar, un Resnet-50 (con 50 capas) se ajustará mejor (ver figura 2.15). Resnet-50 es más pequeño, más rápido y fácil de entrenar, usar e implementar. Este modelo consta de 4 bloques de distintas dimensiones, el primero se repite 3 veces, el segundo 4 veces, el tercero 6 y el cuarto 3. Hay una capa de maxpooling entre un grupo de bloques y otro. Un Resnet-101 (con 101 capas) no solo es más grande, es más lento de entrenar, más fácil de sobreajustar y, en ocasiones, innecesario de usar. Por lo general, depende del conjunto de datos. Y, el Resnet-152 es el más antiguo de todos y casi no se utiliza.

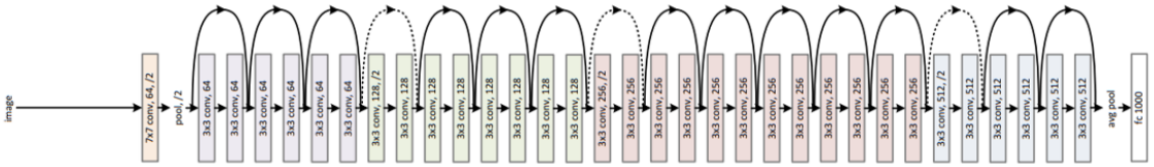


Figura 2.15: Ejemplo de arquitectura ResNet-50 [11].

2.3.2.5 SegNet

Supone una construcción de instrucción profunda para solucionar problemas de segmentación de imágenes. Consta de varias capas de procesamiento (codificador) a las que le siguen un conjunto de decodificadores para segmentación por píxeles, tal y como se puede observar en la imagen de la figura 2.16.

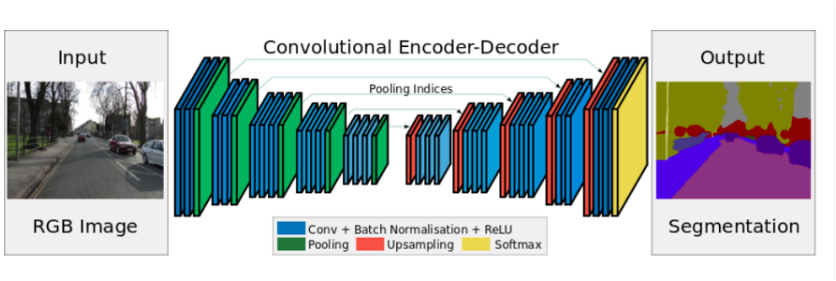


Figura 2.16: Ejemplo de arquitectura SegNet [9].

La característica principal de SegNet es que mantiene un alto nivel de frecuencia en imágenes segmentadas. Esto es debido al índice de agrupamiento de redes del codificador, los cuales están asociados con los índices de decodificadores. En definitiva, la claridad de información es directa en lugar de convolucionarlos. SegNet es uno de los mejores modelos a la hora de tratar problemas de segmentación de imágenes.

2.3.3 Algoritmo de retropropagación o *Backpropagation*

El algoritmo de reducción de error o retropropagación [36] (***Backpropagation*** en inglés) ayuda a entrenar las redes neuronales artificiales dentro del modo **supervisado**. A estos algoritmos se les conoce como los métodos de optimización basados en el gradiente descendente.

En la formación de las redes neuronales artificiales se asignan al azar los valores de los pesos ya que no se tiene conocimiento de los valores correctos. Cuando el valor de los pesos es diferente de la red de retroalimentación esperada, se considera como un error. Por tanto, este algoritmo está configurado de manera que cambie los parámetros cada vez que la salida no resulta ser la esperada. Y, por tanto, el método de retropropagación es utilizado para obtener los pesos más óptimos.

Es por ello que el error está relacionado proporcionalmente con las redes neuronales artificiales. Y es por ello que, cuando el parámetro del algoritmo cambia, el error también cambia hasta que la red neuronal encuentra la salida deseada. El **optimizador**, es el encargado de generar unos pesos cada vez mejores por tanto, su importancia es crucial. Se basa esencialmente en calcular el gradiente de por cada peso de la red. Como lo que se pretende es minimizar el error, se modifica cada peso en la dirección negativa del gradiente. A este cambio se le denomina descenso del gradiente.

Es decir, el **método del Gradiente Descendente** consiste en que cuando el algoritmo aprende del error, comienza a encontrar el mínimo local. Al alejarse negativamente del punto actual del gradiente es cuando encuentra ese mínimo local. Se le denomina gradiente al cálculo que se realiza para saber cómo ajustar los parámetros de la red para que se minimice su desviación a la salida.

De cara a agilizar la convergencia de la red hacia su mínimo, se multiplica el vector de gradiente por un factor llamado factor de entrenamiento (**lr**).

Entre los principales optimizadores disponibles en keras [37] (ver apartado 2.4.1.6) se encuentra:

- El algoritmo de gradiente adaptativo (AdaGrad). Introduce una variación muy interesante en el concepto de factor de entrenamiento. Resulta que en vez de considerar un valor uniforme para todos los pesos, lo que hace es mantener un factor de entrenamiento específico para cada uno de ellos. Es prácticamente imposible calcular este valor de forma específica. Es por ello por lo que AdaGrad lo escala el valor y lo adapta para cada dimensión con respecto al gradiente acumulado en cada una de las iteraciones.
- RMSprop (propagación cuadrática media). Mantiene un factor de entrenamiento diferente para cada dimensión, pero el escalado del factor de entrenamiento lo realiza dividiéndolo por el promedio del cuadrado de los gradientes. Dicho de otra forma, busca corregir los efectos no positivos de la acumulación global de los caches al convertirlo en un promedio movible exponencial con pesos.
- Adam (estimación del momento adaptativo). Este algoritmo combina las bondades de AdaGrad y RMSProp. Es decir, se mantiene un factor de entrenamiento por parámetro. Además de calcular RMSProp, resulta que cada factor de entrenamiento también se ve afectado por la media del impulso del gradiente.
- **AMSgrad** es una de las variantes del optimizador Adam. En lugar del promedio exponencial, utiliza el máximo de los gradientes cuadrados anteriores para actualizar parámetros.

2.3.4 Conjunto de datos: *train, test and validation*

Para la implantación de un sistema destacan 3 conjuntos de datos distintos [38]:

- Conjunto de datos de entrenamiento o ***training***. Recoge un conjunto de datos real que se emplean como muestra para entrenar y ajustar el conjut de pesos en el caso de una red neuronal).
- Conjunto **de datos de validación**. También se conoce como conjunto de desarrollo (ya que este conjunto de datos ayuda durante la etapa de desarrollo del modelo). Compone el conjunto de muestras de datos que se utiliza para proporcionar una evaluación imparcial del ajuste del modelo sobre el conjunto de datos de entrenamiento mientras se ajustan los pesos del modelo. El conjunto de datos de validación se usa para evaluar un modelo dado, pero esto es para una evaluación frecuente. También se usan para ajustar los hiperparámetros del modelo. Es por ello que, el modelo de manera ocasional ve estos datos, pero nunca aprende de ellos. Se utilizan los resultados del conjunto de validación y se actualizan los hiperparámetros de nivel superior. Entonces, el conjunto de validación afecta a un modelo, pero solo de manera indirecta. Muchas veces el conjunto de validación se emplea como conjunto de prueba, pero no es una buena práctica.
- Conjunto de datos de prueba o ***testing***. En este caso, la muestra de datos se emplea para proporcionar una evaluación imparcial de un ajuste de modelo final sobre el conjunto de datos de entrenamiento. El conjunto de datos de prueba es clave para evaluar un modelo. Solo se usa cuando

un modelo está completamente entrenado (gracias a los conjuntos de entrenamiento y validación). El conjunto de prueba, por norma general, contiene datos muestreados cuidadosamente que abarcan las diversas clases a las que se enfrentaría en un futuro el modelo.

2.3.5 Entrenamiento

2.3.5.1 Criterios de parada

El entrenamiento de una CNN que satisfaga las necesidades ya planteadas previamente sobre la clasificación de objetos dentro de una imagen, resulta ser el primer paso que hay que dar para la creación de un sistema completo. Y, a su vez, lo primero que se debe hacer antes de comenzar a entrenar la red es determinar el estado de parada que finaliza el entrenamiento. Puede haber varias razones para esto:

- Se ha alcanzado un margen de error objetivo, que puede ser bastante pequeño.
- Se ha alcanzado el número máximo de repeticiones que se han establecido como límite de entrenamiento.
- Se ha llegado al punto de saturación y por mucho que se siga entrenando, no se puede reducir más el error.

Las redes neuronales se entrenan mediante algoritmos de entrenamiento que siempre intentan encontrar los pesos de las neuronas que dan los mejores resultados [39]. Existen diferentes métodos que se pueden utilizar para llegar a obtener los pesos deseados, como buscar por todo el espacio de pesos hasta lograr encontrar los que tienen el error mínimo, realizar una búsqueda aleatoria o realizar una búsqueda dirigida.

Es decir, lo que se pretende conseguir con el entrenamiento de una red neuronal es establecer valores para el vector de pesos y que el error cometido en la evaluación sea mínimo. La modificación de los valores de los pesos de las neuronas que componen la red se calculan iterativamente mediante algoritmos de corrección del error (ver sección 2.3.3).

Cuando ya se tienen calculados estos pesos, se pasa a probar la red con otros patrones de test (***testing***) con los cuales no ha sido entrenada. Con esta nueva comprobación se busca conocer cómo se comporta la red ante entradas distintas a las usadas para el entrenamiento inicial.

Cuando se entrena una red neuronal con unos ejemplos determinados de entrenamiento (***training***) se intenta minimizar el error y se corre el riesgo de especializar "demasiado" la red. Esto es porque se consigue unos resultados muy positivos con los ejemplos con los cuales ha sido entrenado pero, para el resto con los que no se ha entrenado podemos encontrar una gran variedad de errores considerables. Todo esto lleva a la conclusión de que se deben minimizar los errores, sabiendo que la especialización conlleva pérdidas de generalización. Este concepto toma el nombre de *Overfitting* y el caso contrario es el *Underfitting*.

En el caso de que el **método de validación** falle, se debe elegir la opción de volver a entrenar la red desde cero, pero con pesos iniciales aleatorios que son diferentes a los pesos del entrenamiento anterior. Si los resultados son correctos, la red está lista y podemos garantizar que los resultados serán más o menos correctos para nuevas entradas desconocidas en la red.

Como balance final, mostrar que la fase de entrenamiento es considerada relativamente lenta y tediosa, pero no requiere reentrenamiento una vez completada. Además, estos resultados se obtienen muy rápidamente debido a que el cómputo de los resultados es lineal.

2.3.5.2 Métodos de aprendizaje

Además, las redes neuronales también se pueden clasificar según el método de aprendizaje utilizado durante el entrenamiento [40]:

- Aprendizaje supervisado. Existe un agente externo que controla todo el proceso de aprendizaje y determina para cada entrada la respuesta que debe generar. Se saben los resultados correctos a problemas y se proporcionan a la red neuronal en el periodo de entrenamiento ajustando sus pesos hasta que adaptarse a los valores. Dicho de otra forma, se pueden modificar los pesos de las conexiones para que la salida obtenida se parezca más a la deseada. El aprendizaje supervisado subdivide en:
  - Aprendizaje por corrección de error. Modifica los pesos de las conexiones en función del error cometido mediante reglas o algoritmos de corrección de error (más información en el apartado 2.3.3).
  - Aprendizaje estocástico. Realiza modificaciones aleatorios en los pesos calculando la predicción y mejorando o empeorando con cada cambio, se va quedando con los cambios con mejor resultado.
- Aprendizaje no supervisado o autosupervisado. No exite un agente externo que controla todo el proceso. Busca características, correlaciones, regularidades o categorías relacionadas entre los datos que se presenten como entrada, que se interpretan dependiendo de su estructura y algoritmo de aprendizaje. Al contrario de lo que ocurre en el aprendizaje supervisado, los resultados correctos no se conocen. El grado de similitud entre los datos se representa en la salida. La red ejecuta una compresión o asimilación de los datos por reducción dimensional o agrupamiento. El aprendizaje no supervisado subdivide en:
  - Aprendizaje hebbiano. De los datos de entrada extrae las características y mide la familiaridad.
  - Aprendizaje competitivo y comparativo. De los datos de entrada hace clasificaciones. Añade elementos a una clase, cambiando los pesos si el nuevo elemento es de esta clase, por contrario se crea una clase nueva con la característica asociada a una serie de pesos.
- Aprendizaje por refuerzo. Consiste en un método de aprendizaje automático que penaliza los comportamientos no deseados y recompensar los comportamientos deseados. Es decir, el algoritmo reajusta pesos apoyándose en un mecanismo de probabilidades [41].
- Aprendizaje semi-supervisado. Son a algoritmos que hacer uso de datos de entrenamiento etiquetados y los no etiquetados. Estos algoritmos resultan ser diferentes a los algoritmos que se utilizan en el aprendizaje supervisado ya que solo aprenden de los datos de entrenamiento etiquetados.

2.3.5.3 Entrenamiento supervisado

Para ejecutar el entrenamiento supervisado es necesario un conjunto de **datos etiquetados** con los que decir al modelo qué es lo que queremos que aprenda y en base a unas métricas, tomar decisiones en torno al ajuste de modelos. En estos modelos, se dan las **variables de entrada y salida**. Medidas como son la precisión y la exactitud dan una idea de lo preciso que es el modelo y, los parámetros de ese modelo se ajustan para aumentar estas medidas. Las bajas puntuaciones de precisión significan que necesitar mejorar.

Existen dos tipos de modelo dependiendo del tipo de etiqueta, estos son los modelos de clasificación y de regresión. Algunos de algoritmos para este tipo de entrenamiento consisten en árboles de decisión, modelos de regresión lineal o logística, *Support Vector Machine (SVM)*, *Naïve Bayes* o bosques aleatorios.

En el **método de clasificación** tiene lugar el entrenamiento a un algoritmo para la clasificación de los datos de entrada en variables discretas. Estos modelos reciben datos de entrada con una etiqueta de "clasificación" y producen como salida una etiqueta discreta, que es una etiqueta que pertenece a un conjunto finito de etiquetas posibles. Pueden tener formato binario si se clasifica entre dos clases o etiquetas, o multiclase si es el caso de tener que clasificar más de dos clases. El resultado a predecir es un atributo categórico.

Por su lado, los **modelos de regresión** son un método de aprendizaje supervisado en el que se entrena a un algoritmo para hacer una predicción de la salida de un valor real a partir de un rango continuo de valores posibles. La exactitud del algoritmo de regresión se estima calculando la variación entre la salida precisa y la salida prevista. El resultado a predecir es un atributo numérico

Para ejecutar un entrenamiento supervisado se llevan a cabo los siguientes pasos:

1. Selección de los tipos de datos de entrenamiento. En este primer paso se determina cuál es la naturaleza real de los datos que se van a utilizar para el entrenamiento.

2. Recopilar y procesado de los datos de entrenamiento. En este paso, los datos de entrenamiento se recogen de varias fuentes y se someten a una fase de procesamiento a fin de adaptarlos al formato de entrada a la red.

3. Elegir un modelo utilizando un algoritmo de aprendizaje supervisado. Según la naturaleza que se sabe de los datos de entrada y el uso deseado, se elige un algoritmo de clasificación o de regresión. La decisión de un algoritmo u otro se toma en base a la velocidad de entrenamiento, el uso que se le da a la memoria, la precisión de la predicción de los nuevos datos y la interpretación del algoritmo.

4. Entrenar el modelo. Aquí, la función de ajuste se va perfeccionando a medida que se van haciendo más iteraciones en los datos de entrenamiento. Con ello se mejora la precisión y la velocidad de predicción.

5. Realizar predicciones y evaluar el modelo. Cuando ya se tenga una función de ajuste satisfactoria, se pueden proporcionar nuevos conjuntos de datos al algoritmo y realizar así nuevas predicciones.

6. Optimizar y volver a entrenar el modelo. La degradación de datos resulta ser algo natural. Por ello, los modelos se deben volver a entrenar de forma periódica con datos actualizados para poder garantizar su precisión.

2.3.5.4 Entrenamiento no supervisado

Basándose en el aprendizaje no supervisado, el entrenamiento no supervisado emplea **datos no etiquetados**. Si no se tiene una etiqueta que predecir y no se sabe si se cuenta con patrones ocultos en los datos, se deja al algoritmo que encuentre todo lo que pueda. En estos modelos solo se proporcionarán los **datos de entrada** (no utiliza de salida). Resulta prácticamente imposible obtener una medida razonablemente objetiva de la precisión de su algoritmo. Esto es así debido a que, en comparación con el entrenamiento supervisado, los métodos de aprendizaje no supervisado no se pueden aplicar directamente a un problema de clasificación o regresión ya que no tiene idea de cuáles pueden ser los valores de los datos de salida. Se usan principalmente estos algoritmos en ocasiones donde se necesita analizar los datos con el fin de

extraer un nuevo conocimiento o para agrupar entidades por afinidad es decir, descubrir la estructura subyacente de los datos.

Si continuamos comparándolo con el entrenamiento supervisado, sabemos que el no supervisado permite realizar tareas de procesamiento más complejas. El aprendizaje sin supervisión puede ser bastante más impredecible en comparación con los otros métodos de aprendizaje.

Existen dos tipos de entrenamiento no supervisado centrados en los problemas de agrupación o asociación en su aprendizaje. Para agrupar datos por afinidad, el algoritmo que se utiliza define una métrica que le sirva para comparar los datos por valores de similitud o distancia.

Para agrupar datos por asociación, el algoritmo tiene que encontrar una estructura o patrón en una colección de datos no etiquetados. Los algoritmos de agrupamiento o *clústeres* procesan los datos y encuentran grupos o *clústeres* naturales si existen en los datos. Se puede modificar cuántos grupos deben identificar sus algoritmos. Permite ajustar la granularidad de estos grupos.

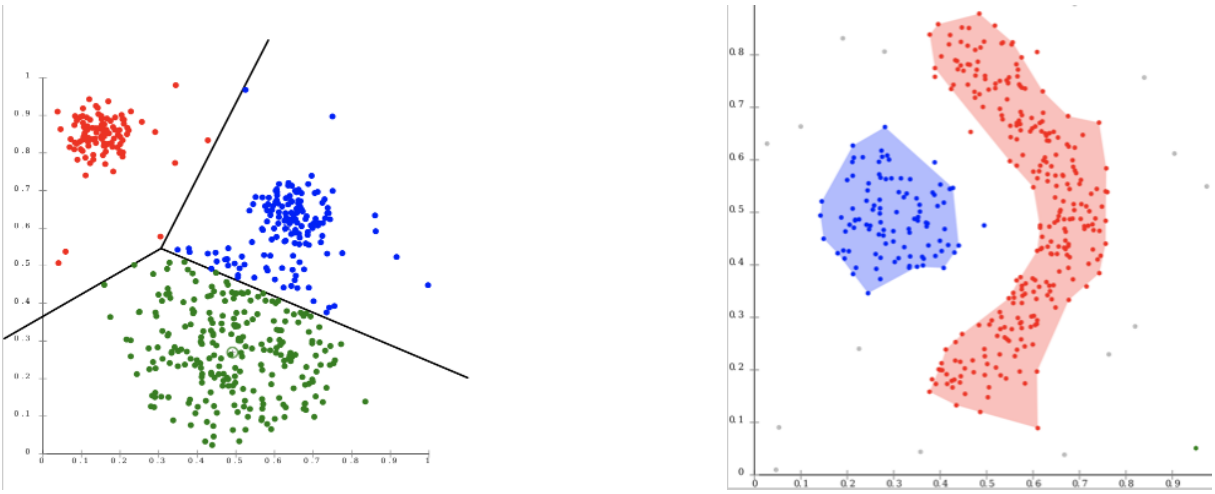
Para emplear el algoritmo de agrupamiento [42] o ***clustering*** hay que proporcionar al algoritmo un gran número de datos de entrada no etiquetados, permitiendo encontrar cualquier agrupación entre los datos que pueda. Se puede definir el concepto de agrupación como un grupo de datos que tienen características similares entre sí. Cuando se empieza con el manejo de estos datos (de los que no se sabe gran cosa), la técnica del agrupamiento debería ser un buen punto de partida para la obtención de información. Este agrupamiento se utiliza principalmente en la ingeniería de características o en el descubrimiento de patrones. Se pueden distinguir varios tipos de algoritmos de agrupamiento:

- Basado en la densidad (solapamiento). Aquí los datos se agrupan juntando las áreas de altas concentraciones de puntos de datos que se encuentran rodeadas de áreas de bajas concentraciones de puntos de datos. Es decir, el algoritmo se dedica a encontrar los lugares que son densos en puntos de datos (grupos). Los grupos pueden tener cualquier forma, no tienen limitaciones. Algunos ejemplos son Agrupamiento Espacial Basado en Densidad de Aplicaciones con Ruido (DBSCAN) (figura 2.17b) y Ordenar Puntos para Identificar la Estructura de Agrupamiento (OPTICS).

- Basado en la distribución (probabilístico). Considera que todos los puntos de datos forman parte de un grupo dependiendo de la probabilidad de que ese punto pertenezca a un grupo en concreto. De manera que hay un punto central dentro del grupo y, cuando va aumentando la distancia de un punto respecto al centro, va disminuyendo la probabilidad de que forme parte del grupo. Ejemplo: algoritmo de Mezcla Gaussiana.

- Basado en Centroides (explosivo). Podría ser probablemente la más conocida y utilizada. Sensible a los parámetros iniciales, pero rápida y eficiente. En función de múltiples centroides en los datos, separan puntos de datos. A cada punto de datos se le asigna a un grupo dependiendo de su distancia al cuadrado del centroide. Los datos se agrupan de tal manera que un dato solo puede pertenecer a un clúster o grupo. Un ejemplo es el algoritmo de agrupamiento de K-means (figura 2.17a).

- Basado en Jerarquías (aglomerativo). Resulta ser más restrictivo que el resto. Utilizado en datos jerárquicos, como los que se obtiene de una base de datos. Lo que hace es construir un árbol de grupos con el fin de organizar todo de arriba hacia abajo. Perfecto para conjuntos de datos con tipos específicos. Cada dato es un clúster y las uniones iterativas entre los dos clústeres más cercanos reducen el número de clústeres. Algunos ejemplos son el algoritmo de agrupamiento de Desplazamiento Medio, el algoritmo de agrupamiento Jerárquico Divisivo y Equilibrio Iterativo de Reducción y Agrupación mediante Jerarquías (BIRCH).



(a) K-means. (b) DBSCAN.

Figura 2.17: Ejemplos de algoritmos de agrupamiento [12].

Este tipo de entrenamiento también se aplica para reducir las dimensiones o simplificar conjuntos de datos. Las **técnicas de reducción de dimensionalidad** (técnicas matemáticas y estadísticas) tiene muchos beneficios, entre los que podemos destacar: implica algo de reducción en el espacio requerido para el almacenaje del conjunto de datos (este también se reduce); por ello, el tiempo de entrenamiento de los modelos va a ser menor para este tipo de dimensiones reducidas; y, también facilita la visualización de datos de forma más rápida gracias a la reducción de ciertas características del conjunto de datos. Aunque la principal de sus desventajas es que se puede perder algunos datos con dicha reducción. Por poner un ejemplo, la técnica de reducción de dimensionalidad más conocida es Análisis de Componentes Principales (PCA). Este método de reducción de dimensionalidad lineal permite simplificar la complejidad de espacios con múltiples dimensiones conservando su información.

2.3.5.5 Entrenamiento semi-supervisado

Este tipo de entrenamiento surge de las complicaciones de disponer de un conjunto de datos completamente etiquetado y tiene un poco de los dos anteriores.

Estos algoritmos intentan hacer uso de datos de entrenamiento etiquetados y no etiquetados. Se comienza etiquetando de forma manual algunos de los datos. Hecho esto, se consigue una pequeña porción de datos etiquetados sobre los que se entrena uno o varios algoritmos de aprendizaje supervisado. Se utilizan los modelos resultantes del entrenamiento para el etiquetado del resto de datos. Al final, se pretende entrenar un algoritmo de aprendizaje supervisado empleando como etiquetas las etiquetadas manualmente y las generadas por los modelos anteriores.

Un ejemplo de este enfoque es el algoritmo de difusión de etiquetas para predictivos de clasificación.

2.4 Lenguaje de programación Python

El Deep Learning es la inclinación en la actualidad para el progreso de Inteligencia Artificial y Python es la herramienta óptima para su desarrollo [43].

Python fue creado por el holandés Guido van Rossum en 1991, y desde entonces es uno de los lenguajes de programación más conocidos de la informática. Muchas de sus aplicaciones son utilizadas cada día,

como por ejemplo: YouTube, Instagram, Dropbox e incluso Google. Sin embargo, Python no solo es rentable para las aplicaciones web, es muy ventajoso del mismo modo, para aplicaciones de aprendizaje profundo debido a tres propiedades muy importantes: la sintaxis, las librerías, y el código abierto

En referencia a la sintaxis, al igual que el español, o cualquier otro idioma, cada lenguaje de programación tiene sus reglas particulares.

Con respecto al código abierto, Python se fundamenta en ser un lenguaje de código libre y abierto. Lo que significa que no está sometido a propagaciones y actualizaciones oficiales para promover el desarrollo. En el estudio de SlashData, muestran que Python tiene $8,2$ millones de desarrolladores, lo que significa que podemos contar con el rápido crecimiento del lenguaje.

2.4.1 Librerías Pyhton

en cuanto a las librerías las cuales, en este contexto informático son como una caja de herramientas en la que cada uno de los objetos tiene una función determinada, Python tiene muchas librerías que se enfocan en el desarrollo del Deep Learning, como por ejemplo: Matplotlib, Numpy, Cdflib , Ffmpeg, SciPy, TensorFlow, Homebrew, etc. Se comentan alguna de ellas a continuación [44].

2.4.1.1 Matplotlib

Además de permitir manipular datos y realizar algoritmos, permite realizar tareas de visualización. Entre sus cualidades destaca que es *open-source* y trabaja a bajo nivel. Está especializada en la creación de gráficos en $2D$ como diagramas de barras, sectores, caja y bigotes, violín, dispersión o puntos, líneas, áreas o de contorno, histogramas y mapas de color. El módulo se donomina: *pyplot*.

2.4.1.2 Numpy

Es una librería de Python para el aprendizaje automático que maneja el procesamiento de arrays. Comprende una gran cantidad de funciones que le autorizan para crear cálculos matemáticos difíciles de arrays multidimensionales. Implementa funciones que le permiten realizar operaciones lógicas de cálculo numérico y analizar datos (redimensionar, buscar y aplicar estadísticas) para un gran volumen de datos. La ventaja de de esta librería frente a las listas predefinidas de Python es que el procesamiento de los arrays se realiza más rápidamente (incluso 50 veces más) lo que la hace muy útil para el procesamiento de vectores y matrices de dimensiones grandes.

A causa de la eficacia y el rendimiento que brindan sus operaciones, Numpy es compatible con otras librerías de *machine learning* como TensorFlow (que usa Numpy principalmente para operaciones de tensores) o Pandas (que permite cálculos matriciales).

2.4.1.3 Ffmpeg

Es considerada la perfecta solución multiplataforma para grabar, transformar y transmitir audio y vídeo a otras fuentes. Se puede implementar en sistemas Unix y Linux aunque también en otros programas y aplicaciones web a través de la fuente.

2.4.1.4 Scipy

Consta de varios módulos que brindan funcionalidad para resolver tareas analíticas y de ciencia computacional. En estos módulos se pueden descubrir soluciones de álgebra lineal, integración, optimización,

interpolación, procesamiento de señales, procesamiento de imágenes y otros. Las estructuras de datos utilizadas por SciPy son matrices multidimensionales facilitadas por el módulo Numpy, es decir, SciPy confía en Numpy para realizar sus operaciones.

En virtud al rápido desarrollo del *machine learning*, los desarrolladores intentan crear diferentes librerías de Python para el aprendizaje automático. Este suceso llevó a los creadores de SciPy en 2001 a combinar y estandarizar varias funciones, dando como resultado esta librería.

2.4.1.5 Tensorflow

Forma parte de una de las biblioteca de código abierto más importantes para el aprendizaje profundo creada por Google. Está orientada al cálculo numérico y el aprendizaje automático a gran escala, es decir, a la computación numérica rápida. TensorFlow fue diseñada para su uso en los campos de investigación/desarrollo y en sistemas de producción,

Tensorflow permite construir y entrenar redes neuronales con el objetivo de detectar patrones y razonamientos usados por los humanos. Facilita la creación y la práctica de modelos *machine learning* mediante su Interfaz de Programación de Aplicaciones (API) que es nominalmente para el lenguaje de programación Python, aunque también valdría para C++. Su ejecución puede ser tanto en sistemas de CPU como de GPU individuales como en dispositivos sistemas distribuidos a gran escala de cientos de máquinas y móviles. Y, su instalación resulta sencilla desde un entorno Python SciPy. TensorFlow opera con Python 3.3+ [45].

2.4.1.6 Keras

Keras [46] es considerada una biblioteca de redes neuronales artificiales de código abierto. Esta biblioteca se desarrolla en el lenguaje de programación de Python y puede ejecutarse sobre diferentes plataformas como TensorFlow o Theano.

Keras está diseñada para ir construyendo por bloques la arquitectura de la red neuronal, las que incluye redes convolucionales y recurrentes, que son las que permiten, entrenar modelos *deep learning*.

2.5 Conclusiones

En este capítulo se ha tratado de recopilar toda la información necesaria para la implantación y el correcto entendimiento del sistema con el que se trabaja. Explicando todos y cada uno de las funciones.

Recoge todas las arquitecturas y tipos de entrenamientos que se utilizan en el apartado 4.2 de desarrollo.

Capítulo 3

Sistemas y bases de datos disponibles

Estar preparado es importante, saber esperarlo es aún más, pero aprovechar el momento adecuado es la clave de la vida.

Arthur Schnitzler

3.1 Introducción

Esta sección recoge todos los sistemas y bases de datos ya disponibles de estimación de pose 3D que han sido hallados tras una profunda búsqueda. La selección tanto de las bases a estudiar como de los sistemas se ha sustentado en el cumplimiento de ciertas características.

La búsqueda realizada para encontrar dichos resultado ha sido apoyada en plataformas como Google y Google Académico para encontrar artículos científicos; GitHub que es un desarrollador *software* para hospedaje de proyectos que utiliza un sistema de control de versiones Git; y *Papers with Code* que se define como un sitio web que tiene la función de organización del acceso a los documentos técnicos, también proporciona las referencias del *software* que ha sido utilizado para la reproducción del los resultados del documento.

3.2 Sistemas disponibles

Para poder valorar cada sistema de los encontrados, ha sido necesario comprobar que éste disponía tanto de datos etiquetados como de código fuente perfectamente explicado. Esto se considera una propiedad imprescindible para más adelante entender el código y poder manipularlo y hacer modificaciones.

Para la detección de los sistemas operativos llevados a estudio, debían analizarse las siguientes características:

- Lenguaje de programación (preferiblemente Python).
- Tecnología (redes neuronales)
- Las dependencias. Conjunto de librerías que necesita para ser ejecutado y sus versiones.
- Tipo de entrada de datos *RGBmonocular*, *RGBbinocular*, *Depth*, *RGB+Depth*...

- Referencias a bibliografía es decir, artículos de revistas, libros u otras páginas donde se describa el sistema o cuente la tecnología que utiliza.
- Considerar si está disponible demostración o test para poder ejecutarlo y ver si funciona. Esto es valorado como un punto importante debido a que en caso contrario habría que montar un sistema y dificultaría en gran medida la tarea.
- Comprobar posible demostración en tiempo real funcionando sobre datos de entrada de cámara RGB/depth o de fichero. Tener en cuenta también las fotogramas por segundo (FPS).
- Contabilizar las articulaciones. Comprobar también el número de *joins* necesarias para las manos y su nivel de detalle.
- ¿Es modificable el conjunto de articulaciones? ¿Requiere re-entrenamiento?
- Listado de bases de datos en las que se ha probado y ver si están disponibles.

Los sistemas más destacables que se encontraron fueron los de MMPose, StridedTransformer-Pose3D y MHFormer. El sistema de MMPose con sus dos implementaciones, VoxelPose y VideoPose3D se fundamentan en una arquitectura CNN como las que hemos explicado anteriormente. Sin embargo, StridedTransformer-Pose3D y MHFormer se componen de una arquitectura de red neuronal Transformers.

Haciendo un breve resumen, Transformers es una arquitectura que está diseñada para manipular datos secuenciales principalmente dentro del campo de procesamiento del lenguaje natural (traducción, predicción de texto y sumarización), o en el campo de las series de tiempo (*forecasting* y/o *trading*). No requiere que los datos estén ordenados es por ello que permite una mayor paralelización que las RNN.

3.2.1 MMPose

MMPose [47] es un proyecto de gran extension del que constan más de 62 colaboradores. Es una caja de herramientas de estimación de poses de código abierto basada en *PyTorch*. Parte del proyecto OpenMMLab. El lenguaje utilizado para escribir el código es *Python*. Las características clave para el desarrollo del proyecto incluyen:

1. Compatibilidad de tareas. Admite una amplia gama de tareas sobre análisis de poses del cuerpo que lideran la comunidad de investigación actual, se le incluye la estimación de pose humana en $2D$, la estimación de pose de las manos en $2D$, la detección de rasgos faciales en $2D$ y 133 posturas de cuerpo completo de puntos clave, búsqueda de mallas humanas $3D$, estimación de poses en animales...
2. Alta eficiencia y precisión. MMPose implementa varios *state-of-the-art (SOTA)* en modelos de *deep learning*, incluidos enfoques de arriba hacia abajo y de abajo hacia arriba. Logran una velocidad de entrenamiento con mayor rapidez y precisión que otras bases de código populares.
3. Compatibilidad con múltiples conjuntos de datos. La *toolbox* admite directamente varias bases de datos populares, incluidos COCO, AIC, MPII, MPII-TRB y OCHuman.
4. Buen diseño, probado y documentado. Se puede descomponer en varios componentes y combinar con varios módulos para construir fácilmente un marco de estimación de pose personalizado. Proporciona documentación detallada y referencia de API, así como diversas pruebas.

Dentro de todas las posibles opciones que nos proporciona MMPose, las más cercanas a la idea inicial y llevadas a estudio son 2: VoxelPose y VideoPose3D.

3.2.1.1 VoxelPose

Sistema creado por Hanyue Tu, Chunyu Wang y Wenjun Zeng. El objetivo principal es la estimación de pose humana en $3D$ de múltiples personas partiendo de múltiples posiciones de cámara. Otros sistemas lo que pretenden hacer es: primero hacer una estimación de pose $2D$ para cada vista de cámara mediante CNN, luego agrupar las diferentes vistas de una misma postura para por último estimar la postura en $3D$ por medio de diferentes métodos (ver figura 3.1).

VoxelPose [48] en su lugar pretende operar directamente en espacios $3D$. Para ello, se toman todas las características de las diferentes vistas de cámara con mapas térmicos de poses en $2D$ para cada una de las vistas para conseguir una codificación de la probabilidad por píxel de las articulaciones. A continuación, se proyectan los mapas térmicos en un espacio común $3D$ de red de cuboides para localizar a las personas y sus articulaciones con precisión comparando los valores con ground truth (GT) en $3D$. Hecho esto, plantea una red de regresión de poses para la estimación de pose detallada para cada propuesta.

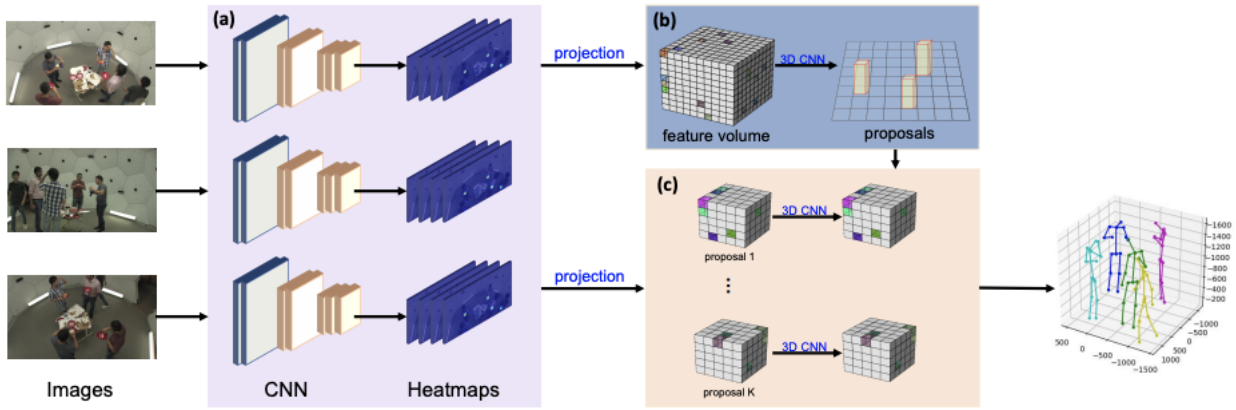


Figura 3.1: Esquema de la visión general del enfoque de VoxelPose [13].

Voxelpose ofrece diferentes opciones en diferentes entornos. En el entorno campus que cuenta con 17 *joints* y 3 cámaras, el entorno *Panoptic* con 15 *joints* y 5 cámaras y el entorno *shelf* con 17 *joints* y 5 cámaras.

3.2.1.2 VideoPose3D

En VideoPose3D se lleva a cabo una investigación de la IA por parde de *facebook*. Los autores del proyecto son: Dario Pavllo, Christoph Feichtenhofer, David Grangier y Michael Auli. En este trabajo se desarrolla la estimación de poses humanas en $3D$ con fuente de datos formato vídeo con circunvoluciones temporales y entrenamiento semisupervisado [49].

Se demuestra que las poses $3D$ en vídeo pueden estimarse de forma eficiente gracias a un modelo totalmente convolucional que se basa en convoluciones temporales dilatadas sobre puntos clave $2D$ predichos para el vídeo no etiquetado. También se introduce la retroproyección que es un método de entrenamiento semisupervisado bastante eficaz y sencillo que funciona con los datos de vídeo no etiquetados. Los trabajos más recientes han demostrado que la predicción de poses en $3D$ es más o menos sencilla si se consideran los puntos clave $2D$; y que, por tanto, la dificultad radica en la predicción de esos puntos.

Aseguran en su documentación ser un entorno que supera el mejor resultado anteriormente dado en 6 mm de error medio en la posición por articulación con la base de datos de Human3.6m, que corresponde a una reducción del error de un porcentaje de 11. También muestra mejoras utilizando la base de datos

de HumanEva-I. Cabe decir que evalúan el sistema sobre dos fuentes de datos de captura de movimiento, Human3.6M (apartado 3.3.1) y HumanEva-I (apartado 3.3.2).

Se realiza un entrenamiento con cinco de los sujetos **S1, S5, S6, S7, S8** (parte de *train*) y se prueba con dos sujetos **S9 y S11** (paste de *test*). Entrenando un único modelo para todas las acciones.

Al ser el sistema seleccionado, se explica en más detalle tanto la arquitectura como su funcionamiento en el apartado 4.2.2 de "Visión general del sistema".

3.2.2 StridedTransformer-Pose3D

StridedTransformer-Pose3D[50] se centra en el problema de que, a pesar de los grandes avances en la estimación de la pose humana 3D a partir de secuencias de vídeos RGB, continúa siendo complicado aprovechar al máximo una secuencia de poses 2D para aprender representaciones y generar una pose 3D. Este sistema fue creado por Wenhao Li, Hong Liu, Runwei Ding, Mengyuan Liu, Pichao Wang y Wenming Yang.

Sus contribuciones se resumen en:

- Propone una nueva arquitectura basada en Transformer para la estimación de la pose humana en 3D denominada Strided Transformer (ver figura 3.2). Consigue llevar de forma simple y efectiva una larga secuencia de poses 2D a una única pose 3D. Se adopta un codificador Vanilla Transformer Encoder (VTE) con objetivo de modelar las dependencias de largo alcance de las secuencias de poses en 2D.
- Para reducir la redundancia de la secuencia como también el coste computacional, se introduce el Strided Transformer Encoder (STE). En cuanto a la disminución de esta redundancia, se sustituyen las capas completamente conectadas de la red *feed-forward* del VTE por convoluciones estriadas con el fin de reducir progresivamente la longitud de la secuencia y poder agregar información de los contextos locales. La VTE modificada se llama STE y se construye sobre las salidas de la VTE. STE además reduce significativamente el coste de cálculo.
- Diseñan un esquema de supervisión completa a simple a escala de secuencia completa y de fotograma objetivo único aplicado a los resultados de VTE y STE, respectivamente. Este esquema impone restricciones adicionales de suavidad temporal junto con la supervisión de un solo fotograma objetivo y, por lo tanto, ayuda a producir poses 3D más suaves y precisas.
- Con los resultados experimentales demuestran que el método propuesto alcanza un rendimiento más avanzado que otros sistemas y con menos parámetros. Esto convierte a este método en una buena base para la estimación de la pose en 3D basada en Transformer. Los experimentos para evaluar el Strided Transformer propuesto, al igual que VideoPose3D, los realiza con dos conjuntos de datos comúnmente utilizados en estimación de la pose humana 3D, como son Human3.6m y HumanEva-I.

3.2.3 MHFormer

Los investigadores que crearon el sistema MHFormer[51] son Wenhao Li, Hong Liu, Hao Tang, Pichao Wang y Luc Van Gool. Surge del hecho de que la estimación de las poses humanas 3D a partir de vídeos monoculares RGB puede resultar algo compleja. Esto se debe a la ambigüedad de la profundidad y la auto-oclusión. La mayoría de los trabajos que existen intentan resolver ambos problemas explotando las relaciones espaciales y temporales. Sin embargo, estos trabajos ignoran que se trata de un problema en

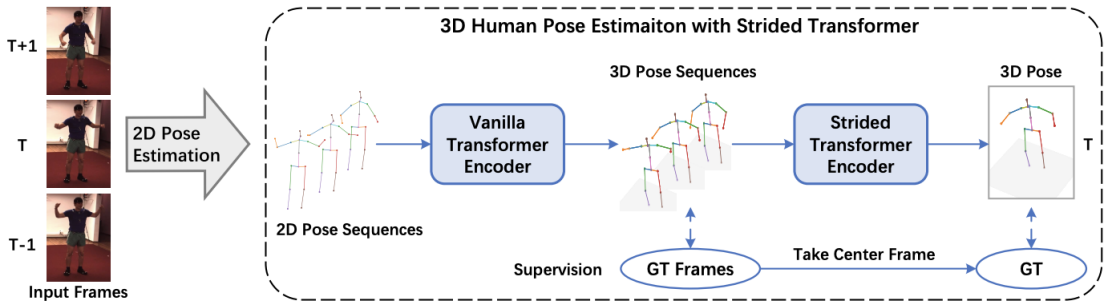


Figura 3.2: Esquema de la arquitectura utilizada en StridedTransformer-Pose3D [14].

el que existen múltiples soluciones factibles (hipótesis). Es por ello que el sistema de MHFormer propone un Multi-Hipótesis Transformer (MHFormer)[52] basado en Transformer que aprende representaciones espacio-temporales de múltiples hipótesis de pose.

Para modelar de manera eficaz las dependencias de las múltiples hipótesis y poder construir relaciones sólidas entre las características de las hipótesis, se pueden realizar en tres etapas. La primera consiste en generar representaciones iniciales de múltiples hipótesis de pose en el dominio espacial. La segunda etapa en modelar la comunicación de las auto-hipótesis, fusionar las múltiples hipótesis en una única representación convergente y luego dividirla en varias hipótesis divergentes. Y la tercera etapa vale para aprender la comunicación entre hipótesis y agregar las características de las múltiples hipótesis para sintetizar la pose 3D final.

Mediante estos procesos anteriores, la representación final de MHFormer se mejora y la pose sintetizada es mucho más precisa que otros sistemas con solo una hipótesis. Los numerosos experimentos realizados demuestran que MHFormer logra grandes resultados en dos conjuntos de datos difíciles como son Human3.6m y MPI-INF-3DHP.

Una de las limitaciones de este método es la complejidad computacional relativamente mayor. El excelente rendimiento de Transformers tiene el precio de un alto coste computacional.

3.3 Bases de datos disponibles

Reconstruir con precisión las poses humanas en $3D$ de las personas a partir de imágenes reales, en una variedad de escenarios tanto interiores como exteriores, tiene un amplio espectro de aplicaciones en el ámbito del entretenimiento, el conocimiento del entorno o la interacción entre el hombre y el ordenador.

Uno de los principales retos de los sistemas entrenables es la insuficiente cobertura de datos. En los últimos 15 años se ha logrado un progreso significativo impulsado por: nuevas optimizaciones y metodologías de modelado, métodos discriminativos, diseños de características y conjuntos de datos estandarizados para el entrenamiento de los modelos. En la actualidad, cualquier sistema de detección humana que tenga éxito, ya sea generativo, discriminativo o combinado, necesitará un componente de formación importante de entrenamiento junto con fuertes restricciones en la medida de la imagen, sobre todo en condiciones de visión monocular y (auto)oclusión.

Es muy estudiado ya que resulta un gran problema de confusión en los resultados la oclusión. Requiriendo un modelado exhaustivo de la escena, más allá de los seres humanos que aparecen en ella. Estos escenarios de comprensión de la imagen amplían la capacidad del sistema de detección de la postura para explotar el conocimiento previo y las correlaciones estructurales, utilizando la información visible incompleta para restringir las estimaciones de las partes del cuerpo no observadas.

Para la detección de las bases de datos llevadas a estudio, debían analizarse las siguientes características:

- Tipo de datos (RGB/depth).
- Resolución y FPS.
- Tamaño de la base de datos (duración, número de grabaciones/secuencias, variabilidad, etc...)

3.3.1 Human3.6m

En cuanto a su **diversidad y tamaño**, presenta un conjunto de datos a gran escala. Contiene $3,6$ millones de poses articuladas en $3D$ diferentes siendo capturadas por un conjunto de actores y actrices profesionales, en concreto 11 actores profesionales de los cuales 6 son hombres y 5 mujeres y con un índice de masa muscular (IMC) de 17 a 19. Proporcionando así gran variedad de formas corporales. Se usan 7 sujetos (3 mujeres y 4 hombres) para entrenamiento y validación, y 4 sujetos (2 mujeres y 2 hombres) para prueba. Human3.6M complementa los conjuntos de datos existentes con una gran variedad de poses humanas típicas de las personas vistas en entornos del mundo real ya que consta de 15 escenarios distintos (discusión, fumar, tomar fotos, hablar por teléfono...).

Sobre la **captura y sincronización** de imágenes precisas, sabemos que la base de datos incluye un conjunto de datos en formato vídeo de alta resolución de 50 Hz con ayuda de 4 cámaras calibradas. Proporciona datos sincronizados en $2D$ y $3D$, a los que se le incluye imagen de alta calidad y datos de captura de movimiento. Consta de modelos humanos $3D$ precisos gracias a los escaneos láser de la superficie del cuerpo de los actores. Es decir, captura las articulares $3D$ así como sus ángulos de forma precisa gracias al sistema de captura de movimiento de alta velocidad consiguiendo así, 24 etiquetas de partes del cuerpo a nivel de píxel para cada configuración.

Se caracteriza también por sus configuraciones de realidad mixta para la evaluación del rendimiento gracias a una sustracción de fondo precisa con fondos realistas, geometría de escena $3D$ y oclusión, con ayuda de cuadros delimitadores de personas.

En cuanto a su **desarrollo**, se proporcionan un *software* para visualización y predicción discriminativa de poses humanas. Este análisis incluye no sólo el vecino más cercano o los métodos de regresión lineal y no lineal estándar, sino también predictores estructurados avanzados y aproximaciones a gran escala a modelos no lineales basados en mapas de características de Fourier explícitos. Posee la capacidad de entrenar aproximaciones complejas a modelos no lineales en millones de ejemplos abre la posibilidad de desarrollar descriptores de características (descriptores de imagen precalculados) y núcleos de correlación alternativos, y probarlos sin problemas a gran escala.

Segmenta el cuerpo humano en un total de 17 articulaciones que en orden son: *'Pelvis', 'RHip', 'RKnee', 'RAnkle', 'LHip', 'LKnee', 'LAnkle', 'Spine1', 'Neck', 'Head', 'Site', 'LShoulder'* y *'LElbow [53]*. Colocadas como en la figura 3.3.

El procedimiento de calibración predeterminado del sistema es bastante simple de realizar, pero el modelo de cámara no contiene parámetros de distorsión radial ni tangencial. Dado que se esfuerzan por obtener información de pose de calidad excepcionalmente alta, han realizado un procedimiento más complejo y sólido que también se ajusta a todos estos parámetros.

Ofrece dos tipos de segmentación de datos de vídeo. Se calculan previamente a partir de datos de imágenes sin procesar para obtener resultados más precisos y están disponibles en la sección de descargas del sitio web.

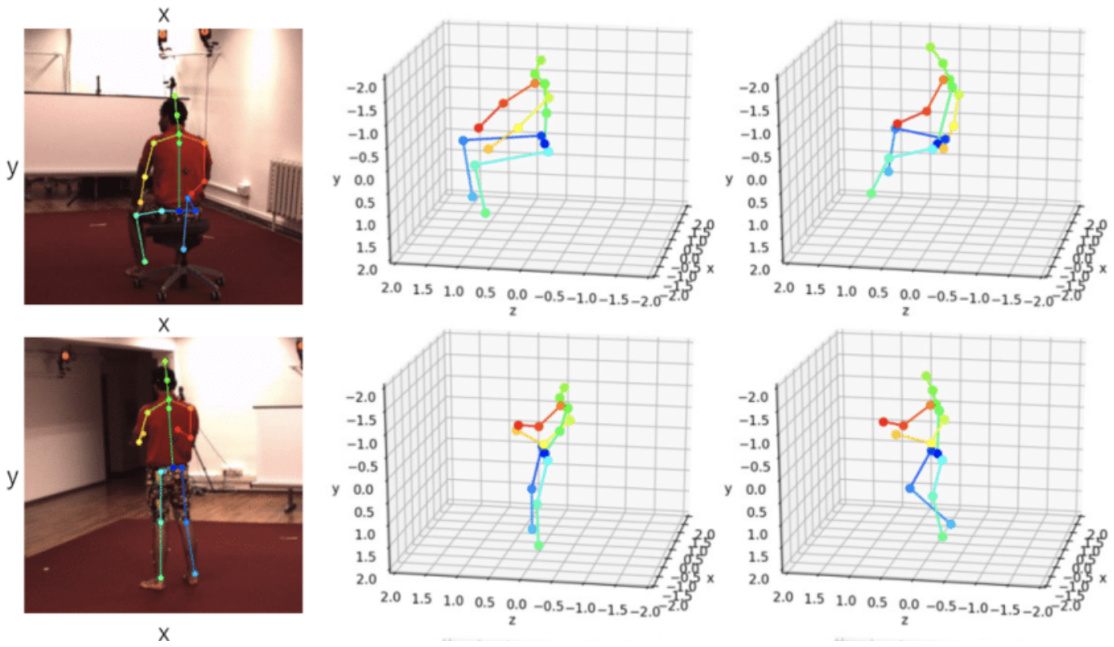


Figura 3.3: Ejemplo de imagem RGB con segmentación de las *joins* [15].

Para obtener un cuadro delimitador muy preciso, se proyecta la pose calibrada y se coloca un rectángulo alrededor de la proyección. Y, la con resta de fondo, el modelo de fondo se obtiene a partir de datos de imagen obtenidos de forma separada para este fin. Se utiliza un gráfico de corte por los bordes para obtener el resultado final de la resta de fondo. El recorte de las etiquetas o *ground truth* se realiza solo en el cuadro delimitador.

En cuanto a la función precalculada, para ambas segmentaciones (cuadro delimitador y resta de fondo), se proporciona HoG (histograma de gradientes orientados) piramidales precalculados que cuenta con diferentes parámetros de extracción en contornos únicos y con bordes internos.

Este conjunto de datos completo ofrece importantes ventajas de rendimiento en comparación con conjuntos de datos equivalentes más pequeños, ofreciendo también un importante margen de mejora. Los datos, los modelos de estructura a gran escala, los descriptores de imagen, así como las herramientas de visualización y de evaluación del software que desarrollan están disponibles de manera gratuita [54].

La base de datos Human3.6m [55] y sus herramientas asociadas quieren estimular la investigación en el campo de la visión por ordenador y el aprendizaje automático, y ayudar en el desarrollo de sistemas mejorados de detección humana en $3D$ que puedan funcionar con solidez en el mundo real.

3.3.2 HumanEva-I

HumanEva-I pretende impulsar a la comunidad de estimación de poses y movimiento humano, un conjunto de datos de desarrollo integral estructurado que posee un código de soporte y métricas de evaluación cuantitativa.

Esta base de datos la compone un conjunto de datos mucho más pequeño que Human3.6M. Consta de 7 secuencias de vídeo que están calibradas 4 en escala de grises y 3 en color y se sincronizan con poses corporales en $3D$ que se obtienen de un sistema de captura de movimiento. Contiene un total de 4 sujetos grabados a partir de 3 vistas de cámara a 60 Hz. Estos sujetos realizan 6 acciones comunes como por ejemplo, caminar, trotar o gesticular y supone, para cada una de ellas, e, entrenamiento de

un modelo diferente. El esqueleto se compone de 15 articulaciones y utiliza la división de entrenamiento proporcionada.

El conjunto de datos HumanEva-I contiene. Las métricas de error para calcular el error en pose $2D$ y $3D$ se proporcionan a los participantes. El conjunto de datos contiene conjuntos de entrenamiento, validación y prueba (con datos reales ocultos).

Las descargas tanto de esta base de datos como de su versión avanzada se puede hacer directamente desde su página web oficial [56].

3.3.3 *ETRI-Activity3D*

A pesar de la gran cantidad de conjuntos de datos que se encuentran disponibles de forma publica, existe una falta de datos enfocados a que los robots reconozcan las actividades diarias de las personas. Es decir, la mayor parte de los conjuntos de datos no contempla el entorno robótico en que la sociedad se encuentra hoy en día donde conviven los humanos y los robots siendo esto un gran impedimento para las investigaciones que engloban toda la inteligencia robótica.

La base de datos *ETRI-Activity3D* [57] surge de la inexistencia de conjuntos de datos visuales centrados en robots y humanos. Y, es por ello que se centra en el comportamiento cotidiano de las personas más mayores para robots de atención primaria.

Esta base de datos se crea a partir de un conjunto de datos RGB-D a gran escala para que los robots reconozcan las actividades diarias de los ancianos como se aprecia en la figura 3.4.

Este conjunto de datos es recopilado por el sensor *Kinect v2* y consta de tres métodos de datos sincronizados: video RGB (con resolución de 1920×1080), mapa de profundidad (se almacenan cuadro por cuadro en 512×424) e instantánea de esqueleto (ubicadas 25 articulaciones corporales en $3D$). Para grabar datos visuales, se juntaron un total de 50 personas mayores en el rango de edades de 64 a 88 años recopilando hasta 55 clases de acción distintas. También, se obtuvo un conjunto de datos de 50 sujetos más jóvenes (20 años) de la misma forma que el de las personas mayores. En total tienen 112,620 conjuntos de datos en $3D$.

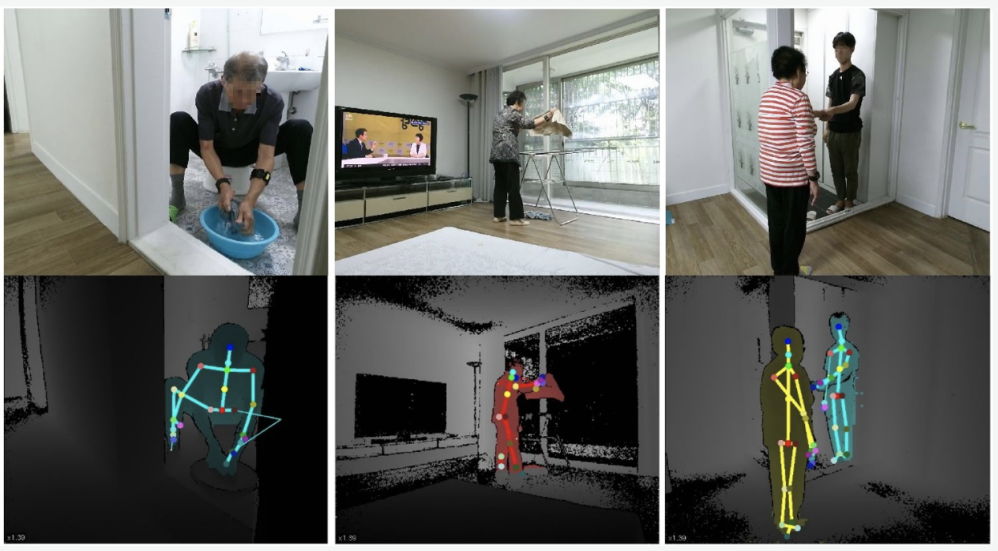


Figura 3.4: Ejemplo base de datos ETRI . Imágenes RGB con las *joins* [16].

Se puede acceder al conjunto de datos directamente desde la web oficial [16].

3.3.4 MPI-INF-3DHP

Es un conjunto de datos de estimación de pose del cuerpo humano en $3D$ [17]. Recoge un gran número de escenas interiores restringidas y escenas exteriores complejas. En las grabaciones aparecen 8 actores realizando hasta 8 actividades diferentes desde 14 vistas de cámara. Compone un total de más de $1,3$ millones de fotogramas capturados entre las 14 cámaras.

Las imágenes son captadas a través de cámaras RGB en 3 escenarios distintos (ver figura 3.5) que corresponden con una habitación de paredes color gris, en una habitación forrada de color verde y en escenas en la naturaleza.

(a) Habitación forrada color verde..

(b) Habitación paredes color gris.

(c) Escenas en la naturaleza.

Figura 3.5: Ejemplos imágenes RGB en los 3 escenarios posibles.

Dada una imagen RGB, se estima la pose $3D$ en el sistema de coordenadas de la cámara. Después se estiman las posiciones globales de las articulaciones del esqueleto representado en la figura3.6 teniendo en cuenta el punto de vista de la cámara, que va más allá de solo estimar en un sistema de coordenadas centrado en la raíz (pelvis).

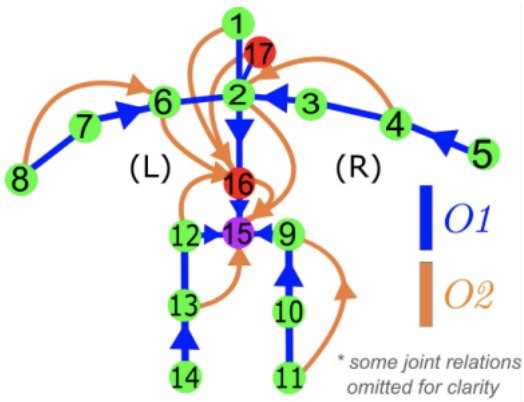


Figura 3.6: Representación de las 17 articulaciones segmentadas de la base de datos MMPI-INF-3DHP [17].

El algoritmo para llevar a cabo el proceso consta de tres pasos. Primero el sujeto se localiza en el marco $2D$ calculado a partir de mapas de calor conjuntos $2D$ obtenido con una CNN llamada 2DPoseNet. Segundo, para encontrar la pose $3D$ centrada en la raíz de una imagen RGB recortada se calcula con una segunda CNN denominada 3DPoseNet. Y tercero, coordenadas de pose $3D$ globales y la corrección de perspectiva se calculan usando la pose $3D$, las ubicaciones de *joins* $2D$ y la calibración de cámara conocida (la red se deriva a Resnet-101).

La base de datos se puede descargar en la siguiente dirección: [58].

3.3.5 Leeds Sports Pose Dataset (LSP)

Esta base de datos [59] contiene un total de 2000 imágenes anotadas que con diferentes poses de personas, en su mayoría deportistas, recopiladas de Flickr[1]. Usa las etiquetas con el nombre del deporte. Las imágenes se han escalado hasta que la persona más destacada tiene una longitud de aproximadamente 150 píxeles. Cada imagen está anotada con 14 ubicaciones conjuntas. Desde un punto de vista centrado en la persona se etiquetan constantemente las articulaciones izquierda y derecha.

Consta de 14 *joins* etiquetadas que en orden son: tobillo derecho, rodilla derecha, cadera derecha, cadera izquierda, rodilla izquierda, tobillo izquierdo, muñeca derecha, codo derecho, hombro derecho, hombro izquierdo, codo izquierdo, muñeca izquierda, cuello y parte superior de la cabeza.

Protocolo experimental que utiliza para el etiquetado viene recogido en el documento de BMVC [60]. El proceso fue el siguiente, primeramente el conjunto de datos se dividió en dos partes para entrenamiento y prueba. Las primeras 1000 imágenes (im0001.jpg a im1000.jpg) se usaron para entrenamiento y selección estricta de todos los parámetros. Las segundas 1000 imágenes (im1001.jpg a im2000.jpg) fueron utilizadas para las pruebas. El método de evaluación utilizado corresponde a la medida del porcentaje de piezas correctas (los puntos finales estimados deben estar dentro del 50 % de la longitud desde los puntos finales de la pieza reales).

Las atribuciones y las *URL* de la página web de Flickr para las imágenes originales se pueden encontrar en el campo de comentarios JPEG de cada archivo de imagen.

3.4 Conclusiones

Tras la evaluación y estudio de los sistemas propuestos (y más), se llegó a la conclusión de que el sistema seleccionado finalmente para la realización de este trabajo fuera el de VideoPose3D.

Para la base de datos con la que se comienza la el proceso de desarrollo del trabajo es Human3.6m.

Una vez hecha esta selección, los pasos siguientes serán poner en marcha el sistema buscado y tratar de comprender el programa en Python que se ofrece.

[1] Flickr es una web de Internet que permite buscar, compartir, almacenar, vender y ordenar fotografías o vídeos en línea

Capítulo 4

Desarrollo

Mamá dice que la vida es como una caja de bombones,
nunca sabes el que te va a tocar.

Forrest Gump

4.1 Introducción

Este capítulo documenta el desarrollo completo del proyecto. Los primeros pasos con el sistema operativo que se emplea, las aplicaciones utilizadas y lenguaje de programación.

Describe todo el funcionamiento del sistema de VideoPose3D, así como las arquitecturas en la que se fundamenta el proyecto y que tienen como objetivo resolver los problemas planteados. También se cuenta en detalle las funciones de activación, la configuración experimental, etc. Recoge una aportación de la base de datos empleada de Human3.6m.

Además, se expresan las propuestas para implementar tras la puesta en marcha del sistema: La modificación y mejora del sistema; y la evaluación de resultados del sistema ante diversos campios en el esquema de entrenamiento utilizado.

4.2 Generación del trabajo

El trabajo se ha llevado a cabo con un ordenador con sistema operativo GNU/Linux proporcionado por el Departamento de Electrónica de la Universidad de Alcalá, con posibilidad de acceso remoto. Esto ha sido posible utilizando el programa de Anidesk que es un software de alto rendimiento que permite compartir el escritorio gozando de un control completo sobre él. Asegura una rápida y segura transmisión de los datos entre los dispositivos y, lo que es más requerido por los usuarios, sin tiempos de latencia. Ha requerido para la puesta en marcha del sistema llevar a cabo descargas de todas las herramientas necesarias como Pycharm, Anaconda, bases de datos y librerías propias.

Para la implantación del código del programa ha sido imprescindible el uso del lenguaje de programación *Python* así como varias de sus librerías externas. Resulta inevitable el uso de estas librerías para el correcto funcionamiento de las tareas de lectura, escritura y ejecución de todo el sistema de *scripts* de *Pyhton*. Las más detacables se encuentran recogidas en el apartado 2.4.1.

Desde la terminal de Linux se procedió a instalar la distribución Anaconda para la gestión de paquetes Python y crear así el entorno virtual desde el que se va a trabajar. El entorno virtual se instala con la versión 3,9 de Python.

Para guardarnos los archivos y código correspondiente con el sistema a implementar (en nuestro caso Videopose3D [49]) y se clona su repositorio Git.

PyCharm es un entorno de desarrollo integrado (IDE) para la programación con Python. Proporciona autocompletado inteligente del código, acceso a línea de comandos, creación entornos virtuales, conexiones a bases de datos y gestión del sistema de control de versiones. En este entorno de desarrollo se crea un nuevo proyecto con acceso al entorno virtual previamente creado.

4.2.1 Selección del sistema

Para la selección del sistema con el que se va a experimentar se ha llevado a cabo un estudio minucioso de todos los sistemas propuestos en el apartado 3.2. Considerando las preferencias basadas en: el tipo de entrada de datos se encuentre en formato vídeo, anteponiendo los grabados con cámaras *red, green, blue (RGB)*); lenguaje de programación empleado Python con dependencias compatibles; con posible demostración en tiempo real; valorar el número de articulaciones segmentadas, considerando razonable un total de entre 15 y 20 y comprobar si hay posibilidades de modificar el número del conjunto de articulaciones a través del código; si el sistema requiere ser re-entrenado y, comprobar si las bases de datos en las que ha sido probado existen y están disponibles.

Dicho esto, el sistema seleccionado corresponde al proyecto **VideoPose3D**.

4.2.2 Visión general del sistema VideoPose3D

El proyecto de VideoPose3D trata de abordar 2 problemas. El primero de ellos consiste en hacer una estimación eficiente de la pose humana en $3D$ en vídeo utilizando trayectorias de puntos clave en $2D$. Y, el segundo de los problemas es el escenario donde los datos de entrenamiento etiquetados son bastante escasos a pesar de los numerosos proyectos de investigación basados en sistemas con redes neuronales equipados con arquitecturas precisas y eficientes que existen en la actualidad.

La solución para abordar el primero de los problemas de **detección de puntos clave en** $2D$ seguida de estimación de pose $3D$, se basa en un enfoque de un método de última generación. Se centra en la existencia del error de asignar el mismo punto clave $2D$ a múltiples poses $3D$. Para resolver esta ambigüedad en vez de usar las RNN, como hacían previamente en otros proyectos para modelar información temporal; se utiliza un enfoque convolucional con ayuda de las CNN para realizar una convolución dilatada en 1D a lo largo del tiempo y cubrir un gran campo receptivo. Ofrece mayor precisión y eficiencia en términos de complejidad computacional y simplicidad en el número de parámetros en comparación con los enfoques basados en RNN. Además, los modelos convolucionales nos permiten procesar múltiples tramas en paralelo, lo que no es posible con redes recurrentes.

Por otro lado, para abordar el segundo problema expuesto se conoce que el **aprendizaje semi-supervisado con retroproyección** tiene una arquitectura altamente precisa y eficaz capaz de abordar entornos que carecen de datos de entrenamiento etiquetados. Así se aprovechan los datos de vídeo sin etiquetar para el entrenamiento semi-supervisado. La configuración de modelos de redes neuronales sobre un volumen de recursos bajo propone ser un gran desafío. Y es que, para poder realizar un entrenamiento razonablemente bueno para la estimación de la pose humana en $3D$, se requiere de grandes cantidades de datos etiquetados y etiquetas, así como de rigurosas configuraciones de captura de movimiento costosas con largas sesiones de grabación.

Es por ello, el modelo que plantea VideoPose3D está inspirado en "la traducción automática no supervisada de textos" que consiste en que las oraciones que están disponibles solo en un idioma se traducen a otro y luego se regresan al idioma original. Haciendo una similitud con esto, el sistema planteado predice puntos clave $2D$ en vídeos sin etiquetar utilizando un detector de puntos clave $2D$, predice las poses en $3D$ para luego mapearlas a un espacio $2D$. En otras palabras, la idea clave es utilizar un estimador de pose $3D$ como codificador para resolver el **problema de codificación automática de datos sin etiquetar**, asignar la pose predicha al espacio $2D$ y calcular las pérdidas de reconstrucción en función de eso. Debido a la proyección en perspectiva de los vídeos, la pose $2D$ en la pantalla depende tanto de la trayectoria (es decir, la posición del marco humano en el espacio en cada paso de tiempo) como de la pose $3D$ (posición de las articulaciones en el marco humano). Por lo tanto, para que la retroproyección a $2D$ se pueda realizar, se hace una regresión de la trayectoria $3D$ de la persona prediciendo una pose $3D$ aplanada.

4.2.3 Formato de la Red: Modelo convolucional temporal dilatado

Para tratar de solventar el primero de los problemas que se proponen, se usa un modelo con una arquitectura totalmente convolucional (CNN) con conexiones residuales. Es decir, el formato de la red es una red residual **ResNet** (se explica en detalle su funcionamiento y arquitectura en la sección 2.3.4 del trabajo). Esta red toma una secuencia de poses en $2D$ como entrada y las transforma por medio de convoluciones temporales.

Los modelos convolucionales de redes neuronales permiten la paralelización en el tiempo cosa que las RNN no. Es beneficioso para la estimación de poses $3D$ el control sobre el campo receptivo temporal y poder modelar las dependencias temporales. En las (CNN) la ruta del gradiente entre salida y entrada no depende de la longitud de la secuencia sino que es de longitud fija. Es más, se emplean circunvoluciones dilatadas para mantener la eficiencia y modelar dependencias a largo plazo. Las arquitecturas que tienen circunvoluciones dilatadas se emplean para la generación de audio, la segmentación semántica y la traducción automática.

La descripción de la **arquitectura** empleada es la siguiente. La capa de entrada coge las coordenadas concatenadas (x, y) que pertenecen a las articulaciones J para cada uno de los fotogramas y emplea una convolución temporal con un tamaño de núcleo (o *kernel*) W y C canales de salida. A esto le sigue los B bloques con la estructura de tipo ResNet que están rodeados por una conexión de salto. Cada bloque primero realiza una convolución 1D con un tamaño de *kernel* W y un factor de dilatación que corresponde a la fórmula de $D = W^B$, seguido de una convolución con un tamaño de núcleo 1. A las convoluciones (excepto la última capa) le sigue una **normalización por lotes**, unidades lineales rectificadas y abandono. Cada bloque aumenta el campo receptivo de forma exponencial por un factor W, mientras que el número de parámetros aumenta linealmente. Los parámetros del filtro W y D se establecen de forma que el campo receptivo para cualquier fotograma de salida forme un árbol que cubra todos los fotogramas de entrada.

Finalmente, la última capa realiza una predicción de las poses $3D$ para todos los fotogramas de la secuencia de entrada utilizando datos pasados y futuros para explotar la información temporal. Para evaluar los escenarios en tiempo real también se experimenta con **convoluciones causales**, es decir, convoluciones que sólo tienen acceso a fotogramas pasados. Los modelos convolucionales de imágenes suelen aplicar el *zeropadding* (esquemas de relleno) para obtener tantas salidas como entradas. Sin embargo, los primeros experimentos mostraron mejores resultados al realizar sólo convoluciones sin relleno mientras se rellena la secuencia de entrada con réplicas de los fotogramas límite a la izquierda y a la derecha.

Para continuar con la explicación de la arquitectura empleada por VideoPose3D, nos apoyamos en la figura 4.1. La entrada consiste en puntos clave $2D$ para un campo recpetivo de 243 cuadros ($B = 4$ bloques) con $J = 17$ articulaciones. Las capas convolucionales están en verde, donde $2J$, $3d1$, 1024 denota 2 articulaciones de entrada, *kernel* de tamaño 3 y dilatación 1, y 1024 canales de salida. Los tamaños de los tensores entre paréntesis para un ejemplo de predicción de 1 fotograma, donde (243, 34) denota 243 fotogramas y 34 canales. Debido a las convoluciones válidas, se cortan los residuos (a la izquierda y a la derecha, simétricamente) para que coincidan con la forma de los tensores posteriores.

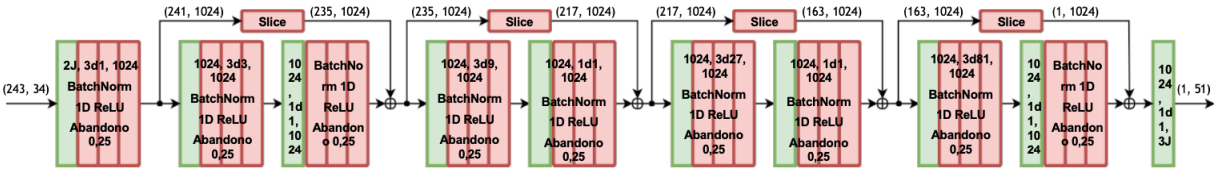


Figura 4.1: Instancia de la arquitectura utilizada por VideoPose3D para un tamaño del campo receptivo de 243 fotogramas ($B = 4$ bloques). Para capas convolucionales, fijando $W = 3$ con $C = 1024$ canales de salida y una tasa de abandono $p = 0,25$[18].

Para el segundo de los problemas planteados, se introduce un método de **entrenamiento semi-supervisado** para mejorar la precisión en entornos en los que la disponibilidad de datos de pose $3D$ etiquetados es limitada. Se aprovecha el vídeo sin etiquetar en combinación con un detector de puntos clave $2D$ para ampliar la función de pérdida supervisada con un término de pérdida de retroproyección. Se resuelve el problema de autocodificación en datos no etiquetados: el codificador (estimador de pose) realiza una estimación de pose $3D$ a partir de coordenadas conjuntas $2D$ y el decodificador proyecta la pose $3D$ de nuevo a coordenadas conjuntas $2D$. El entrenamiento penaliza cuando las coordenadas conjuntas $2D$ del decodificador se alejan de la entrada original.

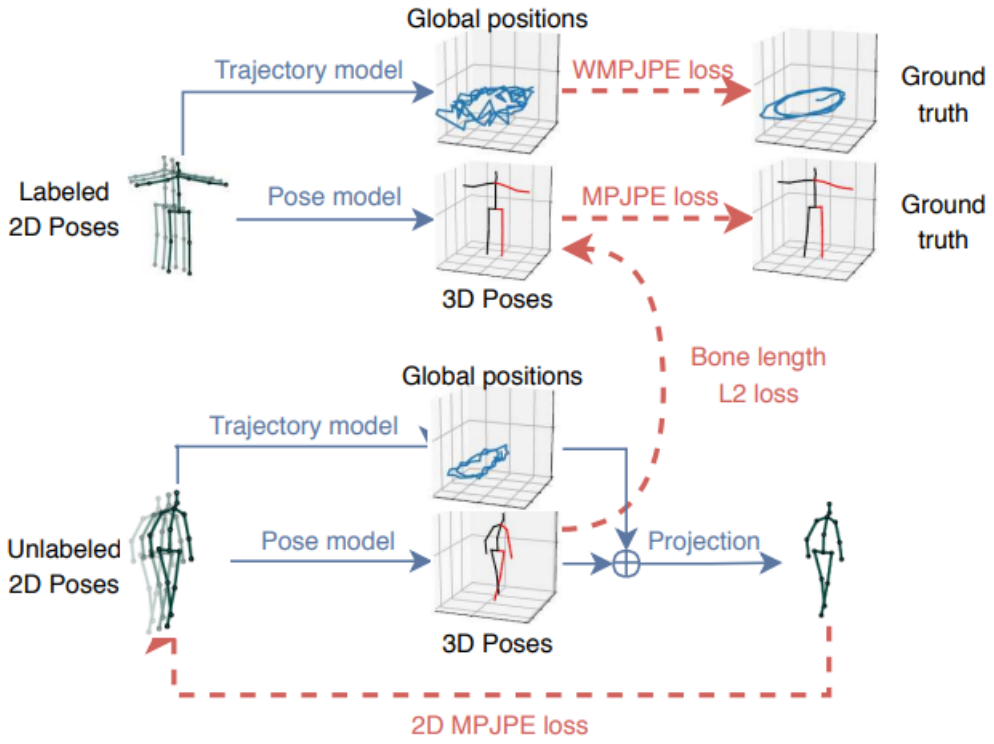


Figura 4.2: Entrenamiento semi-supervisado con un modelo de pose $3D$ que toma como entrada una secuencia de poses $2D$ posiblemente predichas. Se hace una regresión de la trayectoria $3D$ de la persona y se añade una restricción para hacer coincidir las longitudes medias de los huesos de las predicciones re-etiquetadas con las etiquetadas. Todo se entrena conjuntamente. Error medio de posición por articulación (MPJPE)[18].

La figura 4.2 representa el método que combina el componente supervisado con el componente no supervisado (del proyecto), que actúa como regularizador. Los dos objetivos se optimizan conjuntamente, ocupando los datos etiquetados la primera mitad de un lote, y los datos no etiquetados la segunda mitad. Para los datos etiquetados se utilizan las poses $3D$ reales como objetivo y entrenan una pérdida supervisada. Los datos sin etiquetar se utilizan para implementar una pérdida de codificador automático donde las poses $3D$ predichas se proyectan de nuevo a $2D$ y luego se verifica la coherencia con la entrada.

Para el **modelo de trayectoria**. Debido a la proyección en perspectiva, la pose $2D$ en la pantalla depende tanto de la trayectoria (es decir, la posición global de la articulación raíz humana) como de la pose $3D$ (la posición de todas las articulaciones con respecto a la articulación raíz). Sin la posición global, el sujeto se reproyectaría siempre en el centro de la pantalla con una escala fija. Por lo tanto, también se regresa a la trayectoria $3D$ de la persona, para que la retroproyección a $2D$ pueda realizarse correctamente. Para ello, se optimiza una segunda red que hace una regresión de la trayectoria global en el espacio de la cámara. Esta última se añade a la pose antes de volver a proyectarla en $2D$. Las dos redes tienen la misma arquitectura, pero no comparten pesos, ya que se observa que se afectan mutuamente de forma negativa cuando se entrenan de forma multitarea. Como resulta cada vez más difícil recuperar una trayectoria precisa si el sujeto se aleja de la cámara, se emplea una función de pérdida de error de posición media ponderada por articulación MPJPE para la trayectoria:

$$E = \frac{1}{y_z}\|f(x) - y\|$$

Es decir, se pondera cada muestra utilizando la inversa de la profundidad real y_z en el espacio de la cámara. La regresión de una trayectoria precisa para los sujetos lejanos también es innecesaria para

nuestros fines, ya que los puntos clave $2D$ correspondientes tienden a concentrarse alrededor de un área pequeña.

En los experimentos se utiliza la arquitectura descrita en figura 4.2 para mapear poses $2D$ a $3D$. Para proyectar poses $3D$ a $2D$, se utiliza una capa de proyección simple que considera parámetros lineales (distancia focal, punto principal) así como coeficientes de distorsión de lente no lineales (tangencial y radial). Se ha comprobado que las distorsiones de lente de las cámaras utilizadas en la base de datos Human3.6M tienen un impacto insignificante en la métrica de estimación de la pose, pero se incluyen estos términos no obstante porque siempre proporcionan un modelado más preciso de la proyección de la cámara real.

4.2.4 Base de datos seleccionada

VideoPose3D propone la ejecución del sistema sobre dos conjuntos o bases de datos distintas que son Human3.6M y HumanEva-I. Ambas se encuentran explicadas y comentadas en el apartado 3.2.1.2.

Sin embargo, par este proyecto, se ha tomado la elección de continuar con la evaluación del sistema utilizando la base de datos de Human3.6M ya que se adapta mejor a la idea inicial del proyecto.

En resumen, base de datos Human3.6M está compuesta por 3,6 millones de fotogramas que toman formato vídeo. De los 11 sujetos 7 están anotados con poses en $3D$. Cada uno de los sujetos realiza 15 acciones que se graban con 4 cámaras sincronizadas a 50 Hz. El esqueleto completo consta de 17 articulaciones. Se realiza un entrenamiento con cinco de los sujetos **S1, S5, S6, S7, S8** (parte de *train*) y se prueba con dos sujetos **S9 y S11** (parte de *test*). Se entrena un único modelo para todas las acciones.

4.3 Configuración experimental

Tras clonar el repositorio completo de GitHub de VideoPose3D se dispone de toda la documentación necesaria para ejecutar el sistema.

Como ya se ha mencionado, VideoPose3D desarrolla su sistema en dos bases de datos Human3.6M y HumanEva-I. Como este proyecto se construye con Human3.6m, se centra principalmente en configuración con esta base de datos.

Lo que se pretende hacer es **probar la evaluación con los modelos preentrenados** de los que se dispone. Para ello, lo primero es **configurar la base de datos**. Se extraen los archivos nombrados *Poses D3 Positions S X.tgz*(temas 1, 5, 6, 7, 8, 9, 11) (en su interior están los .cdf que se necesitan) de la página wed oficial de Human3.6m a un directorio común. A este conjunto de datos se le ejecuta un *script* de preprocesamiento. Este script convierte el conjunto de datos de la fuente original, utilizando archivos .cdf originales y crea los archivos de pose $2D$ (GT).

Para la **implementación de la estimación de la pose en** $2D$ muchos de los trabajos anteriores a VideoPose3D extraen el sujeto de unos cuadros delimitadores de GT y luego aplican un detector para predecir las ubicaciones de los puntos clave $2D$ dentro del cuadro delimitador del GT. Sin embargo, el enfoque del proyecto VideoPose3D no depende de ningún detector de puntos clave $2D$ particular. Por lo tanto, se investigan varios detectores $2D$ que no dependen de los recuadros de GT, permitiendo el uso de la configuración en la naturaleza.

Además del detector de reloj de arena apilado, se investiga sobre la *Mask R-CNN* que es un algoritmo de segmentación con un *backbone* ResNet-101-FPN. Utiliza su implementación de referencia en **Detectron**, así como la red piramidal en cascada (CPN) que representa una extensión de Función de

red piramidal para detección de objetos (FPN). Para implementar la CPN requiere cajas delimitadoras (cajas de Mask R-CNN). Tanto para Mask R-CNN como para CPN, se comienza con los modelos preentrenados de la base de datos *Microsoft Common Objects in Context (COCO)* y se afinan los detectores en proyecciones $2D$ de Human3.6M, ya que el conjunto de puntos clave en COCO difiere de Human3.6M. En las ejecuciones, también se experimenta aplicando directamente el estimador de pose $3D$ a los puntos clave $2D$ preentrenados de COCO para hacer una estimación de las articulaciones $3D$ de Human3.6M.

La segmentación utilizada se basa en el análisis de una imagen con ayuda de una ventana delimitadora o *bounding box*) y una máscara binaria. Esta máscara determina qué píxeles de dicha ventana pertenecen al objeto que se ha localizado.

Para la *Mask R-CNN*, se adopta un *backbone* ResNet-101 entrenada. Al ajustar el modelo en Human3.6M, se reinicia la última capa de la red de puntos clave y las capas que regresan los mapas de calor para aprender un nuevo conjunto de puntos clave. Se entrena con una tasa de aprendizaje decreciente. En la inferencia, se le aplica la función de activación un *softmax* sobre los mapas térmicos y se extrae el valor esperado de la distribución $2D$ resultante. Esto da lugar a predicciones más suaves y precisas que el *hard-argmax*. El proyecto propone esta opción con la base de datos de Humaneva-I. Al no haber seleccionado esta base de datos este modelo de arquitectura no se llega a poenr en marcha.

Para la **CPN**, se utiliza una *backbone* ResNet-50 con una resolución de 384×288. Para afinar, se reinician las capas finales de GlobalNet y RefineNet. A continuación, se entrena en una GPU con lotes de 32 imágenes con tasa de aprendizaje decreciente. Se mantiene la normalización por lotes activada mientras se hacen ajuntes. Se entrena con cuadros delimitadores reales y se prueba utilizando los cuadros delimitadores predichos por el modelo *Mask R-CNN*. Este modelo es el que pasamos a ejecutar ya que sus creadores recomiendan utilizarlo con la base de datos de Human3.6m.

Estas detecciones CPN han sido producidas por modelos ajustados en Human3.6m. Se adopta el protocolo habitual de ajuste fino en 5 de los sujetos (S1, S5, S6, S7 y S8). También se incluyen detecciones de los sujetos no etiquetados S2, S3, S4, para la experimentación semi-supervisada.

4.4 Evaluación modelos preentrenados

Los modelos preentrenados se pueden descargar de AWS. Permiten reproducir una primera medición de 46,8 mm para Human3.6M, utilizando las detecciones de CPN ajustadas, los cuadros delimitadores de Mask R-CNN y una arquitectura con un campo receptivo de 243 fotogramas.

El **entrenamiento y evaluación en poses** $3D$ se hace en el espacio de la cámara rotando y trasladando las poses de GT de acuerdo con la transformación de la cámara, y no utilizando la trayectoria global (excepto en el ajuste semi-supervisado). Como optimizador de descenso del gradiente se utiliza **Amsgrad** entrenándolo durante 80 épocas. Para Human3.6M, se adopta un programa de tasa de aprendizaje exponencialmente decreciente, empezando por el factor de entrenamiento de $\eta = 0,001$ con de reducción de $\alpha = 0,95$ aplicado a cada época. Se reduce la correlación en las muestras de entrenamiento eligiendo clips de entrenamiento de diferentes segmentos de vídeo. El tamaño del conjunto de clips se ajusta a la anchura del campo receptivo de la arquitectura, de modo que el modelo predice una única pose $3D$ por clip de entrenamiento.

```
[1] time 114.85 lr 0.001000 3d_train 100.681943 3d_eval 49.830340 3d_valid 65.608174
[2] time 116.23 lr 0.000950 3d_train 49.436110 3d_eval 38.033314 3d_valid 56.370583
[3] time 113.54 lr 0.000902 3d_train 43.461873 3d_eval 34.608751 3d_valid 57.182160
[4] time 116.23 lr 0.000857 3d_train 40.286897 3d_eval 31.466340 3d_valid 54.196194
[5] time 114.90 lr 0.000815 3d_train 38.258041 3d_eval 29.516441 3d_valid 53.202255
[6] time 114.31 lr 0.000774 3d_train 36.765702 3d_eval 28.754299 3d_valid 52.613233
[7] time 115.67 lr 0.000735 3d_train 35.655353 3d_eval 27.746945 3d_valid 52.289821
[8] time 115.27 lr 0.000698 3d_train 34.782791 3d_eval 26.187245 3d_valid 52.814355
[9] time 115.41 lr 0.000663 3d_train 34.070475 3d_eval 24.900933 3d_valid 50.992618
[10] time 112.69 lr 0.000630 3d_train 33.529276 3d_eval 24.540537 3d_valid 51.529864
Saving checkpoint to checkpoint/epoch_10.bin
[11] time 114.37 lr 0.000599 3d_train 33.037296 3d_eval 23.898102 3d_valid 51.298898
[12] time 115.40 lr 0.000569 3d_train 32.557230 3d_eval 23.497485 3d_valid 50.732323
[13] time 112.12 lr 0.000540 3d_train 32.201125 3d_eval 22.886760 3d_valid 51.022061
[14] time 111.11 lr 0.000513 3d_train 31.850127 3d_eval 22.827069 3d_valid 51.013113
[15] time 114.77 lr 0.000488 3d_train 31.572318 3d_eval 22.715601 3d_valid 52.270049
[16] time 114.00 lr 0.000463 3d_train 31.321260 3d_eval 22.459663 3d_valid 51.467450
[17] time 113.62 lr 0.000440 3d_train 31.078496 3d_eval 22.304421 3d_valid 50.809220
[18] time 113.98 lr 0.000418 3d_train 30.849119 3d_eval 22.144881 3d_valid 49.814231
[19] time 113.86 lr 0.000397 3d_train 30.640738 3d_eval 21.390776 3d_valid 51.222448
[20] time 112.28 lr 0.000377 3d_train 30.490755 3d_eval 21.443523 3d_valid 50.717960
Saving checkpoint to checkpoint/epoch_20.bin
[21] time 112.81 lr 0.000358 3d_train 30.321047 3d_eval 21.231538 3d_valid 50.444551
[22] time 114.21 lr 0.000341 3d_train 30.189304 3d_eval 20.969347 3d_valid 50.911710
```

(a) Valor lr de la primera época 0.001000.

```
[23] time 113.29 lr 0.000324 3d_train 30.028810 3d_eval 21.329016 3d_valid 50.637359
[24] time 114.21 lr 0.000307 3d_train 29.884488 3d_eval 21.085399 3d_valid 50.445468
[25] time 114.00 lr 0.000292 3d_train 29.789840 3d_eval 20.669957 3d_valid 50.655969
[26] time 114.69 lr 0.000277 3d_train 29.639356 3d_eval 20.445626 3d_valid 50.680642
[27] time 114.59 lr 0.000264 3d_train 29.555976 3d_eval 20.389154 3d_valid 50.255644
[28] time 112.75 lr 0.000250 3d_train 29.456204 3d_eval 20.455821 3d_valid 50.311825
[29] time 115.23 lr 0.000238 3d_train 29.346904 3d_eval 20.299553 3d_valid 50.854379
[30] time 113.09 lr 0.000226 3d_train 29.276563 3d_eval 20.148723 3d_valid 51.002496
Saving checkpoint to checkpoint/epoch_30.bin
[31] time 113.76 lr 0.000215 3d_train 29.186487 3d_eval 19.956072 3d_valid 50.594778
[32] time 111.97 lr 0.000204 3d_train 29.117018 3d_eval 19.687484 3d_valid 50.549329
[33] time 112.24 lr 0.000194 3d_train 29.058219 3d_eval 19.634980 3d_valid 50.807123
[34] time 114.53 lr 0.000184 3d_train 28.987479 3d_eval 19.665922 3d_valid 50.012822
[35] time 111.71 lr 0.000175 3d_train 28.920870 3d_eval 19.383041 3d_valid 50.641834
[36] time 115.43 lr 0.000166 3d_train 28.858250 3d_eval 19.407319 3d_valid 50.837235
[37] time 113.43 lr 0.000158 3d_train 28.803620 3d_eval 19.411685 3d_valid 50.259594
[38] time 115.17 lr 0.000150 3d_train 28.758744 3d_eval 19.362288 3d_valid 49.944227
[39] time 113.98 lr 0.000142 3d_train 28.685536 3d_eval 19.172551 3d_valid 50.089616
[40] time 113.21 lr 0.000135 3d_train 28.643301 3d_eval 19.240761 3d_valid 49.895083
Saving checkpoint to checkpoint/epoch_40.bin
```

(b) Valor lr de la última época (40) 0.000135.

Figura 4.3: Demostración del factor de entrenamiento decreciente **lr** desde la primera época hasta la época 40.

Para optimizar en gran medida la configuración de un solo fotograma se puede hacer sustituyendo las convoluciones dilatadas por convoluciones estriadas en las que el estriado se establece como el factor de dilatación. Esto evita computar estados que nunca se usan y sólo se puede aplicar durante el entrenamiento. En la inferencia, se puede procesar secuencias enteras y reutilizar estados intermedios de otros fotogramas $3D$ para una inferencia más rápida. Esto es posible porque el modelo no utiliza ninguna forma de *pooling* sobre la dimensión temporal. Para evitar la pérdida de fotogramas por convoluciones, se rellena por replicación, pero sólo en los límites de entrada de una secuencia.

Los hiperparámetros por defecto de la normalización por lotes conducen a grandes fluctuaciones del error de prueba de 1 mm, así como a fluctuaciones en las estimaciones de ejecución para la inferencia. Para conseguir unas estadísticas de funcionamiento más estables, se utiliza un programa para el momento de normalización por lotes β: empieza con $\beta = 0,1$, y se baja exponencialmente para que alcance $\beta = 0,001$ en la última época (ver figura 4.3).

Para **visualizar** el resultado por ejemplo, para S11 y la función de "Andando" desde la cámara 0 . Se exportan los primeros fotogramas a una animación GIF con una velocidad de fotogramas de 25 FPS (ver figura 4.4).

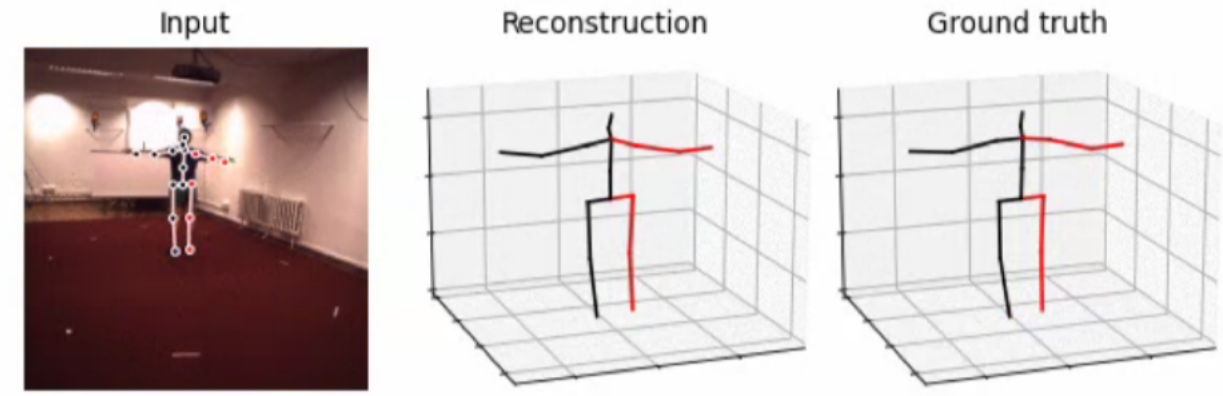


Figura 4.4: Captura de pantalla del GIF generado.

4.4.1 Evaluacion del sistema (*scoring*)

Para evaluar el sistema es necesario testear los resultados numéricos obtenidos. El *scoring* suele ser una función dentro del programa, en este caso la de *loses.py*.

En los experimentos, se consideran 3 protocolos de evaluación:

- **Protocolo 1**. Corresponde con el error en milímetros de posición media por articulación (MPJPE). Distancia media entre las posiciones predichas y las posiciones reales.

- **Protocolo 2**. Busca una transformación óptima en traslación, rotación y escala entre el GT y la estimación. Luego se calcula el error de posición medio entre el GT y las posiciones transformadas de las posiciones estimadas (P-MPJPE).

- **Protocolo 3**. Alinea las posturas predicadas con el GT, al igual que el protocolo 2, pero solo en escala (N-MPJPE).

A continuación, en las tablas 4.1 y 4.2, se recogen los resultados que corresponden al Protocolo 1 y Protocolo 2 del experimento de prueba que se ha ejecutado. En el apartado anterior 4.4 ya se verificó su correcto funcionamiento.

Este experimento es propuesto por VideoPose3D y tiene en su documentación los valores de los resultados que obtuvieron en su momento (con los valores redondeados) [18]. Los desarrolladores de este sistema aseguran que su modelo tiene un error medio más bajo en todos los Protocolos superando así al mejor resultado anterior por 6 mm de media (reducción del error del 11 %).

Los resultados del experimento de prueba son exactamente iguales a los resultados de VideoPose3D.

Actividad	Experimento propuesto
Discussion	46.67
WalkDog	48.95
Sitting	57.29
Phoning	48.09
Walking	32.78
Smoking	47.08
Directions	45.15
WalkTogether	33.87
Photo	55.12
Waiting	43.98
Posing	44.62
Purchases	44.27
Eating	43.30
Greeting	45.61
SittingDown	65.79
Avg	46.8

Tabla 4.1: Resultados del **Protocolo 1** del experimento propuesto. Error medido en mm.

Actividad	Experimento propuesto
Discussion	36.12
WalkDog	37.75
Sitting	45.03
Phoning	36.39
Walking	25.63
Smoking	37.37
Directions	34.11
WalkTogether	27.28
Photo	42.20
Waiting	33.79
Posing	34.42
Purchases	33.59
Eating	34.43
Greeting	37.22
SittingDown	52.53
Avg	36.5

Tabla 4.2: Resultados del **Protocolo 2** del experimento propuesto. Error medido en mm.

4.5 Propuesta de evaluación con una nueva base de datos

Una vez que se ha conseguido montar todo el sistema y se ha comprobado que se ejecuta correctamente, se puso en marcha una de las propuestas del proyecto. Esta propuesta consistía en coger una base de datos diferente con un conjunto de datos completamente distinto y encriptar ese mismo proceso de *joins* comprobando su funcionamiento y nuevos cálculos (como cuánto sube/baja el error por articulación).

Se ha indagado dentro de la búsqueda inicial de sistemas y bases de datos disponibles la más apropiada para el sistema implantado de VideoPose3D. Esta decisión ha sido tomada en cuanto al parecido con la base de datos inicial de Human3.6m. Las condiciones que debía cumplir la nueva base de datos son:

1. La posición de la *joins* tiene que ser similar. Por ejemplo si nuestra base de datos tiene una *join* en el codo, que la nueva base de datos la tenga también ahí y no en mitad del brazo.

2. Tiene q tener las imágenes RGB (color). Por suerte, durante la indagación de bases de datos, sólo dimos por óptimas las grabadas con este tipo de cámara.

3. Etiquetada en espacio tridimensional

4. Como mínimo el mismo número de *joins*. Según el número de articulaciones segmentadas, en este proceso aparecen 3 casos posibles: mayor número de *joins*, menor o igual. El caso más fácil sería aquel en el que el número de articulaciones es igual que en la base de datos de Human3.6m que tiene un total de 17 *joins*. Si tuviera más, nos encontraríamos con una situación algo más compleja ya que habría que modificar la nueva base de datos para conseguir el mismo número de *joins*. Y por último, el tercero de los casos no se llegaba a considerar debida a sus complicaciones, este caso plantea menos número de articulaciones que la base de datos original.

Es por ello, que la base de datos elegida para continuar con las investigaciones propias de este proyecto es la base de datos MPI-INF-3DHP ya que cumplía con todas las condiciones anteriores.

Es muy importante que el tamaño de la imagen y el formato sean el mismo. Pero, cuando se cambia de base de datos puede cambiar muchas cosas. Existe un 90 % de posibilidades de que sea distinta. Primeramente el formato de esta base de datos resulta ser en imágenes y por tanto hay que pasarlas a vídeo. Y, por otro lado, surgió el problema de la calidad de la imagen Human3.6m con 1000×1000 y MPI-INF-3DHP con 2048×2048. Para resolver ambos problemas hizo falta la implementación de una función que lista las imágenes (función *glob.glob*), las redimensiona (función *ffmpeg*) y cambia de formato (función *cv2.resize*). El nombre de la función es *change image video.py*.

Después, toca asegurarse de que las articulaciones se encuentran igual situadas en ambas bases de datos y que tengan el mismo formato (json, txt...) y el mismo eje de coordenadas para identificar las *jois*. Pues bien, llega el punto donde hay que estudiar el posicionamiento de cada una de las *joins*. Para el conjunto de datos de MPI-INF-3DHP es fácil ya que en el mismo *Paper* aparece un dibujo de su distribución (ver figura 4.5). Además, para verificarlo, esta información aparece recogida en un *.mat* (que se descargó con las imágenes anteriormente).

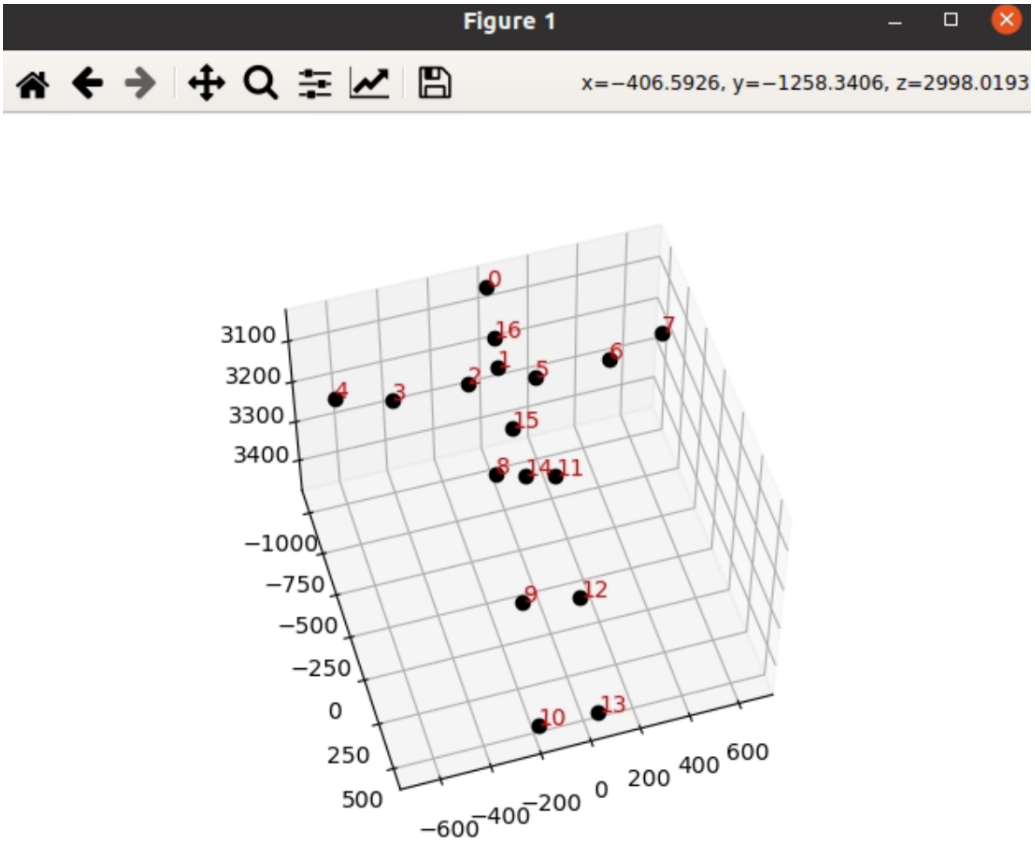


Figura 4.5: Esqueleto generado con el conjunto de datos de MPI-INF-3DHP.

Sin embargo, a pesar de la búsqueda reiterada de este esqueleto en la base de datos Human3.6m no es posible su hallazgo. Siendo la única información útil que orden de sus articulaciones es: *'Pelvis', 'RHip', 'RKnee', 'RAnkle', 'LHip', 'LKnee', 'LAnkle', 'Spine1', 'Neck', 'Head', 'Site', 'LShoulder'* y *'LElbow* [53].

Solo con estos datos no fue posible la reconstrucción del esqueleto. Y por tanto, hasta aquí llegó el intento de implantar una nueva base de datos al sistema.

4.6 Propuesta de cambio del valor de *batch size*

Se plantea hacer modificaciones en el parámetro de entrenamiento de ***batch size***. Se entiende por *batch size* el número de ejemplos que se introducen en una red para que entrene de cada vez. En el caso de que este número sea pequeño, significa que la red tiene en memoria poca cantidad de datos y puede entrenar más rápido. Pero, como desventaja, es posible que no aprenda las características y detalles que se consideran significativos en la predicción. Por el contrario, si este valor es un número grande, ocurre al contrario, siendo más probable que tenga en cuenta los casos más importantes cuando aprenda, pero entrena más lento.

Los resultados se recogen en el apartado siguiente de Resultados 5.2.

4.7 Conclusiones

En la ejecución propuesta por los desarrolladores de VideoPose3D para comprobar el correcto funcionamiento del sistema se obtienen unos resultados favorables ya que corresponden a los de ellos.

Y, tras la primera propuesta de cambio de base de datos fallida, se llevan a cabo un estudio de varias ejecuciones del código para comprobar qué tipo de entrenamiento con qué combinación de hiperparámetros se obtienen mejores resultados. Tanto las explicaciones de los siguientes entrenamientos como sus resultados son recogidos en el apartado siguiente 5.1.

Capítulo 5

Resultados

La esperanza es un estimulante vital muy superior a la suerte.

Friedrich Nietzsche

5.1 Introducción

El capitulo comienza con una presentación de los experimentos que se han llevado a cabo con distintos enfoques en su entrenamiento sobre 15 actividades distintas. El primero de los enfoques corresponde al entrenamiento supervisado que se lleva a cabo en el capítulo anterior verificando así el correcto funcionamiento del sistema. Otro de los enfoques corresponde a un entrenamiento del sistema desde cero. Además, se llevan a cabo otros 3 entrenamientos supervisados sin cambio en los sujetos, el número de fotogramas, la cantidad de épocas de entrenamiento, pero sí en el valor del *batch size*. Y también se realizan 2 entrenamientos semi-supervisados con el mismo número de épocas, fotogramas y sujetos pero cambiando el valor de *batch size*.

Se intenta explicar de forma clara los resultados obtenidos con la ejecución de todos estos entrenamientos. Esto se lleva a cabo con ayuda de unas tablas. Cada tabla aparece triplicada de forma que encontramos una tabla distinta por cada Protocolo. Para hacerlo más visual, a cada Protocolo le corresponde un color: Protocolo 1 amarillo, Protocolo 2 azul y Protocolo 3 naranja.

Incluimos también el término de ***fine-tuning*** o ajuste fino que consiste en hacer un ajuste y entrenar algunas de las capas convolucionales finales del modelo usado para la extracción de características, y entrenar conjuntamente la parte agregada del clasificador y las nuevas capas.

5.2 Presentación de los experimentos

Los experimentos realizados afectan a distintas estrategias de entrenamiento que se describen a continuación:

- El **experimento inicial** que recogemos es el propuesto en la publicación de los autores de VideoPose3D, que se basa en un entrenamiento supervisado que emplea los modelos preentrenados (con COCO). Utiliza un campo receptivo de 243 fotogramas. El entrenamiento es sobre una GPU con

batch size de 32 y durante 80 épocas. Emplea los sujetos de *train* S1, S6, S7 y S8 y los sujetos de *test* S9 y S11.

- El entrenamiento **desde** 0 utiliza una inicialización aleatoria de los pesos. Entrena con un campo receptivo de 27 fotogramas durante 40 épocas. Emplea los sujetos de *train* S1, S6, S7 y S8 y los sujetos de *test* S9 y S11. Toma un valor de *batch size* de 1024 (valor predeterminado por la red).

- En los **entrenamientos supervisados** se parte de la estimación de puntos 2D que luego se mapean a 3D, entrenando con los datos de GT en 3D para luego mapear la imagen a 2D de de nuevo y comprobar así los errores en la estimación. Utiliza un campo receptivo de 27 fotogramas durante 200 épocas. Con sujeto de *train* S1 y sujetos de *test* S9 y S11. Se lleva a cabo 3 entrenamientos distintos cambiando el valor del *batch size* por 32 (*Supervisado B.32*), 64 (*Supervisado B.64*) y 128 (*Supervisado B.128*).

- El **entrenamiento semi-supervisado** actúa como regularizador. Los dos objetivos se optimizan conjuntamente, ocupando los datos etiquetados la primera mitad de un lote, y los datos no etiquetados la segunda mitad. Para los datos etiquetados se utilizan las poses GT en 3D y entrenan una pérdida supervisada. Los datos sin etiquetar se utilizan para implementar una pérdida de codificador automático donde las poses 3D predichas se proyectan de nuevo a 2D y luego se verifica la coherencia con la entrada.

  Se entrenan dos modelos con 27 fotogramas y durante 200 épocas. Utilizan el sujeto de *train* S1, los sujetos de *test* S9 y S11 y los sujetos sin etiquetar S5, S6, S7 y S8. Los valores de *batch size* son de 32 (**SemiSupervisado B.64**) y de 64 (**SemiSupervisado B.64**).

Actividad	Supervisado B.32	Supervisado B.64	Supervisado B.128
Discussion	69.54	70.38	69.22
WalkDog	85.02	87.22	85.05
Sitting	90.69	91.39	88.15
Phoning	84.12	82.88	81.23
Walking	59.78	59.21	57.85
Smoking	73.32	72.62	71.90
Directions	59.62	59.64	58.73
WalkTogether	68.95	67.63	65.86
Photo	115.41	113.55	110.94
Waiting	75.88	76.40	75.00
Posing	69.81	68.69	66.77
Purchases	76.35	79.12	81.48
Eating	74.17	73.94	72.84
Greeting	67.59	68.25	67.82
SittingDown	141.88	146.63	150.45
Avg	80.8	81.2	80.2

Tabla 5.1: Resultados del **Protocolo 1** comparativa de los 3 entrenamientos supervisados con *fine-tuning*. Error medido en mm.

Actividad	Supervisado B.32	Supervisado B.64	Supervisado B.128
Discussion	51.39	51.07	49.87
WalkDog	59.10	59.40	57.70
Sitting	63.48	61.90	61.17
Phoning	61.34	59.52	58.31
Walking	44.32	43.10	42.25
Smoking	54.15	52.71	52.08
Directions	41.53	40.84	40.33
WalkTogether	51.90	50.43	49.10
Photo	73.68	70.47	68.57
Waiting	53.72	53.72	52.24
Posing	50.39	48.85	47.27
Purchases	52.11	51.62	52.32
Eating	50.51	48.27	47.26
Greeting	48.49	48.13	47.59
SittingDown	102.19	99.09	99.46
Avg	57.2	55.9	55.0

Tabla 5.2: Resultados del **Protocolo 2** comparativa de los 3 entrenamientos supervisados con *fine-tuning*. Error medido en mm.

5.2.1 Comparativa de los entrenamientos supervisados

Las tablas 5.1, 5.2 y 5.3 muestra los resultados del Protocolo 1, Protocolo 2 y Protocolo 3, respectivamente. Estos resultados representan el comportamiento del sistema frente al cambio del valor de *batch size* (es el único valor que cambia entre estos 3 entrenamientos).

Como podemos ver, el entrenamiento supervisado con valor de *batch size* 128 muestra los mejores resultados de media en los 3 protocolos. Aunque no siempre es así en actividades por separado por ejemplo, en el Protocolo 1 las actividades de *WalkDog*, *Purchases*, *Greeting* o *SittingDown* no lo cumplen, en el Protocolo 2 solo la actividad de *Purchases* y en Protocolo 3 *Purchases*, *Greeting* y *SittingDown* tampoco. Esto no es relevante ya que para evaluar un entrenamiento de forma adecuada hay que tener en cuenta los resultados, en este caso, de todas las actividades en las que se desarrolla. Y, por ello, el dato importante de estas tablas es el resultado de la media de todas las actividades por tipo de entrenamiento.

Que en este experimento haya salido con mejores valores el entrenamiento supervisado con *batch size* 128 no sorprende ya que corrobora la teoría. Esta teoría dice que los resultados del entrenamiento de un sistema debería ser mejor cuanto mayor es el tamaño del *batch size*. Esto es porque con *batch sizes* más pequeños, el sistema cuenta con menor información para el proceso de adaptación (entrenamiento) de los pesos; es decir puede no aprende bien las características y detalles que se consideran significativos en la predicción. Por el contrario, si este valor es un número grande el sistema cuenta con más información para entrenar siendo más probable que tenga en cuenta características más importantes cuando aprenda.

5.2.2 Comparativa de los entrenamientos semi-supervisados con *fine-tuning*

Este apartado contiene el conjunto de resultados correspondientes al Protocolo 1 (tabla 5.4), Protocolo 2 (tabla 5.5) y Protocolo 3 (tabla 5.6) del comportamiento del sistema en entrenamientos semi-supervisados. Se comparan los resultados obtenidos en 2 modelos de entrenamiento que solo cambian su valor del *batch size* por 32 y 64.

Viendo estas tablas, el entrenamiento semi-supervisado con valor de *batch size* 64 muestra mejores resultados en los 3 protocolos de media.

Actividad	Supervisado B.32	Supervisado B.64	Supervisado B.128
Discussion	64.74	65.34	64.02
WalkDog	80.47	82.02	80.25
Sitting	81.33	80.76	78.68
Phoning	79.65	78.02	76.52
Walking	56.30	55.54	54.26
Smoking	67.39	66.32	65.54
Directions	52.70	52.71	51.65
WalkTogether	63.13	61.81	60.32
Photo	109.94	108.23	105.83
Waiting	71.75	72.38	70.88
Posing	64.75	63.56	61.52
Purchases	70.39	72.33	74.70
Eating	65.27	64.79	63.93
Greeting	60.98	61.62	61.25
SittingDown	140.64	143.81	146.41
Avg	75.3	75.3	74.4

Tabla 5.3: Resultados del **Protocolo 3** comparativa de los 3 entrenamientos supervisados con *fine-tuning*. Error medido en mm.

Es bastante seguro que hubiesen salido aún mejores resultados en el caso de haber realizado un tercer entrenamiento semi-supervisado con valor de *batch size* superior a 64, por ejemplo 128. Ya que, como hemos visto en la comparativa anterior, cuanto mayor es el valor del *batch size*, más información se tiene a la hora de realizar el entrenamiento. Por tanto, las predicciones tienden a ser más exactas.

Actividad	SemiSupervisado B.64	SemiSupervisado B.32
Discussion	54.92	56.23
WalkDog	69.87	70.14
Sitting	67.93	69.35
Phoning	65.12	63.72
Walking	53.34	53.80
Smoking	58.53	55.16
WalkTogether	56.30	56.35
Photo	90.51	90.14
Waiting	63.07	63.81
Posing	58.73	59.24
Purchases	61.70	60.55
Eating	59.62	60.85
Greeting	56.27	56.72
SittingDown	114.84	134.82
Avg	65.4	66.9

Tabla 5.4: Resultados del **Protocolo 1** comparativa de los 2 entrenamientos semi-supervisados con *fine-tuning*. Error medido en mm.

Actividad	SemiSupervisado B.64	SemiSupervisado B.32
Discussion	41.63	41.90
WalkDog	48.98	49.55
Sitting	51.43	53.20
Phoning	48.01	47.48
Walking	39.83	39.54
Smoking	44.30	44.20
Directions	35.50	35.21
WalkTogether	41.30	40.76
Photo	58.43	58.91
Waiting	46.72	46.70
Posing	42.96	42.97
Purchases	41.65	41.82
Eating	40.60	42.20
Greeting	40.97	41.23
SittingDown	95.70	96.00
Avg	47.9	48.1

Tabla 5.5: Resultados del **Protocolo 2** comparativa de los 2 entrenamientos semi-supervisados con *fine-tuning*. Error medido en mm.

Actividad	SemiSupervisado B.64	SemiSupervisado B.32
Discussion	52.11	53.14
WalkDog	66.40	66.54
Sitting	65.39	66.78
Phoning	63.49	61.71
Walking	50.61	50.73
Smoking	55.56	54.76
Directions	44.76	43.82
WalkTogether	52.78	52.26
Photo	88.42	87.90
Waiting	61.01	61.16
Posing	55.12	55.18
Purchases	58.13	56.53
Eating	56.86	57.66
Greeting	52.31	52.29
SittingDown	116.84	130.62
Avg	62.7	63.4

Tabla 5.6: Resultados del **Protocolo 3** comparativa de los 2 entrenamientos semi-supervisados con *fine-tuning*. Error medido en mm.

5.2.3 Comparativa de los entrenamientos supervisados y semi-supervisados con *fine-tuning*

Con cualquier otro sistema, se esperaría obtener mejores resultados con los entrenamientos supervisados ya que se parte de datos etiquetados. Sin embargo, utilizando el sistema de este proyecto para los entrenamientos supervisado y semi-supervisado se obtiene mejores resultados con el semi-supervisado. Esto es debido a que los entrenamientos semi-supervisados es la solución que propone VideoPose3D frente al problema de la escasez de datos de entrenamiento etiquetados.

Revisando las tablas del Protocolo 1 (tabla 5.7), Protocolo 2 (tabla 5.8) y Protocolo 3 (tabla 5.9) se muestra una variación de aproximadamente 15 mm entre los entrenamientos supervisado y semi-supervisado con valores de *batch size* de 64 y 32.

Actividad	Batch size 64		Batch size 32	
	SemiSupervisado B.64	Supervisado B.64	SemiSupervisado B.32	Supervisado B.32
Discussion	54.92	70.38	56.23	69.54
WalkDog	69.87	87.22	70.14	85.02
Sitting	67.93	91.39	69.35	90.69
Phoning	65.12	82.88	63.72	84.12
Walking	53.34	59.21	53.80	59.78
Smoking	58.53	72.62	57.97	73.32
Directions	50.60	59.64	50.16	59.62
WalkTogether	56.30	67.63	56.35	68.95
Photo	90.51	113.55	90.14	115.41
Waiting	63.07	76.40	63.81	75.88
Posing	58.73	68.69	59.24	69.81
Purchases	61.70	79.12	60.55	76.35
Eating	59.62	73.94	60.85	74.17
Greeting	56.27	68.25	56.72	67.59
SittingDown	114.84	146.63	134.82	141.88
Avg	65.4	81.2	66.9	80.8

Tabla 5.7: Resultados del **Protocolo 1** comparativa supervisados y semi-supervisado con *fine-tuning*. Error medido en mm.

5.2.4 Comparativa entrenamiento desde 0 y *fine-tuning*

Comparando los entrenamientos supervisado, semi-supervisado y desde 0 con el mismo número de *batch size* (32), funciona mejor el entrenamiento desde 0. Esto se debe a que reproduce los resultados de los modelos preentrenados con un valor de *batch size* de 1024, muy superior al resto (con 32). Podemos ver los resultados en las tablas del Protocolo 1 (tabla 5.10), Protocolo 2 (tabla 5.11) y Protocolo 3 (tabla 5.12). Suma una diferencia de aproximadamente 16 mm con el siguiente caso mejor (semi-supervisado).

Actividad	Batch size 64		Batch size 32	
	SemiSupervisado B.64	Supervisado B.64	SemiSupervisado B.32	Supervisado B.32
Discussion	41.63	51.07	41.90	51.39
WalkDog	48.98	59.40	49.55	59.10
Sitting	51.43	61.90	53.20	63.48
Phoning	48.01	59.52	47.48	61.34
Walking	39.83	43.10	39.54	44.32
Smoking	44.30	52.71	44.20	54.15
Directions	35.50	40.84	35.21	41.53
WalkTogether	41.30	50.43	40.76	51.90
Photo	58.43	70.47	58.91	73.68
Waiting	46.72	53.72	46.70	53.72
Posing	42.96	48.85	42.97	50.39
Purchases	41.65	51.62	41.82	52.11
Eating	40.60	48.27	42.20	50.51
Greeting	40.97	48.13	41.23	48.49
SittingDown	95.70	99.09	96.00	102.19
Avg	47.9	55.9	48.1	57.2

Tabla 5.8: Resultados del **Protocolo 2** comparativa supervisados y semi-supervisado con *fine-tuning*. Error medido en mm.

Actividad	Batch size 64		Batch size 32	
	SemiSupervisado B.64	Supervisado B.64	SemiSupervisado B.32	Supervisado B.32
Discussion	52.11	65.34	53.14	64.74
WalkDog	66.40	82.02	66.54	80.47
Sitting	65.39	80.76	66.78	81.33
Phoning	63.49	78.02	61.71	79.65
Walking	50.61	55.54	50.73	56.30
Smoking	55.56	66.32	54.76	67.39
Directions	44.76	52.71	43.82	52.70
WalkTogether	52.78	61.81	52.26	63.13
Photo	88.42	108.23	87.90	109.94
Waiting	61.01	72.38	61.16	71.75
Posing	55.12	63.56	55.18	64.75
Purchases	58.13	72.33	56.53	70.39
Eating	56.86	64.79	57.66	65.27
Greeting	52.31	61.62	52.29	60.98
SittingDown	116.84	143.81	130.62	75.3
Avg	62.7	75.3	63.4	75.3

Tabla 5.9: Resultados del **Protocolo 3** comparativa supervisados y semi-supervisados con *fine-tuning*. Error medido en mm.

Actividad	Supervisado B.32	SemiSupervisado B.32	Desde 0
Discussion	51.39	41.90	37.29
WalkDog	59.10	49.55	39.85
Sitting	63.48	53.20	46.48
Phoning	61.34	47.48	38.51
Walking	44.32	39.54	27.54
Smoking	54.15	44.20	39.21
Directions	41.53	35.21	34.50
WalkTogether	51.90	40.76	31.36
Photo	73.68	58.91	44.05
Waiting	53.72	46.70	34.59
Posing	50.39	42.97	35.35
Purchases	52.11	41.82	33.94
Eating	50.51	42.20	35.59
Greeting	48.49	41.23	38.01
SittingDown	102.19	96.00	53.05
Avg	57.2	48.1	38.0

Tabla 5.11: Resultados del **Protocolo 2** comparativa entrenamiento desde 0 y *fine-tuning*. Error medido en mm.

Actividad	Supervisado B.32	SemiSupervisado B.32	Desde 0
Discussion	69.54	56.23	48.38
WalkDog	85.02	70.14	51.44
Sitting	90.69	69.35	57.84
Phoning	84.12	63.72	50.61
Walking	59.78	53.80	35.85
Smoking	73.32	57.97	49.07
Directions	59.62	50.16	45.80
WalkTogether	68.95	56.35	38.47
Photo	115.41	90.14	57.36
Waiting	75.88	63.81	45.76
Posing	69.81	59.24	46.29
Purchases	76.35	60.55	44.35
Eating	74.17	60.85	44.91
Greeting	67.59	56.72	47.100
SittingDown	141.88	134.82	66.83
Avg	80.8	66.9	48.7

Tabla 5.10: Resultados del **Protocolo 1** comparativa entrenamiento desde 0 y *fine-tuning*. Error medido en mm.

5.2.5 Comparativa tiempos de proceso

Mirando la tabla 5.13 verificamos que, como era de esperar, los entrenamientos supervisados son los que menos tardan. Esto es debido a que parte de unos parámetros ya etiquetados por lo que resulta más fácil que el modelo aprenda. El entrenamiento semi-supervisado como cuenta con un pequeño grupo de parámetros sin etiquetar y el resto se encuentran etiquetados, tarda algo más. Y, el entrenamiento desde 0 es el que más tiempo de ejecución emplea.

Actividad	Supervisado B.32	SemiSupervisado B.32	Desde 0
Discussion	64.74	53.14	46.92
WalkDog	80.47	66.54	49.02
Sitting	81.33	66.78	56.26
Phoning	79.65	61.71	48.52
Walking	56.30	50.73	34.39
Smoking	67.39	54.76	47.45
Directions	52.70	43.82	43.37
WalkTogether	63.13	52.26	36.57
Photo	109.94	87.90	55.30
Waiting	71.75	61.16	44.40
Posing	64.75	55.18	44.87
Purchases	70.39	56.53	42.62
Eating	65.27	57.66	43.59
Greeting	60.98	52.29	45.73
SittingDown	140.64	130.62	64.16
Avg	75.3	63.4	46.9

Tabla 5.12: Resultados del **Protocolo 3** comparativa entrenamiento desde 0 y *fine-tuning*. Error medido en mm.

	Tiempo medio 1 época (min)	Total de épocas	Tiempo total aproximado (horas)
Desde 0	114.24	40	76.16
Supervisado B.32	3.54	200	11.8
Supervisado B.64	2.92	200	9.73
Supervisado B.128	2.58	200	8.6
SemiSupervisado B.32	12.77	200	42.56
SemiSupervisado B.64	11.36	200	37.86

Tabla 5.13: Resultados del tiempos de cada entrenamiento.

Con las modificaciones en el parámetro de entrenamiento de ***batch size*** se demuestra que en el caso de que este número sea pequeño puede entrenar más rápido. Por el contrario, si este valor es un número grande, ocurre al contrario, entrena más lento (ver tabla 5.13).

5.3 Conclusiones

Las conclusiones que se obtienen observando todos los valores de las tablas son las mismas que las determinadas por VideoPose3D. Entre el entrenamiento supervisado y semi-supervisado, tiene mejores resultados el semi-supervisado. Sin embargo entre los entrenamientos supervisado, semi-supervisado y desde 0, este último obtiene resultados de error más bajos.

También añadir que todos los casos, el resultado con *batch size* mayor resulta ser el mejor pero el más lento a la hora de entrenar. Y, que el Protocolo 2 es el que mejores resultados obtiene en todas las comparaciones.

Se sabe que para obtener conclusiones más precisas del modelo se necesitarían muchas más pruebas tanto del valor del *batch size* como del resto de valores.

Capítulo 6

Conclusiones y líneas futuras

En este capitulo se puede encontrar un resumen de las conclusiones más importantes de proyecto, con sus metas y desarrollo, así como algunas propuestas para continuar con su investigación.

6.1 Conclusiones

Este trabajo fin de grado ha consistido en la puesta a punto y posterior evaluación de un sistema de segmentación y seguimiento del cuerpo humano.

En los primeros capítulos se recoge tanto la motivación que ha supuesto la investigación del trabajo, como también el estudio teórico previo para su implantación y desarrollo. En este estudio previo se pueden encontrar todos los términos relevantes para poder entender el vocablo de *deep learning*, qué son las redes neuronales por que se componen y cómo funcionan. También se explican las arquitecturas que utiliza el sistema y los tipos de entrenamiento que existen.

Tras la recopilación de toda esta información, se llevó a cabo un estudio de sistemas y bases de datos relacionadas con la intención del proyecto. El fin era elegir las opciones más optimas y desarrollarlas. Se optó por el trabajo de VideoPose3D ya que cumplía con todas las características de demostración de poses 3D que se buscaba: utiliza Python para el desarrollo del código, entrada de datos de vídeo en RGB, demostraciones, etc. Este trabajo contenía un listado de bases de datos con las que se había probado y una de ellas era Human3.6m que fue la que se escogió para este proyecto. El conjunto de datos de vídeo de esta base de datos era en RGB y tenía un tamaño considerable.

Una vez hecha esta selección había que poner el sistema en marcha y analizar el código Python que nos ofrecía. Esta parte fue imprescindible para acabar de profundizar en el entendimiento de la arquitectura de la red y todas sus funciones comentadas y explicadas previamente en el estudio teórico.

Para la estimación eficiente de la pose humana en 3D utilizando trayectorias de puntos clave 2D, Videopose3D propone un enfoque convolucional con una arquitectura CNN. Con esto se consigue realizar una convolución dilatada en 1D a lo largo del tiempo y cubrir un gran campo receptivo, procesando múltiples tramas en paralelo.

A pesar de los numerosos proyectos de investigación basados en sistemas con redes neuronales que existen en la actualidad, los datos de entrenamiento etiquetados son bastante escasos. Es por ello que el proyecto de VideoPose3D para abordar este problema emplea el aprendizaje semi-supervisado con retroproyección. Este tipo de entrenamiento tiene una arquitectura altamente precisa y eficaz capaz de abordar entornos que carecen de datos de entrenamiento etiquetados. La forma de hacerlo es aprovechando

los datos de vídeo sin etiquetar. El sistema planteado se compone de una arquitectura CPN que predice puntos clave 2D en vídeos sin etiquetar utilizando un detector de puntos clave 2D compuesto por cuadros delimitadores llamado Mask R-CNN. Predice las poses en 3D para luego mapearlas a un espacio 2D.

Tras conseguir poner en marcha el modelo preentrenado, se hizo la propuesta de entrenar el sistema con un cambio en la base de datos para comprobar su funcionamiento. Se buscó e hicieron diversas modificaciones con la nueva base de datos de MPI-INF-3DHP; sin embargo, esta propuesta no resultó efectiva debido a problemas con la interpretación de las *joins* de la base de datos de Human3.6m.

Es por ello que se llevó a cabo una nueva propuesta de evaluar el sistema enfrentándose a diversos entrenamientos con diferentes valores de los hiperparámetros, como el número de fotogramas de entrada, épocas de entrenamiento y sobre todo la variable de *batch size*. Marcando el final del proyecto el análisis de estos resultados.

6.2 Líneas futuras

En este apartado final se plantean incluir un conjunto de líneas futuras como ideas para continuar con este TFG.

- Conseguir los valores óptimos del sistema. Continuar con los entrenamientos que se han llevado a cabo en este proyecto cambiando el valor del *batch size*, el número de épocas de entrenamiento así como los fotogramas de entrada, del factor de entrenamiento (lr), el optimizador de descenso del gradiente (en el proyecto se utiliza Amsgrad, probar con otro)... Así como también la función de activación del sistema o modificar el número de capas de la red neuronal que se utiliza.
- Evaluar el sistema con otras bases de datos, en particular Human3.6m y ETRI, la última más próxima al entorno buscado en el proyecto EYEFUL.
- Como objetivo final se propone demostrar en tiempo (casi)real, desde una cámara 3D y RGB [20] el sistema en funcionamiento.

Bibliografía

[1] "Ejemplo pose 2d y 3d," https://zephyrnet.com/es/Estimaci%C3%B3n-de-la-pose-en-3D-para-el-seguimiento-de-atletas-utilizando-videos-2d-y-Amazon-Sagemaker-Studio./ [Último acceso 7/septiembre/2022].

[2] "Neuronas. definición y partes," https://www.cognifit.com/es/cerebro [Último acceso 7/septiembre/2022].

[3] "Equivalencia componentes neuronas biológicas vs neuronas artificiales," https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s4p3.htm [Último acceso 7/septiembre/2022].

[4] "Arquitectura cnn," https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html [Último acceso 7/septiembre/2022].

[5] "Pixeles y neuronas," https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/ [Último acceso 7/septiembre/2022].

[6] "Funciones de activación," https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s6p2.htm [Último acceso 7/septiembre/2022].

[7] "Función de activación relu," https://datascientest.com/es/convolutional-neural-network-es [Último acceso 7/septiembre/2022].

[8] "Documentacion función softmax." https://datapeaker.com/big-data/10-peliculas-ocultas-que-muestran-el-poder-del-aprendizaje-automatico/ [Último acceso 7/septiembre/2022].

[9] "Diferentes arquitecturas cnn," https://datapeaker.com/big-data/arquitecturas-avanzadas-arquitecturas-de-aprendizaje-profundo/ [Último acceso 7/septiembre/2022].

[10] "Arquitectura red vgg." https://datascientest.com/es/vgg-que-es-este-modelo-daniel-te-lo-cuenta-todo [Último acceso 7/septiembre/2022].

[11] "Arquitectura resnet-50." https://jananisbabu.github.io/ResNet50_From_Scratch_Tensorflow/ [Último acceso 7/septiembre/2022].

[12] "Teoría de los algoritmos de agrupamiento," http://www.cs.us.es/~fsancho/?e=230 [Último acceso 7/septiembre/2022].

[13] "Enlace al paper oficial del proyecto voxelpose." https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123460188.pdf [Último acceso 7/septiembre/2022].

[14] "Enlace al paper del sistema de stridedtransformer-pose3d." https://arxiv.org/pdf/2103.14304.pdf [Último acceso 7/septiembre/2022].

[15] "Ejemplo de imágnes y joins human3.6m." https://www.researchgate.net/figure/Qualitative-results-on-the-Human36M-dataset-using-detected-2D-joint-locations-as-input_fig5_323956820 [Último acceso 7/septiembre/2022].

[16] "Página web oficial etri database," https://ai4robot.github.io/etri-activity3d-en/ [Último acceso 7/septiembre/2022].

[17] "Paper mpii database," https://paperswithcode.com/dataset/mpi-inf-3dhp [Último acceso 7/septiembre/2022].

[18] "Enlace al paper oficial del proyecto videopose3d." https://www.ecva.net/papers/eccv_2020/papers_ECCV/papers/123460188.pdf [Último acceso 7/septiembre/2022].

[19] "Cifras discapacidad del observatorio estatal de la discapacidad," https://www.observatoriodeladiscapacidad.info/mujeres-y-hombres-en-espana-2020-cifras-de-discapacidad/ [Último acceso 7/septiembre/2022].

[20] H. H. Pham, H. Salmane, L. Khoudour, A. Crouzil, S. A. Velastin, and P. Zegers, "A unified deep framework for joint 3d pose estimation and action recognition from a single rgb camera," *Sensors*, vol. 20, no. 7, p. 1825, 2020.

[21] "Ia en salud," https://es.wikipedia.org/wiki/Inteligencia_artificial_en_el_campo_de_la_salud [Último acceso 7/septiembre/2022].

[22] "Página del proyecto eyeful," http://www.geintra-uah.org/eyeful/es/resumen [Último acceso 21/diciembre/2021].

[23] Y. W. Hen and R. Paramesran, "Single camera 3d human pose estimation: A review of current techniques," in *2009 International Conference for Technical Postgraduates (TECHPOS)*. IEEE, 2009, pp. 1–8.

[24] Y. Shavit and R. Ferens, "Introduction to camera pose estimation with deep learning," *arXiv preprint arXiv:1907.05272*, 2019.

[25] "Aprendizaje profundo," https://www.hpe.com/es/es/what-is/deep-learning.html [Último acceso 7/septiembre/2022].

[26] "Clasificación redes neuronales," https://inteligencia-artificial.dev/tipos-redes-neuronales/ [Último acceso 7/septiembre/2022].

[27] "Redes adaline y madaline," https://numerentur.org/adaline_madaline_gadaline/ [Último acceso 7/septiembre/2022].

[28] "Configuración conexiones de las neuronas artificiales," https://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s5p1.htm [Último acceso 7/septiembre/2022].

[29] "Cnn. capas," https://bootcampai.medium.com/redes-neuronales-convolucionales-5e0ce960caf8 [Último acceso 7/septiembre/2022].

[30] "Arreglo espacial en al salida de la convolución." https://cs231n.github.io/convolutional-networks/ [Último acceso 7/septiembre/2022].

[31] "Función de activación," https://aiofthings.telefonicatech.com/recursos/datapedia/funcion-activacion [Último acceso 7/septiembre/2022].

[32] "Max-poling," https://www.codificandobits.com/blog/padding-strides-maxpooling-stacking-redes-convolucionales/ [Último acceso 7/septiembre/2022].

[33] "Capa fully-connected," https://datascientest.com/es/convolutional-neural-network-es [Último acceso 7/septiembre/2022].

[34] "Arquitectura alexnet." https://datapeaker.com/big-data/analisis-de-desercion-de-empleados-mediante-regresion-log [Último acceso 7/septiembre/2022].

[35] "Arquitectura googlenet-50." https://lamaquinaoraculo.com/computacion/googlenet [Último acceso 7/septiembre/2022].

[36] "Teoría del método de retropropagación." https://datascience.eu/es/inteligencia-artificial/como-funciona-el-algoritmo-de-retropropagacion/ [Último acceso 7/septiembre/2022].

[37] "Teoría optimizadores." https://velascoluis.medium.com/optimizadores-en-redes-neuronales-profundas-un-enfoque-p C3%A1ctico-819b39a3eb5 [Último acceso 7/septiembre/2022].

[38] "Test, train and validation." https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 [Último acceso 7/septiembre/2022].

[39] "Entrenamiento cnn," https://hmong.es/wiki/Convolutional_neural_network [Último acceso 7/septiembre/2022].

[40] "Neuronas artificiales," https://artyco.com/que-son-las-redes-neuronales-y-cual-es-su-aplicacion-en-el-marketing/ [Último acceso 7/septiembre/2022].

[41] "Clasificación de las redes neuronales según el método de aprendizaje," https://www.diegocalvo.es/clasificacion-de-redes-neuronales-artificiales/ [Último acceso 7/septiembre/2022].

[42] "Teoría de los algoritmos de agrupamiento," https://www.freecodecamp.org/espanol/news/8-algoritmos-de-agrupacion-en-clusteres-en-el-aprendizaje-automatico-que-todos-los-cientificos-de-datos-deben-cono [Último acceso 7/septiembre/2022].

[43] "Teoría python." https://quid.solutions/python-la-principal-herramienta-para-el-deep-learning/ [Último acceso 7/septiembre/2022].

[44] "Teoría librerías python." https://iasolver.es/6-librerias-de-python-para-machine-learning/ [Último acceso 7/septiembre/2022].

[45] "Teoría librería tensorflow," https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/ [Último acceso 7/septiembre/2022].

[46] "Teoría keras." https://piperlab.es/glosario-de-big-data/keras/ [Último acceso 7/septiembre/2022].

[47] "Enlace al git oficial de proyecto mmpose," https://github.com/open-mmlab/mmpose [Último acceso 7/septiembre/2022].

[48] "Enlace al github oficial del proyecto voxelpose." https://github.com/microsoft/voxelpose-pytorch [Último acceso 7/septiembre/2022].

[49] "Enlace al github oficial del proyecto videpose3d." https://github.com/facebookresearch/VideoPose3D [Último acceso 7/septiembre/2022].

[50] "Enlace al github del sistema de stridedtransformer-pose3d." https://github.com/Vegetebird/StridedTransformer-Pose3D [Último acceso 7/septiembre/2022].

[51] "Enlace al github del sistema de mhformer." https://github.com/Vegetebird/MHFormer [Último acceso 7/septiembre/2022].

[52] "Enlace al paper del sistema de mhformer." https://arxiv.org/pdf/2111.12707.pdf [Último acceso 7/septiembre/2022].

[53] "Articulaciones human3.6m," http://vision.imar.ro/human3.6m/readme_challenge.php [Último acceso 7/septiembre/2022].

[54] C. S. Catalin Ionescu, Fuxin Li, "Latent structured models for human pose estimation," in *International Conference on Computer Vision*, 2011.

[55] "Paper human3.6m." http://vision.imar.ro/human3.6m/pami-h36m.pdf [Último acceso 4/septiembre/2021].

[56] "Página web oficial humaneva dataset," http://humaneva.is.tue.mpg.de/datasets_human_1 [Último acceso 4/septiembre/2021].

[57] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrun," *IEEE Transactions on Acoustics, Speech and Signal processing*, vol. ASSPS-34, no. 1, pp. 52–59, February 1986.

[58] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt, "Monocular 3d human pose estimation in the wild using improved cnn supervision," in *3D Vision (3DV), 2017 Fifth International Conference on*. IEEE, 2017. [Online]. Available: http://gvv.mpi-inf.mpg.de/3dhp_dataset

[59] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *Proceedings of the British Machine Vision Conference*, 2010, doi:10.5244/C.24.12.

[60] "Documento de bmv." http://sam.johnson.io/research/publications/johnson10bmvc.pdf [Último acceso 7/septiembre/2022].

[61] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, P. Fua, H.-P. Seidel, H. Rhodin, G. Pons-Moll, and C. Theobalt, "Xnect: Real-time multi-person 3d motion capture with a single rgb camera," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 82–1, 2020.

[62] "Información sobre gnu/linux en wikipedia," http://es.wikipedia.org/wiki/GNU/Linux [Último acceso 21/diciembre/2021].

[63] L. Lamport, *LaTeX: A Document Preparation System, 2nd edition*. Addison Wesley Professional, 1994.

[64] J. D. Blischak, E. R. Davenport, and G. Wilson, "A quick introduction to version control with git and github," *PLoS computational biology*, vol. 12, no. 1, p. e1004668, 2016.

[65] G. Van Rossum *et al.*, "Python programming language." in *USENIX annual technical conference*, vol. 41, no. 1, 2007, pp. 1–36.

[66] Q. Hu, L. Ma, and J. Zhao, "Deepgraph: A pycharm tool for visualizing and understanding deep learning models," in *2018 25th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2018, pp. 628–632.

Apéndice A

Herramientas y recursos

Las herramientas necesarias para la elaboración del proyecto han sido:

- Elementos *hardware*
  - PC compatible
  - Cámaras de vídeo y profundidad (3D y RGB) [61]
- Elementos *software*
  - Sistema operativo GNU/Linux [62]
  - Procesador de textos LaTeX [63]
  - Control de versiones GIT [64]
  - Python y librerías de soporte [65]
  - Entorno de desarrollo PyCharm [66]

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá