UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS TELECOMUNICACIÓN

Reconocimiento de la actividad humana mediante aprendizaje profundo en imágenes de vídeo y sobre dataset multimodal

Autor:

Ángela González de Diego

Tutor/es:

Dr. D. Mario Martínez Zarzuela

Valladolid, Julio de 2022

TÍTULO:	**Reconocimiento de la actividad humana mediante aprendizaje profundo en imágenes de vídeo y sobre dataset multimodal**
AUTOR:	**Ángela González de Diego**
TUTOR:	**Dr. D. Mario Martínez Zarzuela**
DEPARTAMENTO:	**Teoría de la Señal y las Comunicaciones e Ingeniería Telemática**

Tribunal

PRESIDENTE:	**Dra. Dª. Míriam Antón Rodríguez**
SECRETARIO:	**Dr. D. Mario Martínez Zarzuela**
VOCAL:	**Dr. D. David González Ortega**
SUPLENTE 1:	**Dr. D. Francisco Javier Díaz Pernas**
SUPLENTE 2:	**Dr. D. Carlos Gómez Peña**

FECHA:	**Julio de 2022**
CALIFICACIÓN:	

Agradecimientos

En primer lugar, quisiera agradecer a Mario Martínez Zarzuela por su labor como tutor. Particularmente, por los conocimientos aportados sobre la materia de estudio así como por toda la ayuda y apoyo brindado durante la realización del presente trabajo.

Agradecer también al Grupo de Telemática e Imagen (GTI) por la ayuda proporcionada durante los últimos meses.

Por último, a mi familia, por su paciencia y apoyo incondicional a lo largo de los años y a mis amigos, por haberme acompañado y apoyado durante todo este tiempo.

Resumen

El campo del Reconocimiento de la Actividad Humana (HAR) se encuentra en auge debido a la creciente demanda de análisis de vídeo aplicado al ámbito médico. No obstante, la tarea de predicción de actividades en una secuencia de vídeo no es trivial, puesto que existen numerosos factores como la iluminación o el ángulo de captura, que afectan al reconocimiento.

El objetivo del trabajo es poder realizar este Reconocimiento de la Actividad Humana haciendo uso de Aprendizaje Profundo (*Deep Learning*), más concretamente, mediante una Red Neuronal. La red utilizada permite ejercer la tarea de clasificación de secuencias de imágenes. Para la extracción de características de las imágenes se emplean capas convolucionales 3D, asimismo, se emplean bloques residuales para mitigar el problema del desvanecimiento de gradiente observado en redes con un elevado número de capas. Trabajos previos han realizado estimación de poses de las mismas secuencias de vídeo, así como han llevado a cabo el HAR mediante Aprendizaje Profundo haciendo uso de datos provenientes de sensores.

Debido al aumento en el uso de sistemas de captura ópticos para la adquisición de datos, han surgido grandes datasets de refencia. No obstante, el trabajo se centra en el reconocimiento de actividades con relevancia en el ámbito médico, razón por la cual se ha hecho uso del dataset adquirido por el grupo de investigación. En consecuencia, se ha llevado a cabo el reconocimiento de 13 actividades realizadas por 37 sujetos diferentes.

El entrenamiento de la red para dicho dataset ha sido realizado tanto desde cero, como mediante el uso de *transfer learning*. Se ha observado como el empleo de un modelo pre-entrenado permite llegar al punto de convergencia de la red más rápidamente, ahorrando además capacidad computacional. Además, se muestran las dificultades del reconocimiento de datos provenientes de sistemas de captura ópticos, como son la dificultad en clasificación de actividades con movimiento reducido, o actividades bimanuales.

Palabras clave

Aprendizaje Porfundo, Redes Neuronales, Reconocimiento de la actividad humana, Visión Artificial, Aprendizaje por transferencia, TAO Toolkit.

Abstract

Human Activity Recognition (HAR) has garnered a lot of attention due to the growing demand for video analysis applied to the medical field. However, the task of predicting activities in video sequences is not trivial, since there are numerous factors that affect the recognition, such as lighting or the viewpoint.

The purpose of this work is to carry out Human Activity Recognition using Deep Learning, more specifically, through Neural Networks. The network performs the task of classifying image sequences. 3D convolutional layers are used to extract image features, and residual blocks are used to mitigate the problem of gradient vanishing observed in networks with a large number of layers. Previous works have estimated poses in the same video sequences that were employed. Moreover, they have also carried out HAR through Deep Learning using data acquired from sensors.

Due to the growing popularity of optical capture systems for data acquisition, a large number of benchmark datasets have emerged. Nevertheless, this work focuses on the recognition of activities relevant in the medical field, consecuently, the dataset employed has been the one acquired by the research group. Therefore, 13 activities carried out by 37 different subjects have been classified.

The network's training has been conducted both from scratch, and by transferring learning from a previously trained model. It has been observed how the use of a pre-trained model allows reaching convergence faster, thus saving computational cost. In addition, the results exhibit the limitations of recognizing data from optical capture systems, such as the difficulty of classifying activities with reduced movement, or bimanual activities.

Keywords

Deep Learning, Neural Networks, Human Activity Recognition, Computer Vision, Transfer Learning, TAO Toolkit.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El presente Trabajo de Fin de Grado (TFG) ha sido desarrollado dentro del Grupo de Telemática e Imagen (GTI) de la Universidad de Valladolid (UVa). El trabajo se ubica dentro del campo de la Inteligencia Artificial, más concretamente del uso de Aprendizaje Profundo para llevar a cabo reconocimiento de actividades. Para ello se ha hecho uso de una base de datos multimodal formada por 13 actividades relevantes dentro del ámbito médico. Para llevar a cabo el reconocimiento de las acciones se ha hecho uso de Redes Neuronales.

1.1. Motivación

La Visión Artificial (*Computer Vision*) es una rama de la Inteligencia Artificial que permite extraer información relevante de imágenes digitales, vídeos y otros *insputs* visuales. Entre las técnicas empleadas en Visión Artificial, existe la posibilidad de emplear Redes Neuronales Artificiales (*Artificial Neural Networks*) como herramienta para procesar todos los datos obtenidos. Las Redes Neuronales Artificiales son un campo de investigación en auge, gracias a importantes avances alcanzados con nuevas técnicas de Aprendizaje Profundo (*Deep Learning*), que permiten extraer características ocultas a simple vista en nuestros datos, por la cantidad de datos que se manejan. El Aprendizaje Profundo se integra dentro de las disciplinas de Aprendizaje Automático (*Machine Learning*) debido a que son los propios algoritmos los que aprenden y no es una persona quien indica cómo realizar la tarea, al contrario que en otros métodos dentro de la Inteligencia Artificial (Voulodimos et al., 2018).

Recientemente se observan avances en Visión Artificial gracias a técnicas de Aprendizaje Profundo, las cuales nos permiten generar un modelo de cinemática del movimiento del cuerpo humano a partir de una imagen 2D. Encontramos un ejemplo en (Kidziński et al., 2020), donde se estudia el análisis cuantitativo de movimiento a través de Redes Neuronales de Aprendizaje Profundo utilizando vídeos de una única cámara. Este estudio presenta un método para predecir parámetros del movimiento clínicamente relevantes, a través del vídeo de un paciente. La utilización de Redes Neuronales para el reconocimiento de actividades tanto mediante vídeo (Hassan et al., 2018; Kidziński et al., 2020) como a través de sensores (González-Alonso et al., 2020; Ordóñez and Roggen, 2016) es una potente herramienta con un amplio abanico de uso.

El sistema médico ha sido uno de los grandes beneficiados del desarrollo del reconocimiento de acciones y de las innovaciones provenientes del IoT (*Internet of Things*). La investigación ha tendido al desarrollo de sistemas que permitan llevar a cabo la monitorización del paciente de for-

ma remota, dado que cuanto mayor sea el seguimiento físico y psicológico disponible, mejor será la recuperación, rehabilitación y menor tiempo tendrá que permanecer ingresado el paciente en el hospital, mejorando así su calidad de vida.

Entre las aplicaciones más destacables del HAR mediante Aprendizaje Profundo aplicado al ámbito médico encontramos:

- El cuidado de personas mayores en el hogar mediante detección de caídas y situaciones de riesgo (Wu and Xue, 2008) así como la monitorización de sus actividades durante la vida diaria (Najafi et al., 2003).
- Monitorización de personas que necesiten realizar alguna terapia física de rehabilitación debida a trastornos muscoloesqueléticos (fracturas o discapacidades físicas), incluso en su entorno natural (Bartalesi et al., 2005; Walker et al., 1997).
- La monitorización de pacientes y diagnosis médico (Jiang et al., 2008).

Otro punto a destacar, es el gran incremento en la cantidad de datos recopilados y consecuentemente, los requerimientos en la capacidad de cómputo necesaria para el procesamiento de los mismos. Por ende, la investigación en los últimos años ha tendido hacia la búsqueda de sistemas que sean capaces de tomar experiencias de tareas pasadas y utilizarlas para mejorar el rendimiento de una nueva tarea, a la cual no se ha enfrentado con anterioridad. Dicha técnica es denominada *transfer learning*, la cual no sólo proporciona un ahorro computacional importante sino que también permite reducir la cantidad de datos necesarios para llevar a cabo la tarea. Esta técnica puede ser aplicada a muchos campos de estudio, no obstante, su contribución más destacada se encuentra en el reconocimiento de acciones humanas (Cook et al., 2013).

Otra contribución a la reducción del coste computacional ha venido dada por el desarrollo de tecnologías para Computación en el Borde (Edge Computing), las cuales migran la computación y las capacidades y recursos de almacenamiento de la nube, al borde de la red para satisfacer las necesidades de la industria de las TIC. Ejemplos de estas necesidades son los negocios en tiempo real, la optimización de datos, inteligencia de aplicaciones, seguridad y privacidad. Además, cumple con los requisitos de baja latencia y alto ancho de banda en la red. La computación en el borde se ha convertido en un popular tema de investigación en la actualidad (Cao et al., 2020).

1.2. Hipótesis y objetivos

La hipótesis de partida es que la utilización combinada de Deep Learning y Edge Computing es válida para lograr una mejor monitorización de personas de manera remota. Es decir, sumarle a esta base de reconocimiento de actividades todas las ventajas previamente mencionadas de las tecnologías de borde: la mejora en el rendimiento de la transmisión de datos, garantizar el procesamiento en tiempo real, reducir el tiempo de respuesta y proporcionar seguridad y privacidad. Para probar esta hipótesis es necesario desarrollar una base de sistema de monitorización de movimientos y reconocimiento de actividades que pueda ser utilizada en sistemas embebidos.

El objetivo de este proyecto es diseñar un sistema de monitorización de actividades mediante el uso de redes neuronales para la detección del movimiento proveniente de sistemas de captura ópticos. El sistema diseñado podrá ser exportado para su ejecución en sistemas embebidos. Asimismo, para poder llevarlo a cabo se abordarán a su vez los siguientes objetivos secundarios:

- Análisis de la base de datos multimodal aplicada al reconocimiento de actividades así como de las métricas obtenidas.
- Comparación de los datos recogidos mediante el sistema óptico y los análogos recogidos mediante el sistema no óptico (IMUs).
- Estudio de la eficiencia computacional aportada por la técnica de *transfer learning*.

Para abordar los objetivos previamente descritos se ha llevado a cabo el diseño de un modelo de reconocimiento de acciones, el cual puede ser exportado e implementado mediante DeepStream. Las actividades contenidas en la base de datos multimodal utilizada se caracterizan por su relevancia en el ámbito médico.

1.3. Fases y métodos

Para poder alcanzar los objetivos planteados en la sección anterior, ha sido necesario seguir la secuencia de pasos:

i. Familiarización con los conceptos de Inteligencia Artificial y Aprendizaje Profundo. Adquisición de conocimientos base sobre el funcionamiento de las Redes Neuronales: estructura en capas, algoritmos más relevantes así como la elección del valor de los hiperparámetros involucrados en el entrenamiento de redes.

ii. Aprendizaje sobre las disintas arquitecturas de Redes Neuronales así como los campos de aplicación de cada una de ellas.

iii. Búsqueda bibliográfica sobre captura de movimientos y reconocimiento de acciones. Revisión de las bases de datos disponibles para el HAR mediante sistemas de captura ópticos. Elección de la base de datos multimodal a utilizar para llevar a cabo el reconocimineto de acciones.

iv. Adquisición de habilidades para el manejo del software TAO Toolkit, que permite realizar tanto el entrenamiento como evaluación e inferencia de la Red Neuronal ReSNet-18 mediante *notebooks* de Jupyter. Incorporación de código al *notebook* para la obtención de matrices de confusión así como de las métricas asociadas a esta mediante el lenguaje Python.

v. Reacondicionamiento de las bases de datos a utilizar. Descomposición de los vídeos del dataset HDMB51 en fotogramas para su introducción en la Red Neuronal. Separación en *set* de entrenamiento y *set* de validación y *test*. Reorganización del dataset multimodal utilizado en el HAR. Descomposición de los vídeos en fotogramas y separación en *set* de entrenamiento y *set* de validación y *test*.

vi. Realización de diferentes pruebas buscando los valores óptimos de los hiperparámetros de la red.

vii. Análisis de los diferentes resultados obtenidos y comparación con los resultados de trabajos previos.

viii. Extracción de conclusiones a partir de los resultados obtenidos y planteamiento de líneas futuras de investigación.

1.4. Descripción del documento

El presente apartado describe la estructura del TFG, el cual se encuentra dividido en 5 capítulos. El primer capítulo realiza una breve introducción al Trabajo de Fin de Grado y describe las fases y métodos seguidos a lo largo de este, junto con el planteamiento de los objetivos a cumplir. El resto de los capítulos se describen brevemente a continuación:

- **Capítulo 2. Fundamentos teóricos del Aprendizaje Profundo.** En este capítulo se hará una introducción a Redes Neuronales, haciendo un resumen de los algoritmos más relevantes. Después se comentarán los principales parámetros implicados en el entrenamiento, para finalizar comentando las distintas arquitecturas disponibles así como el campo de aplicación de cada una.
- **Capítulo 3. Revisión de datasets para el reconocimiento de la actividad humana.** Para comenzar, se hará una breve introducción a los diferentes sistemas de captura de datos. Posteriormente, se enumeran las bases de datos que marcan un punto de referencia en el reconocimiento de acciones. Por último, se describe la base de datos multimodal utilizada para el entrenamiento e inferencia de la Red Neuronal utilizada.
- **Capítulo 4. Materiales y métodos.** En primer lugar, se describirá la reorganización llevada a cabo en la estructura de la base de datos así como su re-etiquetado. Acto seguido, se explicará brevemente la estructura y características de las red ResNet-18 utilizada y se comentarán los resultados obtenidos para diferentes entrenamientos.
- **Capítulo 5. Conclusiones y Líneas Futuras.** Finalmente se abordan las distintas conclusiones del trabajo y se comentarán la posible continuidad en futuros proyectos de investigación.

1.5. Materiales utilizados

Para poder llevar a cabo los objetivos planteados en secciones previas, se ha requerido hacer uso de una serie de elementos, tanto *hardware* como *software*, los cuales se detallan a continuación.

1. Materiales *hardware* utilizados: PC ubicado en el laboratorio del GTI, situado en la ETSIT.
   - CPU: Intel(R) Core(TM) i5-4690 CPU @ 3.50GHz.
   - GPU: NVIDIA GeForce GTX 1060 6GB.
   - Placa base: Gigabyte B85M-HD3.
   - RAM: 8GB DDR3 kingston.
2. Materiales *software* utilizados.
   - Jupyter Lab.
   - virtualenv.
   - nvidia-tao. Es el *launcher* de TAO Toolkit. La tabla 1.1 muestra los pre-requisitos para la instalación de la herramienta, así como la versión instalada de ellos.

Pre-requisito	Versiones válidas	Versión instalada
Ubuntu LTS	>18.04	18.04.3
Python	⩾3.6.9	3.6.9
docker-ce	>19.03.5	20.10.12
docker-API	>1.40	1.41
nvidia-container-toolkit	>1.3.0-1	1.7.0
nvidia-docker2	2.5.0-1	2.9.0-1
nvidia-driver	>455	470.103.01
python-pip	>21.06	21.3.1

Tabla 1.1. Programas software necesarios para la ejecución de TAO Toolkit así como la versión instalada de los mismos.

Capítulo 2

Fundamentos teóricos del Aprendizaje Profundo

2.1. Introducción al Aprendizaje Profundo

El Aprendizaje Profundo (DL, *Deep Learning*) es un subcampo de la Inteligencia Artificial (AI, *Artificial Intelligence*). La figura 2.1 muestra un diagrama de Venn con la relación entre la Inteligencia Artificial, el Aprendizaje Automático (ML, *Machine Learning*) y el Aprendizaje Profundo.

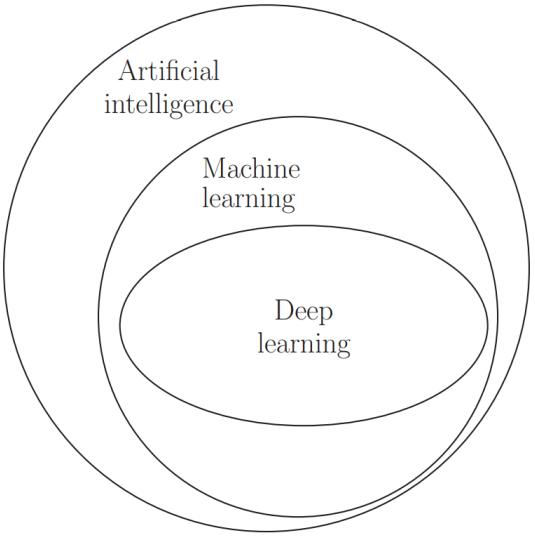


Figura 2.1. Diagrama de Venn representando la relación entre la Inteligencia Artificial, Machine Learning y Deep Learning. Imagen de (Kelleher, 2019).

Un algoritmo de *Machine Learning* es aquel capaz de realizar un aprendizaje a partir de unos datos para poder llevar a cabo una tarea. Esto implica el desarrollo y la evaluación de algoritmos que permiten a un ordenador extraer funciones de un conjunto de datos (*dataset*). Entre las tareas típicamente desempeñadas por los algoritmos de *Machine Learning*, una de las más comunes es la de clasificación. En ella, se busca asignar a cada entrada (*input*), una de k posibles categorías como salida (*output*). Para ello, el algoritmo genera una función de la forma $y = f(\boldsymbol{x})$, que asigna a la entrada descrita por el vector $\boldsymbol{x}$, una categoría identificada por un código numérico y (Goodfellow et al., 2016).

Un modelo de Aprendizaje Profundo (*Deep Learning*) es un modelo capaz de aprender representaciones de datos mediante el aprendizaje a través de diferentes capas, en las que las representaciones cada vez son más significativas. Permite tomar decisiones identificando y extrayendo dichas representaciones a partir de grandes conjuntos de datos, obteniendo así una conversión precisa de un conjunto de entradas a un conjunto de buenos resultados. El aprendizaje de estas representaciones se lleva a cabo mediante el uso de redes neuronales. Este modelo es particularmente de interés cuando se utilizan grandes datasets y se manejan datos complejos (Chollet, 2018).

Una red neuronal consiste en un conjunto de unidades o nodos de procesamiento conectadas de manera unidireccional y estructuradas en capas. La profundidad del modelo (*depth*) hace referencia al número de capas que éste utiliza. Esta red neuronal se encuentra dotada de unos parámetros internos, llamados pesos, que son los que permiten el mapeo de la entrada (*input*) a una salida (*output*), es decir, la transformación de la entrada implementada por una capa viene parametrizada por sus pesos. La primera capa del modelo es la denominada *input layer*, aquella en la cual se definen los nodos de entrada. Esta se caracteriza por estar definida de forma externa. La última capa del modelo es la *output layer*, aquella en la que la salida de los nodos forma la salida de la red. El resto de capas intermedias son las denominadas *hidden layers* y son las encargadas del cómputo. Cada capa se encuentra formada por una serie de unidades de procesamiento trabajando en paralelo. Estas unidades de procesamiento (también llamadas neuronas artificiales) reciben una entrada proveniente de las unidades de la capa anterior y computan una salida, a la que denominamos activación (Kelleher, 2019).

2.1.1. *Feedforward Neural Networks*

La arquitectura *Feedforward Neural Network* o Perceptrón Multicapa, es una red neuronal formada por perceptrones como unidades de procesamiento. En cada capa de la red se lleva a cabo un mayor nivel de abstracción, tomando decisiones cada vez más complejas. El objetivo de esta red neuronal es encontrar una aproximación a la función f^* que permite llevar a cabo la tarea de clasificación de mapear una entrada $\mathbf{x}$ a una categoría de salida $y = f^*(\boldsymbol{x})$. Para ello, la red deberá estimar y aprender los parámetros $\boldsymbol{\Theta}$ que proporcionan la mejor aproximación a esa función, de forma que se obtenga $y = f(x; \Theta)$ (Goodfellow et al., 2016).

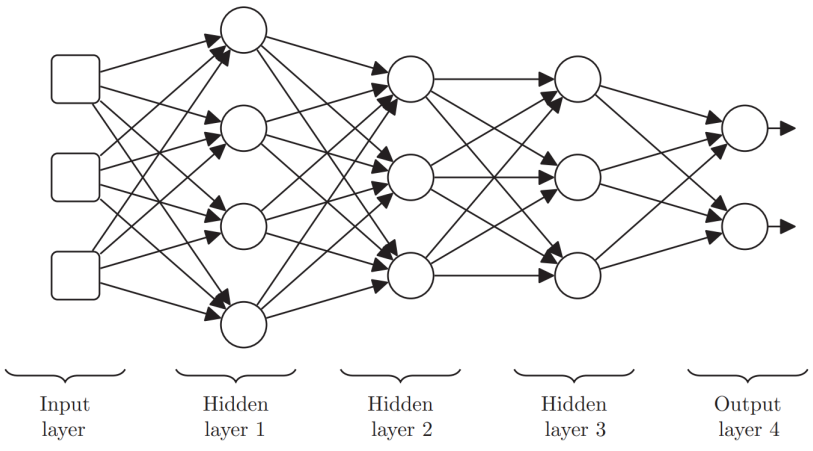


Figura 2.2. Esquema de una arquitectura Feedforward Neural Network. Imagen de (Kelleher, 2019).

El perceptrón es un tipo de unidad de procesamiento básica. Se caracteriza por tener una serie de entradas x_j y pesos w_j, los cuales asignan en función de su valor, mayor o menor relevancia a una determinada entrada. La salida de la neurona, llamada activación, viene dada en función de si la suma ponderada entre las entradas y los pesos supera un determinado umbral. Cabe destacar que, para las capas intermedias, las entradas serán las activaciones de las neuronas de la capa inmediatamente anterior (Aggarwal, 2018).

$$salida = \begin{cases} 0 & \text{si} \quad \sum_j w_j x_j \leq umbral \\ 1 & \text{si} \quad \sum_j w_j x_j > umbral \end{cases} \tag{2.1}$$

Este umbral es llamado sesgo y mide la facilidad con la que el perceptrón se activa (es decir, con cuanta facilidad proporciona una salida de 1). Haciendo el cambio de notación $b \equiv -umbral$, y escribiendo de forma vectorial las entradas y los pesos, obtenemos:

$$salida = \begin{cases} 0 & \text{si} \quad \boldsymbol{w^T} * \boldsymbol{x} + b \leq 0 \\ 1 & \text{si} \quad \boldsymbol{w^T} * \boldsymbol{x} + b > 0 \end{cases} \tag{2.2}$$

Donde $\boldsymbol{x}$ y $\boldsymbol{w}$ son vectores columna de *n* elementos de la forma:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \tag{2.3}$$

Podemos separar este comportamiento del perceptrón en dos funciones. En primer lugar, el cálculo de la suma ponderada de las entradas y los pesos, al que además añadimos un umbral, $z = \boldsymbol{w^T} * \boldsymbol{x} + b$. En segundo lugar, aplicar una función umbral al valor z que permita realizar una decisión, esta función es denominada función de activación. La función que utiliza el perceptrón es la función escalón ilustrada en la figura 2.3. De esta forma la salida de la neurona es (Kelleher, 2019):

$$salida = \begin{cases} 0 & \text{si} \quad g(z) \leq 0 \\ 1 & \text{si} \quad g(z) > 0 \end{cases} \tag{2.4}$$

El objetivo es poder realizar modificaciones de forma gradual tanto en los pesos como en los sesgos para conseguir que la red se acerque al comportamiento deseado, la función f^*. Sin embargo, a diferencia de lo esperado, una pequeña variación en los pesos o sesgos de un perceptrón puede conllevar a un cambio radical en el comportamiento de la red. Esto se debe a que el modelo de red definido es lineal, y la mayoría de las situaciones que se buscan describir son modelos no lineales. Para solventarlo, se introduce una función no lineal como función de activación. Esta permite realizar un mapeado no lineal de la suma ponderada al valor de activación de la neurona.

Consecuentemente, la neurona realiza el proceso de mapeo de la entrada (*input*) a una salida (*output*) en dos fases, las cuales vemos ilustradas en la figura 2.4. En la primera, se realiza la suma ponderada de las entradas con los pesos, de la forma descrita anteriormente. En la figura, esta operación viene descrita por el símbolo de sumatorio $\sum$. Al resultado se le aplica una segunda función, la función de activación la cual denotamos por el símbolo φ, y realiza el mapeo al valor de activación de la neurona. Matemáticamente lo podemos expresar de la forma (Kelleher, 2019):

$$salida = \varphi(\boldsymbol{w^T} * \boldsymbol{x} + b) \tag{2.5}$$

Existe una gran variedad de funciones de activación, entre las cuales cabe destacar la función sigmoide, la función ReLU, la función tangente hiperbólica y la función Softmax.

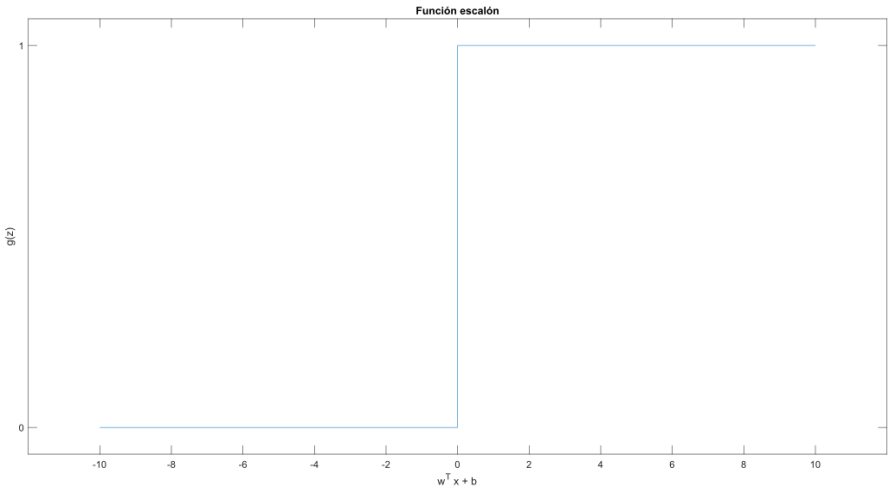


Figura 2.3. Gráfica de la función escalón. Imagen generada mediante la herramienta Matlab.

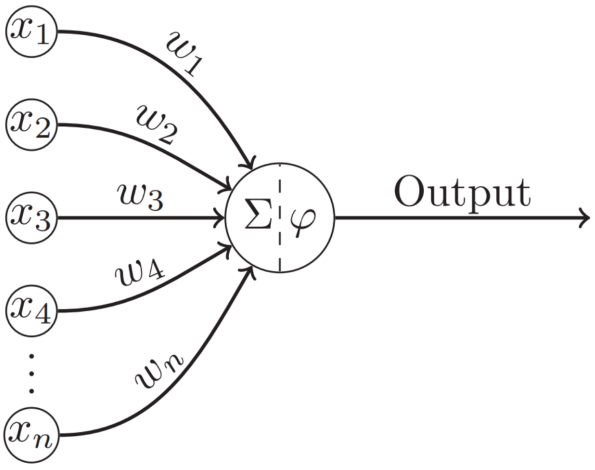


Figura 2.4. Esquema de funcionamiento de una neurona. Imagen de (Kelleher, 2019).

Función sigmoide

También conocida como función logística. En la gráfica de la figura 2.5 se observa que mientras que la función escalón se caracterizaba por tener un cambio brusco de 0 a 1, la función sigmoide tiene una transición suave. A causa de dicha transición, pequeños cambios en los pesos y en el sesgo, Δw y Δb, causarán a su vez pequeños cambios en la salida de la neurona.

La función sigmoide suele ser utilizada en la capa de salida (*output layer*) de la red y se aplica frecuentemente en problemas de clasificación binaria o en modelado de tareas de regresión logística (Kelleher, 2019).

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.6}$$

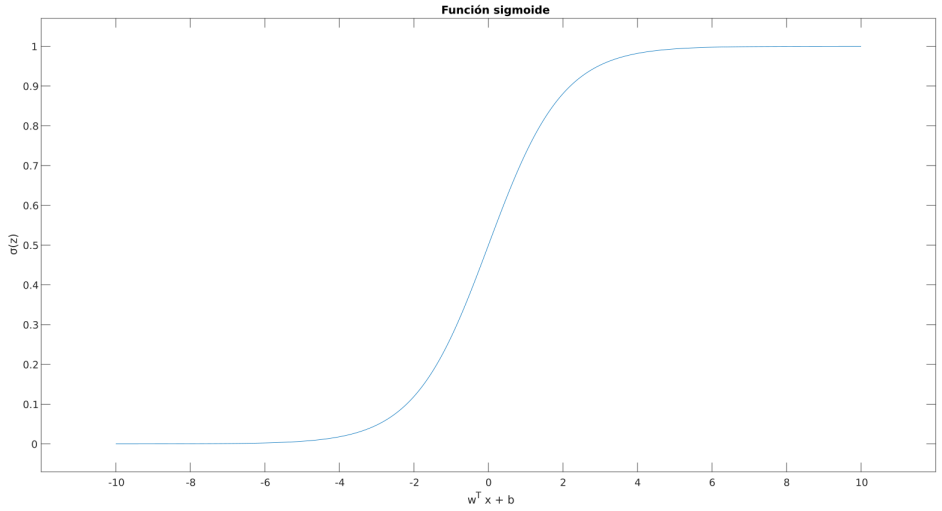


Figura 2.5. Gráfica de la función sigmoide. Imagen generada mediante la herramienta Matlab.

Función tangente hiperbólica

La función tangente hiperbólica, conocida como función *tanh*, viene definida por la ecuación 2.7. Esta se caracteriza por proporcionar una salida entre -1 y 1, estando centrada en cero como vemos en la figura 2.6. Esta característica provoca que la media de las salidas se encuentre con mayor probabilidad entorno a cero, favoreciendo así al algoritmo de *backpropagation*. (Ding et al., 2018; Nwankpa et al., 2018).

El uso más frecuente de la tangente hiperbólica ha sido en Procesado de Lenguaje Natural (NPL, *Natural Language Processing*) mediante Redes Neuronales Recurrentes (RNN, *Recurrent Neural Networks*) (Dauphin et al., 2017) (Dauphin et al., 2017), así como en tareas de reconocimiento del habla (Maas et al., 2013).

$$tanh(z) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.7}$$

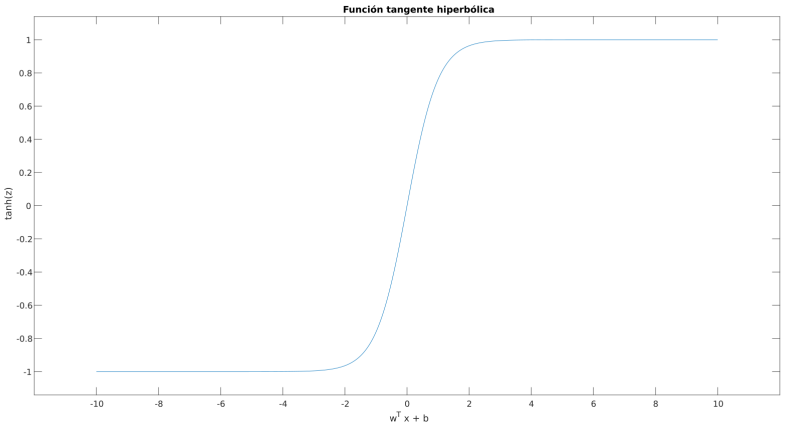


Figura 2.6. Gráfica de la función tangente hiperbólica. Imagen generada mediante la herramienta Matlab.

Función ReLU

La función *Rectified Linear Unit* (ReLU) rectifica a cero todos los valores de z negativos y viene dada por la ecuación 2.8. Esta función fue propuesta por Nair y Hinton en 2010 (Fürnkranz and Society, 2010). La principal ventaja que ofrece es computacionalmente, dado que no lleva a cabo cálculos con exponenciales ni divisiones. Mayoritariamente se utiliza en las neuronas de las capas intermedias (*hidden layers*) en conjunto con alguna otra función de activación para la capa de salida para así realizar tareas de clasificación (Krizhevsky et al., 2012) o de reconocimiento del habla (Maas et al., 2013).

$$f(z) = max(0, z) = \begin{cases} 0 & \text{si} \quad z \leq 0 \\ 1 & \text{si} \quad z > 0 \end{cases} \tag{2.8}$$

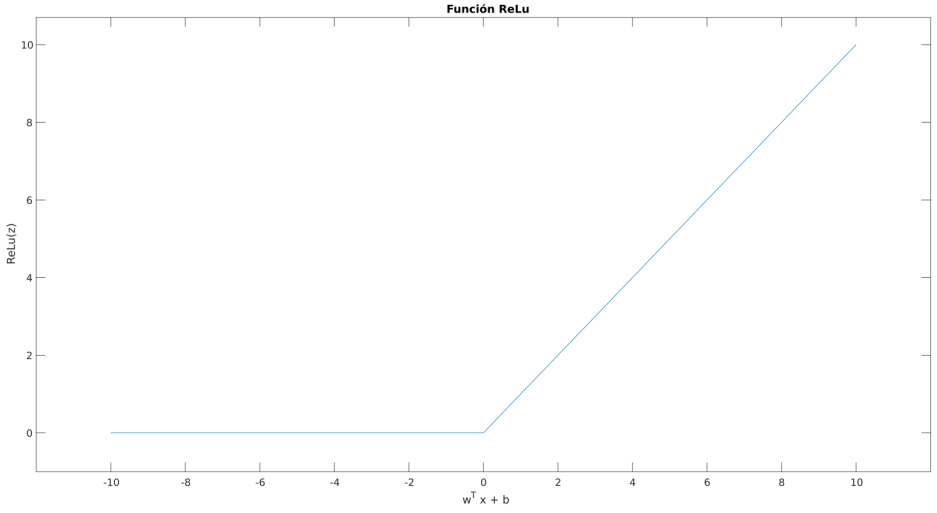


Figura 2.7. Gráfica de la función ReLU. Imagen generada mediante la herramienta Matlab.

Función Softmax

La función Softmax produce salidas entre 0 y 1, caracterizándose por calcular la distribución de probabilidad de un vector formado por números reales. Al realizar cálculos de probabilidades, la suma de todas ellas genera como resultado el valor 1. La expresión de la función viene dada por:

$$f(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} = \frac{e^{z_i}}{\mathbf{1^T} e_j^z} \tag{2.9}$$

Siendo $\mathbf{1}$ y $\boldsymbol{e^z}$ los vectores columna de tamaño n siguientes:

$$\mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \boldsymbol{e^z} = \begin{bmatrix} e^{z_1} \\ e^{z_2} \\ \vdots \\ e^{z_n} \end{bmatrix} \tag{2.10}$$

La función Softmax se emplea en modelos en los que el valor de salida es el valor de la probabilidad de cada clase, siendo la clase objetivo aquella con mayor valor de probabilidad (Nwankpa et al., 2018). En cuanto a la arquitectura de la red neuronal, la función suele ser utilizada en la capa

de salida (Krizhevsky et al., 2012). Su principal diferencia con la función Sigmoide es que esta última es utilizada en tareas de clasificación binaria mientras que la función Softmax es utilizada en tareas de clasificación multivariante (Nwankpa et al., 2018).

2.1.2. *Backpropagation*

En la sección anterior se ha explicado cómo en cada capa de la red neuronal se realiza la transformación de una serie de entradas a una salida mediante la relación de la ecuación 2.5, en la que el vector columna $\boldsymbol{w}$ representa los pesos y b el sesgo.

$$salida = \varphi(\boldsymbol{w^T} * \boldsymbol{x} + b) \tag{2.5}$$

Estos parámetros contienen la información que la red neuronal ha aprendido durante el entrenamiento. No obstante, dichos parámetros se inicializan de manera aleatoria en una primera aproximación, por lo que es de esperar que, al realizar la inicialización aleatoria la salida obtenida carezca de utilidad. Es necesario realizar un ajuste gradual de los pesos y sesgos en función de la salida proporcionada por la red. A esto se le llama entrenar a la red y se lleva a cabo realizando los siguientes pasos en bucle:

- Tomar un set de muestras de entrenamiento con sus correspondientes etiquetas.
- Introducirlas en la red y obtener su predicción.
- Calcular el error cometido por la red para ese set, realizando la diferencia entre las etiquetas verdaderas y las predicciones de la red.
- Actualizar el valor de los pesos y sesgos para minimizar el error previamente calculado.

Tras un número de iteraciones se llegará a un mínimo de la función del error, el cual denota que el error cometido en las predicciones de la red es pequeño. Para conseguir llegar a este mínimo es necesario aplicar un algoritmo de optimización (Chollet, 2018).

2.1.2.1. Función de costes

Una función de costes o función del error proporciona una medición sobre cómo de lejos se encuentras las predicciones realizadas por la red del valor real. En nuestro caso, denotaremos la función de costes como $J(\boldsymbol{\Theta})$, siendo $\boldsymbol{\Theta}$ el parámetro que engloba a nuestros parámetros a estimar, los pesos y sesgos.

MSE

Una forma de medir el error cometido por los estimadores es mediante el MSE (*Mean Squared Error*). El MSE mide la desviación general entre los estimadores $\hat{\theta}_m$ y el verdadero valor θ. En lo que a nuestro problema se refiere, el estimador es la predicción realizada por la red para nuestra muestra y el verdadero valor es la etiqueta de cada muestra.

$$MSE = E\left[\left(\hat{\theta}_m - \theta\right)^2\right] = Sesgo\left(\hat{\theta}_m\right)^2 + Var(\hat{\theta}_m) \tag{2.11}$$

La evaluación del MSE incluye la evaluación de un sesgo y una varianza, como denota la ecuación 2.11. Estas son dos fuentes de error diferentes que se encuentran en un estimador. El sesgo mide la desviación del estimador frente al verdadero valor, mientras que la varianza, mide

la desviación de una muestra frente al valor del estimador calculado (Goodfellow et al., 2016). La función de costes obtenida mediante el MSE es la que menor coste computacional conlleva. Esta función de costes es utilizada en regresión lineal (Zhao et al., 2010). No obstante, el gradiente del MSE tiende a sufrir el problema del desvanecimiento de gradiente cuando se usa en conjunto con la función de activación Softmax (Brox et al., 2019).

Cross-entrophy

La función de costes de la entropía cruzada (*cross entropy loss function*), compara la probabilidad predicha por la red para cada una de las clases con la salida real. En función de cómo de lejos se encuentre la probabilidad del valor real, se aplica una penalización. La característica principal de esta penalización se debe a su naturaleza logarítmica, cuanto mayor es la diferencia entre la predicción y el valor real, mayor es la función de pérdidas. Se puede observar en la figura 2.8, cómo el valor de la función de pérdidas logarítmica disminuye lentamente según la probabilidad de predicción se acerca a 1, y aumenta rápidamente con la disminución de esta (Wang et al., 2022).

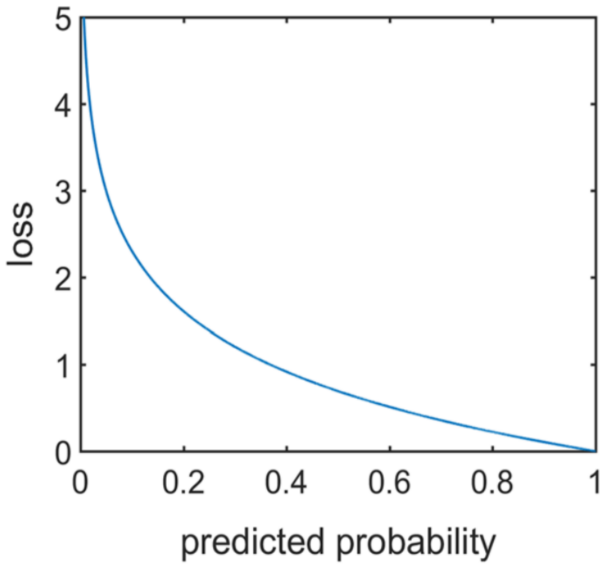


Figura 2.8. Función de pérdidas logarítmica. Imagen de (Wang et al., 2022).

Por consiguiente, el error de la entropía cruzada es menor que el error del MSE en cada iteración lo que conlleva a una convergencia más rápida (Bosman et al., 2020; Sangari and Sethares, 2015). Por esta razón, la entropía cruzada es utilizada en tareas de clasificación multivariante (Zhou et al., 2019).

2.1.2.2. Algoritmos de optimización

La optimización implica la maximización o minimización de una función $f(x)$ mediante la modificación de x. Cuando buscamos minimizar una función, la denominamos función de costes (*cost function*) o función de pérdidas (*loss function*). El algoritmo de *backpropagation* busca la minimización de la función de costes $J(\boldsymbol{\Theta})$, previamente mencionada.

Descenso de gradiente

Una primera aproximación para llevar a cabo la minimización de una función $f(x)$ es mediante el uso de la derivada, dado que esta nos proporciona la dirección de máximo aumento de una función. Haciendo pequeños cambios en x, en la dirección contraria a la derivada (dado que buscamos la dirección en la que la función decrece en mayor medida), es posible alcanzar un mínimo de la función. Este mínimo puede ser un mínimo relativo y no absoluto, no obstante, en el contexto del aprendizaje profundo únicamente se busca que este mínimo proporcione un valor de $f\,(x)$ suficientemente pequeño, pese a no ser un mínimo absoluto.

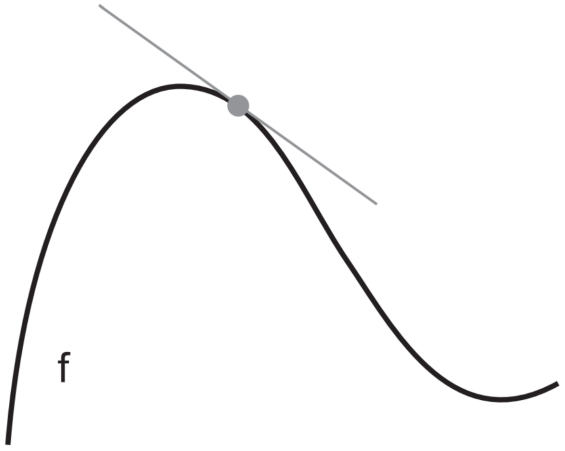


Figura 2.9. Pendiente de una función f en un punto p. Imagen de (Chollet, 2018).

El parámetro de entrada de nuestra función de costes $J(\mathbf{\Theta})$ es vectorial y no escalar como se había definido x previamente. En consecuencia, la operación que proporciona la dirección de máxima variación de la función no es la derivada, sino el gradiente. Para minimizar $J(\mathbf{\Theta})$, se realizarán pequeños cambios en $\mathbf{\Theta}$ en la dirección negativa del gradiente. Este método definido en la ecuación 2.12 se denomina método del descenso de gradiente (*steepest descent* o *gradient descent*). La cantidad de variación introducida en $\mathbf{\Theta}$ viene determinada por ϵ, la tasa de aprendizaje (*learning rate*) (Goodfellow et al., 2016).

$$\mathbf{\Theta}' = \mathbf{\Theta} - \epsilon \nabla_{\mathbf{\Theta}} J(\mathbf{\Theta}) \tag{2.12}$$

Esta tasa deberá tomar un valor suficientemente pequeño como para poder llevar a cabo una buena aproximación mediante la ecuación 2.12, de lo contrario, es posible llegar a obtener valores del gradiente para los que la función de costes aumenta. Debe tenerse en cuenta que también existe un compromiso entre el tamaño de ϵ y el coste computacional, puesto que cuanto menor es el valor de la tasa de aprendizaje más lento es el algoritmo (Aggarwal, 2018).

Stochastic Gradient Descend

Llevar a cabo el cálculo del gradiente para cada una de las entradas es computacionalmente caro, especialmente cuando tenemos un número de entradas elevado. Con esta motivación de acelerar el entrenamiento surge el descenso de gradiente estocástico (SGD, *Stochastic Gradient Descend*). La idea principal es realizar una estimación del gradiente mediante el cálculo de los gradientes de un pequeño número de entradas. Tomando un número de muestras de tamaño m suficiente (al que se denomina *mini-batch*), la media de los gradientes de estas m muestras se aproxima a la media del gradiente para todas las muestras. El funcionamiento del algoritmo de descenso de gradiente

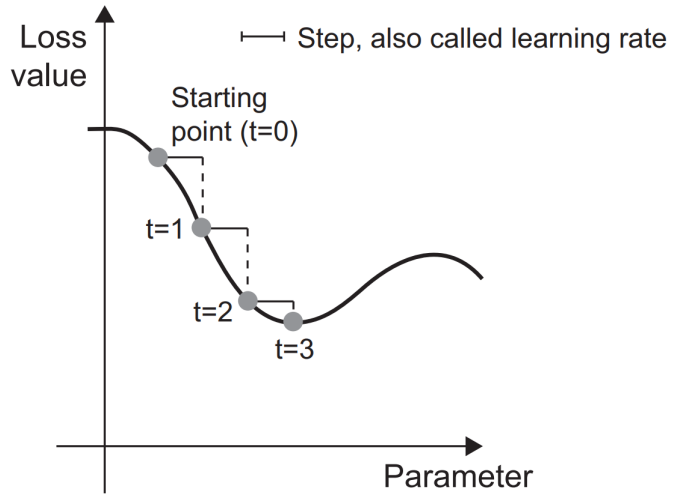


Figura 2.10. GD aplicado a una función 1D. Imagen de (Chollet, 2018).

estocástico se basa en la selección aleatoria de un *mini-batch*, el cual se utiliza para entrenar la red. A continuación, se selecciona de forma aleatoria un nuevo *mini-batch* y se repite el mismo proceso de forma continua hasta agotar las muestras de entrenamiento. En este momento se ha completado una época (*epoch*) de entrenamiento, tras ella se comenzará una nueva época y se realizara de nuevo el proceso descrito (Aggarwal, 2018; Ruder, 2017).

Este método de optimización es generalmente el utilizado por el algoritmo de *backpropagation*. Esto se debe no sólo a que el aprendizaje se realiza de manera más rápida, especialmente cuando se hace uso de grandes datasets, sino que, además, obtiene mejores resultados que el descenso de gradiente.

La estimación realizada del gradiente es ruidosa, lo que conlleva también a introducir ruido en la actualización de los parámetros Θ. Este ruido será el que permitirá a los valores actualizados de Θ realizar saltos y alcanzar diferentes mínimos locales, posiblemente menores que el mínimo más próximo al valor con que se habían inicializado de forma aleatoria los pesos y sesgos (Bottou, 2012). En (Heskes and Kappen, 1993; Orr, 1996), se pueden encontrar demostraciones simplificadas de dicho efecto.

Existen numerosas variantes del descenso del gradiente estocástico que tienen en cuenta los valores previos de los pesos y sesgos a la hora de realizar su actualización. Entre dichas variantes encontramos el SGD con momento, Adagrad, RMSProp, Adam, etc (Aggarwal, 2018).

Desvanecimiento y explosión de gradiente

El desvanecimiento de gradiente se trata del principal inconveniente del algoritmo de *backpropagation*. Cuando se realiza el entrenamiento con una red poco profunda (*shallow network*), formada por una o dos capas intermedias, el algoritmo funciona correctamente. Sin embargo, al utilizar redes más profundas, o bien la red no converge dado que no es capaz de hallar un valor óptimo para los pesos y sesgos, o bien, llegar a ese valor requiere de excesivos tiempos de entrenamiento. El problema, identificado por Sepp Hochreiter, es la forma en la que el algoritmo propaga los errores hacia atrás.

Al propagar el error (el gradiente), de una neurona a la siguiente, este se ve multiplicado por una serie de valores, menores que 1. Consecuentemente, la actualización del valor de los pesos y sesgos depende de los gradientes de las funciones de activación de cada nodo. El valor del error disminuye según se propaga por la red, en muchas ocasiones disminuye de manera exponencial con respecto a la distancia con la capa de salida. Cuando el error llega a las primeras capas de la red, su valor ha sufrido una importante reducción por lo que únicamente modifica los pesos y sesgos de forma leve, o incluso puede llegar a no modificarlos. Las primeras capas de la red son las encargadas de la extracción de las características de la entrada, las cuales las capas más profundas utilizan para el cálculo de las representaciones que determinan la salida de la red. Por esta razón, estas capas son de vital importancia y debido al desvanecimiento del gradiente su entrenamiento es excesivamente lento o ni se llegan a modificar los valores aleatorios con los que se inicializaron (Hochreiter, 1991).

Entre las posibles soluciones a este problema, encontramos la utilización de celdas LSTM (Hochreiter and Schmidhuber, 1997), utilización de funciones de activación que no saturan como la función ReLU (dado que no utiliza términos exponenciales) (Glorot et al., 2011), la inicialización de los pesos de manera adecuada (Glorot et al., 2011), la utilización del *batch normalization* (Ioffe and Szegedy, 2015) o el uso de otras arquitecturas como las ResNet, las cuales utilizan redes residuales. Estas redes incorporan conexiones con accesos directos que saltan una o varias capas, permitiendo que el gradiente fluya entre las diferentes capas.

Por el contrario, la explosión de gradiente se produce, como muestra (Pascanu et al., 2012), cuando pequeños cambios en los pesos y sesgos de una Red Neuronal Recurrente provocan grandes cambios en el comportamiento del sistema. Esto provoca la aparición de gradiente localmente grande el cual aumenta de forma exponencial, debido a que según se propaga por las capas se multiplica por valores mayores a 1.

2.2. Evaluación de modelos

Un modelo es un buen modelo de aprendizaje profundo si es capaz de llevar a cabo una generalización de las representaciones aprendidas para nuevos valores de entrada. Para medir la capacidad de generalización del modelo es necesario realizar una evaluación del mismo (Chollet, 2018).

2.2.1. Selección de los hiperparámetros

Como ya se ha comentado, un algoritmo de Aprendizaje Profundo no es más que una función que produce una salida en base a una entrada. No obstante, existen una serie de parámetros denominados hiperparámetros, que, dado un set de algoritmos de aprendizaje, permiten la selección del más apropiado para nuestro problema. "*Un hiperparámetro que forma parte de un algoritmo de aprendizaje A se define como una variable que debe establecerse antes de la aplicación real de A a los datos y se caracteriza por no ser seleccionado directamente por el propio algoritmo de aprendizaje*"(Bengio, 2012).

La mayoría de los hiperparámetros han sido ya mencionados en el apartado de optimización del descenso de gradiente. Entre ellos encontramos: la tasa de aprendizaje ϵ (*learning rate*), el subconjunto de m muestras del dataset con el que el SGD realiza el entrenamiento (*mini-batch*), el número de iteraciones que realiza el algoritmo durante el entrenamiento sobre el conjunto de

datos de entrenamiento (época) o el *momentum* (Bengio, 2012).

El *momentum* es un método que interviene en la actualización de parámetros del descenso de gradiente estocástico y permite acelerar el entrenamiento. Al calcular cada nuevo valor de $\boldsymbol{\Theta}$, no se utiliza únicamente el valor actual del gradiente, sino que, también afecta la media del gradiente (los valores previos que ha tomado el gradiente) (Alom et al., 2019).

2.2.2. Set de entrenamiento, validación y *test*

Para llevar a cabo la evaluación del modelo, se realiza una división del conjunto de datos de los que se dispone en tres grupos: entrenamiento, validación y *test*. El entrenamiento se lleva a cabo sobre el set de entrenamiento y seguidamente se realiza la evaluación sobre el set de validación. Los datos de validación proporcionan una idea sobre cómo de buena es la generalización. Finalmente, para poder analizar la capacidad de generalización del modelo, lo evaluaremos sobre los datos de *test*. Estos datos con completamente nuevos para el modelo (Chollet, 2018).

2.2.3. *Underfitting y Overfitting*

Si representamos gráficamente el error de generalización del modelo en función de uno de los hiperparámetros, obtendremos una gráfica con una forma parecida a la representada en la figura 2.11. El error comienza con un valor elevado, desciende según se realiza el entrenamiento y llega un momento que comienza a aumentar de nuevo. Esta primera mitad de la gráfica en la que el error de generalización desciende según se lleva a cabo el entrenamiento, se denomina zona de subajuste (*underfitting*), y la zona en la que el error comienza a aumentar nuevamente se denomina zona de sobreajuste (*overfitting*). En el centro de la gráfica encontramos el menor error de generalización.

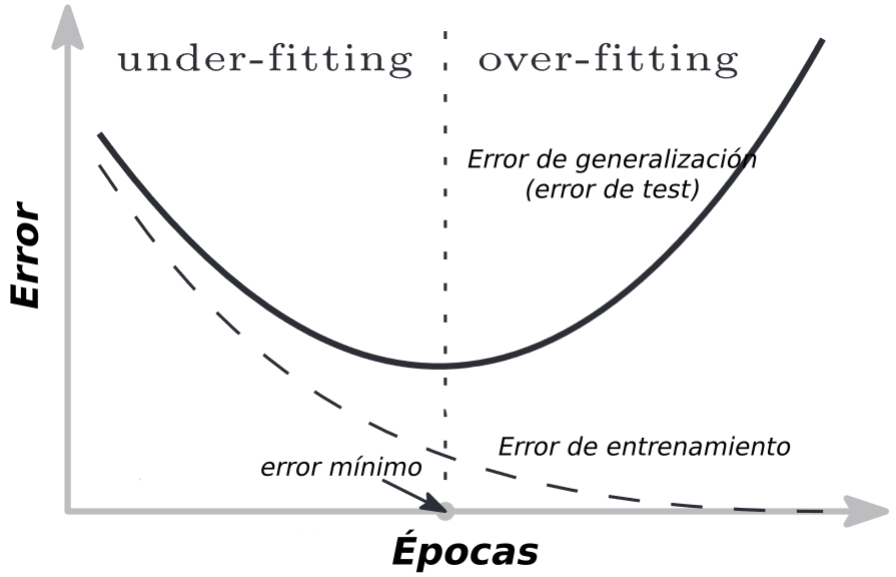


Figura 2.11. Gráfico del error para los sets de entrenamiento y validación. Imagen de (Belkin et al., 2019).

La causa se debe a que el modelo tiene un compromiso entre la optimización (ajustar el modelo a los datos de entrenamiento para obtener el mejor resultado posible) y generalización (cómo de

bien actúa el modelo en datos completamente nuevos). El objetivo es poder llevar a cabo una buena generalización, pudiendo únicamente ajustar el modelo en función de los datos de entrenamiento disponibles.

Al comienzo del entrenamiento, en la zona de *underfitting* existe una correlación entre optimización y generalización. Si disminuye el error en el set de entrenamiento, también disminuye el error del set de test. En esta zona el modelo realiza el aprendizaje de las representaciones de los datos de entrenamiento. En la zona de *overfitting*, la generalización deja de mejorar, el modelo aprende patrones muy específicos de los datos de entrenamiento que son irrelevantes o inducen a errores en la evaluación de nuevos datos.

Para evitar que el modelo se vea afectado por patrones que inducen a errores se implementan técnicas de regularización. Estas técnicas ayudan a que, si el modelo únicamente es capaz de aprender una determinada cantidad de información, memorice los patrones más significativos, y por tanto, sea más probable que pueda realizar una mejor generalización (Chollet, 2018; Goodfellow et al., 2016). A mayores, existen otras posibles técnicas a implementar como:

- La disminución de la capacidad de la red.
- *Batch normalization*: Propuesto por (Ioffe and Szegedy, 2015), ha demostrado mejorar el entrenamiento y *accuracy* de las redes. Su funcionamiento se basa en la estabilización de las distribuciones de las entradas (de cada *mini-batch*), dado que estas cambian con frecuencia, especialmente si se hace uso de funciones de activación como la sigmoide o tanh que introducen saturaciones no lineales. Para ello, introduce una capa adicional que controla la media y varianza de dichas distribuciones (Santurkar et al., 2018; Shrestha and Mahmood, 2019).
- *Dropout*: El término *dropout* se refiere a la eliminación temporal de una neurona de la red, incluyendo sus conexiones entrantes y salientes, como se muestra en la figura 2.12. Esta eliminación se realiza de forma aleatoria, siendo la probabilidad de que la neurona no sea eliminada p. (Srivastava et al., 2014) demuestra cómo la implementación de esta técnica resulta en una mejora significativa en el aprendizaje en visión artificial, biología computacional, en reconocimiento del habla y en problemas de clasificación de documentos.

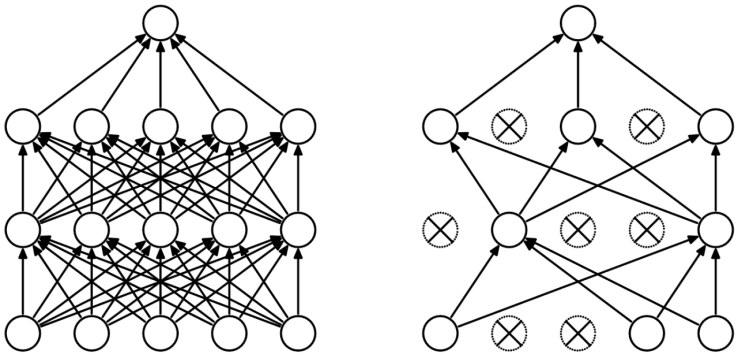


Figura 2.12. Modelo de Red Neuronal con dropout. A la izquierda, un modelo con dos capas intermedias y a la derecha aplicación del dropout al anterior modelo. Las neuronas tachadas han sido eliminadas. Imagen de (Srivastava et al., 2014).

2.3. Arquitecturas de las redes neuronales

2.3.1. *Feedforward Neural Network*

La arquitectura *Feedforward Neural Network*, también conocida como Perceptrón Multicapa (*Multilayer Perceptron* o MPL) es la arquitectura explicada anteriormente. Está formada por una capa de entrada, otra de salida y una o más capas intermedias. En las capas intermedias las salidas de cada neurona están conectadas con todas las neuronas de la capa siguiente, estas se encargan de realizar las labores computacionales. La última capa desempeña la tarea a realizar, ya sea clasificación o predicción. La información fluye desde la capa de entrada a la salida y utiliza el algoritmo de *backpropagation* para el aprendizaje. La arquitectura está diseñada para aproximarse a cualquier función continua y resolver problemas que no son separables linealmente (Munirathinam, 2020).

2.3.2. Red Neuronal Convolucional

Una Red Neuronal Convolucional (CNN, *Convolutional Neural Network*), es un tipo de red neuronal especializada en el reconocimiento de patrones. Para ello realiza un procesado de datos con topologías cuadriculares, como el reconocimiento de imágenes (podemos ver una imagen como una cuadrícula de píxeles 2D). La mayor ventaja ofrecida por las CNNs es la reducción del número de hiperparámetros a utilizar, dado que las redes tradicionales tendían a ser computacionalmente complejas en tareas de reconocimiento de imágenes.

La función llevada a cabo por una CNN se basa en la extracción de "características visuales locales" en las primeras capas de la red para posteriormente combinar dichas características y formar representaciones a mayor nivel en las capas más profundas. De maneara que, podemos dividir la arquitectura de la CNN en dos secciones: en la primera se realiza la extracción de características y en la segunda, se lleva a cabo la clasificación. Para ello, la arquitectura emplea tres tipos de capas: convolucional, *maxpooling* (utilizadas para la extracción de características) y de clasificación, que podemos ver en la figura 2.13 (Alom et al., 2019; Goodfellow et al., 2016; Kelleher, 2019).

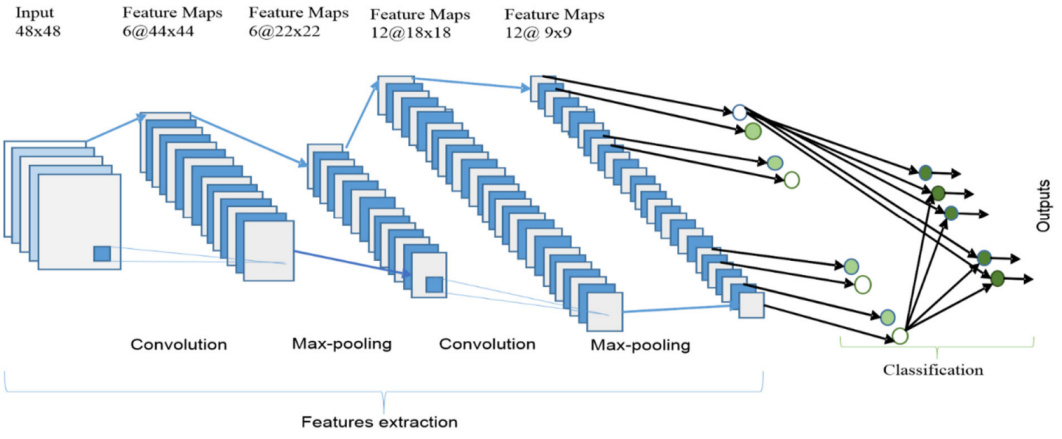


Figura 2.13. Arquitectura de una Red Neuronal Convolucional. Imagen de (Alom et al., 2019).

Una "característica visual local" es aquella que se localiza en una región de la imagen, como segmentos de curvas o líneas en un determinado ángulo. La detección de forma robusta de la presencia o ausencia de características visuales locales en una imagen es la tarea fundamental a realizar en reconocimiento de imágenes. Para ello, la red debe ser capaz de aprender funciones

que sean capaces de detectar dichas características. Sin embargo, la arquitectura de la red debe ser diseñada de manera que la red sea capaz de identificar la presencia de una característica visual en una imagen independientemente de dónde se encuentre localizada (Alom et al., 2019; Goodfellow et al., 2016).

Extracción de características

Podemos dividir esta sección en tres funcionalidades básicas:

I La capa de entrada. Al igual que en las redes descritas anteriormente, esta capa almacena los valores de las entradas. En el caso de las CNNs, cada entrada almacena el valor de un píxel de la imagen (O'Shea and Nash, 2015).

II La capa convolucional. Se caracteriza por el uso de kernels (filtros), como parámetros de aprendizaje. Estos kernels son dimensionalmente de menor tamaño que las imágenes. Al realizar la operación de convolución en una capa, se desplaza el kernel a lo largo de la imagen, realizando el producto escalar entre el kernel y los correspondientes valores de la imagen, generándose un mapa 2D de activaciones (*activation map*) o mapa de características (*feature map*) como ilustra la figura 2.14. La salida de la capa convolucional estará formada por tantos mapas de características como kernels se apliquen. La capa de convolución utiliza comúnmente utiliza la función ReLU como función de activación.

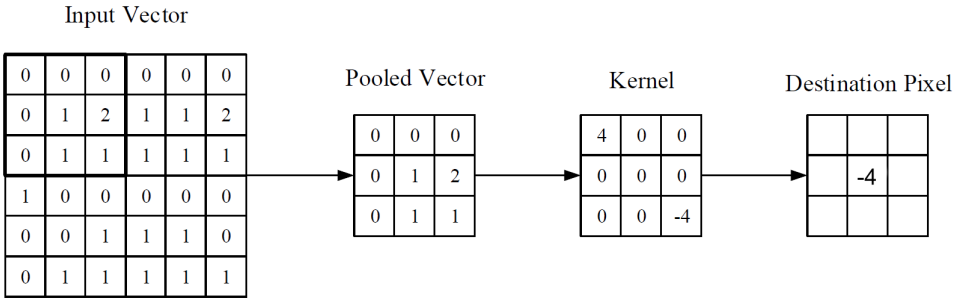


Figura 2.14. . Representación de la operación realizada por la capa convolucional. Imagen de (O'Shea and Nash, 2015).

Una de las características principales de las CNNs se debe a que las neuronas de la capa se encuentran organizadas en tres dimensiones: la dimensión espacial de la entrada (ancho y alto) y su profundidad (si es una imagen en blanco y negro su profundidad será 1, mientras que para imágenes a color será 3). Las capas convolucionales reducen significativamente la complejidad del modelo a través de tres hiperparámetros: la profundidad, el *stride* y el *zero-padding* (O'Shea and Nash, 2015).

III La capa *maxpooling*. Simplifica el funcionamiento del algoritmo. Reduce la dimensión de las entradas, reduciendo a su vez el número de hiperparámetros a utilizar. Esta capa aplica un kernel con la función *max* (quedarse con el valor máximo) en cada mapa de características. En la mayoría de CNNs, este kernel tiene una dimensión 2x2, lo que reduce los mapas de características en un 25 % (O'Shea and Nash, 2015).

Es posible identificar la presencia de una característica visual en una imagen, independientemente de dónde se encuentre localizada dado que la convolución realiza una búsqueda por toda la imagen y registra las detecciones realizadas en el mapa de características. Para realizar la detección de varias características se realiza el entrenamiento con múltiples capas convolucionales en

paralelo, cada una con un filtro que realiza el aprendizaje de una función de extracción de características (kernel). La salida de múltiples filtros se puede combinar de diversas maneras, una de ellas es tomando cada uno de los mapas de características generados y combinarlos en un único mapa de características, convirtiéndose este en la entrada de la siguiente capa convolucional. Otra opción, es la ilustrada en la figura 2.15, donde se utiliza una capa densamente conectada (densely connected layer) para combinar la información proveniente de diferentes filtros. Esta capa realiza la misma función que las capas de la red completamente conectada (Kelleher, 2019).

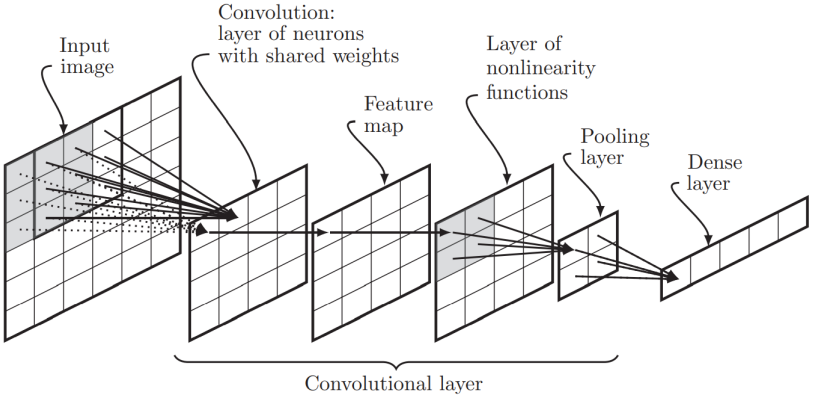


Figura 2.15. Capa convolucional. Imagen de (Kelleher, 2019).

Clasificación

La clasificación es una red completamente conectada (*fully connected network*) cuya entrada es la salida de la última capa convolucional. En la última capa de la red completamente conectada se realiza el cálculo de la puntuación de cada clase mediante la función Softmax. En función del resultado obtenido, el clasificador devuelve un valor para cada una de las clases.

El uso de redes *Feedforward* para la tarea de clasificación se debe a que ofrecen mejores prestaciones (Hinton et al., 2006; Nair and Hinton, 2010). No obstante, las redes completamente conectadas son computacionalmente caras, debido a la cantidad de parámetros que deben aprender. Entre las alternativas a estas redes, podemos encontrar técnicas como *average pooling* y *global average pooling*.

Las Redes Neuronales Convolucionales aportan una importante reducción en la complejidad computacional de la red al llevar a cabo tareas de análisis de datos visuales y no secuenciales. No obstante, en cuanto a la interpretación de datos temporales y bloques de texto (datos secuenciales) se refiere, estas no disponen de un buen comportamiento (Alom et al., 2019).

2.3.3. *Residual Networks*

A medida que avanza el desarrollo de las Redes Neuronales, estas han ido tomando gradualmente mayor profundidad. No obstante, el entrenamiento de redes con tal cantidad de capas tiene diversas dificultades, como el desvanecimiento o explosión de gradiente. Para solventar el problema, el equipo Microsoft Research Asia (MSRA) aplicó una red residual (ResNet) en el conjunto de datos CIFAR-10 y ganó el primer lugar en la competición ILSVRC 2015, con una red de 152 capas (Mandic and Chambers, 2001). Estas redes han demostrado tener una mejor generalización

(Zagoruyko and Komodakis, 2017) además de acelerar la convergencia (Szegedy et al., 2016).

En lugar de colocar secuencialmente una serie de capas y esperar que estas se ajusten al mapeado deseado, dejamos que las capas se ajusten a un mapeado residual. Denotando al mapeado buscado $H(x)$, dejamos que las capas se ajusten al mapeado $F(x)$, que toma la forma $F(x) = H(x) - x$, de forma que es posible obtener $H(x)$ como $H(x) = F(x) + x$. Esta operación puede ser realizada mediante redes neuronales en las que se incorporan "atajos", conexiones con accesos directos que saltan una o varias capas, como muestra la figura 2.16. Estos atajos llevan a cabo un mapeado identidad, agregándose su salida a la de la capa anterior. Otra característica importante es que estas conexiones no conllevan la incorporación de ningún parámetro ni son computacionalmente costosas (He et al., 2015; Zagoruyko and Komodakis, 2017).

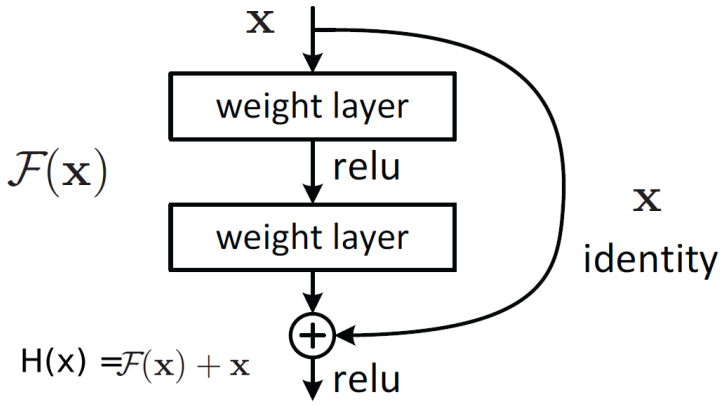


Figura 2.16. Aprendizaje residual. Imagen de (He et al., 2015).

2.3.4. *Recurrent Neural Networks*

Las Redes Neuronales Recurrentes (RNN, *Recurrent Neural Networks*) fueron un modelo de red neuronal propuesto en los años 80 (Elman, 1990; Rumelhart et al., 1986; Werbos, 1988) para procesar secuencias de datos e información temporal. Su arquitectura es análoga a la del Perceptrón Multicapa agregando a mayores conexiones entre neuronas de las capas intermedias ((Pascanu et al., 2013). Cabe mencionar, que las RNNs también sufren el problema del del desvanecimiento y explosión de gradiente, lo que dificulta en gran medida su entrenamiento (Bengio et al., 1994).

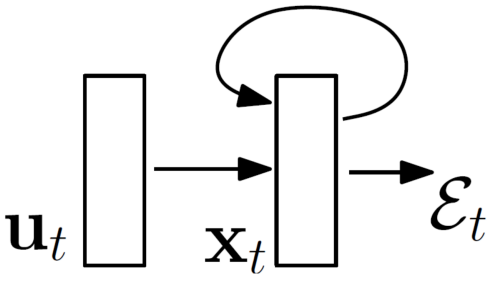


Figura 2.17. Esquema de una RNN. Las conexiones de las capas intermedias permites el mantenimiento de la información. Imagen de (Pascanu et al., 2013).

2.4. *Transfer Learning*

Al llevar a cabo el entrenamiento de una red con una gran cantidad de datos, el modelo aprende los qué valores deben tomar los pesos y el sesgo. A mayores, dentro de una red neuronal entrenada, existen capas que detectan contornos, curvas, líneas y otras características identificativas. La identificación de dichas características fue aprendida en el entrenamiento original del modelo, por lo que es posible reutilizarlas. Aun así, será necesario entrenar a la red para reconocer las clases que busquemos, usando ejemplos debidamente etiquetados, pero como la red ya ha aprendido a reconocer características comunes a la mayoría de los objetos, el entrenamiento está parcialmente hecho. Por lo tanto, el *transfer learning* consiste en introducir en la red el valor de los pesos de un modelo que ha sido previamente entrenado, en lugar de realizar el entrenamiento desde cero. La utilización de modelos pre-entrenados no sólo conlleva ahorro computacional, sino que, permite llegar al punto de convergencia con mayor rapidez debido a que entrenar modelos con grandes datasets puede llevar hasta múltiples semanas (Alom et al., 2019).

Capítulo 3

Revisión de datasets para el reconocimiento de la actividad humana

3.1. Introducción a técnicas de adquisición de movimientos

En los últimos años, el reconocimiento de la actividad humana (HAR, *Human Activity Recognition*) es un tema que progresivamente cobra mayor importancia debido a sus aportaciones en la vida diaria asistida (Philipose et al., 2004) y en la mejora de la calidad de vida de las personas, gracias a sus contribuciones en atención médica así como en la tecnología utilizada en rehabilitación (Liu et al., 2021). Además, realiza aportaciones en numerosos campos como la seguridad gracias a la vigilancia, el aprendizaje automático, la computación móvil (Philipose et al., 2004), o el entretenimiento interactivo (Liu et al., 2021).

Mediante el análisis de las observaciones adquiridas sobre los sujetos, así como del entorno que las rodea, el HAR busca comprender los comportamientos de la vida cotidiana. Para llevar a cabo el reconocimiento de la actividad humana se hace uso de sistemas dotados de capacidad sensorial, los cuales se pueden dividir en dos clases: sistemas ópticos y sistemas no ópticos (Fleron et al., 2019).

Sistemas de adquisición ópticos

Los sistemas ópticos hacen uso de cámaras de vídeo para la adquisición de datos, la cual se lleva a cabo mediante sistemas de captura de movimiento. Encontramos dos tipos de sistemas de captura, los que hacen uso de marcadores, como Vicon y los que no (Patrizi et al., 2016; Perrott et al., 2017).

Los sistemas de HAR mediante vídeo utilizan cámaras RGB, RGB-D, cámaras térmicas, sensores infrarrojos, etc (Das Antar et al., 2019; Yadav et al., 2021). A la hora de llevar a cabo la adquisición de datos, existen determinados factores como la variación en la iluminación, los objetos localizados en el fondo o las articulaciones del cuerpo humano, que influyen en los datos recogidos (Yadav et al., 2021).

Sistemas de adquisición no ópticos

Los sistemas no ópticos permiten realizar la captura del movimiento en condiciones de mala visibilidad. Típicamente se utilizan unidades de medición inerciales (IMUs), que hacen uso de

sensores vestibles.

Entre los sensores vestibles utilizados, podemos encontrar acelerómetros, giroscopios, magnetómetros o la combinación de todos ellos en sensores inerciales integrados. Estos permiten realizar mediciones de parámetros fisiológicos que no se pueden medir mediante el uso de cámaras (Mukhopadhyay, 2015). No obstante, entre sus limitaciones podemos encontrar la pérdida en la precisión de las medidas realizadas (*sensor drift*) que aparecen durante largos periodos de operación, así como la variación en las mediciones realizadas en función de la localización de los sensores en el cuerpo del sujeto (Yadav et al., 2021). La calidad de los datos recogidos por el sensor, y, por tanto, del reconocimiento de la acción, vienen dados en función de la localización de los sensores en el cuerpo del sujeto (Atallah et al., 2011; Martín et al., 3 04; Vahdatpour et al., 2011). Según estudios realizados, las localizaciones más óptimas, son la parte superior del brazo, el antebrazo, el muslo, la espinilla, el pecho o la cintura y la cabeza, y la subdivisión en regiones de menor tamaño no implica una mejora del reconocimiento de la acción (Vahdatpour et al., 2011). No obstante, otros estudios han mostrado que las localizaciones óptimas dependen de la actividad a reconocer (Atallah et al., 2011).

Existen otros métodos para capturar la actividad humana, como la utilización de sensores acústicos o ambientales, pero las soluciones basadas en la visión y dispositivos vestibles son las tecnologías más utilizadas (Cippitelli et al., 2017).

La investigación en los últimos años ha mostrado interés en la combinación tanto de los datos obtenidos mediante sistemas ópticos como no ópticos, buscando una complementación entre ambos (Ehatisham-Ul-Haq et al., 2019; Kwon et al., 2020; Rey et al., 2019). Por una parte, ningún sistema de HAR es capaz de extraer datos precisos en condiciones variables del entorno, pero la información recogida por sistemas ópticos es complementaria a la obtenida con otros métodos (Zhang et al., 2019). Adicionalmente, el disponer de datos provenientes de diferentes sistemas permite al usuario tomar los datos que mejor se ajusten a sus necesidades en cada momento (Roggen et al., 2013).

3.2. Revisión de datasets existentes

La investigación en HAR se lleva a cabo mediante el uso de extensas cantidades de datos. En consecuencia, la correcta recopilación de datos es imprescindible en el reconocimiento de acciones (Liu et al., 2021). A medida que el reconocimiento de la actividad humana ha ido cobrando mayor importancia y se ha ido desarrollando gracias al uso de grandes cantidades de datos, han surgido grandes datasets de referencia (*benchmark* datasets) (Yadav et al., 2021).

En cuanto al HAR mediante sistemas ópticos, Human3.6M (Ionescu et al., 2014) es uno de los datasets más extensos, compuesto por 3.6 millones de poses 3D en 17 escenarios diferentes. Los datos fueron recogidos de 11 sujetos (5 de ellos mujeres y 6 hombres), bajo 4 perspectivas diferentes. Presenta una serie de actividades y poses de la vida diaria, buscando complementar los datasets del estado del arte ya existentes. Estos datos se utilizan en el entrenamiento de sistemas de detección humana así como en la evaluación de modelos y algoritmos de estimación de poses, con el objetivo de mejorar la reconstrucción 3D de poses humanas a partir de imágenes. Los experimentos realizados en (Ionescu et al., 2014) han obtenido una mejora del 20 % en los modelos utilizados en comparación con otros datasets del estado del arte.

ActivityNet (Heilbron et al., 2015) es el *benchmark* dataset más extenso tanto por las 200 actividades que incorpora como por la cantidad total de vídeos que contiene. Proporciona una serie de actividades de la vida diaria caracterizadas por tener movimientos más complejos, puesto que los *benchmark* datasets existentes generalmente están formados tanto por acciones como movimientos básicos, extraídos manualmente de vídeos más largos. La base de datos contiene una media de 137 vídeos por cada una de las clases, a los cuales no se les ha aplicado ningún corte. De media, cada uno de sus vídeos contine 1.41 actividades, estando cada una de ellas limitada temporalmente. En total, los datos recopilados tienen una duración aproximada de 849 horas.

Asimismo, debido al auge de las redes sociales en los últimos años, han surgido datasets como HMDB51 o UCF101 que recogen vídeos de diferentes plataformas. HMDB51 (Kuehne et al., 2011) es un *benchmark* dataset que incorpora 51 actividades con un total de casi 6.500 vídeos, conteniendo cada categoría un mínimo de 101 vídeos. Estos proceden mayoritariamente de películas o bases de datos públicas como Prelinger, Youtube o Google videos. Las actividades que contiene pueden ser divididas en cinco categorías: expresiones faciales, expresiones faciales con manipulación de objetos, movimiento corporal, movimiento corporal con manipulación de objetos y movimiento corporal con interacción humana. Este dataset es de especial interés puesto que ha sido utilizado en este TFG para generar un modelo pre-entrenado, que la Red Neuronal utilizada para el HAR emplea como datos de partida mediante *transfer learning*.

Por otro lado, UCF101 (Soomro et al., 2012) contiene prácticamente 13.200 vídeos de YouTube, que se dividen en 101 categorías de 5 clases diferentes. Los datasets de referencia comúnmente se encuentran bastante preparados, empleando actores para la recolección de datos. Por el contrario, UCF101 es una base de datos formada por vídeos más realistas, que incorporan cambios en la iluminación, movimientos de cámara, diferentes ángulos de captura y escalado de objetos, entre otros factores. Encontramos cinco posibles categorías en las que podemos dividir las diferentes acciones del dataset: interacciones entre humanos y objetos, movimiento corporal, interacciones entre varios humanos, instrumentos musicales y deportes. Adicionalmente, tanto los vídeos de ActivityNet como los de estos dos últimos datasets ya se encuentran divididos en un set de entrenamiento, otro de validación y un último de *test*.

Para finalizar, también cabe destacar el dataset Kinetics 400 (Kay et al., 5 19), compuesto por 400 clases con al menos 400 vídeos cada una, cada uno con una duración media de 10 segundos. Los datos han sido recopilados de YouTube. Entre las acciones que engloba, encontramos tanto interacciones entre varios sujetos como interacciones de un sujeto con objetos.

Análogamente, encontramos datasets adquiridos mediante sistemas no ópticos como Opportunity benchmark database, dentro del cual destaca el subset Opportunity Activity Recognition (Roggen et al., 3 11). Este se encuentra formado por actividades o intervenciones que ocurren en la vida diaria (*naturalistic activities*) de 12 sujetos, utilizando 72 sensores de 10 modalidades diferentes que se encuentran localizados tanto en los sujetos como en objetos del entorno. En total, incluye más de 25 horas de datos provenientes de los sensores. A posteriori, se realizó una extensión multimodal, denominada Opportunity++ (Ciliberto et al., 2021). Este dataset incluye datos adquiridos mediante sistemas ópticos y no ópticos, dando pie a futuras líneas de investigación en las que se fusionen ambos datos. Proporciona grabaciones de 4 sujetos realizando 6 *runs* (secuencias de actividades) diferentes, 5 de ellos ADL (*Activity of Daily Living*) *runs* y un *Drill run*, en el que las actividades se realizaban en un espacio más restringido. Las activades recogidas son las típicamente realizadas en una rutina matinal. Otra importante contribución es la del dataset

REALDISP (REAListic sensor DISPlacement) (Banos et al., 2012), cuyo principal propósito es el estudio de los efectos provocados por el desplazamiento que sufren los sensores. No obstante, éste también es utilizado para evaluar técnicas para el reconocimiento de acciones en condiciones ideales. Contiene a 17 sujetos realizando 33 actividades físicas en 3 escenarios diferentes y emplea 9 sensores.

3.3. Descripción del dataset utilizado

Como se ha comentado previamente, el reconocimiento de la actividad humana conlleva el uso de grandes cantidades de datos. Todos los datasets mencionados en el capítulo anterior son un punto de referencia en el HAR, no obstante, nuestro interés se centra particularmente en el uso de estos en el ámbito médico. Una gran parte de los datasets no presentan variedad en el tipo de actividades que incorporan, siendo la mayoría actividades realizadas con el tronco inferior del cuerpo. Adicionalmente, pese a que la interacción humana entre dos personas es un caso de estudio de gran interés, para el uso de determinadas herramientas como son estimadores de poses 3D como DeepStream, es necesario que aparezca un único sujeto en la escena y que todo su cuerpo sea visible. Estas dos características tampoco son comunes en los *benchmark* datasets. Lo previamente mencionado, sumado al reciente interés por la creación de datasets multimodales conllevó al uso del dataset multimodal adquirido por el grupo de investigación.

El conjunto de datos se encuentra compuesto por 13 actividades de la vida diaria realizadas por 39 sujetos diferentes. De los 39 sujetos, 37 son adultos y 2 son niños. La adquisición de movimientos se llevó a cabo simultáneamente con tecnología de captura de vídeo y sensores portátiles diseñados a medida (González-Alonso et al., 2020, 2021).

Las actividades realizadas se pueden dividir en actividades del tronco inferior (caminar, caminar hacia atrás, caminar sobre una línea, levantarse y sentarse en una silla) y actividades del tronco superior (mover una botella de lado a lado, fingir beber, construir una torre, alcanzar un objeto alto, romper un papel, hacer una bola con él y lanzarlo). Dentro de las actividades de tronco superior, encontramos algunas de carácter bimanual y otras de carácter manual. Las acciones manuales se llevaron a cabo dos veces (una con cada mano) y se encuentran etiquetadas con identificadores diferentes. Las acciones contenidas en la base de datos fueron escogidas por su relevancia clínica, debido al uso de acciones similares para realizar diagnósticos clínicos, puesto que son actividades que se realizan en la vida diaria. En la tabla 3.1 podemos encontrar una descripción más detallada sobre cada una de las actividades.

Cada vídeo está compuesto por un único sujeto realizando una sola acción un número de repeticiones. El número de repeticiones varía en función de la duración de cada actividad. No obstante, todos los sujetos realizaron cada una de las actividades el mismo número de veces. La duración de los vídeos es de aproximadamente 25 segundos.

ID	Acción	Tipo	Número de repeticiones
A01	Caminar	Tronco inferior - pierna	3
A02	Caminar hacia atrás	Tronco inferior - pierna	3
A03	Andar sobre una línea	Tronco inferior - pierna	3
A04	Sentarse en una silla	Tronco inferior - pierna	5
A05	Mover un vaso (mano derecha)	Tronco superior - unimanual	5
A06	Mover un vaso (mano izquierda)	Tronco superior - unimanual	5
A07	Beber (mano derecha)	Tronco superior - unimanual	5
A08	Beber (mano izquierda)	Tronco superior - unimanual	5
A09	Montar una torre	Tronco superior - bimanual	1
A10	Tirar un balón al aire y cogerlo de nuevo	Tronco superior - bimanual	10
A11	Alcanzar un objeto alto (mano derecha)	Tronco superior - unimanual	5
A12	Alcanzar un objeto alto (mano izquierda)	Tronco superior - unimanual	5
A13	Romper un papel, hacer una bola y lanzarlo	Tronco superior - bimanual	1

Tabla 3.1. Descripción de las actividades del dataset.

Capítulo 4

Materiales y métodos

4.1. Limpieza y reorganización del dataset

Los datos recogidos mediante el sistema de captura óptico fueron utilizados en un proyecto anterior, introduciéndolos en sistemas de *tracking* para inferir así la pose de los sujetos (Pérez de la Fuente, 2021). Debido a su anterior uso, el dataset se encontraba organizado en 39 directorios, uno por cada sujeto. Dentro de cada uno se encontraban para cada una de las 13 acciones que desempeñaron los sujetos, cuatro ficheros diferentes:

- La grabación de vídeo con extensión `.mp4` del sujeto realizando la actividad.
- Un fichero con extensión `.json`. Este recoge los puntos de referencia (*keypoints*) que proporciona el programa de *tracking*. Los *keypoints* son las coordenadas de los puntos distintivos de cada fotograma, que usualmente marcan bordes (Uchida, 2013).
- La misma grabación del sujeto con extensión `.mp4`, en la que se representan los *keypoints* inferidos.
- La misma grabación del sujeto con extensión `.mp4`, en la que se representa un *bounding box*. Un *bounding box* es una caja rectangular imaginaria que sirve para delimitar o ”marcar los bordes”del objeto que encierra. Esta herramienta se utiliza en detección de objetos (Mousavian et al., 2017).

Todos los sujetos disponían de al menos una grabación por cada una de las 13 actividades, salvo los sujetos S03 y S04 (los niños), que únicamente realizaron las actividades de tronco superior A01-A04. Por esta razón, los datos de dichos sujetos se ordenaron dentro de la base de datos pero no se utilizaron en el entrenamiento ni inferencia de la Red Neuronal. Adicionalmente, los vídeos se encontraban etiquetados con la fecha y hora en la que realizaron las capturas.

No obstante, para poder llevar a cabo el reconocimiento de acciones mediante la red neuronal ResNet-18, es necesario que la base de datos se encuentre estructurada como se indica en la figura 4.1. Para ello, se llevó a cabo una reorganización de los vídeos. Se crearon 13 directorios, denotados por cada uno de los identificadores de las actividades (A01, A02, A03, ... , A13). En ellos se encuentran los vídeos de todos los sujetos realizando dicha acción.

Cabe destacar que, algunos de los sujetos grabaron dos veces la misma acción. Esto se debe a que durante la adquisición de datos se realizaron segundas grabaciones en los casos en los que se dudaba si el sistema de *tracking* sería capaz de funcionar correctamente. Se llevó a cabo una

```
/Dataset
    /clase1
         /vídeo1
            /rgb
                0000.png
                0001.png
                0002.png
                ...
                ...
                ...
                N.png
```

Figura 4.1. Estructura del dataset requerida para realizar el entrenamiento e inferencia en la red ResNet-18.

revisión de los vídeos y se descartaron aquellos defectuosos. Calificamos de defectuosos, aquellos en los que el sujeto no realiza la acción correctamente o aquellos en los que la grabación se encontraba cortada. Todos aquellos grabados correctamente se mantuvieron en la base de datos etiquetados como `P2` (Prueba 2).

Finalmente, se realizó un renombramiento de los vídeos buscando que el nombre de cada uno fuera autodescriptivo además de que facilitara el entrenamiento y cálculo de métricas de la Red Neuronal utilizada, al tener la etiqueta verdadera de la acción a reconocer en el propio nombre del vídeo. Por dichas razones, se decidió utilizar el formato:

```
numeroSujeto_IDActividad_numeroPrueba
```

La estructura final de la base de datos es la mostrada a continuación:

```
/Dataset
    /A01
        S01_A01_P01.mp4
        S02_A01_P01.mp4
        ...
        ...
        ...
        S39_A01_P01.mp4
    /A02
    ...
    /A13
```

Figura 4.2. Estructura de la base de datos reorganizada.

Una vez la base de datos se encontraba debidamente estructurada, se realizó la conversión de cada uno de los vídeos a fotogramas, los cuales se guardaron con la extensión `.png`. Para ello, se ejecutó un *script* de Python con el *path* a la base de datos y el *path* al directorio donde buscamos guardar los fotogramas resultantes, como argumentos de entrada. El *script* se encarga

de recorrer recursivamente los subdirectorios generando los correspondientes ficheros ordenados de igual manera a la descrita en la figura 4.1.

4.2. Entrenamiento con redes neuronales

4.2.1. Reconocimiento de acciones mediante TAO Toolkit

Haciendo uso de la herramienta TAO Toolkit de NVIDA, se ha llevado a cabo un reconocimiento de las acciones de nuestro dataset mediante Visión Artificial.

NVIDIA TAO Toolkit es una Interfaz de Línea de Comandos (CLI) y notebook de Jupyter que utiliza un *framework* el cual favorece la adaptación de modelos de Inteligencia Artificial. Permite abstraer la complejidad de los *frameworks* de IA, posibilitando realizar modificaciones en la configuración de modelos pre-entrenados, gracias al *transfer learning*.

Esta herramienta posibilita crear modelos personalizados de Visión Artificial y de IA Conversacional con los propios datos del usuario. En particular, el Toolkit es un paquete de Python el cual interactúa con contenedores Docker a bajo nivel. Dentro de los contenedores, se encuentran notebooks de Jupyter desde los cuales se ejecuta la línea de comandos directamente. Los comandos disponibles son: `data augmentation`, `train`, `evaluate`, `infer`, `prune` y `export`. Asimismo, una vez obtenido el modelo entrenado, es posible exportarlo para realizar inferencias sobre él utilizando otros dispositivos de NVIDIA, mediante técnicas como DeepStream (NVIDIA, 2022b).

4.2.2. Modelo empleado para llevar a cabo el HAR

Para poder ser capaces de reconocer acciones no basta con realizar el análisis de un único fotograma, sino que es necesario añadir un contexto temporal, analizando un conjunto de fotogramas consecutivos. Para ello, utilizamos un modelo 3D de una arquitectura ResNet-18, ilustrado en la figura 4.3. Esta red se encuentra formada por una Red Neuronal Convolucional 3D de dimensiones: ancho (W), alto (H), número de canales (C) y dimensión temporal (D). Como entrada, se utilizará una secuencia de fotogramas RGB, cuyo tamaño es 3x32x224x224 (CxDxHxW). La salida de la red convolucional se introduce en una capa *fully connected*, seguida de una capa Softmax utilizada para la predicción de la acción. El algoritmo de entrenamiento *Stochastic Gradient Descend* (SGD) optimiza la red para minimizar el error calculado mediante *cross-entropy* (NVIDIA, 2022a).

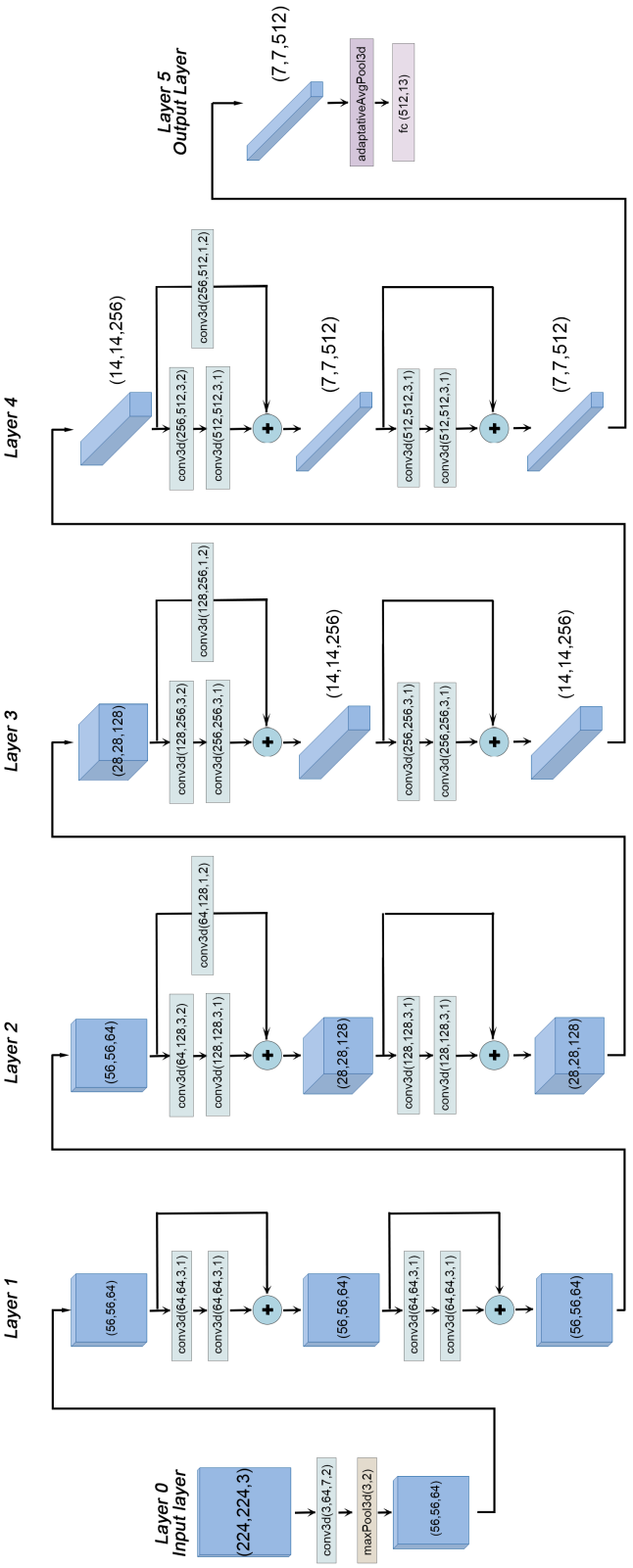


Figura 4.3. Estructura de la red neuronal ResNet-18 utilizada.

Adicionalmente, se dispone de un modelo ResNet-18 pre-entrenado para un conjunto de acciones del dataset HMDB51, con fotogramas RGB como entrada. Dicho modelo se entrenó con 120 épocas, un *batch size* de 1 y una GPU V100 para 5 acciones de HMDB51. Las acciones utilizadas fueron caminar, montar en bicicleta, correr, caerse al suelo y empujar. Los vídeos pertenecientes a estas acciones presentan diversidad en las partes del cuerpo visibles, el movimiento realizado por la cámara, la calidad de vídeo utilizada, el número de personas contenidas y el punto de colocación de la cámara. Las estadísticas proporcionadas sobre los vídeos utilizados se encuentran recogidas en las tablas 4.1 y 4.2 (NVIDIA, 2022a).

Clases	**Número de vídeos**
Caminar	494
Montar en bicicleta	93
Correr	209
Caerse al suelo	123
Empujar	105

Tabla 4.1. Estadísticas de los vídeos del dataset HMDB51 utilizados en el pre-entrenamiento.

Características de los vídeos del dataset HMDB51	
Partes del cuerpo visibles	Tronco superior, tronco inferior, cuerpo completo
Movimiento de la cámara	Movimiento, estática
Punto de colocación de la cámara	Delantero, trasero, izquierda, derecha
Número de personas involucradas en la acción	Una, dos, tres
Calidad del vídeo	Buena, media, baja
Tamaño del vídeo	La mayoría de los vídeos tienen un tamaño de 320x240

Tabla 4.2. Características de los vídeos del dataset HMDB51.

La evaluación del modelo pre-entrenado se llevó a cabo tomando un 10 % de los vídeo totales disponibles por clase del dataset HMDB51, de forma aleatoria. Existen dos modos diferentes con los que es posible realizar la inferencia y evaluación. El primero, *center*, el cual utiliza los frames centrales del vídeo. Si el modelo requiere 32 frames de entrada y el tamaño del vídeo es de 128 frames, únicamente se utilizarán aquellos con los índices 48 a 79. El segundo, *conv*, divide el vídeo uniformemente en 10 secciones y toma el frame central de cada una de estas secciones como frame de partida y finalmente elige los 32 frames consecutivos a ese frame de partida para formar los segmentos de inferencia. La etiqueta final de salida se determina mediante la media de esos 10 segmentos. Para el modo *center*, se obtuvo un *accuracy* del 84.69 %, mientras que para el modo *conv* se obtuvo un *accuracy* del 85.59 % (NVIDIA, 2022a).

Para poder utilizar este modelo entrenado como pesos pre-entrenados se utiliza *transfer learning*.

4.2.3. HAR del tren superior

En este apartado se llevó a cabo el reconocimiento de 9 actividades del tren superior: mover un vaso con el brazo izquierdo y derecho, beber con el brazo izquierdo y derecho, realizar una construcción con Legos, tirar un balón al aire y cogerlo de nuevo, alcanzar un objeto alto con el brazo izquierdo y derecho y romper un papel, hacer una bola y tirarlo, cuyos identificadores en la tabla 3.1 son A05-A13. Para ello se utilizaron 15 sujetos: S01, S02, S05, S29-S39. El entrenamiento ha sido efectuado mediante *k-fold cross validation* utilizando 14 sujetos en el entrenamiento y 1 sujeto para validación y *test*, haciendo uso del modo *center* tanto en la inferencia como en la evaluación. Asimismo, se ha empleado el modelo pre-entrenado con la base de datos HMDB51.

K-fold cross validation es un método estadístico utilizado en la evaluación de algoritmos en situaciones en las que se dispone de una cantidad de datos reducida. Los datos son divididos equitativamente en *k* grupos, realizándose *k* iteraciones de entrenamiento y validación. En cada iteración, *k-1* datos son utilizados para entrenar a la red y el último es empleado en la validación y *test*, obtiendose adicionalmente las métricas del modelo entrenado. Una vez realizadas las ejecuciones, se dispondrá de *k* métricas, para las cuales se buscará obtener un resultado global. Para ello, es posible aplicar diferentes metodologías, como la la media (Refaeilzadeh et al., 2009). En nuestro caso particular, se ha seguido la siguiente secuencia:

1. División de los datos en $k = 15$ grupos.
2. Realización de 15 iteraciones de entrenamiento.
   a) En cada iteración, se toman 14 sujetos como datos de entrenamiento y 1 sujeto (diferente cada vez) como datos de validación y *test*.
   b) Obtención de la matriz de confusión del modelo y las métricas correspondientes a esta.
   c) Descarte del modelo.
3. Obtención de la matriz de confusión para el conjunto así como de las métricas correspondientes. Para ello, se aplica la media aritmética de las 15 métricas almacenadas.

Los hiperparámetros definidos en el fichero de especificaciones para el entrenamiento aparecen en la tabla 4.3.

Parámetro	**Valor**
Horizontal flip	0.0
Épocas	15
Learning rate	0.001
Momentum	0.9
Weight decay	0.0001

Tabla 4.3. Valor de los parámetros configurables del optimizador SGD en el entrenamiento.

En la imagen 4.4 se representa la matriz de confusión resultante de realizar inferencia sobre nuestra red entrenada mediante *k-fold*. Complementariamente, en la tabla 4.4 encontramos las métricas correspondientes a dicha matriz de confusión.

Se pude observar en la matriz de confusión de la figura 4.4 que, en general, no se obtienen buenos resultados con la reducida cantidad de datos introducida en la red.

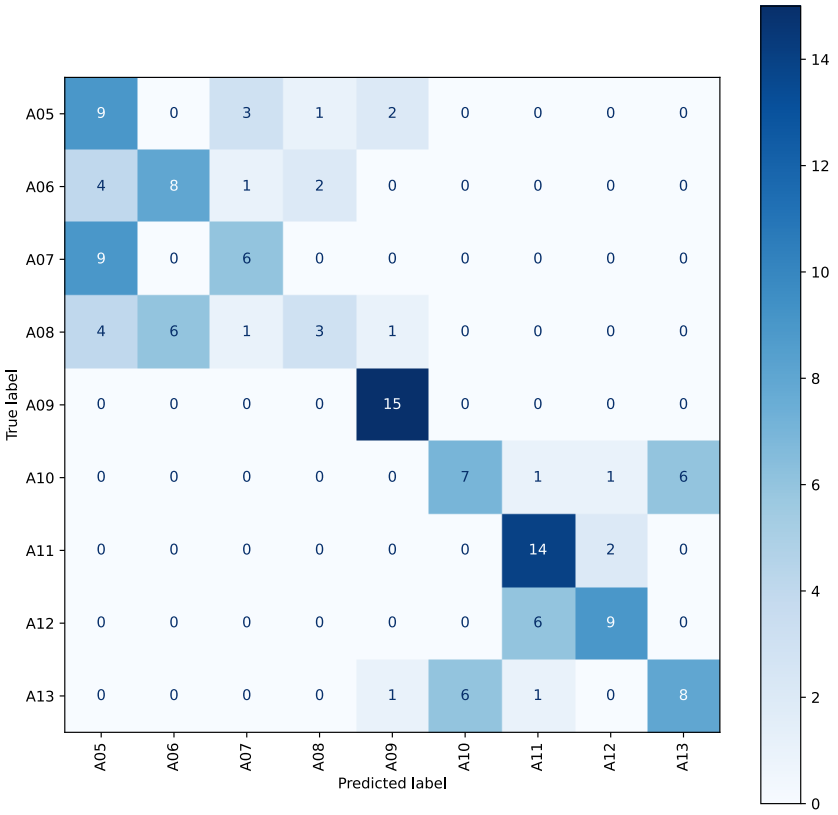


Figura 4.4. Matriz de confusión para el HAR del tren superior.

Sensitivity	Specificity	Precision	Negative predictive value	False positive rate
0.5766	0.9471	0.5766	0.9471	0.0529

False negative rate	False discovery rate	F1	Overall accuracy
0.4236	0.4234	0.5766	0.9059

Tabla 4.4. Métricas de la matriz de confusión del HAR del tren superior.

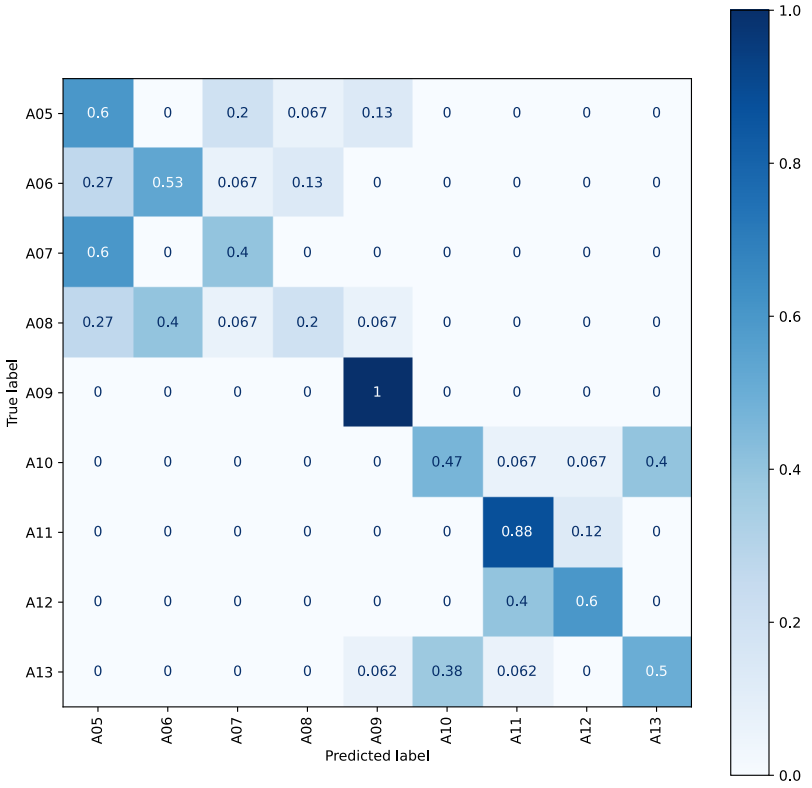
La actividad A09 (montar unos bloques de Lego) es la única que se reconoce razonablemente, ya que se identifican correctamente todas las muestras pertenecientes a dicha clase. Si bien, también se clasifican como pertenecientes a esta clase muestras de otras actividades(andar hacia delante, beber con la mano izquierda y romper un papel, hacer una bola y lanzarlo). No obstante, debe tenerse en cuenta que dicha actividad es una de las tres bimanuales de la base de datos, además de ser la más dispar de las nueve. Cabe destacar los resultados obtenidos para otra actividad bimaual, A13, la cual se confunde en gran medida con la tercera actividad bimanual, A10 (tirar un balón y cogerlo de nuevo). Se observa adicionalmente, que en general la red confunde las acciones unimanuales no con su homóloga realizada con la mano opuesta, sino que se confunde con aquellas de movimientos similares que utilizan la misma mano. Es decir, la red confunde la actividad A07 (beber con el brazo derecho) con A05 (mover un vaso con el brazo derecho) y A08 (beber con el brazo izquierdo) con A06 (mover un vaso con el brazo izquierdo). Para las activi-

dades unimanuales A11-A12, sí que encontramos confusión entre la misma acción realizada con manos opuestas, dado que el sujeto en las actividades anteriores se encontraba sentado tras una mesa, mientras que en las últimas se encuentra de pie.

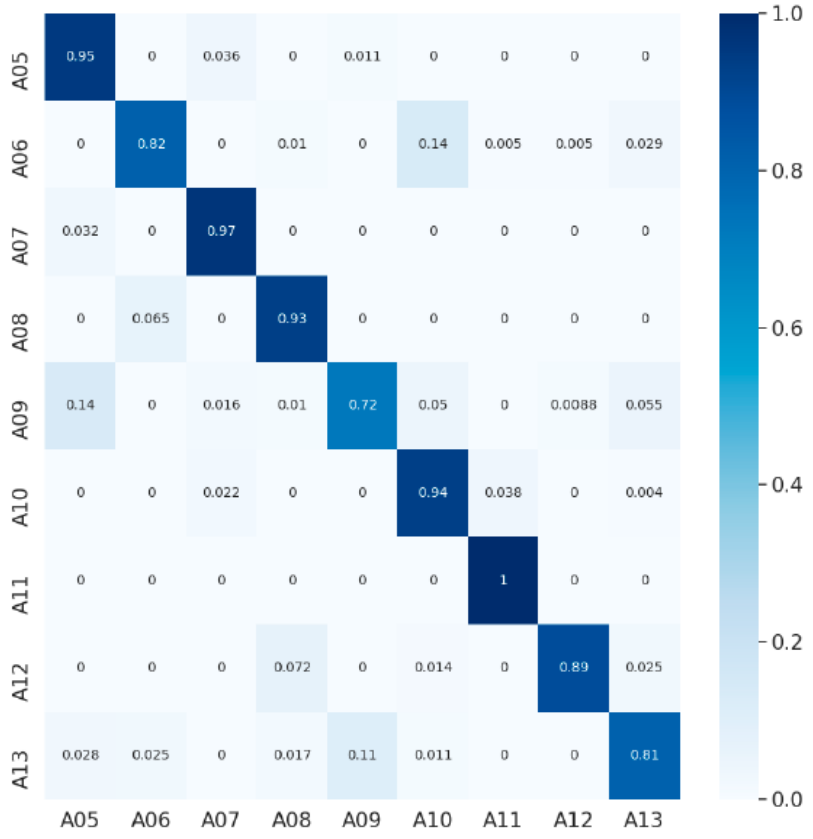
En un trabajo previo, (Arévalo González, 2021) llevó a cabo un estudio en iguales condiciones, realizando el HAR mediante *k-fold cross validation* con los mismos sujetos y acciones pero tomando los datos generados por los IMUs en vez de los obtenidos mediante el sistema de captura óptico. En la tabla 4.5 se muestra la comparativa de los resultados de ambos métodos y en la figura 4.5 comparamos ambas matrices de confusión. En ellas observamos cómo la predicción realizada con los datos procedentes de los IMUs es más precisa que la realizada con imágenes de vídeo en las acciones A05-A08 y A10-A13. En 4.5b también ocurre el mismo fenómeno mencionado previamente; la red confunde actividades unimanuales diferentes realizadas con la misma mano, es decir, confunde A05 con A07 y A06 con A08. Mientras que no observamos la confusión entre A11 y A12. La única actividad que se reconoce correctamente mediante imágenes un mayor número de veces es la A09 (montar unos bloques de Lego).

Modelo	**F1**	**Accuracy**
Imágenes de vídeo	0.5556	0.8222
IMUs	0.87	0.98

Tabla 4.5. Métricas de las matrices de confusión del HAR del tren superior tanto para imágenes de vídeo como para IMUs.



(a) Matriz de confusión normalizada del HAR mediante imágenes de vídeo.



(b) Matriz de confusión normalizada del HAR mediante IMUs (Arévalo González, 2021).

Figura 4.5. Comparativa del HAR mediante imágenes de vídeo y mediante IMUs

4.2.4. HAR mediante un modelo pre-entrenado con 5 acciones

En la siguiente prueba se llevó a cabo el reconocimiento de las 13 acciones de las que se encuentra compuesto el dataset con los 37 sujetos, de los cuales los datos de 32 se utilizaron en el entrenamiento y los 5 restantes en validación y *test*. Como datos de partida, se utilizaron los del modelo pre-entrenado con 5 actividades del dataset HMDB51 mediante *transfer learning*. Los hiperparámetros definidos en el fichero de especificaciones para el entrenamiento han sido los mostrados en la tabla 4.6.

Parámetro	**Valor**
Horizontal flip	0
Épocas	20
Learning rate	0.001
Momentum	0.9
Weight decay	0.0001

Tabla 4.6. Valor de los parámetros configurables del optimizador SGD en el entrenamiento.

La gráfica 4.6 muestra tanto el error de entrenamiento como el error de validación en cada época. Se ha tomado el modelo de la época 16 para realizar la inferencia, la cual tiene un error de 0.64 .

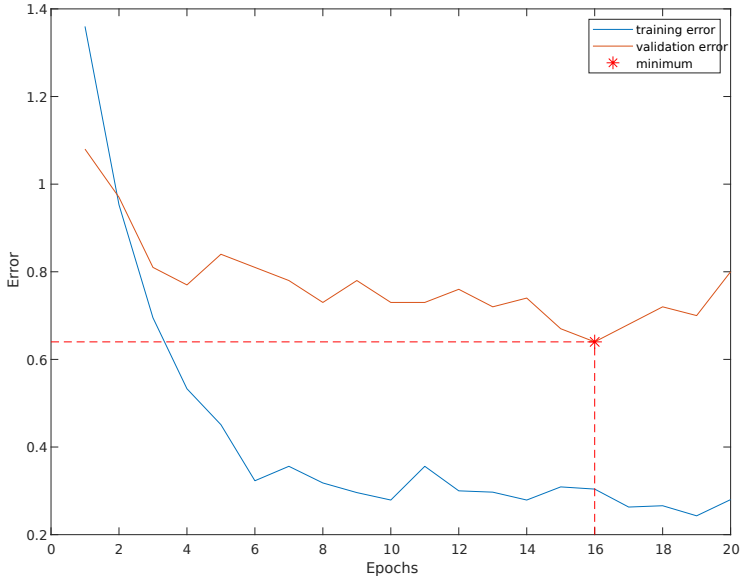


Figura 4.6. Error de entrenamiento y validación durante las 20 épocas de entrenamiento.

Evaluación e inferencia mediante el modo *center*

En primer lugar, se ha utilizado el modo *center* para llevar a cabo tanto la evaluación como la inferencia. En la tabla 4.7 encontramos las métricas resultantes tras realizar inferencia sobre nuestra red entrenada. Los valores de F1 y *accuracy* obtenidos en esta prueba en la que se ha utilizado una mayor cantidad de sujetos (y consesuentemente de datos) en comparación con los utilizados

mediante *k-fold*, son destacablemente mejores.

Sensitivity	Specificity	Precision	Negative predictive value	False positive rate
0.7879	0.9823	0.7879	0.9823	0.0177

False negative rate	False discovery rate	F1	Overall accuracy
0.2121	0.2121	0.7879	0.9674

Tabla 4.7. Métricas del HAR mediante un modelo pre-entrenado con 5 actividades y en modo *center*.

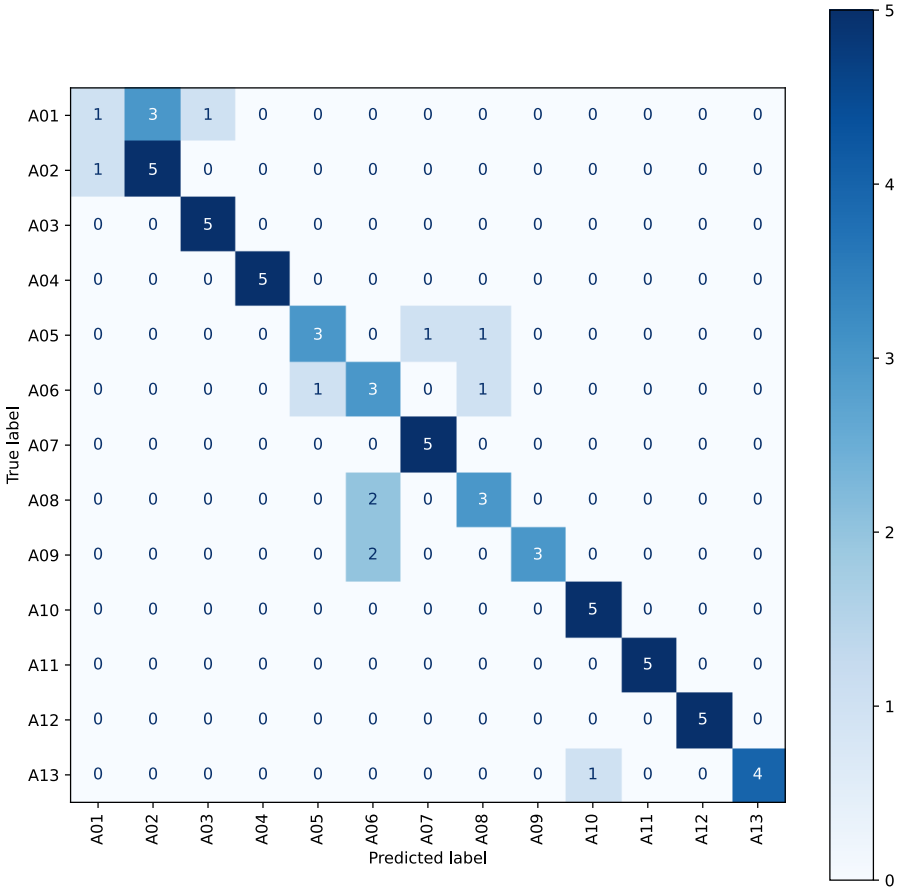


Figura 4.7. Matriz de confusión para el HAR con un modelo pre-entrenado con 5 actividades y en modo *center*.

Complementariamente, en la imagen 4.7 se representa la correspondiente matriz de confusión. Ésta muestra cómo la red mayoritariamente realiza las predicciones de forma correcta, a excepción de A01 (caminar hacia delante). Esta acción se predice como caminar hacia atrás un mayor número de veces de las que se predice correctamente. Asimismo, ocurre nuevamente lo analizado en la prueba anterior. Las actividades unimanuales A05-A07 y A06-A08 sufren de una notable confusión entre ellas mismas, no obstante para A11 (alcanzar un objeto alto con la mano derecha) y A12 (alcanzar un objeto alto con la mano izquierda) no se observa tal confusión. Esto se debe

a que en las actividades A05-A08, el sujeto se encuentra sentado detrás de una mesa, mientras que en las acciones A11 y A12, el sujeto está de pie. Por lo tanto, no se encuentra una similitud tan fuerte entre las acciones realizadas. Cabe destacar que ninguna de las acciones utilizadas en el modelo pre-entrenado son unimanuales por lo que en ese sentido el pre-entrenamiento no aporta ningún beneficio en cuanto al reconocimiento de dichas acciones.

Evaluación e inferencia mediante el modo *conv*

Seguidamente, realizamos de nuevo la inferencia utilizando el modo *conv*, para comprobar de esta manera cuál generaba mejores resultados. En la tabla 4.8 encontramos las métricas resultantes tras realizar inferencia sobre nuestra red entrenada. Los valores de F1 y *accuracy* obtenidos en esta prueba.

Sensitivity	**Specificity**	**Precision**	**Negative predictive value**	**False positive rate**
0.7879	0.9823	0.7879	0.9823	0.0177

False negative rate	**False discovery rate**	**F1**	**Overall accuracy**
0.2121	0.2121	0.7879	0.9674

Tabla 4.8. Métricas del HAR mediante un modelo pre-entrenado con 5 actividades y en modo *conv*.

Podemos observar en la matriz de confusión 4.8 cómo los resultados son exactamente análogos en ambos casos. Para cada uno de los 5 sujetos utilizados en la evaluación, la red ha realizado las mismas predicciones independientemente de los fotogramas seleccionados. Dicho resultado puede deberse a que las actividades realizadas son de corta duración y realmente no supone una importante diferencia en cuanto a los datos introducidos el tomar los fotogramas centrales o el utilizar la técnica de la convolución. En consecuencia, tomamos la decisión de únicamente utilizar uno de los modos en pruebas posteriores.

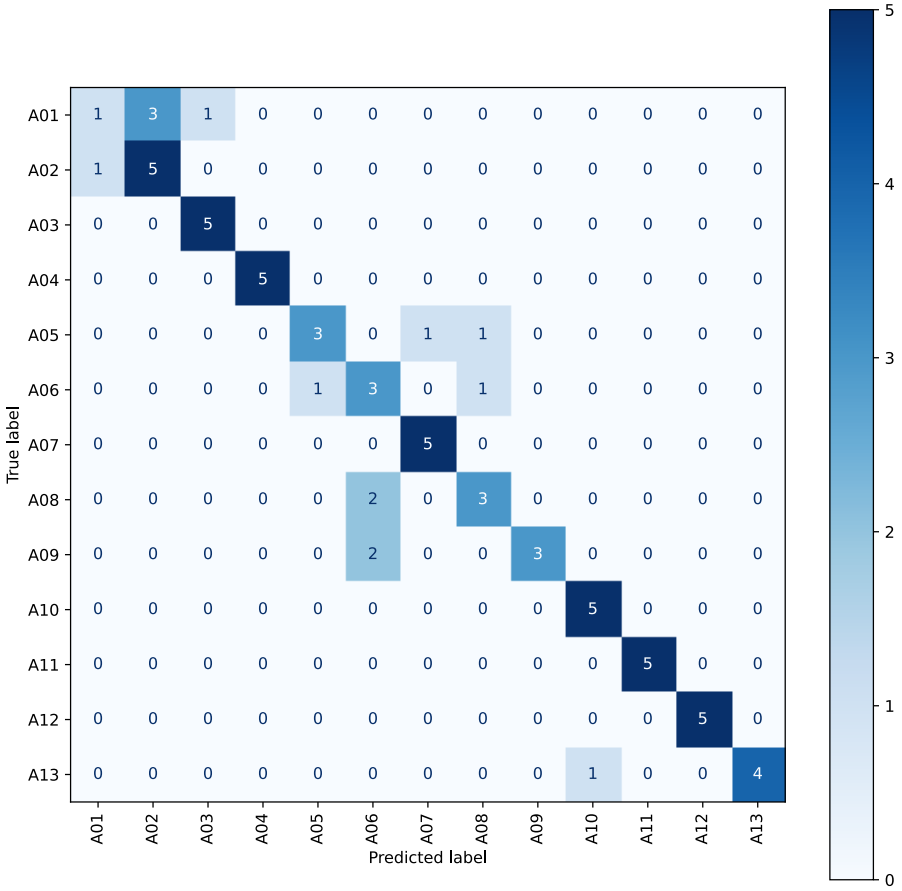


Figura 4.8. Matriz de confusión para el HAR con un modelo pre-entrenado con 5 actividades y en modo *conv*.

4.2.5. HAR mediante un modelo sin pre-entrenamiento

En la siguiente prueba realizada se prescindió del modelo pre-entrenado, para poder hacer un análisis del efecto del *transfer learning* en el modelo. Análogamente, se utilizaron 32 sujetos en el entrenamiento y 5 en validación y test, además de hacer uso de las 13 actividades del dataset. Para llevar a cabo tanto la evaluación como la inferencia, se ha utilizado el modo *center*.

El objetivo de la prueba es el de replicar los valores de F1 y *accuracy* del modelo pre-entrenado que aparecen en la tabla 4.7. Para ello, se han realizado un total de 3 entrenamientos.

Primer entrenamiento

Los hiperparámetros definidos en el fichero de especificaciones para el entrenamiento han sido los mostrados en la tabla 4.9.

Parámetro	**Valor**
Horizontal flip	0
Épocas	20
Learning rate	0.001
Momentum	0.9
Weight decay	0.0001

Tabla 4.9. Valor de los parámetros configurables del optimizador SGD en el entrenamiento.

La gráfica 4.9 muestra tanto el error de entrenamiento como el error de validación en cada época. Se ha tomado el modelo de la época 15, el cual tiene un error de 0.93, para realizar la inferencia.

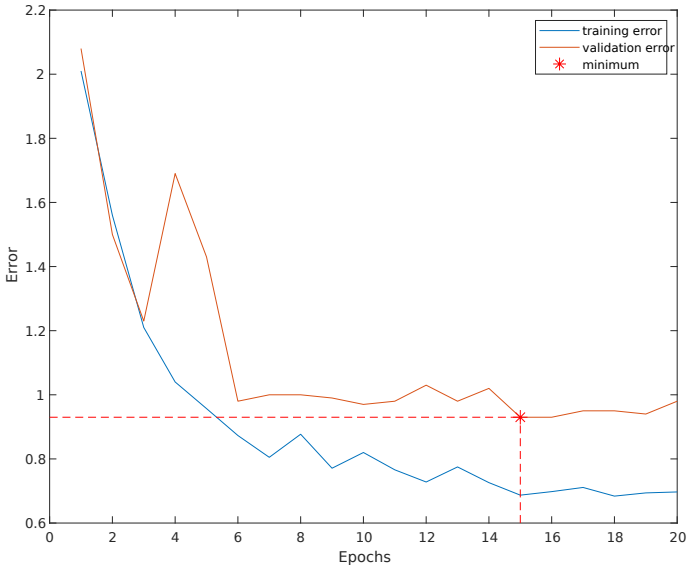


Figura 4.9. Error de entrenamiento y validación durante las primeras 20 épocas de entrenamiento.

En la tabla 4.10 encontramos las métricas resultantes tras realizar inferencia sobre nuestra red entrenada. Complementariamente, en la imagen 4.10 se representa la correspondiente matriz de

confusión. Hemos obtenido mediante este primer entrenamiento con 20 épocas una F1 de 0.67 y un *accuracy* de 0.95. El valor de F1 se encuentra algo distante del obtenido en la prueba anterior, de valor 0.79. Consecuentemente, continuaremos con el entrenamiento de la red. Como datos de partida, tomaremos el modelo obtenido en la época 15 y lo introduciremos en la red como modelo pre-entrenado.

Sensitivity	**Specificity**	**Precision**	**Negative predictive value**	**False positive rate**
0.6666	0.9722	0.6667	0.9722	0.2778

False negative rate	**False discovery rate**	**F1**	**Overall accuracy**
0.3333	0.3333	0.6667	0.9487

Tabla 4.10. Métricas de la matriz de confusión tras el primer entrenamiento.

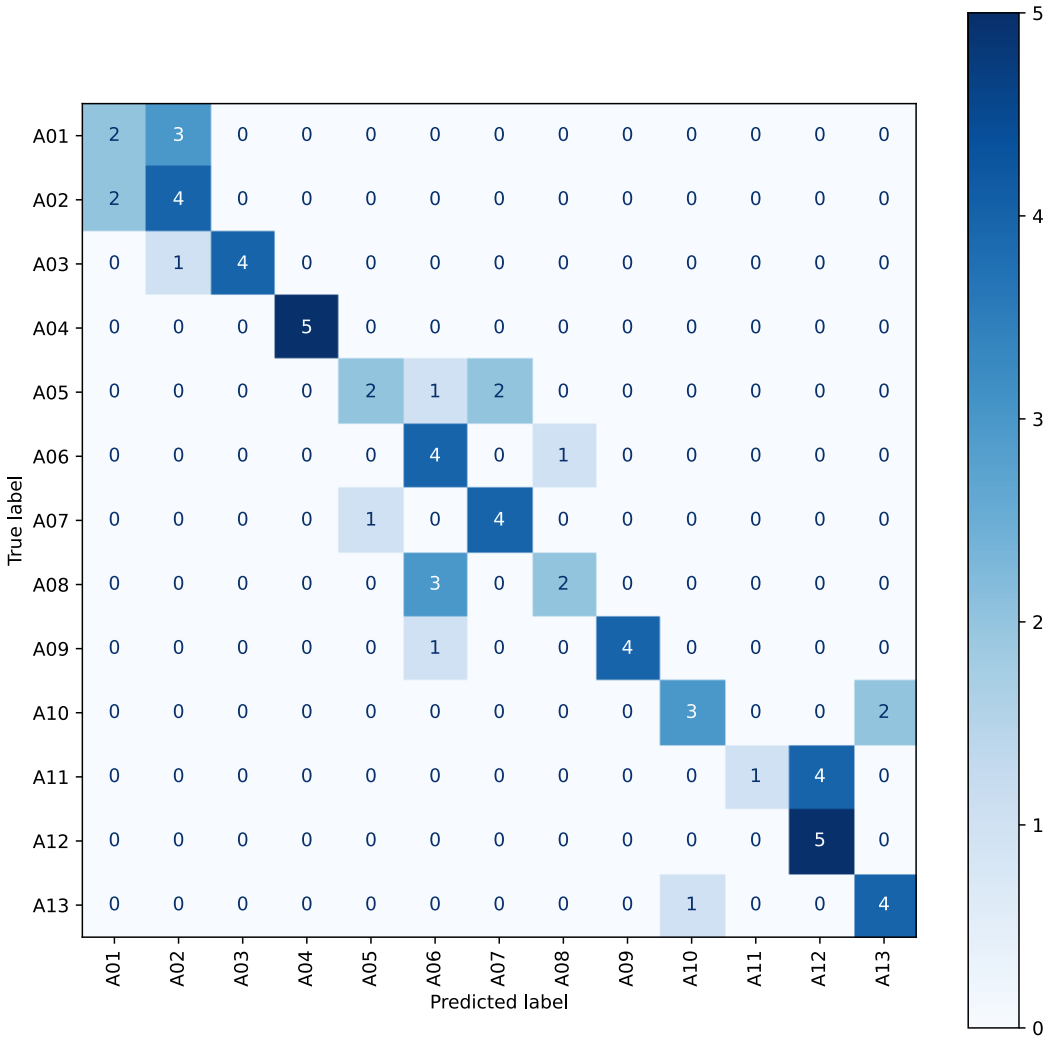


Figura 4.10. Matriz de confusión para el HAR con un modelo pre-entrenado con 5 actividades.

Segundo entrenamiento

En el segundo entrenamiento se ha reducido la tasa de aprendizaje y se ha aumentado el número de épocas utilizadas. La tabla 4.11 recoge todos los hiperparámetros definidos en el fichero de especificaciones para el entrenamiento.

Parámetro	**Valor**
Horizontal flip	0
Épocas	60
Learning rate	0.0001
Momentum	0.9
Weight decay	0.0001

Tabla 4.11. Valor de los parámetros configurables del optimizador SGD en el entrenamiento.

La gráfica 4.11 muestra tanto el error de entrenamiento como el error de validación en cada época. Se ha tomado el modelo de la época 58, el cual tiene un error de 0.84 para realizar la inferencia.

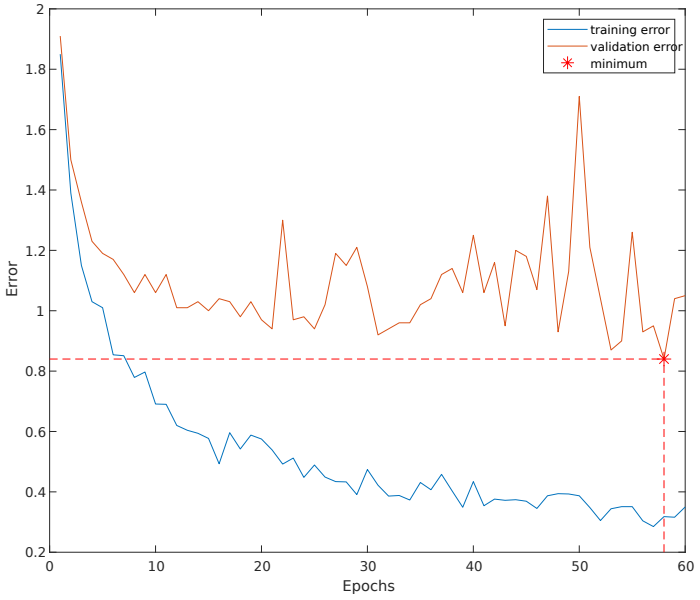


Figura 4.11. Error de entrenamiento y validación durante las 60 épocas de entrenamiento.

En la tabla 4.12 encontramos las métricas resultantes de realizar inferencia en la red entrenada. Complementariamente, en la imagen 4.12 se representa la correspondiente matriz de confusión. Hemos obtenido mediante este segundo entrenamiento con 60 épocas adicionales una F1 de 0.72 y un *accuracy* de 0.96. Se puede observar cómo los valores de F1 y *accuracy* obtenidos han mejorado con respecto a los obtenidos anteriormente, aunque todavía no llegan a lo buscado. Consecuentemente, continuaremos con el entrenamiento de la red. Como datos de partida, tomaremos el modelo obtenido en la época 58 y lo introduciremos en la red como modelo pre-entrenado.

Sensitivity	Specificity	Precision	Negative predictive value	False positive rate
0.7273	0.9773	0.7273	0.9773	0.2778

False negative rate	False discovery rate	F1	Overall accuracy
0.2727	0.2727	0.7273	0.9580

Tabla 4.12. Métricas de la matriz de confusión tras el segundo entrenamiento.

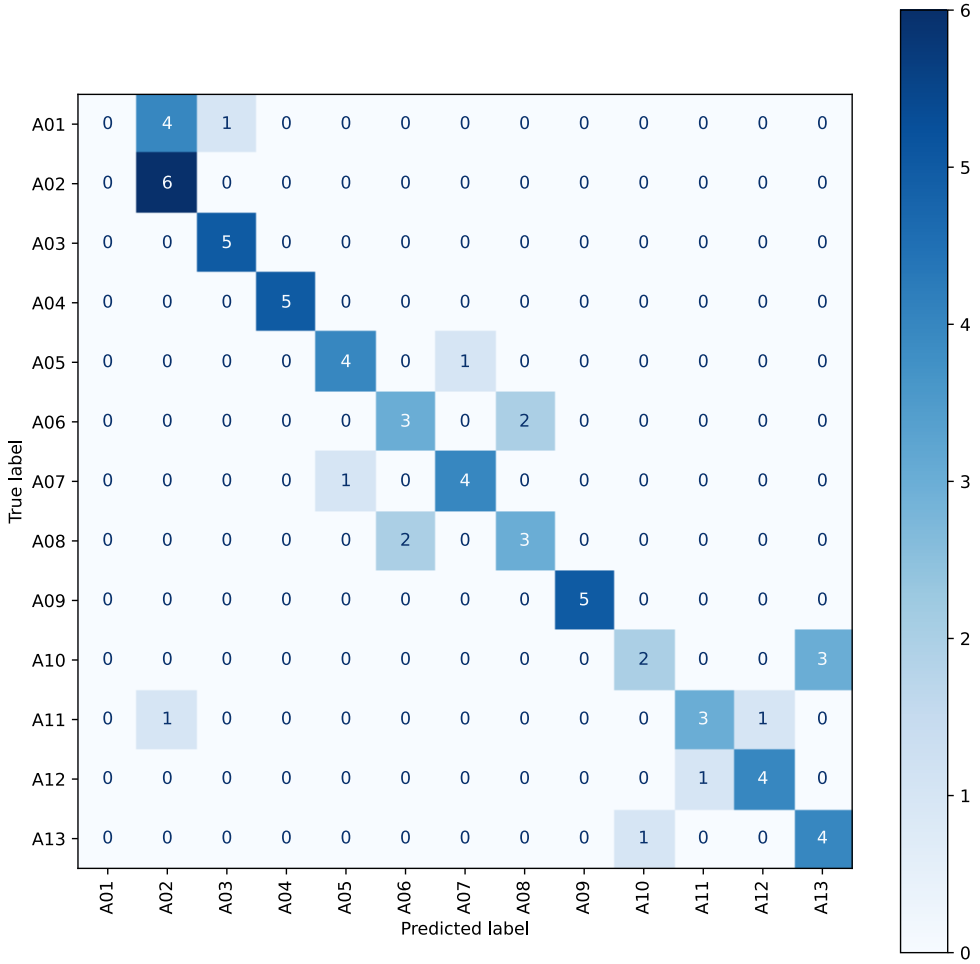


Figura 4.12. Matriz de confusión para el HAR sin modelo pre-entrenado.

Tercer entrenamiento

En la gráfica 4.11 se observa como a lo largo de las 60 épocas de entrenamiento, el error sigue disminuyendo. Por lo tanto, continuaremos entrenando la red con la misma tasa de aprendizaje. Los hiperparámetros definidos en el fichero de especificaciones para el entrenamiento se recogen en la tabla 4.13.

Parámetro	**Valor**
Horizontal flip	0
Épocas	60
Learning rate	0.0001
Momentum	0.9
Weight decay	0.0001

Tabla 4.13. Valor de los parámetros configurables del optimizador SGD en el entrenamiento.

La gráfica 4.13 muestra tanto el error de entrenamiento como el error de validación en cada época. Se ha tomado el modelo de la época 45, el cual tiene un error de 0.80 para realizar la inferencia.

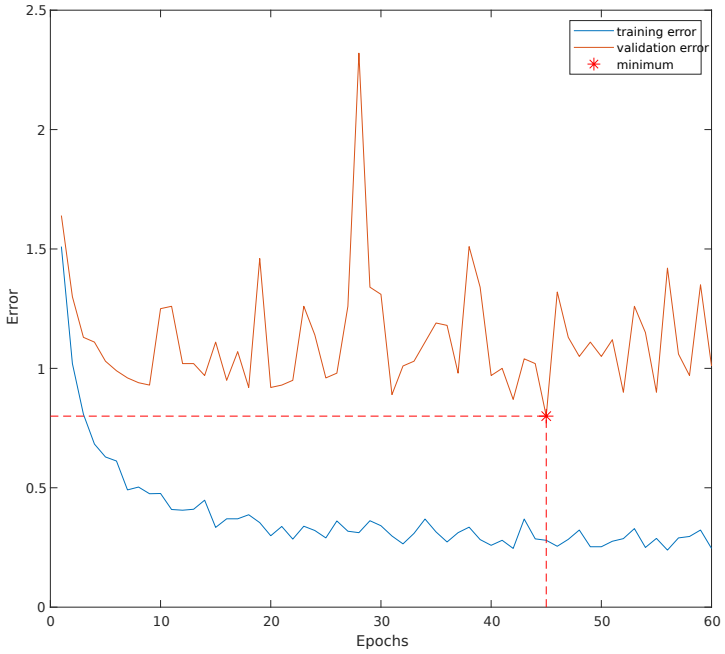


Figura 4.13. Error de entrenamiento y validación durante las 60 épocas de entrenamiento.

En la imagen 4.14 se representa la correspondiente matriz de confusión. En ella, seguimos encontrando características similares a las analizadas en el modelo pre-entrenado. Por una parte, la única actividad que no se reconoce es A01 (caminar), la cual se predice como A02 (caminar hacia atrás) mayoritariamente. Además encontramos bastante confusión entre las actividades unimanuales A05-A08. No obstante, también encontramos notables diferencias. Por un lado, las predicciones para la actividad A09 (montar unos bloques de Lego) son mejores. Además, la red falla en la predicción de las actividades A11 y A12, en las cuales no encontrábamos problema an-

teriormente. Adicionalmente, incluso aparece confusión en las actividades bimanuales A10 (lanzar un balón y cogerlo de nuevo) y A13 (romper un papel con ambas manos, hacer una bola y tirarla).

Complementariamente, en la tabla 4.14 encontramos las métricas resultantes tras realizar inferencia sobre nuestra red entrenada. En este último entrenamiento hemos logrado obtener un valor de F1 de 0.78 además de un *accuracy* de 0.97, similares a los obtenidos mediante el uso del modelo pre-entrenado con 5 actividades.

Sensitivity	**Specificity**	**Precision**	**Negative predictive value**	**False positive rate**
0.7727	0.9811	0.7727	0.9811	0.1894

False negative rate	**False discovery rate**	**F1**	**Overall accuracy**
0.2273	0.2273	0.7727	0.9650

Tabla 4.14. Métricas de la matriz de confusión tras el tercer entrenamiento.

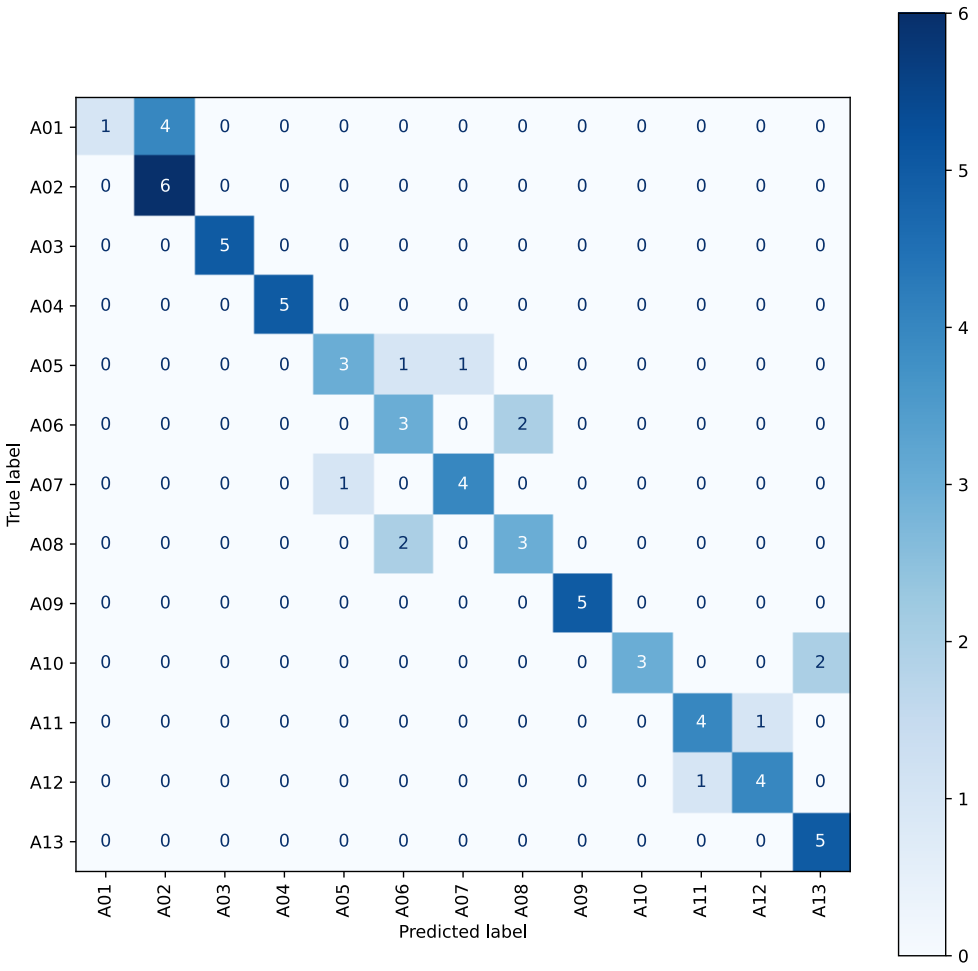


Figura 4.14. Matriz de confusión para el HAR sin modelo pre-entrenado.

Realizando el entrenamiento de la red ResNet-18 con las 13 actividades del dataset, los 37 sujetos y sin utilizar pre-entrenamiento hemos necesitado llevar a cabo tres entrenamientos, con un total de 140 épocas. El resultado obtenido finalmente es análogo a haber utilizado un modelo pre-entrenado y 20 épocas de entrenamiento. Por consiguiente, la utilización de *transfer learning* supone un ahorro computacional notable.

4.2.6. Replicación del modelo pre-entrenado

Previamente analizamos cómo las mejores predicciones de la red se consiguen haciendo uso de un modelo pre-entrenado. No obstante, encontrábamos una importante confusión en las actividades unimanuales. Con el objetivo de mitigar este problema, se llevó a cabo el entrenamiento de un modelo utilizando tanto actividades del tronco superior unimanuales y bimanuales como del tronco inferior del dataset HMDB51, para así introducirlo en la red como modelo pre-entrenado.

4.2.6.1. Pre-entrenamiento de la red

En primer lugar se realizó un entrenamiento de la red con 11 actividades del dataset HMDB51 cuya información se encuentra recogida en la tabla 4.15. Para el entrenamiento se emplearon 70 vídeos de cada clase y para validación y *test* 30.

Clases	**Número total de vídeos**
Caminar	494
Sentarse	142
Levantarse	154
Atrapar un objeto	102
Beber	164
Aplaudir	130
Atrapar un objeto	102
Lanzar un objeto	102
Verter un líquido	106
Fumar	109
Comer	108

Tabla 4.15. Estadísticas de los vídeos del dataset HMDB51 utilizados en el pre-entrenamiento.

Los hiperparámetros definidos en el fichero de especificaciones se muestran en la tabla 4.16.

Parámetro	**Valor**
Horizontal flip	0
Épocas	80
Learning rate	0.01
Momentum	0.9
Weight decay	0.0001

Tabla 4.16. Valor de los parámetros configurables del optimizador SGD en el entrenamiento.

La gráfica 4.15 muestra tanto el error de entrenamiento como el error de validación en cada época. Se ha tomado el modelo de la época 11, el cual tiene un error de 2.14 para realizar la inferencia. A continuación se utilizará *transfer learning*, y se entrenará la red para nuestro dataset de interés.

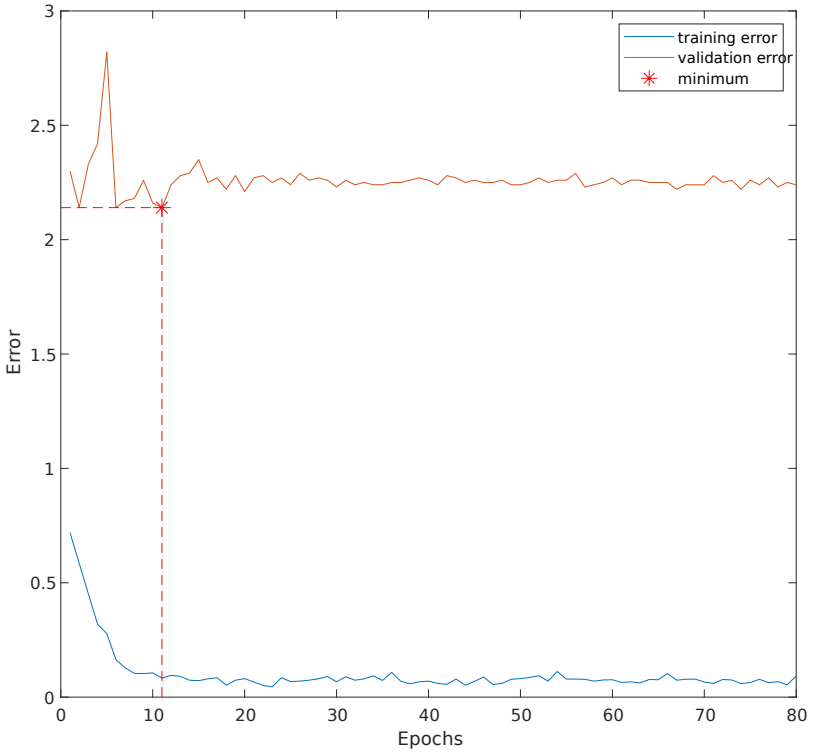


Figura 4.15. Error de entrenamiento y validación durante las 80 épocas de entrenamiento.

4.2.6.2. HAR aplicando el modelo pre-entrenado

Primer entrenamiento

En la siguiente prueba se llevó a cabo el reconocimiento de las 13 acciones de las que se encuentra compuesto el dataset con los 37 sujetos. Se ha hecho uso del modelo pre-entrenado con 11 actividades descrito en el apartado anterior. Para llevar a cabo tanto la evaluación como la inferencia, se ha utilizado el modo *center*.

El objetivo de la prueba es el de replicar los valores de F1 y *accuracy* del modelo pre-entrenado que aparecen en la tabla 4.7. Para ello, se han realizado un total de 2 entrenamientos.

Los hiperparámetros definidos en el fichero de especificaciones se muestran en la tabla 4.17.

Parámetro	**Valor**
Horizontal flip	0
Épocas	30
Learning rate	0.001
Momentum	0.9
Weight decay	0.0001

Tabla 4.17. Valor de los parámetros configurables del optimizador SGD en el entrenamiento.

La gráfica 4.16 muestra tanto el error de entrenamiento como el error de validación en cada época. Se ha tomado el modelo de la época 24, el cual tiene un error de 0.82 para realizar la inferencia.

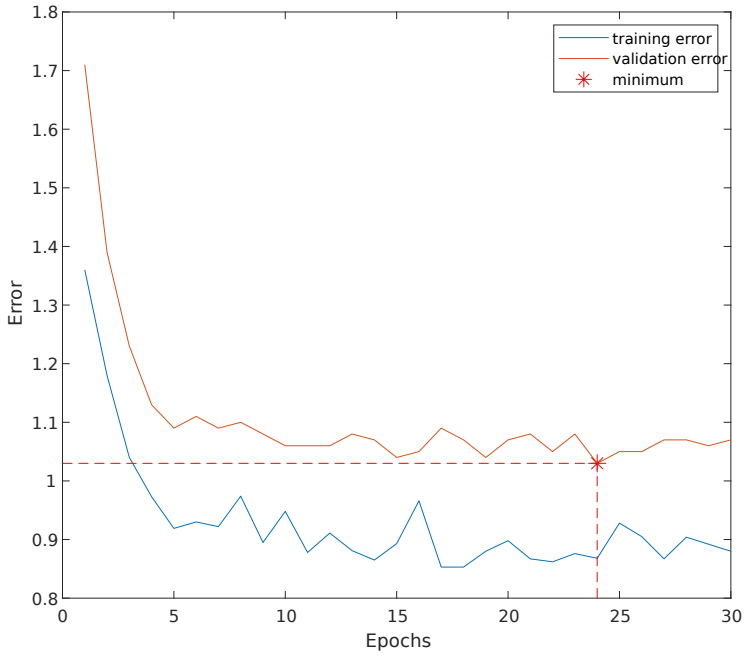


Figura 4.16. Error de entrenamiento y validación durante las 30 épocas de entrenamiento.

En la tabla 4.18 encontramos las métricas resultantes tras realizar inferencia sobre nuestra red

entrenada. Complementariamente, en la imagen 4.17 se representa la correspondiente matriz de confusión. Hemos obtenido mediante este primer entrenamiento con 30 épocas una F1 de 0.74 y un *accuracy* de 0.96. Se observa como a partir de dicha época, el error comienza a aumentar. Por lo tanto, no continuaremos realizando entrenamientos.

Sensitivity	Specificity	Precision	Negative predictive value	False positive rate
0.7424	0.9785	0.7424	0.9785	0.0215

False negative rate	False discovery rate	F1	Overall accuracy
0.2576	0.2576	0.7424	0.9603

Tabla 4.18. Métricas de la matriz de confusión para el primer entrenamiento con el modelo pre-entrenado con 11 actividades.

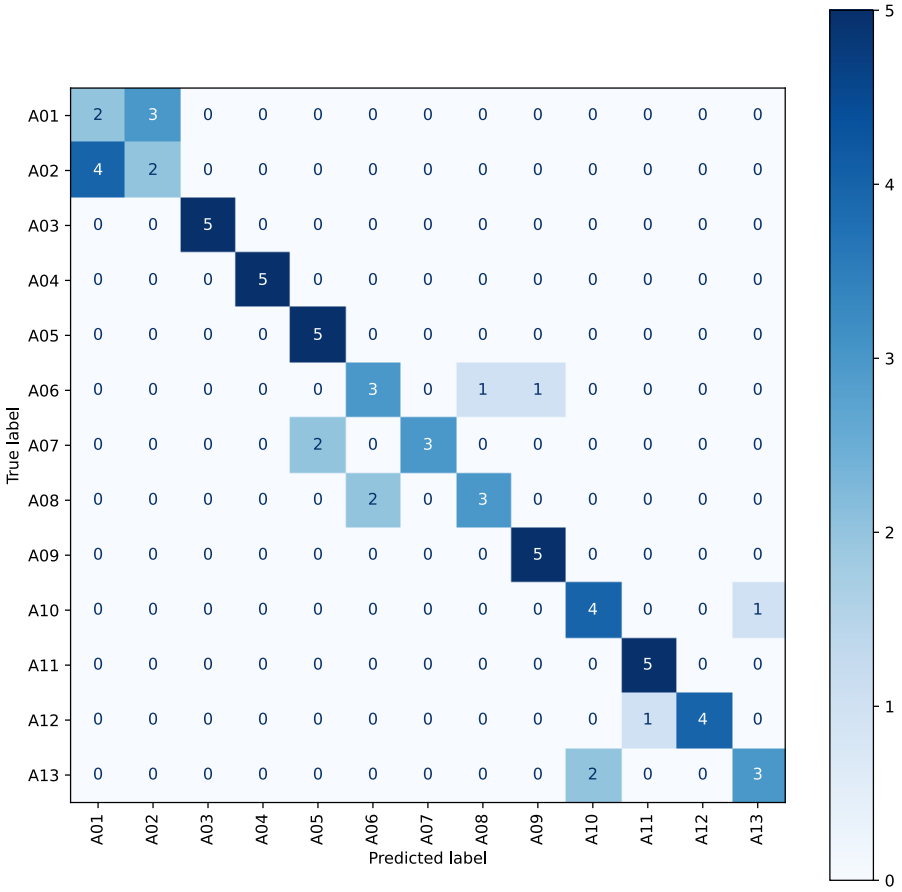


Figura 4.17. Matriz de confusión para el HAR con un modelo pre-entrenado con 11 actividades.

Realizando el entrenamiento de la red ResNet-18 con las 13 actividades del dataset, los 37 sujetos y haciendo uso de un modelo pre-entrenado con 11 actividades, hemos necesitado llevar a cabo un único entrenamiento, con un total de 30 épocas. El resultado obtenido finalmente se aproxima al observado con el modelo pre-entrenado con 5 actividades y 20 épocas de entrenamiento.

4.2.7. Comparación de los modelos

Asimismo, cabe señalar las similitudes y diferencias encontradas en las tres últimas pruebas, para las cuales se utilizó la misma cantidad de datos (32 sujetos para llevar a cabo el entrenamiento y 5 para validación y *test*). La tabla 4.19 recoge las métricas correspondientes a cada uno de los modelos.

El modelo entrenado desde cero, se caracteriza por el alto coste computacional necesitado para obtener resultados. Se llevaron a cabo 3 entrenamientos, con un total de 120 épocas. Los resultados muestran una fuerte confusión en cuatro tipos de actividades.

- Actividades de tronco inferior: caminar hacia delante (A01) y hacia atrás (A02). La red identifica correctamente andar hacia atrás, no obstante, andar hacia delante lo predice mayoritariamente (4 de 5 veces) como andar hacia atrás. Cabe destacar además, que la acción caminar sobre una línea (A03) no es confundida con caminar hacia delante ni hacia atrás, pese a la similitud que guardan en el movimiento realizado las tres actividades.

- Actividades de tronco superior unimanuales: Mover un vaso con la mano derecha (A05) e izquierda (A06) y beber con la mano derecha (A07) e izquierda (A08). Para la realización de las actividades, el sujeto se encuentra sentado tras una mesa. Parte de una posición inicial en la cual sus palmas se encuentras posadas sobre los muslos, realiza el de beber o mover un vaso y retorna a la posición inicial. En estos casos se confunden las actividades bimanuales realizadas con la misma mano mayoritariamente, es decir, se confunde mover un vaso con la mano derecha (A05) con beber con la mano derecha (A07) y viceversa, así como se confunde mover un vaso con la mano izquierda (A06) con beber con la mano izquierda (A08) y viceversa.

- Actividades de tronco superior unimanuales: Alcanzar un objeto alto con la mano derecha (A11) e izquierda (A12). Durante la realización de la actividad, el sujeto se encuentra de pie. Este caso no guarda tanta similitud con las acciones bimanuales A05-A08 debido a que el sujeto se encuentra de pie y la ausencia de la mesa y la silla. Para este caso sí se observa confusión entre la misma actividad realizada con ambas manos.

- Actividades de tronco superior bimanuales: Lanzar un balón y cogerlo de nuevo (A10) y romper un papel, hacer una bola y lanzarlo (A13). Se observa para dichas actividades el mismo caso que el analizado para las actividades caminar hacia delante y hacia atrás. La actividad romper un papel, hacer una bola y lanzarlo se predice correctamente, no obstante, la actividad lanzar un balón y cogerlo de nuevo se predice 2 de 5 veces como la anterior. La posición del sujeto para estas acciones es similar, puesto que éste se encuentra de pie y realiza la acción con ambas manos.

Por otra parte, el modelo pre-entrenado con 5 actividades, destaca por ser el que menor coste computacional requiere, de los tres modelos. Se llevó a cabo un único entrenamiento, con un total de 20 épocas. Asimismo, pese a haber obtenido métricas similares en los tres modelos, los resultados del presente modelo son los más óptimos. En comparación con los resultados previos se observa:

- Actividades de tronco inferior: caminar hacia delante (A01) y hacia atrás (A02). La red identifica correctamente andar hacia atrás, no obstante, andar hacia delante lo predice mayoritariamente (3 de 5 veces) como andar hacia atrás. En contraposición con lo observado anteriormente, si que encontramos confusión a mayores con la acción de caminar sobre una

línea (A03). Pese a que dicha acción se identifica de manera correcta en todos los casos, existe un caso en el que caminar hacia delante se confunde con caminar sobre la línea.

- Actividades de tronco superior unimanuales: Mover un vaso con la mano derecha (A05) e izquierda (A06) y beber con la mano derecha (A07) e izquierda (A08). Se sigue manteniendo la confusión entre actividades diferentes realizadas con la misma mano, sin embargo, también hay confusión entre la misma actividad realizada con ambas manos. Igualmente, también se observa confusión con la actividad montar una torre (A09), la cual se reconocía de correctamente en el anterior modelo. Esta se identifica como mover un vaso con la mano izquierda 2 de 5 veces.

- Actividades de tronco superior unimanuales: Alcanzar un objeto alto con la mano derecha (A11) e izquierda (A12). A diferencia de lo observado en el anterior modelo, los resultados muestran una correcta identificación de ambas acciones.

- Actividades de tronco superior bimanuales: Lanzar un balón y cogerlo de nuevo (A10) y romper un papel, hacer una bola y lanzarlo (A13). Se observa para dichas actividades el mismo caso que en el modelo anterior. Sin embargo, esta vez la actividad lanzar un balón y cogerlo de nuevo se clasifica correctamente todas las veces, mientras que en un caso romper un papel, hacer una bola y lanzarlo se predice como esta última.

Finalmente, el modelo pre-entrenado con 11 actividades presenta un término medio entre los dos modelos analizados previamente. Se llevó a cabo un único entrenamiento, con un total de 30 épocas. Asimismo, pese a haber obtenido métricas similares en los tres modelos, en el presente modelo es en el cual observamos mayor confusión:

- Actividades de tronco inferior: caminar hacia delante (A01) y hacia atrás (A02). A diferencia de lo observado en casos anteriores, la red confunde ambas actividades. 3 de 5 veces identifica caminar hacia delante como caminar hacia atrás y 4 de 6 predice caminar hacia atrás como caminar hacia delante. Al igual de lo observado para el modelo sin pre-entrenamiento, la acción caminar sobre una línea (A03) no es confundida con caminar hacia delante ni hacia atrás.

- Actividades de tronco superior unimanuales: Mover un vaso con la mano derecha (A05) e izquierda (A06) y beber con la mano derecha (A07) e izquierda (A08). En este último modelo también se mantiene la confusión entre actividades diferentes realizadas con la misma mano, sin embargo, esto no se observa entre la misma actividad realizada con ambas manos. Igualmente, también aparece confusión con la actividad montar una torre (A09), la cual se reconoce de correctamente, pero en un caso se identifica mover un vaso con el brazo izquierdo como montar una torre.

- Actividades de tronco superior unimanuales: Alcanzar un objeto alto con la mano derecha (A11) e izquierda (A12). A diferencia de lo observado en el primer modelo, los resultados muestran una correcta identificación de ambas acciones salvo en un único caso en el que la actividad realizada con la mano izquierda se confunde con la análoga de la mano derecha.

- Actividades de tronco superior bimanuales: Lanzar un balón y cogerlo de nuevo (A10) y romper un papel, hacer una bola y lanzarlo (A13). Se observa para dichas actividades el mismo caso que en los modelos anteriores. Lanzar un balón y cogerlo de nuevo se identifica una única vez erróneamente y romper un papel, hacer una bola y lanzarlo se identifica 2 de 5 veces como la anterior actividad.

Pruebas	Métricas				
Modelo	**Sensitivity**	**Specificity**	**Precision**	**F1**	**Overall accuracy**
Sin pre-entreno	0.7727	0.9811	0.7727	0.7727	0.9650
Pre-entrenado (5 act.)	0.7879	0.9823	0.7879	0.7879	0.9674
Pre-entrenado (11 act.)	0.7424	0.9785	0.7424	0.7424	0.9603

Tabla 4.19. Comparación de las métricas obtenidas para cada uno de los tres modelos utilizados para llevar a cabo el HAR.

Capítulo 5

Conclusiones y líneas futuras

5.1. Cumplimiento de los objetivos del trabajo fin de grado

El objetivo de este TFG ha sido realizar el Reconocimiento de la Actividad Humana para los datos adquiridos mediante sistemas de captura ópticos del dataset del grupo de investigación. Estos datos se componen de una serie de sujetos realizando actividades de la vida diara, las cuales se caracterizan por su relevancia clínica. Para ello, se ha entrenado una Red Neuronal ResNet-18 y posteriormente se ha llevado a cabo inferencia sobre la misma.

Para poder llevarlo a cabo, se describieron en el Capítulo 1 una serie de fases a cumplir:

i. Se ha asistido a las clases de Técnicas Computacionales en Biomedicina impartidas por el Dr. D. Mario Martínez Zarzuela sobre Inteligencia Artificial, Aprendizaje Profundo y Redes Neuronales, realizando así una introducción a dichos conceptos.

ii. Se ha hecho una lectura tanto de libros como artículos relacionados con las distintas arquitecturas de las redes neuronales así como los algoritmos implementados y sus características más importantes.

iii. Se ha realizado una lectura y revisión de artículos en los que se describen los *benchmark* datasets existentes.

iv. Se ha trabajado tanto con el *sofware* Jupyther Lab como con TAO Toolkit y adquirido el dominio necesario para poder ejecutar redes neuronales que lleven a cabo el reconocimiento de actividades.

v. Se han ejecutado *scripts* en Python para llevar a cabo la organización de los datos a introducir a la Red Neuronal.

vi. Se han analizado los resultados obtenido y modificado el valor de los hiperparámetros convenientemente para alcanzar los resultados más favorables.

vii. Se ha hecho un análisis de los resultados finales obtenidos para cada uno de los tres modelos empleados para el HAR.

viii. Una vez obtenidos los resultados, se han extraído las conclusiones pertinentes así como las limitaciones encontradas en este TFG.

5.2. Conclusiones

A partir de los resultados obtenidos se extraen las conclusiones descritas a continuación:

- Los resultados obtenidos de aplicar la Red Neuronal mediante *k-fold* a los datos obtenidos mediante el sistema de captura óptico, para situaciones en las que se dispone de un número reducido de datos son significantemente peores que los obtenidos llevando a cabo el análisis de los datos adquiridos por los sensores.
- Existe una fuerte confusión entre las actividades bimanuales, las cuales además se caracterizan por el reducido movimiento realizado por el sujeto, característica por la cual son difíciles de predecir correctamente. Adicionalmente, los modos *center* y *conv* utilizados en la inferencia no son los más óptimos a la hora de escoger los fotogramas introducidos a la red. Esto se debe a que los sujetos realizan una serie de repeticiones para cada actividad, por lo que es probable en algún caso los fotogramas tomados coincidan con el final o comienzo de la realización de la actividad. Para el caso de las actividades de beber con una mano y mover una botella, el sujeto comienza y termina en la misma posición, sentado y con las palmas sobre los muslos.
- Se ha observado el gran coste computacional necesario para realizar los entrenamientos de la red neuronal desde cero. Asimismo, se demuestra la eficacia del *tranfer learning* a la hora de alcanzar la convergencia de manera más temprada, ahorrando recursos computacionales.

5.3. Líneas futuras

Los resultados de este trabajo pueden ser ampliados, solucionando algunas de las limitaciones encontradas. En primer lugar, modificando el modo en el que se realiza la inferencia. Como se ha analizado previamente, dada la naturaleza de los vídeos en los que se realizan varias repeticiones de las actividades, es posible que el modo *center* y el *conv* no tomen los fotogramas adecuados para poder llevar a cabo una correcta clasificación de la actividad. Por esta razón, sería conveniente disponer de alguna herramienta que sea capaz de escoger de manera óptima los fotogramas de cada segmento de vídeo a introducir a la red.

En segundo lugar, para casos de aplicación concretos podría hacerse un estudio de las actividades relevantes en dicho caso de estudio. Realizando las pruebas nuevamente para las actividades pertinentes, situación en la cual es altamente probable que se eliminen actividades de similar naturaleza, y finalmente obteniendo las conclusiones pertinentes.

Otra oportunidad de investigación pasaría por realizar una búsqueda de datasets compuestos tanto por activiades de tronco inferior como por actividades bimanuales de tronco superior, en las que los sujetos no realizen gran movimiento, puesto que se ha comprobado que las acciones del dataset HMDB51 no guardan especial similitud con el utilizado para el HAR. Tomando ese dataset, podría entrenarse la red y utilizar el modelo resultante como modelo pre-entrenado mediante *transfer learning*.

Por último, otra línea de investigación interesante sería aprovechar el carácter multimodal de la base de datos utilizada. Es posible llevar a cabo el Reconocimiento de la Actividad Humana con la fusión tanto de los datos provenientes de los sensores como los provenientes del sistema de captura óptico.

Glosario de siglas y acrónimos

AI:	Artificial Intelligence
DL:	Deep Learning
ML:	Machine Learning
NPL:	Natural Language Processing
RNN:	Recurrent Neural Network
CNN:	Convolutional Neural Network
ReLU:	Rectified Linear Unit
MSE:	Mean Squared Error
SGD:	Stochastic Gradient Descend
MPL:	Multilayer Perceptron
HAR:	Human Activity Recognition
ADL:	Activity of Daily Living

Bibliografía

Aggarwal, C. C. (2018). *Neural networks and deep learning: a textbook*. Springer.

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Hasan, M., Van Essen, B. C., Awwal, A. A. S., and Asari, V. K. (2019). A state-of-the-art survey on deep learning theory and architectures. *Electronics*, 8(3).

Arévalo González, D. (2021). Diseño e implementación de redes neuronales de aprendizaje profundo para clasificación y análisis de movimientos corporales capturados mediante dispositivos vestibles. Technical report, Universidad de Valladolid.

Atallah, L., Lo, B., King, R., and Yang, G.-Z. (2011). Sensor positioning for activity recognition using wearable accelerometers. *IEEE Transactions on Biomedical Circuits and Systems*, 5(4):320–329.

Banos, O., Tóth, M., and Amft, O. (2012). REALDISP dataset.

Bartalesi, R., Lorussi, F., Tesconi, M., Tognetti, A., Zupone, G., and de rossi, D. (2005). Wearable kinesthetic system for capturing and classifying upper limb gesture. pages 535– 536.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *arXiv:1206.5533 [cs]*.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Bosman, A., Engelbrecht, A., and Helbig, M. (2020). Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *Neurocomputing*, 400.

Bottou, L. (2012). Stochastic gradient descent tricks. In Montavon, G., Orr, G. B., and Müller, K.-R., editors, *Neural Networks: Tricks of the Trade: Second Edition*, pages 421–436. Springer.

Brox, T., Bruhn, A., and Fritz, M., editors (2019). *Pattern Recognition - 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings*, volume 11269 of *Lecture Notes in Computer Science*. Springer.

Cao, K., Liu, Y., Meng, G., and Sun, Q. (2020). An overview on edge computing research. *IEEE Access*, PP:1–1.

Chollet, F. (2018). *Deep learning with Python*. Manning.

Ciliberto, M., Fortes Rey, V., Calatroni, A., Lukowicz, P., and Roggen, D. (2021). Opportunity++: A multimodal dataset forVideo- and wearable, object and ambientsensors-based human activity recognition.

Cippitelli, E., Gambi, E., and Spinsante, S. (2017). Human action recognition with RGB-d sensors. In Travieso-Gonzalez, C. M., editor, *Motion Tracking and Gesture Recognition*. IntechOpen.

Cook, D., Feuz, K., and Krishnan, N. (2013). Transfer learning for activity recognition: A survey. *Knowledge and information systems*, 36:537–556.

Das Antar, A., Ahmed, M., and Ahad, M. A. R. (2019). Challenges in sensor-based human activity recognition and a comparative analysis of benchmark datasets: A review. In *2019 Joint 8th International Conference on Informatics, Electronics Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision Pattern Recognition (icIVPR)*, pages 134–139.

Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. (2017). Language modeling with gated convolutional networks. *PMLR*, 70:933–941.

Ding, B., Qian, H., and Zhou, J. (2018). Activation functions and their characteristics in deep neural networks. In *2018 Chinese Control And Decision Conference (CCDC)*, pages 1836–1841. IEEE.

Ehatisham-Ul-Haq, M., Javed, A., Azam, M. A., Malik, H. M. A., Irtaza, A., Lee, I. H., and Mahmood, M. T. (2019). Robust human activity recognition using multimodal feature-level fusion. *IEEE Access*, 7:60736–60751.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Fleron, M. K., Ubbesen, N. C. H., Battistella, F., Dejtiar, D. L., and Oliveira, A. S. (2019). Accuracy between optical and inertial motion capture systems for assessing trunk speed during preferred gait and transition periods. *Sports Biomechanics*, 18(4):366–377.

Fürnkranz, J. and Society, I. M. L., editors (2010). *Proceedings, Twenty-Seventh International Conference on Machine Learning: held June 21 - June 25 in Haifa, Israel*. ACM.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323. PMLR.

González-Alonso, J., Martínez-Zarzuela, M., Aguado, H., and Díaz-Pernas, F. J. (2020). Validación de prototipo con sensores vestibles para captura y análisis de movimientos del cuerpo humano con aplicación en medicina. In *XXXVIII Congreso Anual de la Sociedad Española de Ingeniería Biomédica. CASEIB 2020*.

González-Alonso, J., Oviedo-Pastor, D., Aguado, H. J., Díaz-Pernas, F. J., González-Ortega, D., and Martínez-Zarzuela, M. (2021). Custom imu-based wearable system for robust 2.4 ghz wireless human body parts orientation tracking and 3d movement visualization on an avatar. *Sensors*, 21(19).

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. Adaptive computation and machine learning. The MIT Press.

Hassan, M. M., Huda, S., Uddin, M. Z., Almogren, A., and Alrubaian, M. (2018). Human activity recognition from body sensor data using deep learning. *J. Med. Syst.*, 42(6):1–8.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv:1512.03385 [cs]*.

Heilbron, F. C., Escorcia, V., Ghanem, B., and Niebles, J. C. (2015). ActivityNet: A large-scale video benchmark for human activity understanding. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–970.

Heskes, T. M. and Kappen, B. (1993). Online learning processes in artificial neural networks. In Taylor, J. G., editor, *NorthHolland Mathematical Library*, volume 51, pages 199–233. Elsevier.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554.

Hochreiter, S. (1991). *Untersuchungen zu dynamischen neuronalen Netzen.* phdthesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167 [cs]*.

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339.

Jiang, S., Cao, Y., Iyengar, S., Kuryloski, P., Jafari, R., Xue, Y., Bajcsy, R., and Wicker, S. (2008). Carenet: An integrated wireless sensor networking environment for remote healthcare.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. (2017-05-19). The kinetics human action video dataset. *arXiv:1705.06950 [cs]*.

Kelleher, J. D. (2019). *Deep learning*. The MIT press essential knowledge series. The MIT Press.

Kidziński, L., Yang, B., Hicks, J., Rajagopal, A., Delp, S., and Schwartz, M. (2020). Deep neural networks enable quantitative movement analysis using single-camera videos. *Nature Communications*, 11:4054.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., and Serre, T. (2011). HMDB: A large video database for human motion recognition. In *2011 International Conference on Computer Vision*, pages 2556–2563.

Kwon, H., Tong, C., Haresamudram, H., Gao, Y., Abowd, G. D., Lane, N. D., and Plötz, T. (2020). IMUTube: Automatic extraction of virtual on-body accelerometry from video for human activity recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 4(3).

Liu, H., Hartmann, Y., and Schultz, T. (2021). CSL-SHARE: A multimodal wearable sensor-based human activity dataset. *Frontiers in Computer Science*, 3.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *Proc. icml*, 30(1):3.

Mandic, D. P. and Chambers, J. A. (2001). *Recurrent neural networks for prediction: learning algorithms, architectures and stability*. Wiley series on adaptive and learning systems for signal processing, communications, and control. John Wiley.

Martín, H., Bernardos, A. M., Iglesias, J., and Casar, J. R. (2013-04). Activity logging using lightweight classification techniques in mobile devices. *Personal Ubiquitous Comput.*, 17(4):675–695.

Mousavian, A., Anguelov, D., Flynn, J., and Kosecka, J. (2017). 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mukhopadhyay, S. C. (2015). Wearable sensors for human activity monitoring: A review. *IEEE Sensors Journal*, 15(3):1321–1330.

Munirathinam, S. (2020). *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, volume 117 of *Advances in computers*. Elsevier.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, Haifa, Israel. Omnipress.

Najafi, B., Aminian, K., Paraschiv-Ionescu, A., Loew, F., Büla, C., and Prince, R. (2003). Ambulatory system for human motion analysis using a kinematic sensor: Monitoring of daily physical activity in the elderly. *IEEE transactions on bio-medical engineering*, 50:711–23.

NVIDIA (2022a). ActionRecognitionNet Model Card.

NVIDIA (2022b). TAO toolkit 3.22.02 documentation.

Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv:1811.03378 [cs]*.

Ordóñez, F. and Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16:115.

Orr, G. B. (1996). *Dynamics and Algorithms for Stochastic Search*. phdthesis, Oregon Graduate Institute of Science & Technology.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv:1511.08458 [cs]*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). Understanding the exploding gradient problem. *ArXiv*, abs/1211.5063.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318. PMLR.

Patrizi, A., Pennestrì, E., and Valentini, P. P. (2016). Comparison between low-cost marker-less and high-end marker-based motion capture systems for the computer-aided assessment of working ergonomics. *Ergonomics*, 59(1):155–162.

Perrott, M. A., Pizzari, T., Cook, J., and McClelland, J. A. (2017). Comparison of lower limb and trunk kinematics between markerless and marker-based motion capture systems. *Gait & Posture*, 52:57–61.

Philipose, M., Fishkin, K., Perkowitz, M., Patterson, D., Fox, D., Kautz, H., and Hahnel, D. (2004). Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57.

Pérez de la Fuente, D. (2021). Análisis y clasificación de movimientos en actividades humanas a partir de vídeo 2d: adquisición de base de datos propia y comparativa de técnicas con aprendizaje profundo. Technical report, Universidad de Valladolid.

Refaeilzadeh, P., Tang, L., and Liu, H. (2009). Cross-validation. *Encyclopedia of database systems*, 5:532–538.

Rey, V. F., Hevesi, P., Kovalenko, O., and Lukowicz, P. (2019). Let there be IMU data: Generating training data for wearable, motion sensor based activity recognition from monocular RGB videos. In *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, UbiComp/ISWC '19 Adjunct, pages 699–708, London, United Kingdom. Association for Computing Machinery.

Roggen, D., Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T., Tröster, G., and Millán, J. d. R. (2013-11). The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042.

Roggen, D., Tröster, G., Lukowicz, P., Ferscha, A., del R. Millán, J., and Chavarriaga, R. (2013). Opportunistic human activity and context recognition. *Computer*, 46(2):36–45.

Ruder, S. (2017). An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Sangari, A. and Sethares, W. (2015). Convergence analysis of two loss functions in soft-max regression. *IEEE Transactions on Signal Processing*, 64:1–1.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. (2018). How does batch normalization help optimization? In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Shrestha, A. and Mahmood, A. (2019). Review of deep learning algorithms and architectures. *IEEE Access*, PP:1–1.

Soomro, K., Zamir, A. R., and Shah, M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402 [cs]*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception-v4, inception-ResNet and the impact of residual connections on learning. *arXiv:1602.07261 [cs]*.

Uchida, S. (2013). Image processing and recognition for biological images. *Development, Growth & Differentiation*, 55(4):523–549.

Vahdatpour, A., Amini, N., and Sarrafzadeh, M. (2011). On-body device localization for health and medical monitoring applications. In *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications*, PERCOM '11, pages 37–44. IEEE Computer Society.

Voulodimos, A., Doulamis, N., Doulamis, A., and Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. 2018:1–13.

Walker, D. J., Heslop, P. S., Plummer, C. J., Essex, T., and Chandler, S. (1997). A continuous patient activity monitor: validation and relation to disability. 18(1):49–59.

Wang, Q., Ma, Y., Zhao, K., and Tian, Y. (2022). A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2):187–212.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356.

Wu, G. and Xue, S. (2008). Portable preimpact fall detector with inertial sensors. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 16:178–83.

Yadav, S. K., Tiwari, K., Pandey, H. M., and Akbar, S. A. (2021). A review of multimodal human activity recognition with special emphasis on classification, applications, challenges and future directions. *Knowledge-Based Systems*, 223:106970.

Zagoruyko, S. and Komodakis, N. (2017). Wide residual networks. *arXiv:1605.07146 [cs]*.

Zhang, H.-B., Zhang, Y.-X., Zhong, B., Lei, Q., Yang, L., Du, J.-X., and Chen, D.-S. (2019). A comprehensive survey of vision-based human action recognition methods. *Sensors*, 19(5).

Zhao, L., Mammadov, M., and datewood, J. (2010). From convex to nonconvex: A loss function analysis for binary classification. In *2010 IEEE International Conference on Data Mining Workshops*, pages 1281–1288.

Zhou, Y., Wang, X., Zhang, M., Zhu, J., Zheng, R., and Wu, Q. (2019). MPCE: A maximum probability based cross entropy loss function for neural network classification. *IEEE Access*, PP:1–1.