Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Computación

Estudio del uso de redes convolucionales en el reconocimiento biométrico del usuario por su forma de andar

---

Alumno: D. Martín Manuel González Olivera

Tutores: Dña. María Aránzazu Simón Hurtado
D. Carlos Enrique Vivaracho Pascual

"La tecnología lo suficientemente avanzada es indistinguible de la magia."
-Arthur C. Clarke

Agradecimientos

La universidad de Valladolid tiene 784 años, la ciudad de Valladolid fue fundada hace 951 años, la raza humana puebla este planeta desde hace más de 200.000 años y el universo lleva en expansión desde hace más de 13.800 millones de años. Al lado de todas estas cifras, 21 años no parecen gran cosa, sin embargo, ha sido tiempo más que suficiente para crecer, aprender, equivocarme, volver a aprender y, sobre todo, conocer a gente extraordinaria que me ha permitido llegar hasta donde ahora estoy.

En primer lugar, me gustaría agradecer a Carlos y Arancha, mis tutores, por haberme dado la oportunidad de formar parte de la aventura que culmina con este trabajo y por haberme guiado hasta aquí. Aprovecho para agradecer también a todos los profesores que han aportado su conocimiento para formarme y llevarme a ser quien soy; muchas gracias.

A mi abuela, de la que heredo el carácter tenaz al que atribuyo gran parte de mi éxito. Por hacer de mi infancia una base sólida desde la que crecer sano y fuerte, por inspirarme y por enseñarme a creer en mí mismo, espero que estés orgullosa de mí; muchas gracias.

A mis padres y mi hermana que han sido mi más preciado bien durante estos 21 años. Por quererme, por criarme, por cuidarme y porque jamás podré devolver todo lo que han hecho y hacen día a día por mí; muchas gracias.

A Lucía, por llegar a mi vida cuando más la necesitaba y darme fuerzas para seguir adelante. Gracias por estar a mi lado y por dejarme estar al tuyo, gracias por animarme a seguir, apoyarme y dotarme de la confianza que me faltaba; muchas gracias, te amo.

A mis compañeros y amigos, a las personas que me ayudaron a tomar las decisiones correctas y a toda la gente que trabaja en la sombra todos los días para que yo pueda vivir mi vida; muchas gracias.

Resumen

El presente Trabajo de Fin de Grado tiene como principal objetivo explorar la viabilidad del uso de redes neuronales convolucionales (CNN por sus siglas en inglés) en la verificación biométrica basada en la forma de andar a partir de los datos inerciales obtenidos de los sensores de dispositivos ponibles de uso comercial. La verificación mediante el análisis de la marcha constituye una técnica no invasiva y de creciente interés dentro del campo de la biometría, especialmente en entornos en los que otras formas de autenticación pueden resultar intrusivas o inadecuadas.

Este estudio parte de una línea de trabajo previa desarrollada en el Departamento de Informática de la Universidad de Valladolid. Por tanto, se cuenta con un conjunto de datos reales recopilado empleando los sensores de dos dispositivos distintos. A partir de estos datos, se diseñó una línea de preprocesamiento que tiene como resultado un conjunto de imágenes RGB específicamente diseñadas para codificar espacialmente las series temporales.

El núcleo experimental del trabajo consiste en comparar el rendimiento de una serie de arquitecturas de red convolucional —tanto preexistentes como de creación propia— en una tarea de verificación binaria usuario genuino/impostor. Se han entrenado y comparado cinco arquitecturas: ResNet18, ResNet50, EfficientNet-B0, MobileNetV2 y una CNN de diseño propio; empleando la tasa de equierror (EER) como métrica principal de evaluación. El sistema ha sido implementado mediante PyTorch, haciendo uso de técnicas de transferencia de aprendizaje, y la experimentación se ha llevado a cabo en una combinación de entornos locales y máquinas virtuales proporcionadas por la universidad.

Los resultados obtenidos permiten estudiar la factibilidad de este tipo de redes en entornos interactivos y comprobar si pueden alcanzar tasas de error competitivas utilizando exclusivamente dispositivos comerciales.

Abstract

The main objective of this thesis is to explore the feasibility of using Convolutional Neural Networks (CNN) in gait-based biometric verification from inertial data obtained from commercial wearable devices. The identification of people by analyzing their gait is a non-invasive technique of growing interest within the field of biometrics, especially in environments where other forms of authentication may be intrusive or inappropriate.

This study is based on a previous line of work developed in the Department of Computer Science of the University of Valladolid, where a set of real data was collected using the sensors of two different devices. From this data, a preprocessing process was developed, resulting in a set of RGB images specifically designed to spatially encode the time series.

The experimental core of the work consists of comparing the performance of a number of convolutional network architectures - both pre-existing and self-created - in a binarial genuine user/impostor verification task. Five architectures have been trained and compared: ResNet18, ResNet50, EfficientNetB0, MobileNetV2, and a self-designed CNN; using the Equal Error Rate (EER) as the main evaluation metric. The system has been implemented using PyTorch, making use of learning transfer techniques, and the experimentation has been carried out in a combination of local environments and virtual machines provided by the university.

The results obtained allow us to study the feasibility of this type of networks in interactive environments and to check if they can achieve competitive error rates using exclusively commercial devices.

Índice general

Agradecimientos **III**

Resumen **V**

Abstract **VII**

Lista de figuras **XIII**

Lista de tablas **XV**

1. Introducción **1**

1.1. Contexto 1

1.2. Motivación 2

1.3. Objetivos 3

1.3.1. Objetivo General 3

1.3.2. Objetivos Específicos 3

1.4. Estructura de la memoria 4

2. Gestión del Proyecto **5**

2.1. Metodología empleada 5

2.2. Planificación 7

2.3. Riesgos 14

2.3.1. Análisis de Riesgos 15

2.4. Seguimiento Real 28

2.5. Estimación de costes 30
2.5.1. Presupuesto en un entorno profesional 31
2.5.2. Presupuesto Real 33

3. Contexto de la investigación **35**
3.1. Biometría: Fundamentos y Aplicaciones 36
3.2. Dispositivos ponibles en la biometría 38
3.3. Redes Neuronales Artificiales 40
3.4. Aprendizaje profundo en biometría 42
3.5. Transferencia de aprendizaje en biometría 43
3.6. Redes Neuronales en Biometría de la marcha 44
3.6.1. Síntesis de estudios relevantes 45

4. Tecnologías empleadas **47**
4.1. Herramientas de desarrollo y gestión 47
4.2. Bibliotecas y frameworks empleados 49
4.2.1. Bibliotecas auxiliares 49

5. Conjuntos de datos **51**
5.1. Fuente y Características de los Datos 51
5.2. Preprocesamiento de Datos y Generación de Imágenes 52
5.2.1. Eliminación de valores incorrectos 52
5.2.2. Ruido 53
5.2.3. Interpolación y normalización de datos 56
5.2.4. Segmentación en ventanas 56
5.2.5. Normalización de Datos 57
5.2.6. Generación de Imágenes 57
5.3. Formación de los conjuntos de entrenamiento y prueba 60
5.3.1. Construcción del D ataset 60
5.3.2. Selección de sesiones y muestras 61

6. Modelos **63**

6.1. Redes Residuales (ResNet) 67

6.2. EfficientNet 70

6.3. MobileNetV2 72

6.4. Arquitectura de Red Propuesta 74

6.4.1. Extracción de Características 75

6.4.2. Pooling Adaptativo 76

6.4.3. Clasificador 76

7. Pruebas **79**

7.1. Pruebas durante la generación de imágenes 79

7.2. Pruebas durante la evaluación de las arquitecturas 80

8. Resultados **83**

8.1. Diseño Experimental 83

8.2. Experimento 1: Estudio preliminar 85

8.2.1. Configuraciones evaluadas 85

8.2.2. Metodología 85

8.2.3. Resultados 86

8.2.4. Discusión 86

8.3. Experimento 2: Estudio comparativo 88

8.3.1. Criterios de selección 88

8.3.2. Resultados 88

8.3.3. Discusión 88

8.4. Experimento 3: Estudio de hiperparámetros 91

8.4.1. Discusión 91

8.5. Discusión Final y Comparación de Resultados 95

9. Conclusiones **97**

9.1. Líneas de trabajo futuras 98

Bibliografía **105**

Lista de Figuras

2.1. Diagrama de las fases del ciclo CRISP-DM original 6

2.2. Diagrama de las fases del ciclo CRISP-DM adaptado 8

2.3. Diagrama de Gantt de la etapa I 10

2.4. Diagrama de Gantt de la etapa II 11

2.5. Diagrama de Gantt de la etapa III 13

2.6. Diagrama de Gantt de la etapa IV 13

3.1. Identificación (1:N) vs. Verificación (1:1). Fuente: [3] 36

3.2. Dispositivos ponibles empleados en la investigación 39

3.3. Estructura básica de una ANN. Fuente: [49] 40

5.1. Pérdida de conexión al comienzo de la muestra. Fuente: [57] 54

5.2. Ejemplos de pérdidas intermedias de conexión 54

5.3. Bloqueo grave de la aplicación: señal corta y pérdida severa de datos. Fuente: [57] 55

5.4. Presencia de valores negativos. Fuente: [57] 55

5.5. Ejemplo de gráfico de líneas para la señal de acelerómetro. 58

5.6. Ejemplo de mapa de calor a color. 58

5.7. Ejemplo de mapa de calor en escala de grises. 59

5.8. Esquema gráfico de los criterios mono-session y cross-session. Elaboración propia 62

6.1. Jerarquía conceptual de la inteligencia artificial, el aprendizaje automático y el aprendizaje profundo. Fuente: [23] 63

6.2. Estructura típica de una red neuronal convolucional. Fuente: [56] 64

6.3. Estructura de un bloque residual básico. Fuente: [27] 67

6.4. Análisis comparativo de las variantes de ResNet (ResNet-18 frente a ResNet-50) (a) conjunto de datos Office-Home [76] (b) conjunto de datos PACS [46] 68

6.5. Esquema del escalado compuesto en EfficientNet. Fuente: [74] 70

6.6. Izquierda: Capa convolucional estándar. Derecha: convoluciones separables en profundidad con ambas capas *Depthwise* y *Pointwise* . Fuente: [30] 72

6.7. Estructura interna de un bloque *Inverted Residual* de MobileNetV2. Fuente: [71]. 73

6.8. Diagrama de la arquitectura de red propupesta 77

8.1. EER por arquitectura para cada combinación de sensor y tipo de imágenes 86

8.2. EER por usuario para cada red 89

8.3. EER promedio para cada configuración probada 91

8.4. Distribución del EER por número de capas ocultas 92

8.6. Distribución del EER por tamaño de lote 92

8.5. Distribución del EER por tamaño de la primera capa oculta 93

8.7. EER por usuario para la configuración final. 94

Lista de Tablas

2.1. Registro estimado de actividades - Etapa I: Inicio del proyecto 9
2.2. Registro estimado de actividades - Etapa II: Análisis y procesamiento de los datos 10
2.3. Registro estimado de actividades - Etapa III: Creación y evaluación de los modelos 12
2.4. Registro estimado de actividades - Etapa IV: Cierre del Proyecto 13
2.5. Tabla de posibilidad e impacto 14
2.6. Tabla de Evaluación de Riesgos 14
2.7. R-01 Falta de experiencia previa 15
2.8. R-02 Problemas de salud o estrés 16
2.9. R-03 Desmotivación 17
2.10. R-04 Falta de tiempo 18
2.11. R-05 Escasez de datos 18
2.12. R-06 Limitaciones del hardware 19
2.13. R-07 Problemas con las bibliotecas software 20
2.14. R-8 Pérdida de datos o código 21
2.15. R-9 Fallos en las plataformas de trabajo 22
2.16. R-10 Mala distribución del tiempo 23
2.17. R-11 Definición inadecuada de los objetivos 24
2.18. R-12 Falta de retroalimentación oportuna 25
2.19. R-13 Dificultad al interpretar resultados 26
2.20. R-14 Falta de documentación adecuada 27

2.21. Registro real de actividades - Etapa I: Inicio del proyecto 28

2.22. Registro real de actividades - Etapa II: Análisis y procesamiento de los datos 29

2.24. Registro real de actividades - Etapa IV: Cierre del Proyecto 29

2.25. Salarios brutos anuales por puesto 31

2.26. Costes estimados de hardware 32

2.27. Presupuesto total estimado 32

2.28. Comparación de costes entre entorno profesional y académico 33

2.23. Registro real de actividades - Etapa III: Creación y evaluación de los modelos (las horas en cursiva corresponden a actividades que se realizan en paralelo) 34

3.1. Estudios recientes sobre biometría de marcha utilizando redes neuronales 45

8.1. Resultados preliminares: EER por red, sensor y tipo de representación visual. En negrita se indican los mejores resultados. 86

8.2. Evaluación de EER por usuario. Los valores faltantes corresponden a usuarios con una sola muestra para la primera sesión. 90

8.3. Configuraciones óptimas según EER medio. 91

Capítulo 1

Introducción

1.1. Contexto

El reconocimiento biométrico es una disciplina cuyo interés se remonta siglos atrás. El uso de la forma de andar para tareas de esta naturaleza ha cobrado un creciente interés en los últimos años debido a su potencial en numerosas disciplinas. A diferencia de otros métodos biométricos tradicionales como las huellas dactilares o el reconocimiento facial, permite la autenticación de manera no intrusiva y continua, lo que lo hace atractivo para aplicaciones en dispositivos ponibles (*wearable*).

Tradicionalmente, este tipo de verificación ha sido abordado mediante técnicas basadas en modelos estadísticos, redes neuronales recurrentes (RNNs), arquitecturas LSTM, *transformers* o incluso sistemas basados en lógica difusa. Sin embargo, el uso de redes neuronales convolucionales (CNNs) en este contexto sigue siendo un área poco explorada.

En este trabajo se estudia la viabilidad de aplicar redes neuronales convolucionales (CNNs) para la verificación biométrica a partir de señales inerciales, obtenidas mediante los acelerómetros y giroscopios integrados en relojes inteligentes comerciales. Los datos empleados provienen de un conjunto recopilado en la Universidad de Valladolid (UVa), lo que garantiza su realismo y aplicabilidad práctica.

Si bien el enfoque principal de esta investigación es la biometría, los resultados obtenidos podrían tener implicaciones en otros campos, como el análisis de patrones de marcha en pacientes con trastornos motores o la optimización de sistemas de monitorización en el ámbito deportivo y de la salud.

1.2. Motivación

El uso de la forma de andar como rasgo biométrico ofrece múltiples aplicaciones y ventajas que lo diferencian de otros más clásicos. La marcha permite una autenticación continua y no intrusiva, especialmente relevante en el contexto de los dispositivos ponibles.

La motivación principal de este trabajo es explorar el potencial de las CNNs en este ámbito, dado su éxito en la extracción de características espaciales en otras áreas. Al aplicar estas técnicas a señales de acelerómetros y giroscopios integrados en dispositivos comerciales, se busca desarrollar un sistema de verificación biométrica de la marcha que sea eficiente, preciso y fácilmente integrable en soluciones existentes.

1.3. Objetivos

Para garantizar que la investigación sea clara, alcanzable y medible, se definen los objetivos siguiendo la metodología *SMART* [16], que propone describir objetivos Específicos, Medibles, Alcanzables, Relevantes y con un tiempo determinado. Dicho esto, se presenta un objetivo troncal en torno al cual estructurar el trabajo, y una serie de objetivos específicos a modo de hitos que permiten monitorizar el progreso del proyecto.

1.3.1. Objetivo General

A. Explorar y evaluar la viabilidad de las redes neuronales convolucionales (CNNs) para la verificación biométrica mediante el análisis de la marcha a partir de señales inerciales registradas con dispositivos ponibles comerciales.

1.3.2. Objetivos Específicos

B1. Revisión del estado del arte, análisis de estudios previos sobre verificación biométrica basada en la marcha, identificando enfoques existentes y posibles vías de investigación.

B2. Selección y preprocesamiento de datos, empleando el conjunto UVa, aplicando técnicas de limpieza, normalización y segmentación.

B3. Creación de imágenes a partir de los datos preprocesados para emplear como entrada a los diferentes modelos. Búsqueda del tipo de imagen más adecuado para el problema presentado.

B4. Diseño y entrenamiento de modelos CNN. Implementación y entrenamiento de distintas arquitecturas de redes neuronales convolucionales, tanto preexistentes como de propia creación.

B5. Evaluación del rendimiento, precisión, sensibilidad, especificidad y demás métricas relevantes para comparar la efectividad de las CNNs entre sí y con otras arquitecturas ya propuestas.

B6. Evaluación de la aplicabilidad del sistema en relojes inteligentes comerciales, considerando su rendimiento en entornos reales.

B7. Discusión y conclusiones. Extracción de aprendizajes clave, identificación de limitaciones del estudio y propuesta de futuras líneas de investigación.

Este conjunto de objetivos permite estructurar el estudio de manera rigurosa y alineada con el propósito del TFG, asegurando resultados que puedan ser comparados con investigaciones previas y aplicados a contextos reales.

1.4. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 2. Gestión del Proyecto: Describe la metodología escogida para el desarrollo del trabajo. Incluye también el proceso de planificación de calendario, presupuesto y riesgos. También se relata el seguimiento de la ejecución real del proyecto.

Capítulo 3. Contexto de la Investigación: Presenta una visión del estado del arte desde los conceptos más generales empleados en el trabajo hasta el problema concreto que se aborda.

Capítulo 4. Tecnologías empleadas: Lista las tecnologías que se han utilizado a lo largo de la realización del proyecto, así como el porqué de su elección.

Capítulo 5. Conjuntos de datos: Explica el origen y tratamiento de los datos, además de su organización para formar los conjuntos de datos empleados como entrada de los clasificadores.

Capítulo 6. Modelos: Enuncia y describe detalladamente los modelos escogidos para su comparación, incluyendo su funcionamiento y el porqué de su elección.

Capítulo 7. Pruebas: Se exponen las diferentes comprobaciones que se han realizado a lo largo de la implementación del sistema para asegurar la corrección y eficacia del código.

Capítulo 8. Resultados: Presenta de manera objetiva los resultados obtenidos, así como la metodología experimental empleada para su obtención.

Capítulo 9. Conclusiones: Incluye las conclusiones extraídas de los datos presentados y las posibles futuras vías de investigación.

Capítulo 2

Gestión del Proyecto

Todo proyecto de cierta envergadura requiere una planificación estructurada y una gestión eficiente de recursos y tareas para garantizar su éxito. Para ello, en este trabajo se definen una serie de estrategias que permiten organizarlo, minimizar riesgos y asegurar el cumplimiento de los objetivos en el tiempo establecido.

2.1. Metodología empleada

Para el desarrollo de este trabajo, se ha empleado la metodología CRISP-DM [9] (Cross-Industry Standard Process for Data Mining), ampliamente utilizada en proyectos de minería de datos y aprendizaje automático [70]. CRISP-DM proporciona un marco estructurado y flexible que guía el ciclo de vida de un proyecto basado en datos como lo es este, asegurando una ejecución organizada y enfocada en la obtención de resultados fiables.

Esta metodología se compone habitualmente de seis fases interconectadas, que pueden desarrollarse de manera iterativa según las necesidades del proyecto. Dichas fases son:

1. **Comprensión del negocio** (*Business Understanding*): Se define el problema de negocio, los objetivos del proyecto y se desarrolla un plan inicial.
2. **Comprensión de los datos** (*Data Understanding*): Se recopilan, exploran y analizan los datos disponibles para evaluar su calidad y relevancia.
3. **Preparación de los datos** (*Data Preparation*): Se limpian, transforman y seleccionan los datos necesarios para la modelización.
4. **Modelado** (*Modeling*): Se eligen y entrenan modelos de minería de datos, ajustando sus parámetros para optimizar su rendimiento.
5. **Evaluación** (*Evaluation*): Se analizan los modelos generados para determinar su idoneidad y alineación con los objetivos del negocio.
6. **Despliegue** (*Deployment*): Se implementa la solución en un entorno real y se documentan los resultados para su uso futuro.

En la Figura 2.1 se ilustra el flujo de trabajo clásico de un proyecto CRISP-DM.

Figura 2.1: Diagrama de las fases del ciclo CRISP-DM original

2.2. Planificación

Para la realización de este estudio, se ha decidido adoptar una vertiente personalmente adaptada de la metodología que defina de manera más precisa el proceso que se seguirá. Las fases resultantes serán las siguientes:

1. **Definición del problema y objetivos** (*Problem Understanding*): Se identifica la necesidad de un sistema de verificación biométrica basado en la marcha, analizando su viabilidad y buscando cubrir huecos en el estado del arte. Se establecen los objetivos específicos, los riesgos y las limitaciones. Dada la naturaleza del proyecto se abandona la terminología "negocio".

2. **Análisis de los datos** (*Data Understanding*): Se examinan los datos de acelerómetros y giroscopios recopilados mediante relojes inteligentes comerciales (Microsoft y Motorola). Esta fase incluye la exploración inicial de los datos, y la detección de problemas como ruido, datos faltantes o variabilidad entre usuarios. Coincide con la fase de la metodología original.

3. **Preprocesamiento y transformación de datos** (*Data Preprocessing*): Se aplican técnicas de limpieza de datos, normalización y segmentación de las señales inerciales en ventanas temporales. También se consideran diversos métodos para representar dichas ventanas a modo de imágenes (dichos métodos se presentarán más adelante). Una fase más refinada y precisa que la original.

4. **Diseño y entrenamiento del modelo** (*Model selection and training*): Se implementan distintas arquitecturas de CNNs y se entrenan utilizando los datos preprocesados. En esta fase se prueban diferentes hiperparámetros y configuraciones para optimizar el rendimiento del modelo en la verificación biométrica. Similar a la fase de modelado.

5. **Evaluación del modelo y comparación con otros enfoques** (*Model Evaluation*): Se analizan métricas como tasa de aciertos, área bajo la curva (AUC) y tasa de equierror (EER) para determinar la efectividad del modelo y permitir su comparación con los existentes.

Como se observa en la Figura 2.2, en el caso de la metodología propuesta tras evaluar un modelo, no se vuelve necesariamente al primer paso. Lo más frecuente es definir un nuevo modelo o procesar los datos otra vez para generar otro conjunto de imágenes. De esta manera se consigue un proceso de desarrollo mucho más ágil y centrado en la experimentación.

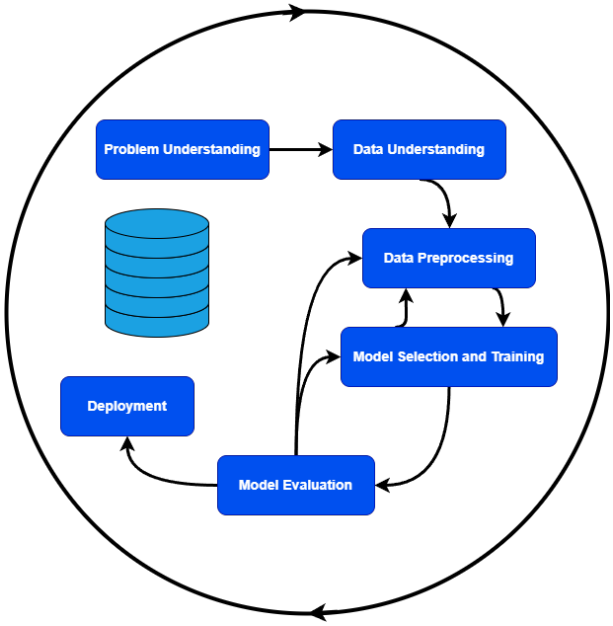


Figura 2.2: Diagrama de las fases del ciclo CRISP-DM adaptado

No obstante, para garantizar dicha agilidad, es necesario establecer un método de experimentación estándar para poder comparar de manera cuantitativa cada modelo y cada conjunto de imágenes.

La fase de experimentación constará, por tanto, de los siguientes pasos:

1. **Selección del Modelo:** Puede consistir en la elaboración teórica de un modelo propio o en la búsqueda de modelos preexistentes reconocidos por la comunidad. En ambos casos se realiza un estudio sobre las ventajas del modelo y las razones de su selección, generando la documentación pertinente.

2. **Implementación:** Se elabora una clase *python* que se pueda emplear mediante una interfaz genérica propuesta diseñada para facilitar las pruebas y que se expondrá más adelante.

3. **Pruebas y Ajustes:** Se somete el modelo a una batería de pruebas establecida. Durante este proceso se realizan varias iteraciones alterando diversos parámetros para observar la variabilidad. Las pruebas incluyen diversos conjuntos de imágenes.

4. **Obtención de resultados:** Finalmente, se tabulan los resultados obtenidos y se almacenan para su posterior comparación.

Como resultado de estas iteraciones de selección, entrenamiento y obtención de resultados, se generan una serie de documentos que se presentan a los tutores para favorecer el seguimiento del proyecto. Estos entregables marcan un conjunto de hitos que, junto con las fases previas de estudio del estado del arte, preprocesamiento de datos y elaboración de los programas de experimentación, permiten estructurar el desarrollo del trabajo y facilitar su distribución temporal.

Una vez finalizada la fase de experimentación y obtención de resultados, se procede a la último paso que correspondería con el despliegue. Esta parte, consistirá en la elaboración de la presente memoria y en la defensa del proyecto ante el tribunal.

Ya establecidas las diferentes fases por las que puede pasar el proyecto y cómo se relacionan entre ellas, se distribuye el trabajo en diversas etapas que agrupan dichas fases para facilitar la distribución temporal y el seguimiento de las tareas. Para cada una de estas etapas se elaboran dos registros con las actividades que la componen. De cada actividad se incluye un título, fechas de inicio y final estimadas y reales, y duración estimada y real en horas, respectivamente, en cada registro. Los registros correspondientes a las fechas y duraciones reales se presentan en la Sección 2.4

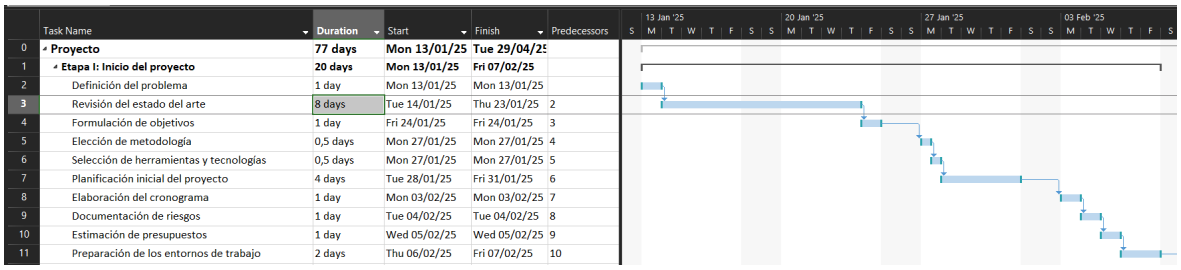
Estableceremos primero las fechas y duraciones estimadas para la etapa de inicio con su correspondiente diagrama de Gantt. La Tabla 2.1 reúne dichas fechas y horarios.

Actividad	**Inicio estimado**	**Fin estimado**	**Duración estimada (h)**
Definición del problema	Día 1, Semana 1	Día 1, Semana 1	3
Revisión del estado del arte	Día 2, Semana 1	Día 4, Semana 2	20
Formulación de objetivos	Día 5, Semana 2	Día 5, Semana 2	5
Elección de metodología	Día 1, Semana 3	Día 1, Semana 3	2
Selección de herramientas y tecnologías	Día 1, Semana 3	Día 1, Semana 3	3
Planificación inicial del proyecto	Día 2, Semana 3	Día 5, Semana 3	10
Elaboración del cronograma	Día 1, Semana 4	Día 1, Semana 4	6
Documentación de riesgos	Día 2, Semana 4	Día 2, Semana 4	6
Estimación de presupuestos	Día 3, Semana 4	Día 3, Semana 4	5
Preparación de los entornos de trabajo	Día 4, Semana 4	Día 5, Semana 4	8

Tabla 2.1: Registro estimado de actividades - Etapa I: Inicio del proyecto

Esta etapa tendrá una duración total de unas 68 horas que se distribuirán a lo largo de las cuatro primeras semanas de trabajo, teniendo como fecha de comienzo el día 13 de enero de 2025. Como se observa en la Figura 2.3, en la que se representa el diagrama de Gantt de la Etapa I, dada la dependencia directa entre las diversas tareas y la naturaleza de las mismas, se propone adoptar una metodología en cascada.

Además, esta primera etapa permite establecer la dinámica general que se seguirá a lo largo del proyecto. Dado que el alumno compagina la investigación con la asistencia a clases y otras actividades, y la realización de numerosos proyectos de carácter tanto profesional como privado, se propone extender la duración del trabajo en el tiempo, dedicando períodos de unas 3 - 5 horas diarias que permitan dejar espacio para otras obligaciones. Asimismo, se propone una carga de trabajo de cinco días semanales, si bien los días en los que no se trabaja en este proyecto concreto no tienen por qué coincidir siempre con el fin de semana.



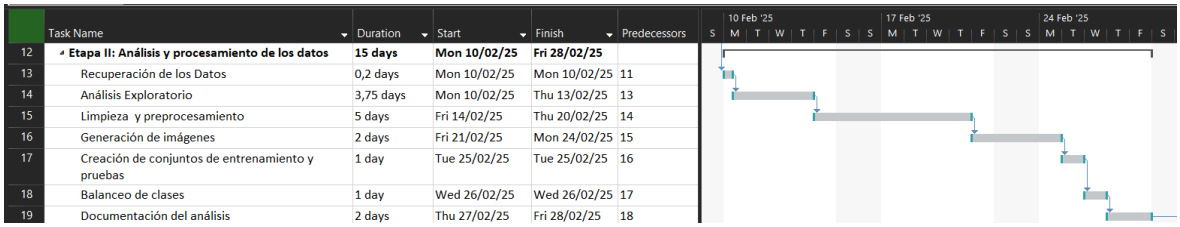
	Task Name	Duration	Start	Finish	Predecessors
0	**Proyecto**	**77 days**	**Mon 13/01/25**	**Tue 29/04/25**	
1	**Etapa I: Inicio del proyecto**	**20 days**	**Mon 13/01/25**	**Fri 07/02/25**	
2	Definición del problema	1 day	Mon 13/01/25	Mon 13/01/25	
3	Revisión del estado del arte	8 days	Tue 14/01/25	Thu 23/01/25	2
4	Formulación de objetivos	1 day	Fri 24/01/25	Fri 24/01/25	3
5	Elección de metodología	0,5 days	Mon 27/01/25	Mon 27/01/25	4
6	Selección de herramientas y tecnologías	0,5 days	Mon 27/01/25	Mon 27/01/25	5
7	Planificación inicial del proyecto	4 days	Tue 28/01/25	Fri 31/01/25	6
8	Elaboración del cronograma	1 day	Mon 03/02/25	Mon 03/02/25	7
9	Documentación de riesgos	1 day	Tue 04/02/25	Tue 04/02/25	8
10	Estimación de presupuestos	1 day	Wed 05/02/25	Wed 05/02/25	9
11	Preparación de los entornos de trabajo	2 days	Thu 06/02/25	Fri 07/02/25	10

Figura 2.3: Diagrama de Gantt de la etapa I

Seguidamente se presentan en la Tabla 2.2 las fechas y duraciones aproximadas de la segunda etapa del proyecto, correspondiente con el análisis y procesamiento de los datos.

Actividad	**Inicio estimado**	**Fin estimado**	**Duración estimada (h)**
Recuperación de los datos	Día 1, Semana 5	Día 1, Semana 5	2
Análisis exploratorio (EDA)	Día 1, Semana 5	Día 4, Semana 5	15
Limpieza y preprocesamiento	Día 5, Semana 5	Día 4, Semana 6	20
Generación del conjunto de imágenes	Día 5, Semana 6	Día 1, Semana 7	10
Creación de conjuntos de entrenamiento y prueba	Día 2, Semana 7	Día 2, Semana 7	5
Balanceo de clases	Día 3, Semana 7	Día 3, Semana 8	3
Documentación del análisis	Día 4, Semana 7	Día 5, Semana 7	5

Tabla 2.2: Registro estimado de actividades - Etapa II: Análisis y procesamiento de los datos



	Task Name	Duration	Start	Finish	Predecessors
12	**Etapa II: Análisis y procesamiento de los datos**	**15 days**	**Mon 10/02/25**	**Fri 28/02/25**	
13	Recuperación de los Datos	0,2 days	Mon 10/02/25	Mon 10/02/25	11
14	Análisis Exploratorio	3,75 days	Mon 10/02/25	Thu 13/02/25	13
15	Limpieza y preprocesamiento	5 days	Fri 14/02/25	Thu 20/02/25	14
16	Generación de imágenes	2 days	Fri 21/02/25	Mon 24/02/25	15
17	Creación de conjuntos de entrenamiento y pruebas	1 day	Tue 25/02/25	Tue 25/02/25	16
18	Balanceo de clases	1 day	Wed 26/02/25	Wed 26/02/25	17
19	Documentación del análisis	2 days	Thu 27/02/25	Fri 28/02/25	18

Figura 2.4: Diagrama de Gantt de la etapa II

La segunda etapa (Figura 2.4) consta de unas 60 horas de trabajo que se distribuyen a lo largo de 3 semanas aproximadamente, coincidiendo el final de esta etapa con el final del mes de febrero. Cabe destacar que se prevé que la tarea de generación del conjunto de datos pueda tener que repetirse en varias ocasiones en etapas futuras del proyecto.

Una vez se haya conseguido crear un conjunto de datos acorde con lo esperado, se procede con la siguiente etapa, que consiste en un conjunto de iteraciones; durante cada una se realizarán las mismas tareas pero para cada uno de los cinco modelos de red convolucional diferentes que se proponen (Figura 2.6).

Como se puede observar en la Tabla 2.3, la implementación y la obtención de resultados para el primer modelo se estima que llevará una cantidad de tiempo mucho superior al resto de modelos pre-entrenados. Esto se debe a que se pretende reutilizar una gran parte del código empleado. Además, para mejor comprensión del diagrama de Gantt resultante, se propone que la tarea de ajustar parámetros dure lo mismo que la de obtener resultados. Lo que se quiere representar es que a lo largo de esas horas se llevan a cabo una serie de iteraciones durante las que se varían diversos hiperparámetros del modelo y se realiza un *fine tunning* del mismo para dictaminar la mejor configuración posible.

La cuarta y última etapa del trabajo se corresponde con el cierre del proyecto. En esta parte se comparan todos los resultados, se extraen conclusiones y se crea la documentación pertinente, incluida la presente memoria. Para finalizar el proyecto, se elabora el material necesario para la presentación ante el tribunal y se lleva a cabo la misma.

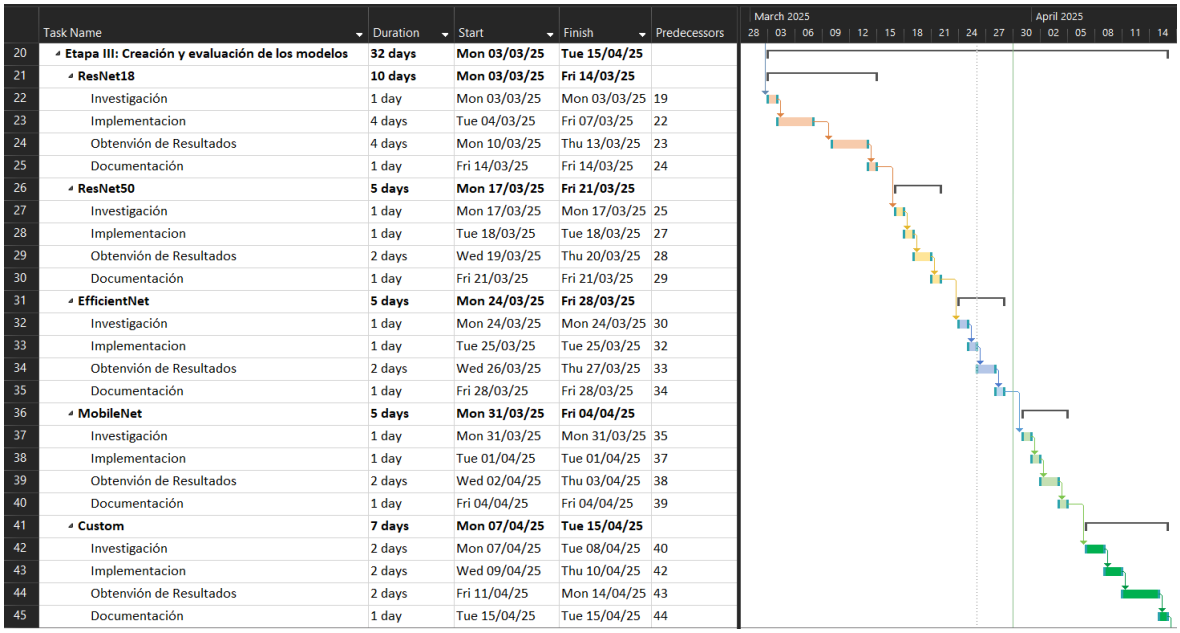
Como se ve en la Tabla 2.4, se estima que la etapa final ocupe un total de 35 horas durante las dos últimas semanas del proyecto. Finalizará en torno al día 29 de abril de 2025.

Finalmente, se incluye en la Figura 2.6 el fragmento de diagrama que corresponde a la última etapa del proyecto.

Esta estimación supone un total de 77 días laborales, en torno a unas 300 horas distribuidas entre enero y abril del año en curso, como es adecuado para un TFG al uso. Se espera que la realización pueda reducirse en ciertas tareas gracias a la reutilización de código y a la automatización de procesos. Sin embargo, al compaginar el trabajo con otras tareas y tras posibles imprevistos, es probable que se reduzca en algunos casos el tiempo diario dedicado, suponiendo un retraso en las fechas.

Actividad	Inicio estimado	Fin estimado	Duración estimada (h)
ResNet18	Día 1, Semana 8	Día 5, Semana 9	**48**
Investigación	Día 1, Semana 8	Día 1, Semana 8	4
Implementación	Día 2, Semana 8	Día 5, Semana 8	20
Obtención de resultados	Día 1, Semana 9	Día 4, Semana 9	20
Ajuste de parámetros	Día 1, Semana 9	Día 4, Semana 9	*20*
Documentación	Día 5, Semana 9	Día 5, Semana 9	4
ResNet50	Día 1, Semana 10	Día 5, Semana 10	**20**
Investigación	Día 1, Semana 10	Día 1, Semana 10	3
Implementación	Día 2, Semana 10	Día 2, Semana 10	3
Obtención de resultados	Día 3, Semana 10	Día 4, Semana 10	10
Ajuste de parámetros	Día 3, Semana 10	Día 4, Semana 10	*10*
Documentación	Día 5, Semana 10	Día 5, Semana 10	4
EfficientNet	Día 1, Semana 11	Día 5, Semana 11	**20**
Investigación	Día 1, Semana 11	Día 1, Semana 11	3
Implementación	Día 2, Semana 11	Día 2, Semana 11	3
Obtención de resultados	Día 3, Semana 11	Día 4, Semana 11	10
Ajuste de parámetros	Día 3, Semana 11	Día 4, Semana 11	*10*
Documentación	Día 5, Semana 11	Día 5, Semana 11	4
MobileNet	Día 1, Semana 12	Día 5, Semana 12	**20**
Investigación	Día 1, Semana 12	Día 1, Semana 12	3
Implementación	Día 2, Semana 12	Día 2, Semana 12	3
Obtención de resultados	Día 3, Semana 12	Día 4, Semana 12	10
Ajuste de parámetros	Día 3, Semana 12	Día 4, Semana 12	*10*
Documentación	Día 5, Semana 12	Día 5, Semana 12	4
Custom	Día 1, Semana 13	Día 2, Semana 14	**30**
Investigación	Día 1, Semana 13	Día 2, Semana 13	8
Implementación	Día 3, Semana 13	Día 4, Semana 13	8
Obtención de resultados	Día 5, Semana 13	Día 1, Semana 14	10
Ajuste de parámetros	Día 5, Semana 13	Día 1, Semana 14	*10*
Documentación	Día 2, Semana 14	Día 2, Semana 14	4

Tabla 2.3: Registro estimado de actividades - Etapa III: Creación y evaluación de los modelos

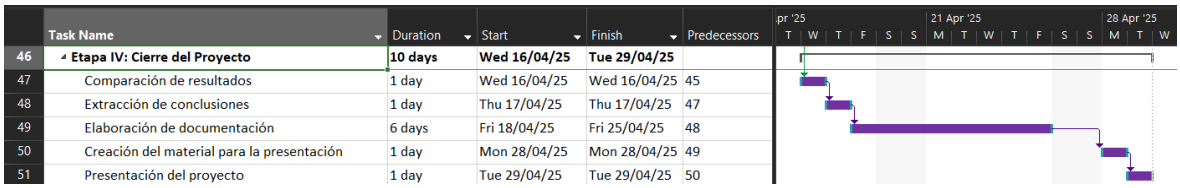


	Task Name	Duration	Start	Finish	Predecessors
20	**Etapa III: Creación y evaluación de los modelos**	**32 days**	**Mon 03/03/25**	**Tue 15/04/25**	
21	**ResNet18**	**10 days**	**Mon 03/03/25**	**Fri 14/03/25**	
22	Investigación	1 day	Mon 03/03/25	Mon 03/03/25	19
23	Implementacion	4 days	Tue 04/03/25	Fri 07/03/25	22
24	Obtenvión de Resultados	4 days	Mon 10/03/25	Thu 13/03/25	23
25	Documentación	1 day	Fri 14/03/25	Fri 14/03/25	24
26	**ResNet50**	**5 days**	**Mon 17/03/25**	**Fri 21/03/25**	
27	Investigación	1 day	Mon 17/03/25	Mon 17/03/25	25
28	Implementacion	1 day	Tue 18/03/25	Tue 18/03/25	27
29	Obtenvión de Resultados	2 days	Wed 19/03/25	Thu 20/03/25	28
30	Documentación	1 day	Fri 21/03/25	Fri 21/03/25	29
31	**EfficientNet**	**5 days**	**Mon 24/03/25**	**Fri 28/03/25**	
32	Investigación	1 day	Mon 24/03/25	Mon 24/03/25	30
33	Implementacion	1 day	Tue 25/03/25	Tue 25/03/25	32
34	Obtenvión de Resultados	2 days	Wed 26/03/25	Thu 27/03/25	33
35	Documentación	1 day	Fri 28/03/25	Fri 28/03/25	34
36	**MobileNet**	**5 days**	**Mon 31/03/25**	**Fri 04/04/25**	
37	Investigación	1 day	Mon 31/03/25	Mon 31/03/25	35
38	Implementacion	1 day	Tue 01/04/25	Tue 01/04/25	37
39	Obtenvión de Resultados	2 days	Wed 02/04/25	Thu 03/04/25	38
40	Documentación	1 day	Fri 04/04/25	Fri 04/04/25	39
41	**Custom**	**7 days**	**Mon 07/04/25**	**Tue 15/04/25**	
42	Investigación	2 days	Mon 07/04/25	Tue 08/04/25	40
43	Implementacion	2 days	Wed 09/04/25	Thu 10/04/25	42
44	Obtenvión de Resultados	2 days	Fri 11/04/25	Mon 14/04/25	43
45	Documentación	1 day	Tue 15/04/25	Tue 15/04/25	44

Figura 2.5: Diagrama de Gantt de la etapa III

Actividad	**Inicio estimado**	**Fin estimado**	**Duración estimada (h)**
Comparación de resultados	Día 3, Semana 14	Día 3, Semana 14	5
Extracción de conclusiones	Día 4, Semana 14	Día 4, Semana 14	2
Elaboración de documentación final	Día 5, Semana 15	Día 5, Semana 15	10
Creación del material para la presentación	Día 1, Semana 16	Día 1, Semana 16	5
Presentación del proyecto	Semana asignada la presentación del TFG	Semana asignada la presentación del TFG	2

Tabla 2.4: Registro estimado de actividades - Etapa IV: Cierre del Proyecto



	Task Name	Duration	Start	Finish	Predecessors
46	**Etapa IV: Cierre del Proyecto**	**10 days**	**Wed 16/04/25**	**Tue 29/04/25**	
47	Comparación de resultados	1 day	Wed 16/04/25	Wed 16/04/25	45
48	Extracción de conclusiones	1 day	Thu 17/04/25	Thu 17/04/25	47
49	Elaboración de documentación	6 days	Fri 18/04/25	Fri 25/04/25	48
50	Creación del material para la presentación	1 day	Mon 28/04/25	Mon 28/04/25	49
51	Presentación del proyecto	1 day	Tue 29/04/25	Tue 29/04/25	50

Figura 2.6: Diagrama de Gantt de la etapa IV

2.3. Riesgos

En el ámbito de la gestión de proyectos, el *Project Management Institute* (PMI) [34] define el riesgo como un evento o condición incierta (amenaza) que, de ocurrir, puede tener un efecto positivo o negativo en uno o más de los objetivos del proyecto, tales como el alcance, el cronograma, el coste o la calidad.

El nivel de riesgo es una estimación de lo que puede ocurrir y se valora, de forma cuantitativa, como el producto del impacto asociado a una amenaza, por la posibilidad de que esta se materialice. La Tabla 2.5 presenta tres valores indicativos para evaluar las posibilidades y los impactos de las posibles amenazas al desarrollo del proyecto.

Calificación (1-3)	**Posibilidad**
1	La amenaza se materializa a lo sumo una vez durante el proyecto
2	La amenaza se materializa a lo sumo una vez cada mes durante el proyecto
3	La amenaza se materializa a lo sumo una vez cada semana durante el proyecto
Calificación (1-3)	**Impacto**
1	El daño derivado de la materialización de la amenaza no tiene consecuencias relevantes.
2	El daño derivado de la materialización de la amenaza tiene consecuencias reseñables.
3	El daño derivado de la materialización de la amenaza tiene consecuencias graves reseñables.

Tabla 2.5: Tabla de posibilidad e impacto

Al multiplicar estas calificaciones, se obtienen los valores numéricos correspondientes a los siguientes riesgos. Estos valores se pueden organizar según su prioridad, como se muestra en la Tabla 2.6

	POSIBILIDAD		
IMPACTO	**1**	**2**	**3**
3	**3**	**6**	**9**
2	2	4	6
1	1	2	3

Tabla 2.6: Tabla de Evaluación de Riesgos

2.3.1. Análisis de Riesgos

Para cada riesgo identificado. se elabora una tabla a modo de registro que incluya la siguiente información:

1. **Identificador**: Identifica el riesgo de forma para poder referenciarlo.
2. **Nombre**
3. **Clase**: Personal, tecnológico o de gestión según su naturaleza.
4. **Descripción**: Breve explicación del riesgo.
5. **Plan de mitigación**: Medida/s para prevenir el riesgo.
6. **Plan de contingencia**: Medida/s para limitar el impacto en caso de que el riesgo se haya materializado.
7. **Posibilidad e impacto**: Valores de riesgo antes y después de aplicar el plan de mitigación.

A continuación se presentan las tablas correspondientes para cada riesgo.

Identificador	R-01			
Nombre	Falta de experiencia previa			
Clase	Personal			
Descripción	El investigador tiene limitada experiencia previa en redes neuronales convolucionales y, sobre todo, en procesamiento de datos biométricos, lo que puede ralentizar el desarrollo del proyecto.			
Plan de mitigación	Realizar un estudio previo del estado del arte, consultar documentación y buenas prácticas de uso de las bibliotecas y buscar apoyo de los tutores o comunidad científica.			
Plan de contingencia	Redefinir los objetivos del proyecto para simplificar el modelo si el aprendizaje resulta complicado y buscar modelos preentrenados para evitar partir desde cero.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	3	3	Crítico (9)
	Postmitigación	2	2	Medio (4)

Tabla 2.7: R-01 Falta de experiencia previa

Identificador	R-02			
Nombre	Problemas de salud o estrés			
Clase	Personal			
Descripción	La carga de trabajo del TFG, combinada con otras responsabilidades académicas o personales, puede generar estrés, ansiedad o problemas de salud que afecten el rendimiento y la productividad.			
Plan de mitigación	Implementar una planificación adecuada del tiempo, establecer descansos regulares, realizar actividad física y mantener hábitos saludables de sueño y alimentación. También se recomienda evitar sobrecarga de trabajo y buscar apoyo emocional si es necesario.			
Plan de contingencia	En caso de experimentar un impacto significativo en la salud, priorizar el bienestar, ajustar el cronograma del proyecto y, si es necesario, solicitar una extensión de plazo o reestructurar objetivos.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	2	2	Medio (4)
	Postmitigación	1	2	Bajo (2)

Tabla 2.8: R-02 Problemas de salud o estrés

<table>
<tr><td>Identificador</td><td colspan="4">R-03</td></tr>
<tr><td>Nombre</td><td colspan="4">Desmotivación</td></tr>
<tr><td>Clase</td><td colspan="4">Personal</td></tr>
<tr><td>Descripción</td><td colspan="4">La falta de avances significativos, dificultades técnicas o la sobrecarga de trabajo pueden generar pérdida de interés y motivación en el desarrollo del TFG, afectando el rendimiento y la calidad del trabajo.</td></tr>
<tr><td>Plan de mitigación</td><td colspan="4">Dividir el proyecto en objetivos pequeños y alcanzables para mantener la sensación de progreso. Buscar apoyo en los tutores y compañeros, así como recordar la relevancia y aplicaciones del trabajo para mantener la motivación.</td></tr>
<tr><td>Plan de contingencia</td><td colspan="4">Si la desmotivación afecta significativamente al rendimiento, reajustar el enfoque del trabajo para hacerlo más estimulante o cambiar la estrategia de desarrollo. También se puede establecer un sistema de recompensas por hitos alcanzados.</td></tr>
<tr><td rowspan="3">Evaluación de Riesgo</td><td></td><td>Posibilidad</td><td>Impacto</td><td>Riesgo Total</td></tr>
<tr><td>Premitigación</td><td>1</td><td>3</td><td>Medio (3)</td></tr>
<tr><td>Postmitigación</td><td>1</td><td>1</td><td>Muy Bajo (1)</td></tr>
</table>

Tabla 2.9: R-03 Desmotivación

Identificador	R-04			
Nombre	Falta de tiempo			
Clase	Personal			
Descripción	La carga académica, responsabilidades personales u otros compromisos pueden reducir el tiempo disponible para dedicar al TFG, retrasando su avance y afectando su calidad.			
Plan de mitigación	Elaborar un cronograma realista con hitos claros y fechas de entrega. Priorizar tareas esenciales, evitar la procrastinación y reservar bloques de tiempo específicos para trabajar en el TFG.			
Plan de contingencia	Si el tiempo se vuelve insuficiente, se puede simplificar el alcance del proyecto, optimizar el tiempo eliminando tareas no críticas o solicitar una extensión de la fecha de entrega si es posible.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	2	3	Alto (6)
	Postmitigación	1	3	Medio (3)

Tabla 2.10: R-04 Falta de tiempo

Identificador	R-05			
Nombre	Escasez de datos			
Clase	Tecnológico			
Descripción	Un conjunto de datos insuficiente puede afectar la capacidad del modelo para generalizar correctamente, lo que impacta en su precisión y fiabilidad.			
Plan de mitigación	Aumentar el tamaño del conjunto de datos recopilando más muestras si es posible o utilizar técnicas de aumento de datos, como el agregado de ruido sintético.			
Plan de contingencia	Si no es posible obtener más datos, considerar transfer learning con modelos preentrenados o modificar la arquitectura del modelo para que requiera menos datos de entrenamiento.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	2	2	Medio (4)
	Postmitigación	1	1	Muy Bajo (1)

Tabla 2.11: R-05 Escasez de datos

<table>
<tr><td>Identificador</td><td colspan="4">R-06</td></tr>
<tr><td>Nombre</td><td colspan="4">Limitaciones del hardware</td></tr>
<tr><td>Clase</td><td colspan="4">Tecnológico</td></tr>
<tr><td>Descripción</td><td colspan="4">El entrenamiento de redes neuronales convolucionales puede requerir una gran cantidad de recursos computacionales, lo que podría superar la capacidad del hardware disponible, afectando los tiempos de entrenamiento y la viabilidad del proyecto.</td></tr>
<tr><td>Plan de mitigación</td><td colspan="4">Recurrir al uso de una máquina virtual de la escuela. Pedir más recursos para la máquina en caso necesario.</td></tr>
<tr><td>Plan de contingencia</td><td colspan="4">Si las limitaciones persisten, reducir la complejidad del modelo, utilizar técnicas de entrenamiento distribuido o cambiar a una arquitectura más eficiente. También se puede buscar acceso a hardware de alto rendimiento en la universidad o a través de plataformas de cómputo en la nube.</td></tr>
<tr><td rowspan="3">Evaluación de Riesgo</td><td></td><td>Posibilidad</td><td>Impacto</td><td>Riesgo Total</td></tr>
<tr><td>Premitigación</td><td>2</td><td>2</td><td>Medio (4)</td></tr>
<tr><td>Postmitigación</td><td>2</td><td>1</td><td>Bajo (2)</td></tr>
</table>

Tabla 2.12: R-06 Limitaciones del hardware

Identificador	R-07			
Nombre	Problemas con las bibliotecas software			
Clase	Tecnológico			
Descripción	Durante el desarrollo del proyecto pueden surgir incompatibilidades entre versiones de bibliotecas utilizadas, especialmente aquellas relacionadas con frameworks de machine learning (TensorFlow, PyTorch) y bibliotecas de procesamiento de datos. Además, algunas versiones pueden no ser compatibles con el hardware disponible debido a su antigüedad, lo que puede afectar al rendimiento o incluso impedir la ejecución del modelo.			
Plan de mitigación	Utilizar entornos virtuales o contenedores (como Conda o Docker) para aislar dependencias y asegurar compatibilidad. Mantener un control estricto de las versiones utilizadas y documentarlas correctamente. Consultar foros y documentación oficial para evitar conflictos entre bibliotecas.			
Plan de contingencia	Si se presentan problemas graves de compatibilidad, evaluar la posibilidad de cambiar de versión de la biblioteca o utilizar alternativas compatibles. En caso de incompatibilidad con el hardware, considerar el uso de máquinas virtuales.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	3	3	Crítico (9)
	Postmitigación	2	2	Medio (4)

Tabla 2.13: R-07 Problemas con las bibliotecas software

<table>
<tr><td>Identificador</td><td colspan="4">R-8</td></tr>
<tr><td>Nombre</td><td colspan="4">Pérdida de datos o código</td></tr>
<tr><td>Clase</td><td colspan="4">Tecnológico</td></tr>
<tr><td>Descripción</td><td colspan="4">Existe el riesgo de perder datos críticos o código debido a fallos en el almacenamiento, corrupción de archivos, eliminación accidental o fallos en plataformas en la nube. Esto podría retrasar significativamente el desarrollo del proyecto y comprometer los resultados.</td></tr>
<tr><td>Plan de mitigación</td><td colspan="4">Implementar un sistema de copias de seguridad automáticas tanto en local como en la nube. Usar control de versiones con GitHub para el código y asegurar que los datos originales se almacenan en repositorios seguros. Documentar adecuadamente los cambios para evitar confusión y pérdida de trabajo.</td></tr>
<tr><td>Plan de contingencia</td><td colspan="4">En caso de pérdida de datos, recurrir a copias de seguridad previas o recuperar versiones anteriores desde el control de versiones. Si el problema es grave, evaluar la posibilidad de regenerar o recopilar nuevamente los datos perdidos.</td></tr>
<tr><td rowspan="3">Evaluación de Riesgo</td><td></td><td>Posibilidad</td><td>Impacto</td><td>Riesgo Total</td></tr>
<tr><td>Premitigación</td><td>2</td><td>3</td><td>alto (6)</td></tr>
<tr><td>Postmitigación</td><td>2</td><td>1</td><td>Bajo (2)</td></tr>
</table>

Tabla 2.14: R-8 Pérdida de datos o código

<table>
<tr><td>Identificador</td><td colspan="4">R-9</td></tr>
<tr><td>Nombre</td><td colspan="4">Fallos en las plataformas de trabajo</td></tr>
<tr><td>Clase</td><td colspan="4">Tecnológico</td></tr>
<tr><td>Descripción</td><td colspan="4">El uso de plataformas en la nube (GitHub, Overleaf etc.) y herramientas externas (máquina virtual) conlleva el riesgo de fallos en su disponibilidad, errores técnicos, incompatibilidades o problemas de acceso, lo que puede interrumpir el desarrollo del proyecto.</td></tr>
<tr><td>Plan de mitigación</td><td colspan="4">Disponer de copias de seguridad locales del código y los documentos en caso de caída de GitHub o Overleaf. Mantener un entorno de desarrollo alternativo por si la máquina virtual falla. Descargar versiones offline de los archivos críticos y configurar sincronización periódica.</td></tr>
<tr><td>Plan de contingencia</td><td colspan="4">En caso de fallo en GitHub, utilizar otro servicio de repositorio (como GitLab o Bitbucket) o trabajar temporalmente en local. Si Overleaf no funciona, editar los documentos en un editor LaTeX offline. Si la máquina virtual falla, usar una instalación local o un servidor alternativo.</td></tr>
<tr><td rowspan="3">Evaluación de Riesgo</td><td></td><td>Posibilidad</td><td>Impacto</td><td>Riesgo Total</td></tr>
<tr><td>Premitigación</td><td>2</td><td>3</td><td>Alto (6)</td></tr>
<tr><td>Postmitigación</td><td>1</td><td>1</td><td>Muy Bajo (1)</td></tr>
</table>

Tabla 2.15: R-9 Fallos en las plataformas de trabajo

<table>
<tr><td>Identificador</td><td colspan="4">R-10</td></tr>
<tr><td>Nombre</td><td colspan="4">Mala distribución del tiempo</td></tr>
<tr><td>Clase</td><td colspan="4">Gestión</td></tr>
<tr><td>Descripción</td><td colspan="4">Una planificación deficiente del tiempo disponible puede llevar a acumulación de tareas, retrasos en los hitos del proyecto y presión innecesaria en las etapas finales, afectando la calidad del trabajo.</td></tr>
<tr><td>Plan de mitigación</td><td colspan="4">Elaborar un cronograma detallado con fechas clave y objetivos parciales. Utilizar metodologías de gestión del tiempo como la técnica Pomodoro o la matriz de Eisenhower. Hacer revisiones periódicas del avance y ajustar la planificación si es necesario.</td></tr>
<tr><td>Plan de contingencia</td><td colspan="4">Si se detectan retrasos, priorizar tareas críticas, eliminar actividades secundarias o solicitar ayuda a los tutores para redefinir los objetivos. Reasignar horas extra en semanas posteriores para compensar el tiempo perdido sin comprometer la calidad.</td></tr>
<tr><td rowspan="3">Evaluación de Riesgo</td><td></td><td>Posibilidad</td><td>Impacto</td><td>Riesgo Total</td></tr>
<tr><td>Premitigación</td><td>3</td><td>3</td><td>Crítico (9)</td></tr>
<tr><td>Postmitigación</td><td>2</td><td>2</td><td>Medio (4)</td></tr>
</table>

Tabla 2.16: R-10 Mala distribución del tiempo

Identificador	R-11			
Nombre	Definición inadecuada de los objetivos			
Clase	Gestión			
Descripción	Si los objetivos del proyecto no están bien definidos desde el inicio, pueden surgir desviaciones en el trabajo, dificultando la obtención de resultados relevantes y generando un esfuerzo innecesario en tareas irrelevantes.			
Plan de mitigación	Definir los objetivos siguiendo la metodología SMART. Validar los objetivos con los tutores y otros expertos antes de avanzar en el desarrollo. Realizar revisiones periódicas para asegurar que los objetivos siguen siendo adecuados.			
Plan de contingencia	Si se detectan problemas en la definición de los objetivos, se reestructurará el proyecto para ajustarlo a un alcance realista. Se priorizarán los objetivos más relevantes y se redefinirán aquellos que no aporten valor al estudio.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	2	2	Medio (3)
	Postmitigación	1	2	Bajo (2)

Tabla 2.17: R-11 Definición inadecuada de los objetivos

Identificador	R-12			
Nombre	Falta de retroalimentación oportuna			
Clase	Gestión			
Descripción	La ausencia de revisiones periódicas y comentarios de los tutores por falta de disponibilidad puede llevar a desviaciones en el desarrollo del proyecto, dando lugar a errores que podrían haberse corregido a tiempo.			
Plan de mitigación	Planificar reuniones periódicas con los tutores para discutir avances y recibir sugerencias. Enviar informes de progreso de forma regular y solicitar retroalimentación.			
Plan de contingencia	En caso de que no se reciba retroalimentación a tiempo, se recurrirá a otras fuentes de validación, como autoevaluaciones basadas en literatura, consulta con compañeros o profesionales del área, y revisión de proyectos similares.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	1	2	Bajo (2)
	Postmitigación	1	1	Muy Bajo (1)

Tabla 2.18: R-12 Falta de retroalimentación oportuna

Identificador	R-13			
Nombre	Dificultad al interpretar resultados			
Clase	Gestión			
Descripción	La complejidad de los modelos puede dificultar la interpretación de los resultados obtenidos, afectando la toma de decisiones y la validez de las conclusiones.			
Plan de mitigación	Realizar pruebas con conjuntos de datos más pequeños y comprensibles antes de aplicar el modelo final. Utilizar herramientas de visualización de datos y métricas estándar para evaluar el rendimiento. Consultar literatura científica y buscar asesoramiento.			
Plan de contingencia	En caso de que los resultados sean difíciles de interpretar, se simplificará el modelo o se explorarán métodos alternativos de evaluación. Además, se documentará cada paso detalladamente para facilitar la revisión y comprensión posterior.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	2	2	Medio (4)
	Postmitigación	1	1	Bajo (1)

Tabla 2.19: R-13 Dificultad al interpretar resultados

Identificador	R-14			
Nombre	Falta de documentación adecuada			
Clase	Gestión			
Descripción	La ausencia de una documentación clara y estructurada sobre el desarrollo del proyecto puede generar dificultades en la reproducibilidad, comprensión y mantenimiento del trabajo. Esto puede llevar a errores, pérdida de tiempo y dificultades en la evaluación final.			
Plan de mitigación	Establecer una estructura de documentación desde el inicio del proyecto, incluyendo bitácoras de avances, comentarios en el código y reportes periódicos. Utilizar herramientas de control de versiones como GitHub para registrar cambios y mantener un historial del desarrollo.			
Plan de contingencia	En caso de detectar carencias en la documentación avanzada la fecha de entrega, se priorizará la redacción de los aspectos clave del proyecto, asegurando que al menos los procedimientos esenciales estén bien explicados. Se recurrirá a registros en GitHub y notas previas para recuperar información.			
Evaluación de Riesgo		**Posibilidad**	**Impacto**	**Riesgo Total**
	Premitigación	2	3	Alto (6)
	Postmitigación	1	2	Bajo (2)

Tabla 2.20: R-14 Falta de documentación adecuada

2.4. Seguimiento Real

Para finalizar esta sección, en las siguientes tablas se presenta el seguimiento real del proyecto, se incluyen entre paréntesis las desviaciones con respecto a las duraciones estimadas (positivo representa aumento de duración). El fin real de la primera etapa coincide con el que se planificó (Tabla 2.1). Aunque algunas de las actividades concretas llevaron menos tiempo, debido a problemas con las versiones de TensorFlow y la migración del sistema a PyTorch (riesgo R-07), se produjeron retrasos considerables en otras actividades que llevaron a requerir una mayor carga de trabajo para esta primera etapa, por lo que el cómputo final del tiempo empleado es mayor que el planificado. En la Tabla 2.21, se presentan las fechas y tiempos reales de esta primera etapa.

Actividad	**Inicio real**	**Fin real**	**Duración real (h)**
Definición del problema	Día 1, Semana 1	Día 1, Semana 1	1 (-2)
Revisión del estado del arte	Día 1, Semana 1	Día 3, Semana 2	30 (+10)
Formulación de objetivos	Día 4, Semana 2	Día 4, Semana 2	5 (0)
Elección de metodología	Día 5, Semana 2	Día 5, Semana 2	2 (0)
Selección de herramientas	Día 5, Semana 2	Día 5, Semana 2	3 (0)
Planificación inicial	Día 1, Semana 3	Día 5, Semana 3	25 (+15)
Elaboración del cronograma	Día 1, Semana 4	Día 2, Semana 4	5 (-1)
Documentación de riesgos	Día 2, Semana 4	Día 3, Semana 4	7 (+1)
Estimación de presupuestos	Día 3, Semana 4	Día 3, Semana 4	3 (-2)
Entornos de trabajo	Día 4, Semana 4	Día 5, Semana 4	10 (+2)

Tabla 2.21: Registro real de actividades - Etapa I: Inicio del proyecto

En la Tabla 2.22 se exponen las duraciones reales de las actividades de la segunda etapa. Diversos motivos, como el cambio en el dispositivo de trabajo (riesgo R-06) debido a problemas con el hardware, han llevado a que esta etapa se retrase una semana con respecto a la planificación original (Tabla 2.2). Cabe destacar que la actividad "Generación del conjunto de imágenes" se ha extendido en el tiempo debido a que se han generado más conjuntos de los inicialmente proyectados.

Dada la inesperada complejidad de la implementación de algunos modelos, y la larga duración de las pruebas ejecutadas, la tercera etapa del proyecto ha pasado de durar 154 horas, como se estimaba (Tabla 2.3), a 206 en la realidad, extendiéndose hasta la semana 17. Entra en juego en este retraso el riesgo R-01, ya que gran parte del tiempo extra que se ha empleado en esta etapa se debe a la falta de experiencia en el uso de este tipo de arquitecturas. Se pueden ver en la Tabla 2.23 las duraciones reales de las actividades que conforman esta etapa.

Actividad	Inicio real	Fin real	Duración real (h)
Recuperación de los datos	Día 1, Semana 5	Día 1, Semana 5	5 (+3)
Análisis exploratorio (EDA)	Día 2, Semana 5	Día 5, Semana 5	20 (+5)
Limpieza y procesamiento	Día 1, Semana 6	Día 5, Semana 6	25 (+5)
Generación del conjunto de imágenes	Día 1, Semana 7	Día 3, Semana 7	15 (+5)
Conjuntos de train y test	Día 4, Semana 7	Día 5, Semana 7	10 (+5)
Balanceo de clases	Día 1, Semana 8	Día 1, Semana 8	3 (0)
Documentación del análisis	Día 2, Semana 8	Día 5, Semana 8	10 (+5)

Tabla 2.22: Registro real de actividades - Etapa II: Análisis y procesamiento de los datos

Actividad	Inicio real	Fin real	Duración real (h)
Comparación de resultados	Día 3, Semana 17	Día 3, Semana 17	5 (0)
Extracción de conclusiones	Día 4, Semana 17	Día 4, Semana 17	5 (0)
Elaboración de documentación final	Día 5, Semana 17	Día 5, Semana 18	25 (-5)
Creación del material para la presentación	Día 1, Semana 19	Día 1, Semana 19	5 (0)
Presentación del proyecto	Semana de presentación	Semana de presentación	2 (0)

Tabla 2.24: Registro real de actividades - Etapa IV: Cierre del Proyecto

La última etapa correspondiente al cierre de proyecto no presentó retrasos significativos con respecto a la planificación inicial (Tabla 2.4), como se observa en la Tabla 2.24. De hecho, gracias a la documentación que se ha ido recopilando a lo largo de la ejecución, la elaboración de la memoria ha llevado menos tiempo del esperado. El proyecto real se retrasó unas tres semanas, extendiéndose hasta el día 20 de mayo de 2025. En total, una duración aproximada de 400 horas, 100 más de las que se proyectaron, todo esto sin contar con el tiempo que el sistema ha estado funcionando por su cuenta durante la obtención de resultados.

2.5. Estimación de costes

El desarrollo de un proyecto tecnológico de esta envergadura conlleva una serie de costes asociados que deben ser tenidos en cuenta para garantizar su viabilidad. En esta sección se presenta una estimación del presupuesto necesario para llevar a cabo el trabajo en un entorno profesional, considerando todos los gastos inherentes a la ejecución de una investigación de este tipo aplicada en el ámbito del reconocimiento biométrico. Posteriormente, se comparará esta estimación con el presupuesto real requerido para la realización de este Trabajo de Fin de Grado, en el que muchos de los recursos han sido proporcionados por la universidad o asumidos personalmente por el alumno, con la correspondiente reducción en los costes.

En un escenario realista, la implementación de un sistema de este tipo basado en CNNs y dispositivos ponibles implicaría una inversión significativa en distintos campos. Entre ellos destacan los costes de personal, incluyendo salarios y beneficios sociales de investigadores y desarrolladores; la adquisición de hardware específico; las licencias de software especializadas; los costes de infraestructura computacional, tales como servidores en la nube o estaciones de trabajo; y gastos administrativos asociados a la gestión del proyecto, como impuestos y posibles costes legales.

Sin embargo, en el contexto de este TFG, muchos de estos costes han sido minimizados o eliminados gracias a la disponibilidad de recursos proporcionados por la Universidad de Valladolid. En particular, el acceso a licencias de software como MS Project, herramientas computacionales y una máquina virtual dedicada ha permitido reducir significativamente la inversión necesaria en infraestructura. Asimismo, al tratarse de un trabajo académico desarrollado individualmente, no se han considerado costes laborales, lo que representa una diferencia sustancial con respecto a un escenario profesional.

A continuación, se presentan dos análisis presupuestarios: en primer lugar, un presupuesto teórico basado en los costes reales y, en segundo lugar, una estimación del presupuesto real empleado para la elaboración de este trabajo.

2.5.1. Presupuesto en un entorno profesional

Para elaborar el presupuesto estimado para llevar a cabo el proyecto, se han considerado diversos factores clave que se detallan a continuación.

Para comenzar, se propone el equipo necesario para el desarrollo y los costes de personal asociados. Se considera que se debe contar con un jefe de proyecto que se encargue de la organización del mismo, la coordinación de tareas y miembros del equipo, así como de otros trabajos de administración y documentación. Un arquitecto de IA puede ser clave para poder diseñar las arquitecturas de red necesarias para que el programador pueda implementar el resto del sistema. Finalmente, al ser un sistema en tiempo real y que integra numerosos subsistemas, es necesario contar con una persona encargada de probar que todo funciona de manera adecuada.

A continuación, se detallan en la tabla 2.25 los salarios brutos anuales promedio para cada puesto:

Puesto	**Salario Bruto Anual (€)**
Jefe de Proyecto [20]	41.000
Arquitecto de IA [33]	37.660
Programador [40]	29.800
Tester [21]	24.000

Tabla 2.25: Salarios brutos anuales por puesto

Como se estableció en la sección 2.3, la duración estimada del proyecto es de en torno a 4 meses. El coste total de personal para 4 meses se calcula entonces como:

$$\text{Coste total} = \sum_{i=1}^{n} \left(\frac{\text{Salario anual del puesto}_i}{12} \times 4 \right) \tag{2.1}$$

donde n es el número de puestos. Lo que supone un total de 44.153 €; no obstante, hay dos gastos extras a tener en cuenta a la hora de calcular el presupuesto necesario para pagar el sueldo de un empleado.

En España, el coste de la Seguridad Social a cargo de la empresa es aproximadamente el 30 % del salario bruto del empleado, repartido entre varios factores como la cotización por contingencias comunes, por formación, por desempleo o por accidentes de trabajo, entre otros. Adicionalmente, es necesario tener en cuenta el IRPF (Impuesto sobre la Renta de las Personas Físicas), por el cual se aplicará la retención correspondiente según el tramo impositivo, en este caso del 2 %.

La cifra final será, por tanto, 58.252 €. Es evidente que se podrían reducir gastos empleando a menos personas, o contratando personal menos cualificado, o incluso firmando contratos por horas en lugar de por meses y calculando las horas trabajadas por cada empleado. Sin embargo, el objetivo de este presupuesto es presentar un supuesto ideal.

Por otro lado, se presentan los costes en material hardware necesario para llevar a cabo la investigación. Se propone adquirir temporalmente tres equipos con sus respectivos periféricos, para que todos los trabajadores puedan llevar a cabo sus tareas, todo ello en alquiler durante los cuatro meses que dura el proyecto. Finalmente, se emplearán dos dispositivos ponibles de diferentes marcas para la toma de datos. Los precios de la Tabla 2.26 son una aproximación basada en una revisión del mercado actual.

Equipo	**Coste (€)**
Smartwatch Microsoft	250
Smartwatch Motorola	200
Estaciones de trabajo (3)	600
Total	1.050

Tabla 2.26: Costes estimados de hardware

Los precios de las estaciones de trabajo se han extraído de [62]. Dada la naturaleza del trabajo, se selecciona el plan profesional.

En cuanto a los costes de software y licencias, todo el software empleado es de acceso gratuito, a excepción de Microsoft Project, que se empleará para la gestión del proyecto, con un coste estimado de 1.659 €[53] al año, es decir, unos 553 €para los cuatro meses del proyecto.

Para la infraestructura en la nube, se prevé el uso de Google Cloud Platform (GCP) con una instancia NVIDIA Tesla T4. El coste estimado es de 0,35 €/hora [10]. Se supone un uso de 20 horas semanales durante 16 semanas, lo que da lugar a un coste total de 112 €.

Para concluir, en la Tabla 2.27 se presenta el desglose de los costes y el total que asciende hasta los 58.997 €.

Concepto	**Coste Estimado (€)**
Costes de Personal	58.252
Costes de Hardware	1.050
Costes de Software y Licencias	553
Costes de Infraestructura en la Nube	112
Total	58.997

Tabla 2.27: Presupuesto total estimado

2.5.2. Presupuesto Real

En este apartado, se analiza la reducción de costes en el desarrollo del proyecto debido a su naturaleza académica y al apoyo proporcionado por la Universidad. A diferencia del presupuesto profesional, donde se contemplaban costes laborales, de infraestructura y software, en el ámbito universitario se dispone de diversos recursos que minimizan la inversión necesaria.

Dado que este proyecto se desarrolla como un Trabajo de Fin de Grado, no se incurre en gastos salariales. El trabajo es realizado por el estudiante de manera no remunerada, con la supervisión de los tutores académicos, quienes ya forman parte del personal de la Universidad y no representan un coste adicional.

La Universidad proporciona acceso a licencias de software especializado (MS Project), la infraestructura computacional, incluyendo servidores con GPU para entrenamiento de modelos, laboratorios con estaciones de trabajo adecuadas para el desarrollo (aunque se empleará el equipo personal principalmente) así como la base de datos preexistente, por lo que los dispositivos ponibles no serán necesarios. Esto elimina la necesidad de adquirir hardware adicional y licencias comerciales.

En lugar de contratar servicios en la nube como Google Cloud, se empleará JupyterLab para el desarrollo y GitHub para el almacenamiento, lo que evita costes de almacenamiento y procesamiento.

Se presenta en la Tabla 2.28 una comparativa entre los costes estimados en un entorno profesional y los costes reales en el entorno académico. Como se observa, se alcanza un coste total de 0 €, lo que hace viable el desarrollo del proyecto dentro del contexto académico.

Concepto	**Coste Profesional**	**Coste Académico**
Personal	Alto	0 €
Hardware	Medio	0 €(ya adquirido)
Software	Medio	0 €(licencias de la Universidad)
Cloud Computing	Medio	0 €(máquina virtual)

Tabla 2.28: Comparación de costes entre entorno profesional y académico

Actividad	Inicio real	Fin real	Duración real (h)
ResNet18	Día 1, Semana 9	Día 1, Semana 11	**45 (-3)**
Investigación	Día 1, Semana 9	Día 1, Semana 9	2 (-2)
Implementación	Día 1, Semana 9	Día 5, Semana 9	17 (-3)
Obtención de resultados	Día 1, Semana 10	Día 5, Semana 10	25 (+5)
Ajuste de parámetros	Día 1, Semana 10	Día 5, Semana 10	*25*
Documentación	Día 1, Semana 11	Día 1, Semana 11	1 (-3)
ResNet50	Día 1, Semana 11	Día 3, Semana 12	**32 (+12)**
Investigación	Día 1, Semana 11	Día 1, Semana 11	3 (0)
Implementación	Día 2, Semana 11	Día 2, Semana 11	5 (+2)
Obtención de resultados	Día 3, Semana 11	Día 2, Semana 12	20 (+10)
Ajuste de parámetros	Día 3, Semana 11	Día 2, Semana 12	*20*
Documentación	Día 3, Semana 12	Día 3, Semana 12	4 (0)
EfficientNet	Día 4, Semana 12	Día 5, Semana 13	**32 (+12)**
Investigación	Día 4, Semana 12	Día 4, Semana 12	3 (0)
Implementación	Día 5, Semana 12	Día 5, Semana 12	4 (+1)
Obtención de resultados	Día 1, Semana 13	Día 4, Semana 13	20 (+10)
Ajuste de parámetros	Día 3, Semana 13	Día 4, Semana 13	*20*
Documentación	Día 5, Semana 13	Día 5, Semana 13	5 (+1)
MobileNet	Día 2, Semana 15	Día 5, Semana 12	**33 (+13)**
Investigación	Día 1, Semana 14	Día 1, Semana 14	3 (0)
Implementación	Día 2, Semana 14	Día 2, Semana 14	5 (+2)
Obtención de resultados	Día 3, Semana 14	Día 1, Semana 15	20 (+10)
Ajuste de parámetros	Día 3, Semana 14	Día 1, Semana 15	*20*
Documentación	Día 2, Semana 15	Día 2, Semana 15	5 (+1)
Custom	Día 3, Semana 15	Día 2, Semana 17	**54 (+24)**
Investigación	Día 3, Semana 15	Día 4, Semana 15	10 (+2)
Implementación	Día 5, Semana 15	Día 1, Semana 16	15 (+7)
Obtención de resultados	Día 2, Semana 16	Día 1, Semana 17	25 (+15)
Ajuste de parámetros	Día 2, Semana 16	Día 1, Semana 17	*25*
Documentación	Día 2, Semana 17	Día 2, Semana 17	4 (0)

Tabla 2.23: Registro real de actividades - Etapa III: Creación y evaluación de los modelos (las horas en cursiva corresponden a actividades que se realizan en paralelo)

Capítulo 3

Contexto de la investigación

En las últimas décadas, el reconocimiento biométrico se ha convertido en una herramienta fundamental para la autenticación de usuarios. La necesidad de métodos más seguros, precisos y no invasivos ha impulsado el desarrollo de técnicas avanzadas basadas en inteligencia artificial y dispositivos ponibles. Es por esto que el reconocimiento biométrico mediante la forma de andar ha cobrado un creciente interés recientemente y se ha postulado como una interesante alternativa frente a otros métodos biométricos como las huellas dactilares o el reconocimiento facial.

Los principales estudios que han abordado esta materia se han basado en técnicas enfocadas en modelos estadísticos, redes neuronales recurrentes (RNNs), arquitecturas LSTM, *transformers* o incluso lógica difusa [66, 52, 60]. Sin embargo, el uso de imágenes a partir de los datos inerciales como entrada para CNNs en este contexto sigue siendo un área poco explorada.

En el presente capítulo se realiza una revisión del estado del arte en las tecnologías y metodologías empleadas en este Trabajo de Fin de Grado. Dichas metodologías incluyen técnicas de aprendizaje profundo, así como el uso de modelos preentrenados a través de transferencia de aprendizaje.

En primer lugar, se presentan los fundamentos de la biometría y sus aplicaciones más relevantes. A continuación, se describe el papel de los dispositivos ponibles (*wearables*) como mecanismo de adquisición de datos. Seguidamente, se revisan los principios de las redes neuronales artificiales y su especialización en arquitecturas convolucionales. Posteriormente, se abordan las aplicaciones de estas redes en el ámbito biométrico. Además, se analiza la transferencia de aprendizaje como técnica clave para aprovechar el conocimiento adquirido en grandes conjuntos de datos, permitiendo adaptar modelos preentrenados a nuevos dominios con un coste computacional y temporal significativamente menor. Finalmente, se profundiza en el campo específico de la biometría basada en la marcha.

Si bien el enfoque principal de esta investigación es la biometría, también se quieren destacar las posibles implicaciones que se pueden derivar en otros campos, como el análisis de patrones de marcha en pacientes con trastornos motores o la optimización de sistemas de monitorización en el ámbito deportivo y de la salud.

3.1. Biometría: Fundamentos y Aplicaciones

El concepto de biometría se puede definir como el estudio de métodos automáticos para el reconocimiento de individuos basados en sus características físicas o conductuales. A diferencia de otros sistemas tradicionales de autenticación, como contraseñas o tarjetas de identificación, los sistemas biométricos proporcionan un mecanismo más natural, difícil de falsificar y que no requiere que el usuario recuerde información específica.

Se pueden clasificar los rasgos biométricos en dos grandes categorías: biometría física y biometría conductual [37]. La biometría física incluye características invariables como las huellas dactilares, el rostro, el iris o el ADN. Por otro lado, la biometría conductual engloba rasgos que dependen de la manera en que un individuo realiza una cierta acción, tales como la manera de firmar, la voz o la forma de andar y que pueden estar sujetas a cambios con el paso del tiempo.

Los sistemas biométricos clásicos constan de cinco módulos bien diferenciados: adquisición de datos, preprocesamiento, extracción de características, comparación con plantillas almacenadas y decisión. Además, dependiendo de su objetivo, se puede hablar de dos familias de sistemas biométricos: los que se basan en la identificación (identificar a un usuario entre un grupo de muchos posibles) o los que pretenden lograr la verificación (verificar si un usuario es quien dice ser) [48] (Figura 3.1).

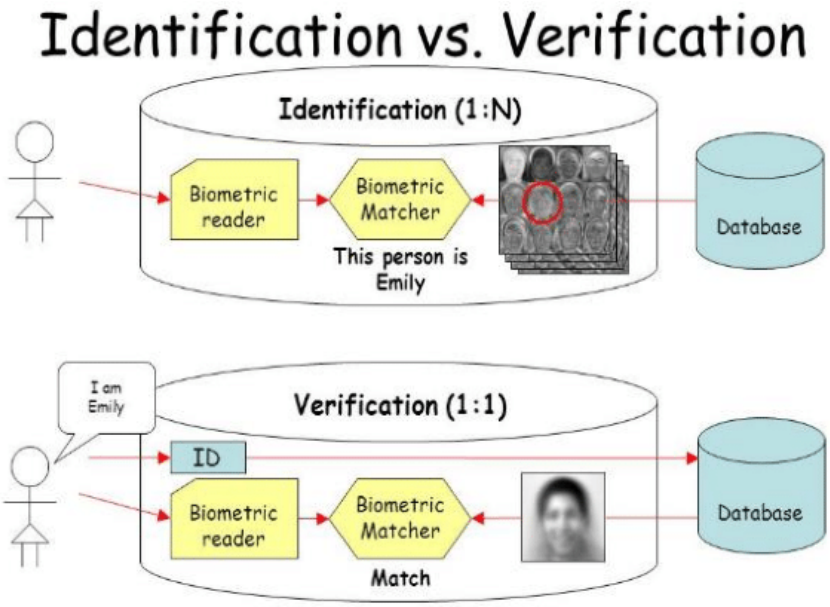


Figura 3.1: Identificación (1:N) vs. Verificación (1:1). Fuente: [3]

La biometría cuenta con una serie de ventajas para su uso en autenticación como la no transferibilidad del rasgo biométrico (es decir, la 'clave' empleada en la autenticación es personal e intransferible), la comodidad de uso y su aplicación en sistemas de seguridad de alto nivel. No obstante, existen desafíos importantes, como la variabilidad intraclase, la susceptibilidad al ruido, los problemas de privacidad y la posibilidad de ataques de suplantación o *spoofing* [58] que dificultan la implementación de este tipo de sistemas.

Existe una amplia variedad de contextos en los que la biometría se ha aplicado, algunos de ellos pertenecen a los sectores más importantes de la sociedad actual: desde controles de acceso hasta aplicaciones en banca, sanidad, justicia y fronteras internacionales. En el contexto de los dispositivos móviles, tecnologías como el reconocimiento facial o dactilar se han convertido en estándares de facto para la autenticación del usuario [50].

En entornos industriales y laborales, la biometría se utiliza para el control de presencia, la supervisión de tareas o la gestión de recursos humanos. En el ámbito forense, juega un papel relevante en la identificación de víctimas o sospechosos. En los últimos años, ha surgido también un creciente interés por emplear biometría en el ámbito de la salud para monitorizar patrones motores, síntomas neurológicos o cambios fisiológicos.

Entre todos estos, uno de los campos emergentes que ha generado más repercusión dentro de la biometría conductual es el análisis de patrones de movimiento, en particular la marcha humana o forma de andar. Este tipo de biometría tiene la ventaja de poder ser capturada de manera no intrusiva y continua mediante sensores ponibles o cámaras, lo que la hace ideal para aplicaciones en entornos reales [4].

Cabe remarcar que, a pesar de lo que pueda parecer, la marcha se considera un rasgo biométrico viable, dado que posee características únicas y estables para cada individuo, aunque también puede verse influida por factores externos como el calzado, el estado emocional o el entorno físico, lo que introduce retos adicionales en su análisis [1, 11].

3.2. Dispositivos ponibles en la biometría

El desarrollo de la biometría en entornos reales ha traído consigo la aparición de nuevos sistemas de adquisición de datos. Entre ellos se encuentran los dispositivos ponibles (*wearables*), que han cobrado especial relevancia debido a su capacidad para capturar información de manera continua, cómoda y no intrusiva.

Se define dispositivo ponible como cualquier aparato electrónico que puede vestirse de forma natural (por ejemplo, en la muñeca, cintura o pie), y que incorpora sensores capaces de medir variables del entorno o del propio usuario [24]. En los últimos años ha habido una gran evolución en este sector, desde simples contadores de pasos o monitores de ritmo cardíaco hasta complejos sistemas de adquisición multicanal, con capacidad de conexión a redes y procesamiento local o remoto de la información.

Gracias al avance en miniaturización de sensores y otros circuitos y al desarrollo de tecnologías inalámbricas de bajo consumo, este tipo de dispositivos ha logrado hacerse un hueco en el mercado en campos como la medicina, el deporte, la vigilancia del estado físico y, más recientemente, la biometría.

Los dispositivos ponibles modernos integran una gran variedad de sensores que permiten la captura de datos biométricos en diferentes dominios:

- **Acelerómetros y giroscopios**: miden aceleración lineal y rotacional en tres ejes. Son la base de la biometría basada en marcha, postura o gestos [15] y serán los empleados para este trabajo.
- **Sensores de frecuencia cardíaca (PPG)**: útiles para autenticación basada en señales fisiológicas, como la variabilidad de la frecuencia cardíaca [61].
- **Electrodos de ECG y EMG**: empleados en autenticación mediante patrones eléctricos del corazón o de la actividad muscular.
- **Micrófonos y sensores de vibración**: para biometría de voz o características relacionadas con patrones de tos, habla o respiración.
- **Sensores ambientales**: presión, temperatura, luz, humedad, que pueden emplearse para enriquecer el contexto de adquisición.

El uso en biometría de este tipo de dispositivos presenta múltiples ventajas frente a otros métodos más tradicionales de captura directa de datos biométricos: permiten recoger datos en tiempo real sin requerir la participación activa del usuario, lo que se conoce como biometría transparente o pasiva; su formato similar al de dispositivos de uso diario (relojes, pulseras, zapatillas) aumenta la aceptación social del sistema; pueden integrarse en entornos móviles o sistemas de seguridad sin necesidad de infraestructuras complejas y empleando lenguajes conocidos; la combinación de señales fisiológicas y conductuales medidas en tiempo real ofrece una mayor resistencia frente a ataques de tipo *spoofing* o *replay* [50].

A pesar de estas ventajas, también se plantean una serie de retos: Al tratarse de dispositivos comerciales no especializados, las diferencias entre modelos o fabricantes pueden introducir sesgos en los datos y dificultades en la implementación; pueden presentar exceso de ruido debido al movimiento, a interferencias electromagnéticas o a errores de sincronización; los datos biométricos capturados son altamente sensibles y requieren de estrictas medidas de protección y encriptación [82]; finalmente, el procesamiento local de los datos puede limitarse por la batería y la capacidad computacional del dispositivo, lo cual es una de las principales limitaciones actualmente.

En este trabajo se pretende aprovechar los sensores inerciales de los dispositivos ponibles, en concreto un Motorola Watch y una Microsoft Band (Figuras 3.2a, 3.2b), para desarrollar un sistema de verificación biométrica basado en marcha, transparente para el usuario, portable y eficiente.

(a) Dispositivo motorola

(b) Dispositivo microsoft

Figura 3.2: Dispositivos ponibles empleados en la investigación

3.3. Redes Neuronales Artificiales

Las redes neuronales artificiales (ANN, por sus siglas en inglés) constituyen uno de los pilares fundamentales del aprendizaje automático moderno. Su creación fue inspirada por el funcionamiento del cerebro humano, más concretamente de las neuronas y sus conexiones. Estas redes permiten construir modelos computacionales capaces de aprender representaciones complejas a partir de los datos y resolver tareas como clasificación, regresión, detección de patrones o segmentación.

El concepto de red neuronal artificial tiene su origen en los trabajos de McCulloch y Pitts en la década de 1940, quienes propusieron un modelo matemático basado en funciones lógicas [51]. Posteriormente, Rosenblatt desarrolló el perceptrón en 1958, considerado el primer modelo de aprendizaje supervisado con capacidad de clasificación lineal [63]. Aunque limitadas, estas primeras aproximaciones sentaron las bases del desarrollo posterior.

La idea fundamental detrás de las redes neuronales es la de imitar la estructura del sistema nervioso biológico, donde un conjunto de unidades (neuronas) se conectan mediante sinapsis (pesos), procesando señales de entrada y generando respuestas. Las redes se componen de capas de neuronas interconectadas. Estas capas se organizan habitualmente en tres tipos (Figura 3.3):

- **Capa de entrada**: recibe las variables del problema como vectores numéricos.
- **Capas ocultas**: realizan transformaciones sucesivas mediante funciones lineales y no lineales.
- **Capa de salida**: proporciona el resultado final del modelo (habitualmente una etiqueta o una probabilidad).

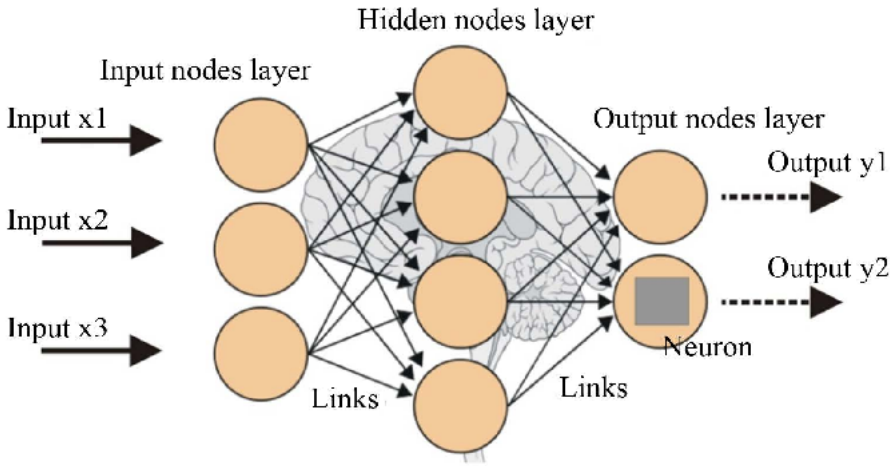


Figura 3.3: Estructura básica de una ANN. Fuente: [49]

Cada neurona realiza una combinación lineal de sus entradas ponderadas por los pesos y aplica una función de activación no lineal, como ReLU, sigmoide o tangente hiperbólica, para introducir no linealidad en el modelo [22] y permitir representaciones de mayor complejidad.

El conjunto de pesos y sesgos de la red se ajusta durante cada etapa de entrenamiento mediante algoritmos de optimización como la retropropagación del error (*backpropagation*) y métodos basados en el descenso de gradiente, como SGD, Adam o RMSprop [42].

Las redes neuronales profundas (*deep neural networks*, DNN) simplemente son un subtipo de red que contiene múltiples capas ocultas. Gracias a esto, pueden aprender representaciones jerárquicas de los datos mucho más complejas, extrayendo características de alto nivel a partir de patrones simples. Según el teorema de aproximación universal, una red neuronal con una sola capa oculta puede aproximar cualquier función continua, pero las redes profundas lo hacen de manera más eficiente y con menor complejidad computacional [29].

Dependiendo del tipo de datos y del problema, existen distintas variantes arquitectónicas de redes neuronales:

- **Perceptrón Multicapa (MLP)**: red densa con conexiones entre todas las neuronas de capas consecutivas. Apta para problemas cuyas entradas son vectores de caracteríasticas unidimensionales, en los que no se tienen en cuenta relaciones espaciales ni temporales.
- **Redes Convolucionales (CNN)**: especializadas en procesamiento de datos con estructura espacial, como imágenes. Utilizan filtros matriciales que permiten extraer características con invarianza espacial.
- **Redes Recurrentes (RNN, LSTM, GRU)**: adecuadas para datos secuenciales como texto o series temporales. Incorporan conexiones hacia atrás que permiten mantener una "memoria" del pasado.
- **Redes de tipo Transformer**: inicialmente aplicadas en procesamiento de lenguaje natural, han sido adoptadas recientemente para visión por computador con arquitecturas como Vision Transformer (ViT) [17].

Las redes neuronales artificiales han revolucionado múltiples disciplinas, como el reconocimiento de imágenes y vídeo, el procesamiento del lenguaje natural (traducción, resumen, análisis de sentimiento), el diagnóstico médico asistido por IA, la conducción autónoma o sistemas biométricos y de seguridad, entre otros.

3.4. Aprendizaje profundo en biometría

El aprendizaje profundo ha supuesto un avance revolucionario en el desarrollo de sistemas biométricos. Frente a otras técnicas tradicionales, basadas en la extracción manual de características y clasificadores estadísticos como SVM, k-NN o árboles de decisión, los modelos de aprendizaje profundo son capaces de aprender representaciones más complejas directamente a partir de los datos brutos, lo que ha incrementado notablemente la precisión, robustez y generalización de los sistemas biométricos actuales [39].

El uso de este tipo de redes ha tenido un impacto especialmente significativo en modalidades biométricas como:

- **Reconocimiento facial**: con modelos como DeepFace [73], FaceNet [69] o ArcFace [14], capaces de alcanzar tasas de precisión superiores al 99 % en bases de datos exigentes como LFW o MegaFace.
- **Reconocimiento de voz**: mediante el uso de redes recurrentes o arquitecturas basadas en espectrogramas y CNN, se han desarrollado sistemas de autenticación robustos al ruido y al habla espontánea [55, 80].
- **Reconocimiento de huellas e iris**: las CNN permiten extraer características altamente discriminativas a partir de imágenes, incluso en condiciones de baja calidad o ángulos no ideales [54, 19, 75].
- **Firma manuscrita**: redes recurrentes y convolucionales se han combinado con éxito para capturar tanto la dinámica como la forma estática de firmas trazadas en tabletas digitalizadoras [25].

El uso de aprendizaje profundo ofrece múltiples ventajas, ya que este tipo de arquitecturas: eliminan la necesidad de diseñar descriptores manualmente, reduciendo el sesgo humano introducido; pueden modelar relaciones complejas no lineales y adaptarse a grandes volúmenes de datos con alta variabilidad; gracias a la redundancia en múltiples niveles de representación, son menos sensibles a la variación intrausuario, iluminación, oclusión, etc.; además, la misma arquitectura puede aplicarse a múltiples formas de biometría con pequeñas adaptaciones.

A pesar de todas estas ventajas, el uso de este tipo de sistemas también plantea varios retos: el entrenamiento requiere conjuntos de datos extensos y balanceados, algo que no siempre es posible; en modalidades con poca variabilidad o usuarios limitados, los modelos pueden memorizar en lugar de generalizar, incurriendo en lo que se conoce como sobreajuste; las decisiones de las redes neuronales profundas son difíciles de interpretar, llegando a ser sistemas de caja negra, lo que complica su adopción en entornos sensibles como la justicia o la salud; finalmente, se ha demostrado que los modelos de aprendizaje profundo pueden ser engañados mediante pequeñas perturbaciones diseñadas específicamente (ej. ataques de tipo adversarial) [7].

3.5. Transferencia de aprendizaje en biometría

La transferencia de aprendizaje (*transfer learning*) es una técnica ampliamente utilizada en el contexto del aprendizaje automático que permite reutilizar modelos previamente entrenados en un dominio o tarea (fuente) para resolver una tarea diferente pero relacionada (destino). En lugar de entrenar una red neuronal desde cero —lo que requiere grandes cantidades de datos y, lo más condicionante en este caso, recursos computacionales—, se aprovechan los pesos aprendidos por una red pre-entrenada sobre un conjunto de datos extenso y se adaptan al objetivo mediante técnicas de *fine-tuning*.

Existen varias razones que apoyan el uso de la transferencia de aprendizaje, entre las que destacan algunas como: la dificultad para contar con conjuntos de datos biométricos grandes y diversos, el elevado coste de adquisición, etiquetado y validación de dichos datos, la gran calidad y la alta capacidad de generalización de modelos preentrenados sobre tareas visuales, que aprenden representaciones jerárquicas reutilizables.

Cabe destacar que numerosos estudios han demostrado que las primeras capas de redes convolucionales entrenadas sobre imágenes naturales extraen características genéricas (como bordes, texturas o formas básicas) útiles en una amplia gama de tareas [81].

Existen dos estrategias principales para aplicar esta técnica:

- **Extracción de características**: se utiliza el modelo preentrenado como extractor de características. Las capas convolucionales se congelan y solo se entrena un clasificador (por ejemplo, una capa densa o un MLP) sobre las salidas intermedias del modelo.
- **Fine-tuning**: se toma un modelo preentrenado y se reentrenan algunas de sus capas —normalmente las superiores— junto con las nuevas capas añadidas, permitiendo adaptar parcialmente los pesos al nuevo dominio [32].

La elección entre una u otra técnica depende del tamaño y similitud del conjunto de datos destino con respecto al de origen. En el presente trabajo, se han explorado ambos enfoques con diversos resultados.

Algunos estudios recientes han demostrado que, en ciertas situaciones, la transferencia de aprendizaje permite obtener resultados competitivos incluso cuando el conjunto de datos destino es pequeño o presenta ruido [54].

3.6. Redes Neuronales en Biometría de la marcha

La marcha humana o *gait* es una característica biométrica conductual que ha resultado prometedora en los últimos años, especialmente por ser no intrusiva, poder ser capturada de diversas formas y por su capacidad para operar de forma ininterrumpida. Este rasgo se ha convertido en una alternativa viable a otros mecanismos biométricos tradicionales, especialmente en escenarios donde se desea una autenticación pasiva y transparente.

Cada individuo tiene una forma única de caminar debido a diversos factores como la anatomía, la neuromusculatura y el comportamiento. Esto permite construir modelos capaces de identificar o verificar usuarios, ya sea mediante análisis de señales capturadas por cámaras o sensores inerciales, entre otros. Aunque inicialmente se exploró en el ámbito de la videovigilancia, el avance de los dispositivos ponibles ha ampliado su aplicabilidad.

Con el auge del *deep learning*, las redes neuronales convolucionales se han convertido en una herramienta prominente para su aplicación en el campo de la biometría; sin embargo, el uso de este tipo de redes en señales relacionadas con la marcha es una perspectiva escasamente explorada en la bibliografía.

En el presente trabajo, se espera que este tipo de redes permita abstraer patrones discriminativos a partir de datos de acelerómetros y giroscopios al convertir estas señales en representaciones bidimensionales.

A día de hoy, diversos estudios han abordado el problema del reconocimiento por marcha con redes neuronales. A continuación se describen algunos de los más relevantes:

- **Luo et al. (2022)** [47] emplean datos multimodales de sensores ponibles y redes recurrentes con capas *Squeeze-and-Excite* para identificación basada en actividades físicas, incluyendo la marcha.
- **Delgado-Santos et al. (2022)** [12] proponen las arquitecturas *Transformer* para biometría conductual con datos de marcha, destacando la robustez de los modelos basados en atención.
- **M-GaitFormer (2023)** [13] introduce una arquitectura propia basada en Transformer optimizada, enfocada en verificación biométrica con alta eficiencia energética.
- **GaitSet (Chao et al., 2018)** [8] propone un enfoque de conjunto de pasos que mejora la invarianza a vistas en datos de vídeo.
- **Said et al. (2018)** [65] utilizan RNN entrenadas sobre señales de aceleración adquiridas por sensores ponibles en diferentes partes del cuerpo, logrando tasas de verificación prometedoras.

- **Sun et al. (2018)** [72] desarrollan una RNN con una segmentación del ciclo de marcha adaptable a velocidad de marcha para autenticación continua en dispositivos IoT portables.

3.6.1. Síntesis de estudios relevantes

En la Tabla 3.1 se presenta una recopilación de trabajos recientes que utilizan redes neuronales para tareas de verificación o identificación biométrica a partir de la marcha humana, destacando el año, el dispositivo empleado en la toma de datos, el tipo de modelo y la tarea abordada.

Autor(es)	**Año**	**Dispositivo**	**Modelo**	**Tarea**
Luo et al. [47]	2022	Wearables	RNN	Identificación
Delgado-Santos et al. [12]	2022	Wearables	Transformer	Verificación
Delgado-Santos et al. (M-GaitFormer) [13]	2023	Móviles	Transformer	Verificación
Chao et al. (GaitSet) [8]	2019	Vídeo multivista	CNN 2D-Set	Identificación
Said et al. [65]	2018	Acelerómetro ponible	RNN	Identificación
Sun et al. [72]	2018	IoT wearable	RNN adaptable	Verificación
Wan et al. (Survey) [77]	2018	Varios	CNN, RNN, LSTM, HMM, Transformer	Revisión
He et al. [27]	2015	Cámaras RGB	3D-CNN + LSTM	Identificación

Tabla 3.1: Estudios recientes sobre biometría de marcha utilizando redes neuronales

Incluso dada la diversidad de enfoques que se presenta en la bibliografía, ningún trabajo encontrado durante las etapas de investigación del proyecto propone combinar el uso de dispositivos ponibles de uso diario para la extracción de datos, con la elección de CNNs como arquitectura de red para la verificación. Esto demuestra que el presente TFG propone una idea innovadora que supone la apertura de un camino inexplorado hasta ahora en la literatura.

Además, el objetivo principal no es tanto ni el reconocimiento de actividades ni la identificación, sino la verificación biométrica, lo que permite confirmar la identidad de un usuario previamente registrado frente a un usuario ajeno al sistema, añadiendo una capa adicional de seguridad a las aplicaciones.

La motivación principal es, por tanto, explorar el potencial de las CNNs en este ámbito, dado su éxito en la extracción de características espaciales en otras áreas. Así como aplicar estas técnicas a señales de acelerómetros y giroscopios integrados en dispositivos comerciales, buscando desarrollar un sistema de verificación biométrica de la marcha que sea eficiente, preciso y fácilmente integrable en soluciones existentes.

Capítulo 4

Tecnologías empleadas

4.1. Herramientas de desarrollo y gestión

Para la realización de este proyecto se emplearon diversas herramientas destinadas tanto a la implementación como a la gestión del código. A continuación se detallan las principales.

El entorno principal de desarrollo fue JupyterLab, utilizado para escribir y ejecutar el código de manera interactiva. Gracias a su estructura basada en celdas, facilita la experimentación con pequeños subconjuntos de datos y el ajuste iterativo del modelo. Sin embargo, debido a la capacidad de procesamiento limitada del equipo local, las pruebas a gran escala y con el conjunto de datos completo se realizaron en una máquina virtual proporcionada por la Escuela de Ingeniería Informática de Valladolid (UVa).

En la máquina virtual se utilizó Vim en su versión básica para realizar ajustes rápidos de hiperparámetros, comprobaciones y modificaciones menores en el código. Su elección se debió a su eficiencia en entornos remotos y su bajo consumo de recursos, además de que es nativo en la máquina virtual.

Para la gestión del código y los resultados del proyecto se utilizó GitHub como sistema de control de versiones. Se optó por un repositorio sencillo organizado en carpetas, donde se almacenaron el código fuente, los resultados obtenidos en distintas ejecuciones y algunos subconjuntos de imágenes procesadas, evitando la necesidad de almacenarlas en el dispositivo local o regenerarlas en cada ejecución.

Dado que el desarrollo fue en solitario, no se implementó un flujo complejo de trabajo como Git Flow, sino una estructura simple con un control manual de los cambios.

Para la documentación del proyecto se utilizó Overleaf, un entorno en línea basado en LaTeX que permitió la redacción y la gestión centralizada del documento. Aunque el trabajo se realizó de manera individual, Overleaf facilitó la organización de las referencias y la generación automática de bibliografía. Además, se empleó la licencia de MS Project otorgada por la universidad para la gestión del proyecto y la realización de los diagramas de Gantt. Finalmente, se emplearon diversas IAs generativas como ChatGPT para revisiones de redacción y corrección, búsqueda de referencias, reformulación de textos y para facilitar el aprendizaje de algunas herramientas como Overleaf que no se habían usado antes.

4.2. Bibliotecas y frameworks empleados

A lo largo del proyecto se emplearon diversas bibliotecas de Python para el procesamiento de datos, el entrenamiento de modelos y la evaluación de resultados.

Inicialmente se desarrolló una primera versión del sistema utilizando TensorFlow, debido a su accesibilidad y la falta de experiencia. Sin embargo, al cambiar de dispositivo, surgieron problemas de compatibilidad con las versiones antiguas de TensorFlow, que eran necesarias para utilizar TensorFlow Hub en la transferencia de aprendizaje (riesgo R-07). Las versiones más recientes de TensorFlow no permitían emplear esta funcionalidad de la misma manera, lo que obstaculizó el desarrollo.

Ante esta situación y con base en lo estudiado en la asignatura de Minería de Datos, se optó por PyTorch, una biblioteca que ofrece mayor control a bajo nivel sobre la arquitectura de los modelos, una API más intuitiva y flexible, integración sencilla con herramientas como Torchvision para el preprocesamiento de imágenes y un sistema de cálculo de gradientes dinámico que facilita la depuración.

4.2.1. Bibliotecas auxiliares

Además de PyTorch, se emplearon otras bibliotecas fundamentales:

- **NumPy**: Manipulación eficiente de datos numéricos y operaciones matriciales.
- **Matplotlib**: Generación de las imágenes y visualización de los resultados, incluyendo gráficos de evolución del entrenamiento.
- **tqdm**: Generación de barras de progreso para facilitar el seguimiento de iteraciones largas.
- **glob**: Manejo de rutas y archivos dentro de los conjuntos de datos y carga de las imágenes en los DataSets.
- **scikit-learn (sklearn)**: Cálculo de métricas de evaluación y validación de modelos.
- **Pandas**: Organización y análisis de datos tabulares, permitiendo facilitar la limpieza y normalización y una mejor interpretación de los resultados.

Capítulo 5

Conjuntos de datos

El presente estudio se basa en el uso de datos previamente recopilados por un grupo de investigación del Departamento de Informática de la Universidad de Valladolid [57], que durante varias fases ha adquirido un nutrido conjunto de datos. Estos datos han sido generados mediante la captura de señales de sensores incorporados en dispositivos comerciales de muñeca, concretamente una Microsoft Band y un Motorola Moto 360.

Estos dispositivos cuentan con múltiples sensores integrados, entre ellos acelerómetros y giroscopios, que permiten registrar información detallada sobre el movimiento del usuario a través de una app Android desarrollada por el grupo de investigación. La adquisición de estos datos tuvo como objetivo original la detección de patrones en las señales inerciales que permitieran verificar la identidad de un usuario empleando métodos estadísticos y de minería de datos tradicionales.

A lo largo de este capítulo se describirá en detalle la estructura de los datos, los procesos de preprocesamiento aplicados, así como la transformación de las señales en las imágenes que serán posteriormente utilizadas como entrada para los modelos de aprendizaje profundo desarrollados.

5.1. Fuente y Características de los Datos

La recopilación de datos se realizó con la participación de un conjunto de 38 usuarios, que llevaron puestos los dispositivos en la muñeca mientras realizaban una serie de recorridos. Los datos fueron almacenados en archivos tabulados con las siguientes columnas principales:

- **timestamp**: Marca de tiempo de cada medición.
- **dato1, dato2, dato3**: Representan las lecturas en los ejes X, Y y Z obtenidas de los sensores.

Cada usuario generó dos sesiones de registro, y dentro de cada sesión se realizaron varias muestras (entre una y tres dependiendo del usuario). Así, la estructura de la base de datos se organiza en función del usuario, dispositivo, sensor, sesión y muestra, dando lugar a un conjunto extenso de archivos de datos. Posteriormente, se separa cada muestra en ventanas con solapamiento correspondiente a unos pocos ciclos de la marcha.

5.2. Preprocesamiento de Datos y Generación de Imágenes

El preprocesamiento es una etapa fundamental en el desarrollo de cualquier proyecto de minería de datos, ya que garantiza la calidad de las hipótesis generadas por los clasificadores y empleadas para la construcción de los modelos. En esta sección se describen los distintos procedimientos aplicados a los datos antes de ser utilizados en el entrenamiento.

5.2.1. Eliminación de valores incorrectos

El primer paso en el preprocesamiento de datos consiste en la detección y eliminación de valores incorrectos, en particular aquellos registros que contienen valores negativos para la columna `timestamp`. La presencia de estos valores puede deberse a errores en la sincronización de los sensores o en la captura de datos.

Para corregir estos problemas, se sigue el siguiente procedimiento:

- Se carga el archivo CSV correspondiente utilizando la biblioteca `pandas`.
- Se calcula la columna de tiempo acumulado, que es la suma acumulada de los valores de los timestamp.
- Se verifica la existencia de valores negativos en timestamp.
- Se ordenan los datos por tiempo acumulado y se eliminan aquellos registros con valores negativos.
- Se recalcula la columna de `timestamp` como la diferencia entre valores consecutivos de tiempo acumulado, asegurando que el primer valor sea 0.
- Se vuelve a calcular tiempo acumulado como la suma acumulada de la nueva columna de timestamps.

Este proceso garantiza que los datos sean consistentes antes de proceder con las siguientes etapas del preprocesamiento.

5.2.2. Ruido

El ruido es uno de los mayores retos a los que se enfrentan los sistemas biométricos, por ser una de las principales fuentes que introducen incertidumbre en los clasificadores, tanto en fase de entrenamiento como en la de evaluación. Este fenómeno puede definirse como cualquier tipo de distorsión, error o artefacto que altera la representación verdadera de la señal fisiológica o conductual del individuo, afectando potencialmente al rendimiento del sistema de verificación. Existen dos tipos principales de ruido según la literatura: ruido sistemático y ruido asistemático [43].

Ruido sistemático. Se refiere a errores que se repiten consistentemente bajo condiciones específicas, normalmente derivados del proceso de adquisición, de limitaciones inherentes al sensor o de problemas estructurales en el protocolo de captura. Este tipo de ruido no es aleatorio y, por lo tanto, no es recomendable eliminarlo mediante técnicas estadísticas convencionales. En el ámbito biométrico, el ruido sistemático puede surgir por latencias en los dispositivos, retardo en el almacenamiento de muestras, o errores causados por la interacción del usuario con el sistema [38].

Ruido asistemático. Por el contrario, el ruido asistemático (también conocido como ruido aleatorio) aparece de forma impredecible y puede deberse a factores externos como condiciones ambientales, movimientos involuntarios del usuario o errores de comunicación. Aunque no puede evitarse completamente, su efecto puede mitigarse mediante técnicas de normalización, filtrado o aumento de datos [18] y es recomendable emplear dichas técnicas para intentar eliminar los registros con este tipo de ruido.

Ruido en el corpus de datos utilizado

En el caso concreto del presente trabajo, se parte de un conjunto de datos ya adquiridos. A pesar de haberse realizado la recolección de manera supervisada, el análisis visual realizado sobre el conjunto ha evidenciado la presencia de varios tipos de ruido, principalmente sistemático, que afecta directamente a la calidad de las muestras y, en consecuencia, a los resultados.

Las siguientes anomalías se han observado de manera reiterada:

- **Pérdidas de conexión al inicio o final de la muestra.** Este fenómeno ocurre cuando la aplicación tarda en iniciar o detener la grabación después de que el usuario pulsa el botón correspondiente. Es un claro ejemplo de *ruido sistemático*, ya que depende de la lógica interna de la aplicación y se reproduce con alta frecuencia en múltiples usuarios. Estas pérdidas, aunque afectan al comienzo o al final de las muestras, no comprometen la integridad del segmento central de la señal, por lo que no se descartan por completo sino que se consideran una manifestación inherente a la naturaleza del sistema.

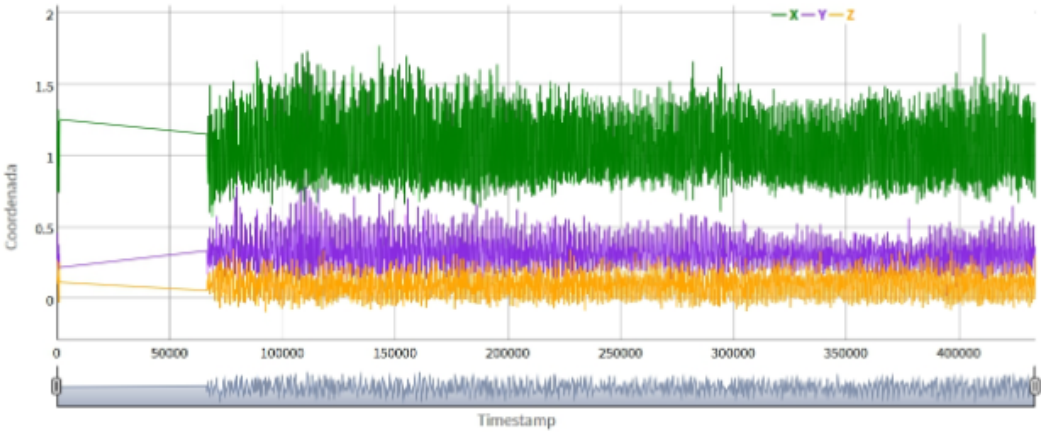
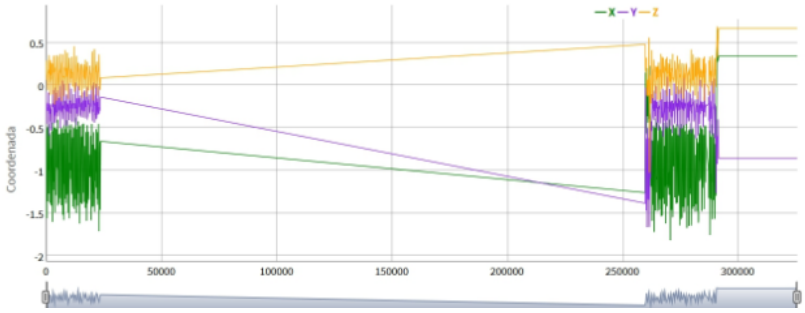
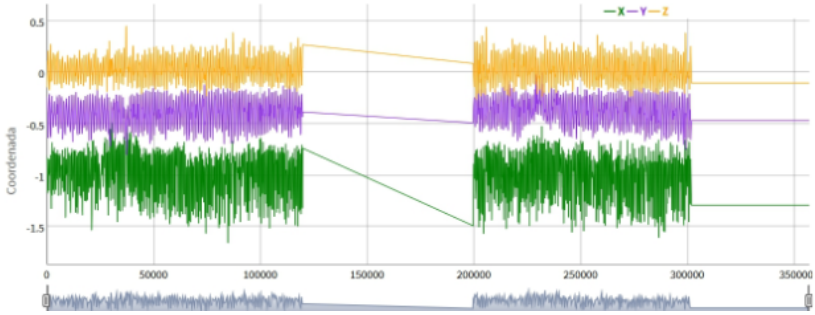


Figura 5.1: Pérdida de conexión al comienzo de la muestra. Fuente: [57]

- **Pérdidas intermedias de conexión.** Se trata de un comportamiento menos frecuente. Su aparición interrumpe la continuidad temporal de la muestra y, en algunos casos, justifica el descarte total del usuario, como se muestra en las Figuras 5.2a y 5.2b. Estas interrupciones son particularmente problemáticas desde el punto de vista del modelado temporal de la señal.



(a) Pérdida intermedia 1. Fuente: [57]



(b) Pérdida intermedia 2. Fuente: [57]

Figura 5.2: Ejemplos de pérdidas intermedias de conexión

- **Señales de duración reducida.** La aplicación puede bloquearse de manera silenciosa durante la obtención de los datos, generando muestras que contienen una fracción insuficiente con respecto a lo esperado (por ejemplo, menos del 30 %). En los casos documentados, esto justifica el descarte de las muestras afectadas. Nuevamente se trata de un tipo de *ruido sistemático* derivado de errores persistentes de software. La pérdida de información resulta evidente al representar las muestras afectadas (Figura 5.3).

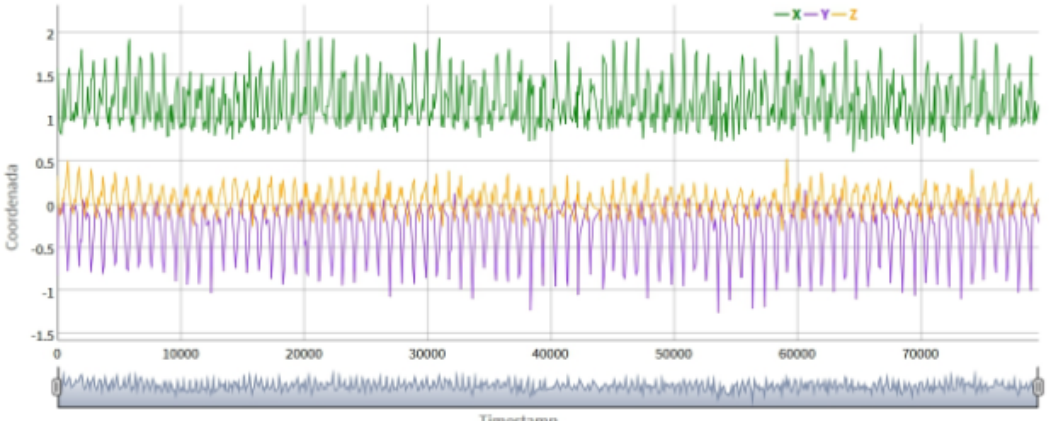


Figura 5.3: Bloqueo grave de la aplicación: señal corta y pérdida severa de datos. Fuente: [57]

- **Presencia de valores negativos.** En algunas muestras aparece un único valor negativo en la serie, indicando una inversión en el orden cronológico de las capturas. Esto suele deberse a errores puntuales de sincronización entre el hardware de adquisición y la aplicación, por lo que se clasifica como ruido asistemático. Al tratarse de ruido asistemático, se eliminarán estos valores para evitar sesgos. Los gráficos resultantes evidencian la presencia de valores negativos en los timestamps (Figura 5.4).

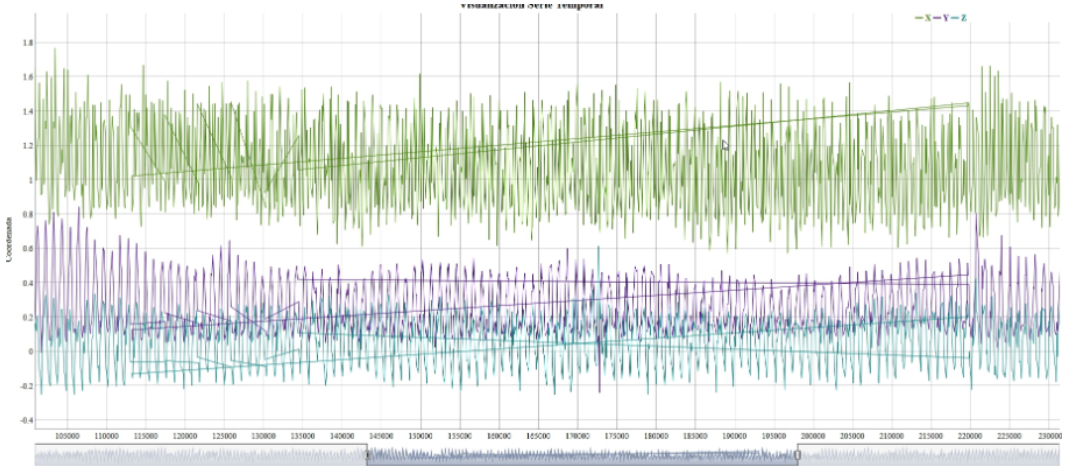


Figura 5.4: Presencia de valores negativos. Fuente: [57]

Desde el punto de vista de la biometría conductual, es importante entender que ciertos tipos de ruido no deben eliminarse, ya que forman parte del comportamiento natural del sistema o de la interacción del usuario. Tal es el caso del ruido sistemático que se observa en las pérdidas iniciales o finales de las muestras. Este tipo de irregularidades, lejos de ser un defecto, puede aportar información adicional sobre la estabilidad del sistema o sobre el estilo de interacción del usuario.

Por esta razón, se ha decidido mantener las muestras con pérdidas iniciales o finales siempre que el segmento principal sea suficientemente representativo. Solo en casos de corrupción grave (como en la Figura 5.3) o pérdida sustancial de la señal, se opta por eliminar los datos completos del usuario de la base de datos.

Esta estrategia se basa en las recomendaciones establecidas en estudios previos de la bibliografía sobre análisis de robustez en sistemas biométricos [41, 68], que sugieren que no es aconsejable realizar un filtrado agresivo del ruido cuando este es inherente al canal o al comportamiento del usuario, ya que esto podría conducir a un sesgo artificial o inducido en la evaluación del sistema.

La gestión del ruido en este trabajo corresponde con lo previamente desarrollado por el grupo y ha sido abordada con un enfoque dual: por un lado, conservando aquellas anomalías que se consideran parte inherente del canal biométrico (ruido sistemático); por otro, eliminando casos extremos que comprometen la validez de la muestra. Esta metodología respeta el principio de fidelidad en la representación del comportamiento biométrico del usuario, al tiempo que garantiza la calidad mínima requerida para una correcta evaluación de los modelos entrenados.

5.2.3. Interpolación y normalización de datos

Para asegurar un muestreo uniforme, se realiza un proceso de interpolación sobre los valores de los sensores, lo que permite generar datos a intervalos regulares, independientemente de las diferencias en la frecuencia de muestreo original. Para ello, primero se genera una nueva secuencia de tiempos con un período de muestreo constante; posteriormente, se emplea interpolación lineal integrada en la biblioteca `numpy` para estimar los valores de los datos correspondientes a cada coordenada en los tiempos generados y, finalmente, se construye un nuevo DataFrame con los datos interpolados, asegurando que los valores estén distribuidos de manera uniforme en el tiempo.

5.2.4. Segmentación en ventanas

Para facilitar el análisis y entrenamiento de modelos de aprendizaje automático, los datos se dividen en segmentos o ventanas de un tamaño predefinido, permitiendo además un solapamiento de muestras.

El procedimiento consiste en: obtener el número total de muestras en la zona de interés, eliminar registros con valores faltantes, recorrer los datos en segmentos de tamaño de la ventana avanzando en cada iteración según el número de muestras solapadas y si los valores dentro de una ventana son válidos (es decir, no todos los valores son idénticos), se genera una imagen correspondiente a la ventana. El proceso se repite hasta recorrer todas las muestras disponibles.

Esta técnica permite que los datos sean manejables y adecuados para ser utilizados en modelos de clasificación, garantizando que cada ventana contenga información representativa de los movimientos capturados por los sensores, abarcando lo que se conoce en la bibliografía como un ciclo de la marcha.

5.2.5. Normalización de Datos

Para mejorar la comparabilidad de los valores y evitar que ciertas características dominen el entrenamiento de los modelos, los datos se normalizan. Se aplica una normalización min-max, ajustando los valores en un rango entre 0 y 1. Este proceso permite que todas las características (en este caso las coordenadas) contribuyan de manera equilibrada al aprendizaje del modelo.

5.2.6. Generación de Imágenes

Uno de los enfoques innovadores en este trabajo es la conversión de los datos de sensores en representaciones visuales. Para ello se utilizaron transformaciones que convierten las secuencias temporales en imágenes de diversos tipos, permitiendo el uso de redes neuronales convolucionales en su análisis. Se exploraron tres enfoques principales para generar imágenes a partir de los datos:

Gráficos de Líneas para cada Coordenada

El primer método consiste en representar cada coordenada (x, y, z) en una gráfica de líneas y combinarlas en una misma imagen (Figura 5.5). Este enfoque permite visualizar la evolución de las señales a lo largo del tiempo, destacando patrones y correlaciones entre las coordenadas. Es la más sencilla y directa de todas.

La ventaja de este método radica en su capacidad para mantener la relación temporal de los datos. Esto puede resultar útil para modelos CNN que sean capaces de identificar patrones en el tiempo sin perder información contextual.

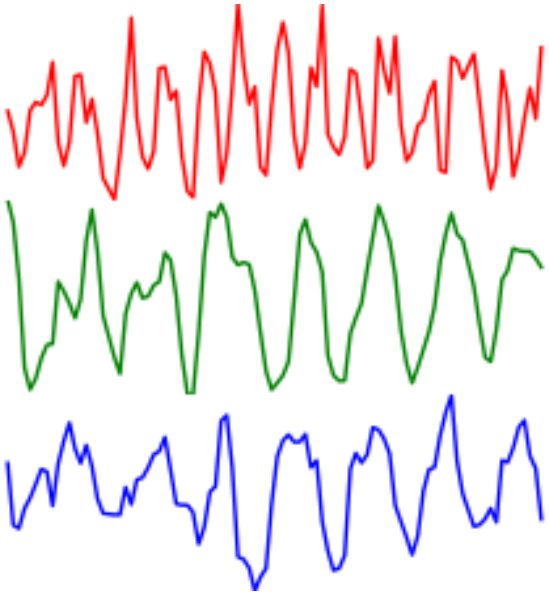


Figura 5.5: Ejemplo de gráfico de líneas para la señal de acelerómetro.

Mapas de Calor en Color

Este enfoque de creación propia, propuesto durante el desarrollo del proyecto, representa los valores de las coordenadas en forma de barras coloreadas según la magnitud de los datos. Cada franja de la imagen representa una coordenada (x, y, z) y el color codifica la intensidad del valor correspondiente a lo largo del tiempo (Figura 5.6).

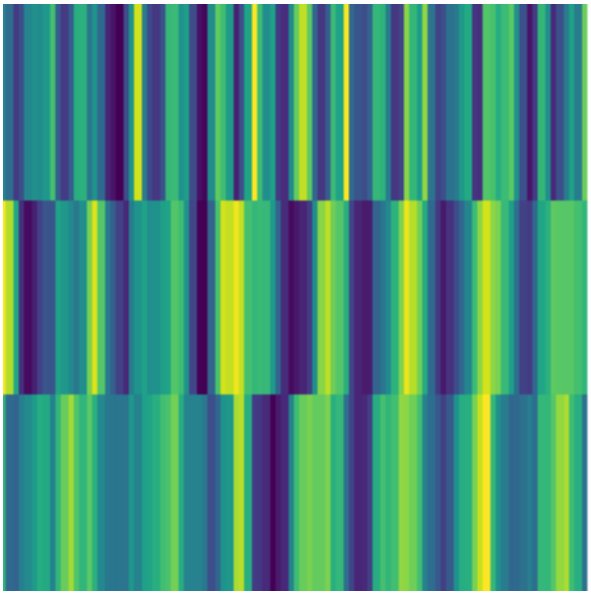


Figura 5.6: Ejemplo de mapa de calor a color.

Este tipo de representación permite que la CNN identifique patrones espaciales en las variaciones de los datos sin necesidad de interpretar las coordenadas como señales temporales. Es un método similar al usado en visión artificial para representar características en imágenes. Se pretende que se puedan comparar los valores en cuanto a su evolución temporal y entre sí.

Mapas de Calor en Escala de Grises

El tercer enfoque es una variante del anterior, pero en escala de grises (Figura 5.7). Se emplea para simplificar los modelos, reduciendo la dimensionalidad de los datos y facilitando el entrenamiento de redes con menor complejidad computacional.

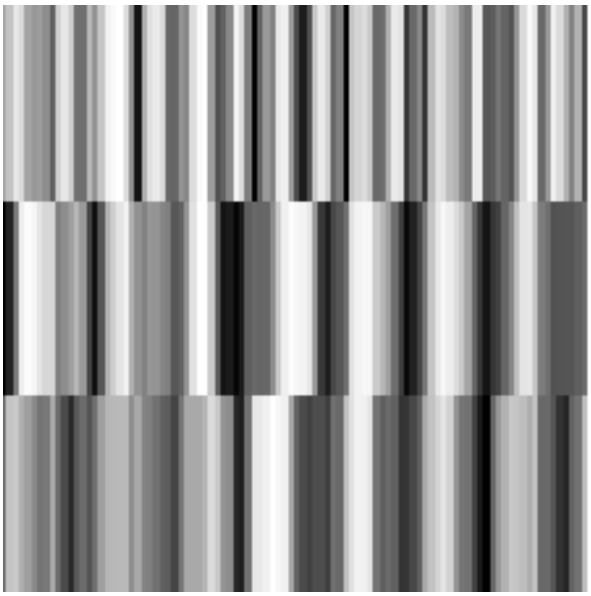


Figura 5.7: Ejemplo de mapa de calor en escala de grises.

Al eliminar la información de color, los modelos pueden concentrarse en detectar la estructura y los cambios de los datos sin verse afectados por la variabilidad cromática. Esto puede ser beneficioso para reducir el ruido en el entrenamiento y mejorar la generalización del modelo.

Para seleccionar la mejor representación, se llevaron a cabo pruebas con los diferentes enfoques, evaluando su desempeño en términos de precisión y velocidad de entrenamiento.

5.3. Formación de los conjuntos de entrenamiento y prueba

En el campo de la biometría, uno de los aspectos más críticos y al que más importancia se le da en la literatura es el diseño de los conjuntos de datos empleados para entrenar y evaluar los modelos [78]. A diferencia de otros basados en redes neuronales, con los sistemas de verificación biométrica se debe tener muy en cuenta la procedencia de cada muestra al dividir los datos.

Además, en el presente trabajo y debido a la naturaleza de los clasificadores empleados, cabe destacar que se entrena un modelo por cada usuario. Este modelo se encarga de llevar a cabo una tarea de clasificación binaria, distinguiendo entre muestras del usuario o clase genuina y muestras del usuario o clase impostor.

Es necesario simular un entorno real, en el que se lleva a cabo una inscripción (*enrollment*) del usuario en el sistema con una primera muestra y se entrena con los datos de una serie de usuarios que ya se encuentran dados de alta. Por otro lado, a la hora de probar, se necesita utilizar muestras que correspondan a otras tomas de datos (otras sesiones), simulando un usuario dado de alta que intenta verificarse en el sistema, y enfrentarlas contra muestras de otros usuarios impostores que intentan verificarse como el genuino, y que pueden o no estar dados de alta en el sistema.

Las imágenes, previamente preprocesadas, se almacenan siguiendo una estructura jerárquica de carpetas por usuario. Las muestras están etiquetadas en función de la sesión en la que se tomaron (1 o 2), la muestra (ya que en cada sesión se tomaron entre una y tres), el número de ventana al que pertenece, el dispositivo con el que se tomaron (Micro o Moto) y el sensor empleado (GYR o ACC para giroscopio o acelerómetro, respectivamente).

5.3.1. Construcción del D ataset

Para construir los conjuntos de entrenamiento y prueba, se ha diseñado una clase personalizada que extiende la interfaz `torch.utils.data.Dataset`. Esta clase se encarga de generar un conjunto equilibrado de imágenes con sus correspondientes etiquetas, 1 para la clase genuina y 0 para la clase impostora. No obstante, es importante destacar que, debido a la naturaleza del problema, se presenta un notorio desbalanceo de clases [26] ya que se tienen muchas más muestras impostoras que genuinas.

El balance de clases se controla duplicando las muestras del usuario objetivo tantas veces como sea necesario para igualar la proporción frente a los impostores en lo que se conoce como sobremuestreo u *oversampling* [6]. En este caso se realiza, por tanto, sobremuestreo por duplicación.

La selección de este tipo de balanceo de clases se debe a la experiencia empírica adquirida por el grupo de investigación. Por simple que parezca, esta forma de alcanzar el balance entre las clases ha demostrado un mejor rendimiento que las alternativas.

5.3.2. Selección de sesiones y muestras

Para cada usuario, las muestras del resto de usuarios se dividen en dos conjuntos disjuntos. Uno de ellos se usa para entrenar el sistema con ejemplos de la clase no genuina o impostor, y las muestras del otro se usan para las pruebas de impostor. Es importante que ambos conjuntos sean disjuntos, para no emplear muestras de un impostor concreto tanto en entrenamiento como en prueba. Además, para cada prueba y sujeto, ambos conjuntos son siempre los mismos, para poder comparar de manera objetiva los resultados.

En este trabajo se han evaluado dos escenarios experimentales clásicos en biometría:

Mono-session Para el conjunto de entrenamiento se utiliza la primera muestra de la primera sesión del usuario objetivo, junto con las mismas muestras de usuarios con paridad opuesta (para asegurar separación de clases). Para el conjunto de prueba se emplea la segunda muestra de la misma sesión para el usuario genuino, y la segunda muestra de la primera sesión de los usuarios distintos del genuino que no se emplearon para entrenar.

Este protocolo es útil para evaluar la capacidad del modelo de distinguir entre usuarios en condiciones controladas y homogéneas. Aunque presenta una menor variabilidad intraclase, permite validar si la arquitectura base es capaz de aprender características discriminativas. Se trata de un escenario muy realista, ya que no incorpora en las pruebas la variación de la característica biométrica en el tiempo.

Cross-session Este escenario es más exigente y representa mejor los desafíos reales a los que se enfrentan este tipo de sistemas. Aquí, para el entrenamiento, también se utiliza la primera muestra de la primera sesión; sin embargo, para probar se emplean ambas muestras de la segunda sesión, lo que introduce una variabilidad temporal (intrausuario) ya que entre ambas sesiones hay un período de tiempo significativo; todo esto se aplica tanto al usuario genuino como a los impostores. La distribución de los usuarios impostores es idéntica a la presentada en la modalidad *mono-session*.

Este planteamiento refleja situaciones prácticas en las que el sistema de verificación debe reconocer a un usuario meses o años después del registro inicial. Tal escenario simula el concepto de permanencia, una propiedad deseable en sistemas biométricos, definida como la estabilidad de los rasgos a lo largo del tiempo [78].

La distribución final de las muestras para cada conjunto bajo cada uno de los protocolos se refleja en el diagrama de la Figura 5.8.

En aplicaciones en tiempo real de verificación biométrica, como la autenticación en dispositivos móviles o el control de acceso sin contacto, es esencial que el sistema sea capaz de reconocer a un usuario genuino frente a cualquier otro individuo impostor, ajeno o no al sistema, sin requerir entrenamiento adicional, aparte de la sesión de *enrollment*.

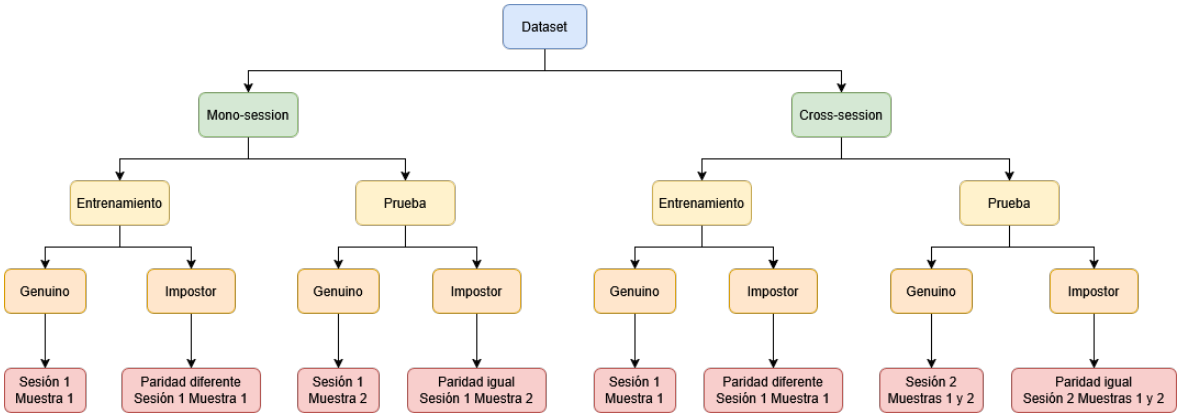


Figura 5.8: Esquema gráfico de los criterios mono-session y cross-session. Elaboración propia

En este trabajo se propone seguir una estrategia que simule precisamente ese escenario: se entrena un modelo para cada usuario, únicamente con sus propias muestras pertenecientes a una sola toma de datos y probado contra una colección de posibles impostores. Esta arquitectura de sistema se alinea con el concepto de verificación 1:1 definido en los estándares biométricos internacionales como ISO/IEC 19795-1 [35].

El diseño del dataset incluye una serie de técnicas comunes en la literatura. Entre ellas destacan algunas como: selección de usuarios alternos por paridad para evitar el solapamiento entre los conjuntos de entrenamiento y prueba, balanceo de clases dinámico para evitar sesgos de aprendizaje, normalización y transformación de las imágenes con los parámetros estándar de las redes preentrenadas para facilitar la transferencia de aprendizaje y una aleatorización controlada para garantizar la reproductibilidad de los experimentos, empleando semillas.

Estos métodos garantizan un entorno experimental sólido y reproducible, en el que es posible comparar diferentes arquitecturas de redes neuronales entre ellas y con otras propuestas en la literatura para evaluar su robustez, precisión y generalización a largo plazo.

Capítulo 6

Modelos

El desarrollo del presente trabajo se enmarca en el ámbito de la inteligencia artificial, un campo de la informática que tiene como objetivo dotar a las máquinas de la capacidad de imitar comportamientos inteligentes característicos del ser humano, como el razonamiento, el aprendizaje o la toma de decisiones [64]. Dentro de la IA, se encuentra una subdisciplina denominada aprendizaje automático (*machine learning*, ML), que permite a los sistemas aprender patrones a partir de datos sin ser programados explícitamente para cada tarea [5].

A su vez, el aprendizaje automático engloba el aprendizaje profundo (*deep learning*, DL), una rama que se basa en arquitecturas de redes neuronales artificiales de múltiples capas capaces de extraer representaciones jerárquicas de alto nivel a partir de los datos. Este enfoque ha experimentado un crecimiento significativo en los últimos años, impulsado por el aumento de la capacidad computacional, los avances en cálculos matriciales gracias a las nuevas GPUs y la disponibilidad de grandes volúmenes de datos con el desarrollo del Big Data[44].

La Figura 6.1 representa visualmente la jerarquía conceptual que forman los tres conceptos que se acaban de presentar.

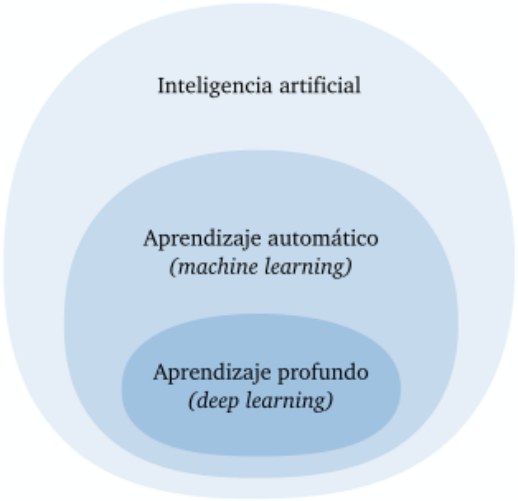


Figura 6.1: Jerarquía conceptual de la inteligencia artificial, el aprendizaje automático y el aprendizaje profundo. Fuente: [23]

Dentro del aprendizaje profundo, uno de los pilares fundamentales son las redes neuronales convolucionales, ampliamente utilizadas en tareas de clasificación, segmentación y análisis de datos en formato de imagen o estructurado espacialmente. Estas redes fueron introducidas inicialmente por Yann LeCun en los años 80 para el reconocimiento de caracteres manuscritos [45], y han evolucionado desde entonces hasta convertirse en el estándar de facto en problemas de visión por computadora.

Las CNN se inspiran en la organización del córtex visual de los mamíferos y utilizan una arquitectura en capas que alterna operaciones de convolución (matriciales), activación no lineal (normalmente ReLU), normalización y *pooling*, permitiendo capturar patrones espaciales de los datos de entrada con una alta eficiencia (Figura 6.2). A diferencia de los modelos tradicionales que requerían un diseño manual de las características, las CNN son capaces de aprender automáticamente representaciones discriminativas directamente a partir de los datos en bruto.

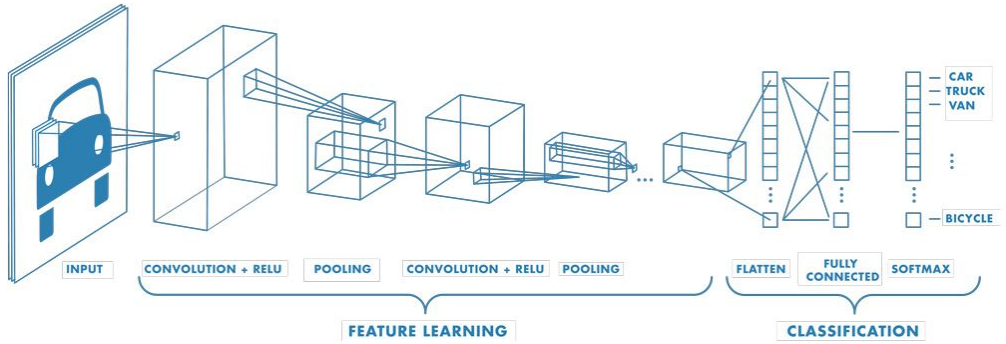


Figura 6.2: Estructura típica de una red neuronal convolucional. Fuente: [56]

El componente fundamental de una CNN es la capa convolucional, que aplica matrices llamadas filtros (*kernels*) con pesos a la entrada. Dichos pesos se inicializan de manera aleatoria al comienzo del aprendizaje y varían a lo largo de las iteraciones, de ahí que la red tenga capacidad de aprendizaje. Dada una imagen de entrada I de tamaño $H \times W \times C$ (altura, anchura y canales), y un filtro K de tamaño $k \times k \times C$, la operación de convolución se define como:

$$(I * K)(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \sum_{c=0}^{C-1} I(i+m, j+n, c) \cdot K(m,n,c) \tag{6.1}$$

Esta operación se realiza de manera iterativa deslizando el kernel sobre la imagen de entrada, generando un *mapa de activación* (o *feature map*) que puede resaltar ciertas características como bordes, texturas o patrones espaciales según la naturaleza de la red y la profundidad de la neurona (a mayor profundidad se aprenden conceptos más complejos y concretos).

Tras la convolución se aplica una función de activación no lineal, que es la que introduce la capacidad de aprender representaciones complejas, no lineales, y modelar relaciones sofisticadas en los datos. Normalmente la más empleada en literatura es la función ReLU (Rectified Linear Unit), definida como:

$$\text{ReLU}(x) = \text{máx}(0, x) \tag{6.2}$$

Las capas de *pooling* tienen como objetivo reducir la dimensionalidad, reduciendo los mapas de activación y el número de parámetros para disminuir la complejidad y evitar el sobreajuste. La más común es el *max-pooling*, que para una ventana de tamaño $p \times p$, toma el valor máximo en dicha región:

$$y_{i,j} = \underset{m,n \in [0,p-1]}{\text{máx}} x_{i+m,j+n} \tag{6.3}$$

Este proceso preserva las características más relevantes y otorga a la red cierta invariancia frente a pequeñas traslaciones o distorsiones.

En arquitecturas modernas, es común emplear técnicas de normalización como *Batch Normalization* o normalización por lotes. Este concepto fue introducido por Ioffe y Szegedy en 2015 [36] y consiste en normalizar las activaciones de una capa para tener media nula y varianza unitaria, estabilizando y acelerando el entrenamiento:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{6.4}$$

donde ϵ es un pequeño valor positivo constante que se añade para evitar divisiones por cero y μ_B y σ_B son la media y desviación estándar del canal en el minibatch que se calculan como:

$$\mu_B = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{6.5}$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{6.6}$$

Finalmente, se emplean los parámetros aprendibles β y γ para desplazar las activaciones normalizadas mediante la siguiente fórmula:

$$y_i = \gamma \hat{x}_i + \beta \tag{6.7}$$

Por otro lado, la *Layer Normalization* (LayerNorm) es una técnica de normalización introducida por Ba et al. (2016) [2] como alternativa a la normalización por lotes, especialmente útil en contextos donde los tamaños de lote pueden variar o ser pequeños.

A diferencia de la *Batch Normalization*, que normaliza a lo largo del eje del lote, la *Layer Normalization* normaliza los valores de activación a lo largo de las características del propio ejemplo individual. Dado un vector de activaciones $\mathbf{x} = [x_1, x_2, ..., x_H]$ para una instancia con H características, LayerNorm realiza la siguiente transformación:

$$\mu = \frac{1}{H}\sum_{i=1}^{H} x_i, \quad \sigma = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(x_i - \mu)^2}$$

$$\text{LayerNorm}(x_i) = \gamma_i \cdot \frac{x_i - \mu}{\sigma + \epsilon} + \beta_i$$

donde γ_i y β_i son parámetros aprendibles que permiten a la red restaurar la capacidad representativa tras la normalización, y ϵ es un pequeño valor constante añadido para evitar divisiones por cero.

Para evitar el sobreajuste también se aplican estrategias como Dropout, que desactiva aleatoriamente un porcentaje de neuronas durante el entrenamiento.

Tras las capas convolucionales y de pooling, es habitual emplear una o varias capas densamente conectadas (*fully connected*, FC) donde cada neurona está conectada a todas las activaciones anteriores. Esta parte actúa como un clasificador tradicional, transformando las características extraídas por las capas anteriores en una predicción final.

El proceso de entrenamiento de una CNN se realiza mediante el algoritmo de *back-propagation* o propagación hacia atrás, que calcula el gradiente del error con respecto a cada parámetro de la red utilizando la regla de la cadena. Estos gradientes se utilizan para actualizar los pesos mediante algoritmos de optimización como SGD o Adam.

El error se mide mediante la función de pérdida, habitualmente la entropía cruzada en tareas de clasificación multiclase:

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic} \log \hat{y}_{ic} \tag{6.8}$$

donde y_{ic} es la etiqueta real y $\hat{y}_{ic}$ la probabilidad predicha para la clase c del ejemplo presentado i.

El éxito de las CNN ha llevado al diseño de arquitecturas más profundas y eficientes como VGGNet, GoogLeNet, ResNet, EfficientNet, entre otras, que introducen innovaciones como bloques residuales, conexiones de salto y escalado compuesto.

6.1. Redes Residuales (ResNet)

Las Redes Neuronales Residuales (ResNet), introducidas por He et al. en 2015 [28], suponen un hito en el desarrollo del DL. A diferencia de redes tradicionales, ResNet logra entrenar arquitecturas con cientos o incluso miles de capas mediante un mecanismo conocido como conexiones residuales.

El principal obstáculo para entrenar redes muy profundas es el problema de la degradación del rendimiento: a partir de cierta profundidad, añadir más capas no mejora la precisión e incluso puede empeorarla debido a la dificultad de propagar gradientes. Esto se debe a lo que se conoce como el problema de la evanescencia del gradiente, un suceso que tiene lugar cuando, debido al alto número de capas, las neuronas de las capas profundas se ven muy ligeramente afectadas por las salidas de las capas anteriores. ResNet se propone resolver este problema introduciendo conexiones hacia atrás (*skip connection*) que permiten que el flujo de gradientes atraviese la red sin degradarse:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x} \tag{6.9}$$

donde $\mathbf{x}$ es la entrada, $\mathcal{F}(\mathbf{x})$ representa el bloque residual compuesto normalmente por dos o tres capas convolucionales y la suma es una operación elemento a elemento. Esta formulación permite que la red aprenda una función residual, en lugar de la transformación completa.

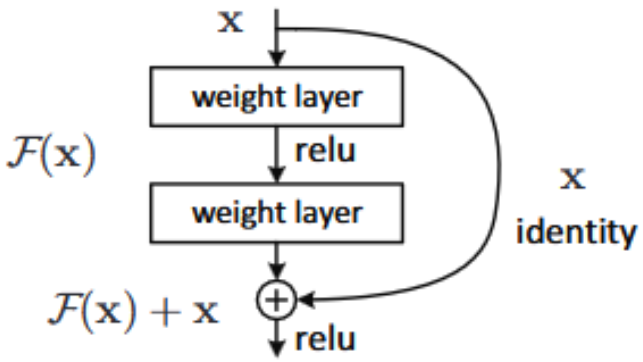


Figura 6.3: Estructura de un bloque residual básico. Fuente: [27]

Esta estructura modular permite el diseño de redes muy profundas, como ResNet18, ResNet34, ResNet50, ResNet101 o ResNet152, siendo las versiones más utilizadas ResNet18 y ResNet50.

ResNet18 es una versión relativamente ligera que consiste en 18 capas con parámetros entrenables, organizadas en 8 bloques residuales. Es adecuada para tareas de clasificación con conjuntos de datos medianos o para aplicaciones con recursos computacionales limitados.

ResNet50, en la misma línea, incrementa la profundidad a 50 capas haciendo uso de bloques *bottleneck*, compuestos por tres capas en lugar de dos. Esta arquitectura es más potente y obtiene mejores resultados en tareas complejas, como se puede ver en la Figura 6.4 procedente de [59], aunque a costa de un mayor consumo computacional.

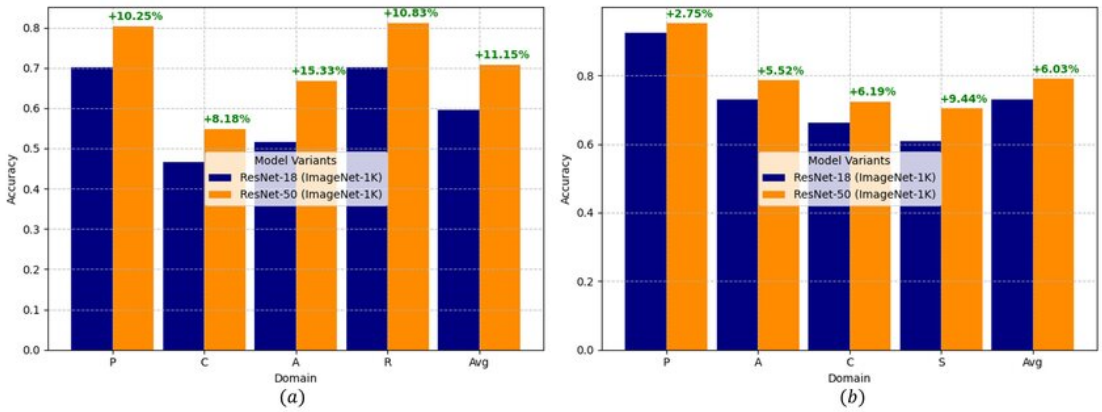


Figura 6.4: Análisis comparativo de las variantes de ResNet (ResNet-18 frente a ResNet-50) (a) conjunto de datos Office-Home [76] (b) conjunto de datos PACS [46]

Las principales ventajas de ResNet incluyen:

- **Facilidad de entrenamiento**: gracias a las conexiones residuales se evita el problema de la evanescencia del gradiente en redes profundas.
- **Generalización**: las redes residuales muestran una gran capacidad de generalización a nuevos conjuntos de datos.
- **Modularidad**: los bloques residuales pueden ensamblarse para construir arquitecturas de distinta profundidad.

PyTorch proporciona implementaciones preentrenadas de ResNet dentro del módulo `torchvision.models`. En el listado 6.1 se muestra un ejemplo básico para cargar la red y modificar la capa de salida para ajustarla al problema de clasificación binaria:

Listado 6.1: Carga de ResNet18 preentrenada con PyTorch

```
import torchvision.models as models
import torch.nn as nn

class ResNet18Binary(nn.Module):
    def __init__(self):
        super(ResNet18Binary, self).__init__()
        # Carga del modelo preentrenado
        self.resnet = models.resnet18(weights="DEFAULT")
        # Se congelan todos los parametros entrenables
        # excepto los de la ultima capa
        # y la completamente conectada
        for name, param in self.resnet.named_parameters():
            if "layer4" not in name or "fc" not in name:
                param.requires_grad = False

        # Modificar la capa de pooling y la capa fully connected
        # Reduce la dimensionalidad
        self.resnet.avgpool = nn.AdaptiveAvgPool2d(output_size=(1, 1))
        num_ftrs = self.resnet.fc.in_features
        self.resnet.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(num_ftrs, 256),  # Reduccion de dimensiones
            nn.BatchNorm1d(256),  # Normalizacion para estabilidad
            nn.ReLU(),
            nn.Dropout(0.3),  # Evita sobreajuste
            nn.Linear(256, 128),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(128, 1),  # Capa de salida
        )

    def forward(self, x):
        return self.resnet(x)
```

Para ResNet50, la estructura es similar, pero cambiando el modelo seleccionado y las capas que se congelan. Ambos modelos se adaptan fácilmente a nuevas tareas mediante *fine-tuning*, permitiendo entrenar solo las últimas capas mientras se aprovechan los pesos aprendidos previamente.

6.2. EfficientNet

La familia de modelos EfficientNet, propuesta por Tan y Le en 2019 [74], representa un avance significativo en la optimización de las CNNs tanto en precisión como en eficiencia. A diferencia de enfoques tradicionales que escalan únicamente en profundidad o ancho, EfficientNet introduce un método sistemático llamado *compound scaling* para escalar en tres dimensiones clave: *depth*, *width* y *resolution* (Figura 6.5).

La motivación principal detrás de EfficientNet es mejorar el rendimiento sin sacrificar precisión. Primero se desarrolló una arquitectura base (EfficientNet-B0) utilizando un algoritmo de búsqueda de arquitectura neuronal (*Neural Architecture Search*) y posteriormente escalaron esta red de forma compuesta para crear versiones con mayor potencia (EfficientNet-B1 a EfficientNet-B7).

El método de *compound scaling* emplea un coeficiente ϕ para escalar las tres dimensiones de manera uniforme:

$$\begin{aligned} \text{depth:} \quad & d = \alpha^\phi \\ \text{width:} \quad & w = \beta^\phi \\ \text{resolution:} \quad & r = \gamma^\phi \end{aligned} \tag{6.10}$$

sujeto a la restricción $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, donde α, β, γ son constantes determinadas por búsqueda empírica.

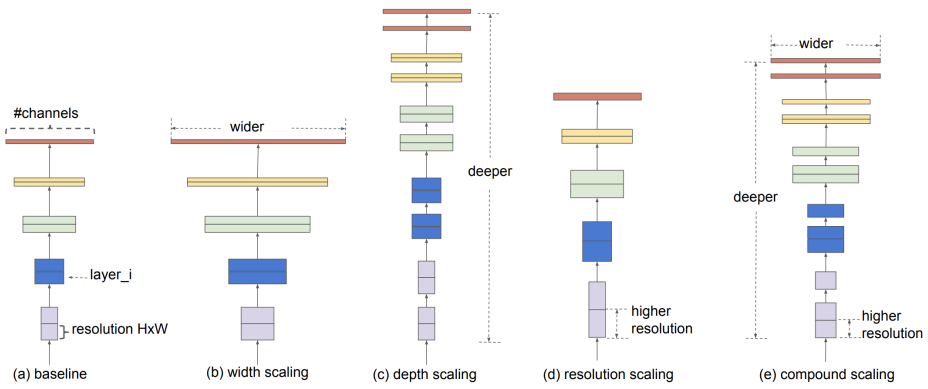


Figura 6.5: Esquema del escalado compuesto en EfficientNet. Fuente: [74]

Además del escalado compuesto, EfficientNet incluye bloques modulares conocidos como *MBConv* (Mobile Inverted Bottleneck Convolution), una evolución de los bloques utilizados originalmente en MobileNetV2 (que se introducirá más adelante). Estos bloques combinan varias estrategias para mejorar la eficiencia y la capacidad expresiva de la red.

En primer lugar, el término *inverted bottleneck* hace referencia a un patrón arquitectónico donde se expande la dimensionalidad del espacio de características antes de aplicar convoluciones y se reduce nuevamente al final del bloque. A diferencia de otros bloques tipo *bottleneck* donde se reduce la dimensionalidad antes de realizar las operaciones más costosas, en un bloque invertido se hace lo contrario: se proyecta a una dimensión mayor para poder realizar las operaciones no lineales en un espacio latente más rico y se reduce de nuevo la dimensionalidad mediante una proyección lineal. Esto permite conservar la información crítica de la entrada, a la par que optimizar el coste computacional.

El segundo componente clave de los bloques *MBConv* es la *depthwise separable convolution*, una técnica que separa la operación de convolución en dos fases: una convolución *depthwise* y una *pointwise*. La convolución *depthwise* aplica un filtro independiente a cada canal de entrada, sin mezclar información entre canales, lo que reduce el número de operaciones. Por otro lado, la convolución *pointwise* (1x1) se encarga de combinar la información de todos los canales. Con esto se logra reducir significativamente los parámetros y el coste computacional.

Por último, cada bloque *MBConv* incorpora una operación adicional denominada *squeeze-and-excitation* (SE), propuesta originalmente por Hu et al. [31]. Esta técnica propone una forma de atención canal a canal, con el fin de que la red pueda aprender qué canales son más relevantes para cada tarea. La operación consta de tres pasos: *squeeze*, que aplica un *global average pooling* para resumir la información global por canal; *excitation*, que emplea una red neuronal pequeña (MLP) para generar pesos y *recalibration*, que multiplica los canales originales por estos pesos nuevos. Este mecanismo mejora la sensibilidad de la red a características discriminativas y mejora el rendimiento sin aumentar significativamente el coste computacional.

La combinación de estas tres técnicas dota a la red de eficiencia y expresividad. En el contexto de EfficientNet, estos bloques son utilizados como bloques constructivos básicos en todas las etapas de la red, optimizando la capacidad de aprendizaje y minimizando los recursos requeridos.

Entre las principales ventajas de EfficientNet se encuentran:

- **Eficiencia computacional**: logra resultados competitivos con menor número de parámetros y operaciones en punto flotante (*Floating Point Operations* o FLOPs) comparado con otras arquitecturas como ResNet.
- **Escalabilidad**: el escalado compuesto permite adaptar la arquitectura a distintos recursos computacionales.
- **Versatilidad**: aplicable tanto a dispositivos embebidos (B0–B2) como a entornos de alto rendimiento (B6–B7).

A través del módulo `torchvision.models` (o bien `efficientnet-pytorch`), se pueden cargar modelos EfficientNet preentrenados, empleando una estructura de código similar a la usada con ResNet.

6.3. MobileNetV2

MobileNet [30] es una arquitectura de red más sencilla diseñada específicamente para dispositivos con recursos limitados, como smartphones o sistemas embebidos (o dispositivos ponibles). Su principal característica es la utilización de convoluciones separables en profundidad (*depthwise separable convolutions*, Figura 6.6) similares a las realizadas por los bloques MBConv presentados en EfficientNet, que permiten reducir drásticamente el número de parámetros y la complejidad, sin poner en riesgo la precisión en tareas de clasificación.

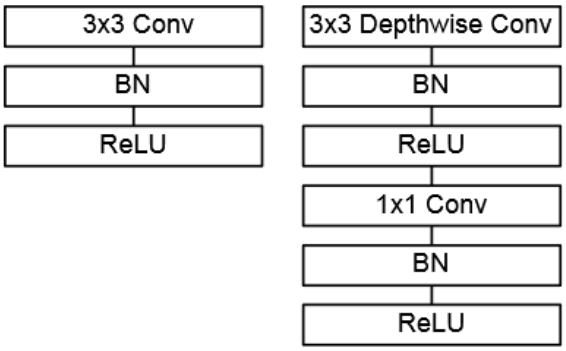


Figura 6.6: Izquierda: Capa convolucional estándar. Derecha: convoluciones separables en profundidad con ambas capas *Depthwise* y *Pointwise* . Fuente: [30]

La descomposición de la convolución estándar en dos etapas reduce significativamente el número de operaciones requeridas. Matemáticamente, si se considera una convolución estándar con un kernel de tamaño $k \times k$ y M canales de entrada y N canales de salida, el coste computacional es:

$$C_{\text{std}} = k \times k \times M \times N \times H \times W$$

donde H y W son las dimensiones espaciales de la entrada. En cambio, utilizando una convolución separable en profundidad, el coste se divide en:

$$C_{\text{depthwise}} = k \times k \times M \times H \times W$$

$$C_{\text{pointwise}} = M \times N \times H \times W$$

y el coste total es:

$$C_{\text{separable}} = C_{\text{depthwise}} + C_{\text{pointwise}}$$

lo que demuestra una reducción significativa en la complejidad operacional, especialmente cuando k es mayor que 1.

MobileNetV2 [67] surge como evolución directa de MobileNet [30], manteniendo su enfoque en la eficiencia. Mientras que la primera versión introdujo las convoluciones separables en profundidad como alternativa eficiente a las convoluciones tradicionales, MobileNetV2 incorpora ciertas mejoras en la arquitectura que permiten aumentar la transmisión de la información y alcanzar un uso aún más eficiente de los parámetros del modelo.

La primera versión de MobileNet logró reducir significativamente la complejidad de los modelos CNN; sin embargo, presentaba ciertas limitaciones. La separación espacial y de canal puede provocar pérdida de información y, consecuentemente, de poder representacional. Además, el diseño no favorece la propagación eficiente del gradiente a través de muchas capas, especialmente en arquitecturas con gran número de ellas.

Para superar estas limitaciones, MobileNetV2 introduce dos innovaciones clave:

1. **Bloques invertidos con conexiones residuales** (*Inverted Residuals*): Similar a lo presentado en EfficientNet, en lugar del patrón clásico de *bottleneck* en redes como ResNet, MobileNetV2 lo invierte expandiendo primero la dimensionalidad (a través de una convolución 1x1), aplicando la convolución separable en profundidad y finalmente proyectando de nuevo a una dimensión menor, lo que permite una mayor capacidad de representación con menos parámetros (Figura: 6.7).

2. **Capas lineales sin activación al final del bloque**: Para evitar la pérdida de información en espacios de dimensionalidad baja, se eliminan las funciones de activación (ReLU) en la última capa de cada bloque, antes de la proyección. Esto se debe a que aplicar la función ReLU tras reducir la dimensionalidad puede destruir información útil, especialmente en tareas de clasificación donde se requiere una representación densa.

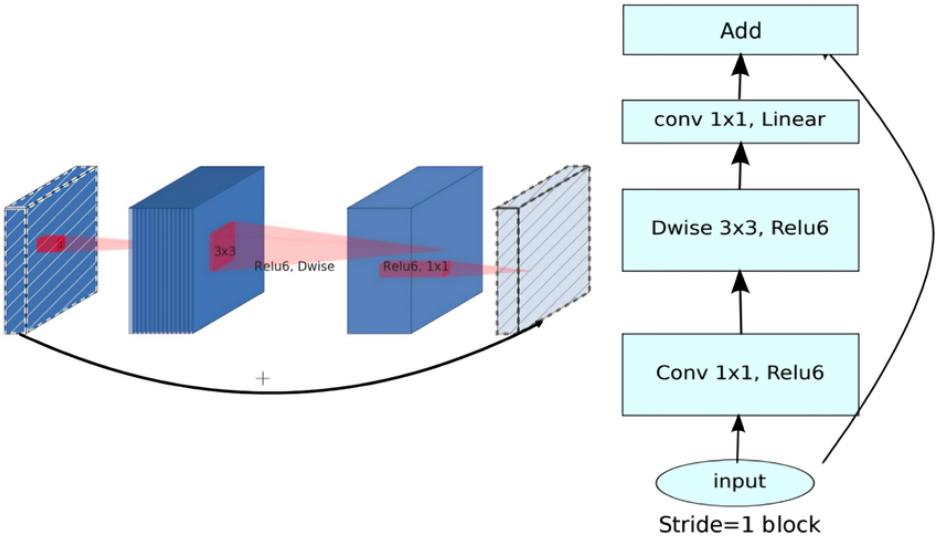


Figura 6.7: Estructura interna de un bloque *Inverted Residual* de MobileNetV2. Fuente: [71].

Estas mejoras permiten a MobileNetV2 mejorar la precisión con respecto a MobileNetV1 sin comprometer el coste computacional y profundizar la red sin penalizar la capacidad de aprendizaje ni dificultar la propagación del gradiente. Todo ello conservando la eficiencia en dispositivos con capacidades limitadas.

En la práctica, MobileNetV2 logra resultados competitivos en benchmarks como ImageNet, manteniendo una fracción del número de parámetros y operaciones requeridas por modelos convencionales.

Entre las principales ventajas de MobileNet destacan:

- **Eficiencia en el uso de recursos**: Se trata de una arquitectura optimizada para funcionar en dispositivos móviles y entornos con limitaciones de hardware.
- **Bajo número de parámetros**: La estrategia de convoluciones separables permite reducir la cantidad de parámetros.
- **Flexibilidad**: MobileNet puede ser ajustada mediante un hiperparámetro de anchura (width multiplier) y un factor de resolución, lo que permite un alto grado de personalización.

La implementación de MobileNetV2 en PyTorch se facilita gracias a la disponibilidad de modelos preentrenados en `torchvision`.

6.4. Arquitectura de Red Propuesta

El diseño de red que se propone en esta sección ha sido diseñado específicamente para abordar el problema que se presenta en este trabajo a partir de los conocimientos obtenidos durante la carrera, la investigación realizada y varias iteraciones de depuración. La red se implementa en PyTorch y se compone de varios bloques convolucionales, seguidos de una capa de pooling adaptativo y una sección de clasificación completamente conectada. A continuación se describe detalladamente cada uno de los componentes y su relevancia en el contexto del proyecto.

La red se define mediante una clase genérica `Network`, que hereda de `nn.Module`, la clase base a partir de la que se construyen todos los módulos de redes neuronales. Su estructura se puede dividir en dos secciones principales: la extracción de características y el clasificador.

6.4.1. Extracción de Características

La sección de extracción de características se implementa mediante un bloque secuencial denominado *features* que consta de tres sub-bloques principales:

Bloque 1:

- Dos capas convolucionales con filtros de tamaño 3×3, cada una seguida de una capa de *Batch Normalization* y una función de activación *ReLU*. La normalización por lotes estabiliza y acelera el entrenamiento, permitiendo que la red aprenda de manera más robusta.
- Una operación de *MaxPooling* con kernel de 2×2 y stride 2, que reduce las dimensiones espaciales a la mitad y resalta las características más prominentes.
- Se aplica un *Dropout* del 25 % para prevenir el sobreajuste, forzando a la red a aprender representaciones redundantes.

Bloque 2:

- Dos capas convolucionales, nuevamente acompañadas de Batch Normalization y ReLU, lo que permite capturar patrones más complejos a medida que aumenta la profundidad.
- MaxPooling con el mismo parámetro, reduciendo la resolución y manteniendo la información relevante.
- Un *Dropout* del 30 % se incorpora para aumentar la robustez del modelo.

Bloque 3:

- Se utilizan dos capas convolucionales con 128 canales, incrementando la capacidad de la red para extraer características de alto nivel.
- La normalización y activación se mantienen constantes, seguida de una reducción espacial mediante MaxPooling.
- Se aplica un *Dropout* del 40 %, lo que es especialmente útil en las capas profundas donde la probabilidad de sobreajuste aumenta.

6.4.2. Pooling Adaptativo

Una vez extraídas las características mediante los bloques anteriores, se utiliza una capa de *Adaptive Average Pooling* (con salida fija de 7×7) para permitir que la red acepte imágenes de entrada de dimensiones variables, generando una representación de tamaño fijo. Matemáticamente, si $X \in R^{C \times H \times W}$ es la salida del bloque de características, la operación de pooling se define como:

$$Y_c(i,j) = \frac{1}{|R_{ij}|} \sum_{(p,q) \in R_{ij}} X_c(p,q)$$

donde R_{ij} es la región de entrada mapeada a la posición (i,j) en la salida.

6.4.3. Clasificador

La parte final de la red es el clasificador, compuesto por: una capa lineal que transforma la representación a un vector de 512 dimensiones, seguida de una activación ReLU para introducir no linealidad; un Dropout del 50 % que actúa como regularizador en esta etapa crítica y una capa lineal final con salida de una dimensión, que se utiliza para la clasificación binaria (genuino vs. impostor). Dado que se trata de un problema de verificación, la salida puede pasarse por una función sigmoide que realiza una transformación para interpretarla como un *score* o probabilidad, sobre la cual se aplica un umbral para la decisión final.

La red ha sido diseñada con el objetivo de lograr un equilibrio entre la capacidad de representación y la eficiencia, aspectos cruciales en aplicaciones de biometría en tiempo real. El diagrama completo de la red se presenta en la Figura 6.8. Los aspectos destacados son:

1. **Profundidad y Capacidad de Extracción de Características:** La red incrementa progresivamente el número de canales (32, 64 y 128) en cada bloque, lo que permite aprender representaciones cada vez más complejas y abstractas. Esta característica es esencial para capturar sutilezas.

2. **Regularización:** La incorporación de capas de Dropout en cada bloque ayuda a mitigar el riesgo de sobreajuste, especialmente importante dado que el conjunto de datos puede tener variaciones significativas entre usuarios y sesiones. Además, el uso de normalizacion por lotes estabiliza el proceso de entrenamiento y permite utilizar tasas de aprendizaje más arriesgadas.

3. **Pooling Adaptativo:** El uso de Adaptive Average Pooling reduce la dimensionalidad, y permite trabajar con imágenes de diferentes tamaños.

4. **Clasificación Binaria:** La salida de la red se reduce a una dimensión, adaptándose al enfoque de verificación.

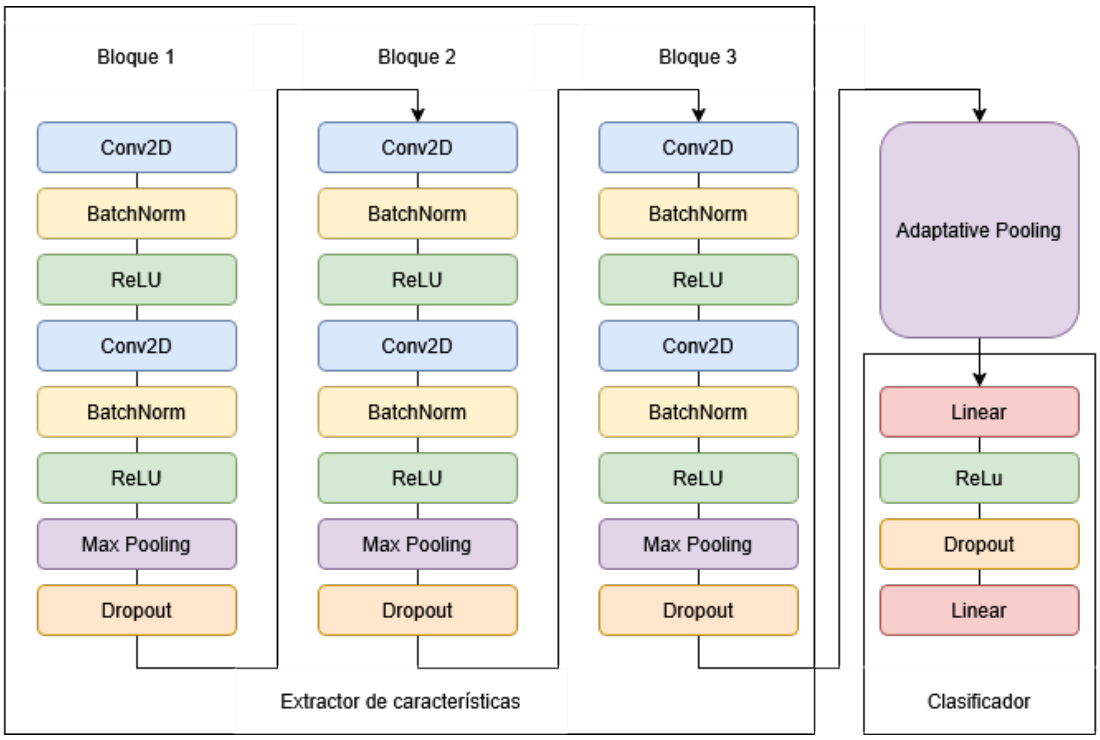


Figura 6.8: Diagrama de la arquitectura de red propupesta

Capítulo 7

Pruebas

Para garantizar la corrección y la efectividad del desarrollo software en este TFG, se han llevado a cabo dos tipos de pruebas independientes pero complementarias: las asociadas a la generación de las imágenes, y las asociadas a la evaluación de las arquitecturas. A continuación, se describen en detalle ambas fases de prueba.

7.1. Pruebas durante la generación de imágenes

La fase de generación de las imágenes constituye un punto crítico, ya que cualquier error en el preprocesamiento o en la transformación de la señal comprometería toda la evaluación posterior. Para verificarla, se aplicaron las siguientes pruebas:

1. **Inspección de DataFrames intermedios.**
   - Se imprimieron por consola los primeros y últimos registros de cada `DataFrame` tras la lectura y antes de la normalización, para comprobar que los valores de cada columna correspondían a los esperados.
   - Tras aplicar la interpolación y normalización, se volvieron a imprimir muestras para asegurar que los rangos de valores se habían escalado correctamente y no existen valores faltantes o negativos.
2. **Visualización de series temporales.**
   - Se generaron gráficos de líneas de las tres componentes antes y después de limpieza para verificar que los picos y valles de la marcha se mantenían coherentes.
   - Se guardaron ejemplos de estos gráficos como PNG y se inspeccionaron manualmente para detectar artefactos o desplazamientos temporales incorrectos.

3. **Generación y almacenamiento de imágenes.**
    - En cada iteración de la función donde se generan las imágenes, se muestra por pantalla un subconjunto reducido de ellas, para comprobar visualmente que el mapa de color se aplica correctamente y las imágenes no presentan artefactos inesperados.
    - Al finalizar la ejecución completa, se listaron los archivos generados en el directorio de salida y se comparó el número de imágenes esperadas contra las efectivamente creadas, asegurando que no hay pérdidas ni duplicados inesperados.
4. **Pruebas de robustez ante datos corruptos.**
    - Se introdujeron manualmente algunos ficheros con `Timestamp` negativos o con filas incompletas, comprobando que el código detecta los valores negativos y los elimina correctamente y no se detiene ante archivos malformados, sino que informa por consola y continúa procesando el resto.

7.2. Pruebas durante la evaluación de las arquitecturas

Una vez generadas las imágenes, se validó la fase de entrenamiento y evaluación de modelos CNN aplicando las siguientes comprobaciones:

1. **Validación de la carga de modelos.**
    - Se utilizó la función `summary` (de `torchsummary`) para imprimir por pantalla la arquitectura completa de cada red, incluyendo número de capas, parámetros entrenables y formas de los tensores de entrada/salida de cada bloque.
    - Se comprobó que las capas finales sustituidas para clasificación binaria tenían las dimensiones correctas.
2. **Monitorización de pérdidas y métricas por época.**
    - Durante el entrenamiento se imprimió en cada época el valor promedio de la función de pérdida de cada lote, permitiendo verificar la tendencia decreciente y detectar rápidamente oscilaciones o divergencias.
    - Tras cada época, se invoca la función de evaluación y se muestra en consola el `Accuracy`, el `EER` y el `AUC`, comprobando que las métricas son coherentes.

3. **Balanceo de clases en los conjuntos de datos.**

   - Se imprimieron recuentos de etiquetas positivas y negativas en los conjuntos antes de comenzar el entrenamiento, para garantizar un reparto equilibrado entre muestras de usuario genuino y muestras de impostor.
   - Se comprobó que, tras aplicar las estrategias de replicación de ejemplos de la clase minoritaria, la proporción se mantenía según lo especificado.

4. **Comprobación de forma de tensores y salidas de red.**

   - En la fase de entrenamiento se añadieron comprobaciones en puntos clave para verificar que las dimensiones de los tensores coincidían con lo esperado tras cada bloque (por ejemplo, antes y después de la capa de *flatten* y del clasificador).
   - Se inspeccionaron manualmente las primeras salidas de las redes para confirmar que los *logits* producidos están en un rango razonable y la conversión a probabilidades y etiquetas binarias funciona correctamente al aplicar el umbral seleccionado.

5. **Tests unitarios de componentes críticos.**

   - Se redactaron pequeños tests unitarios en `pytest` para verificar que la función de cálculo de EER devuelve resultados simétricos (EER(FAR,FRR) = EER(FRR,FAR)), comprobar que los DataLoaders devuelven lotes del tamaño correcto y segurar que, con entradas sintéticas de prueba, las arquitecturas devuelven siempre tensores de forma $[B, 1]$.

Gracias a esta batería de pruebas se verifica la correcta generación de imágenes, la integridad de los datos intermedios y el adecuado funcionamiento de las arquitecturas de red en todas las etapas de entrenamiento y evaluación.

Capítulo 8

Resultados

En este capítulo se presentan y analizan los resultados obtenidos a lo largo de un proceso experimental específicamente diseñado para evaluar la eficacia de las arquitecturas propuestas. El objetivo principal es comparar de manera sistemática el rendimiento de las configuraciones, tipos de representación visual de los datos y sensores utilizados, con el fin de identificar la combinación óptima que maximice la capacidad de discriminación entre usuarios genuinos e impostores.

8.1. Diseño Experimental

El proceso propuesto cuenta con tres fases claramente diferenciadas:

1. **Estudio preliminar**: Se emplea un subconjunto reducido y aleatoriamente seleccionado de usuarios con el objetivo de comparar el comportamiento de cinco arquitecturas distintas, evaluadas sobre tres representaciones visuales diferentes generadas a partir de los datos de los dos sensores disponibles (acelerómetro y giroscopio). Esta fase permite identificar combinaciones poco prometedoras y descartarlas sin incurrir en costes computacionales o temporales elevados.

2. **Evaluación de configuraciones óptimas**: A partir de los resultados del estudio preliminar se seleccionan las dos configuraciones más prometedoras. La primera corresponde a la combinación red + sensor + tipo de imagen que obtiene el mejor rendimiento global. La segunda se construye a partir de las mejores medias individuales: la arquitectura con mejor rendimiento promedio, el sensor con mejor comportamiento medio y la representación visual que genera mejores resultados en términos agregados. Estas configuraciones se evalúan utilizando el conjunto completo de datos.

3. **Optimización de hiperparámetros**: Finalmente, se realiza una serie de experimentos sobre la configuración seleccionada para afinar el modelo final. En esta fase se explora el efecto de distintos hiperparámetros sobre el rendimiento, incluyendo el tamaño de lote, la tasa de aprendizaje, la arquitectura del clasificador final y el empleo de técnicas de regularización como *dropout*. Esta etapa busca no solo mejorar la métrica objetivo, sino también estudiar la sensibilidad del sistema a dichas variaciones.

En todas las fases el rendimiento de los modelos se evalúa utilizando métricas estándar en el campo de la biometría. En particular, se empleará como métrica principal la *Equal Error Rate* (EER) o *tasa de equierror*, ampliamente utilizada en tareas de verificación biométrica. Esta métrica representa el punto de equilibrio entre la tasa de falsas aceptaciones (*False Acceptance Rate*, FAR) y la tasa de falsos rechazos (*False Rejection Rate*, FRR) y se define como el valor en el que ambas tasas son iguales:

$$\text{EER} \equiv \text{FAR}(t^*) = \text{FRR}(t^*) \quad \text{para algún umbral } t^*$$

El valor de EER se obtiene trazando la curva ROC (Receiver Operating Characteristic) del clasificador, y representa una estimación compacta del punto de operación donde los errores tipo I (FAR) y tipo II (FRR) se equilibran. Cuanto más bajo es el valor de la EER, mejor es la capacidad discriminativa del sistema, pues indica que se puede alcanzar un compromiso entre seguridad (evitar accesos no autorizados) y usabilidad (evitar rechazos indebidos) con menor error total.

La elección de la EER como métrica principal está justificada por su independencia del umbral de decisión elegido y porque resume de manera intuitiva el rendimiento del sistema en contextos de verificación uno a uno, donde el balance entre falsas aceptaciones y falsos rechazos es crucial [79]. Además, su uso está estandarizado en evaluaciones biométricas como las realizadas por el NIST o en competiciones internacionales como las organizadas por el IEEE BIOSIG o FVC.

Cabe destacar que el protocolo *mono-session* no se empleará para comparar en ninguno de los experimentos. La dinámica *cross-session* se considera el escenario más interesante desde un punto de vista práctico. Sin embargo, en la segunda etapa se muestran los resultados con *mono-session*, para mostrar los resultados en este caso y ver el deterioro en las pruebas al emplear un protocolo más estricto pero realista.

8.2. Experimento 1: Estudio preliminar

Con el objetivo de reducir la complejidad computacional de las primeras pruebas y validar la viabilidad de las distintas configuraciones del sistema, se ha llevado a cabo un estudio preliminar empleando un subconjunto aleatorio de cinco usuarios de la base de datos.

8.2.1. Configuraciones evaluadas

En este estudio se han considerado tres dimensiones de variación experimental:

- **Modelo de red neuronal**: se han evaluado las cinco arquitecturas convolucionales propuestas: ResNet18, ResNet50, EfficientNet-B0, MobileNetV2 y la CNN de diseño propio (Custom).
- **Tipo de sensor**: los sensores empleados en la adquisición de los datos (acelerómetro - ACC y giróscopo - GYR), proporcionan señales de diferente naturaleza y sensibilidad al movimiento corporal. Por tanto, se analizará por separado el rendimiento de las muestras obtenidas de cada uno de los sensores.
- **Tipo de representación visual**: cada señal ha sido transformada en una representación bidimensional mediante los distintos enfoques gráficos presentados anteriormente.

8.2.2. Metodología

Para cada una de las combinaciones posibles entre arquitectura, tipo de imagen y sensor (un total de $5 \times 3 \times 2 = 30$ configuraciones), se ha entrenado un modelo independiente para cada usuario. La tarea consiste en verificar si una muestra corresponde al usuario objetivo o a un impostor, empleando el conjunto de imágenes generadas a partir de las sesiones de entrenamiento y prueba predefinidas.

8.2.3. Resultados

En la Tabla 8.1 se presentan los resultados obtenidos, expresados como el EER medio por configuración en el subconjunto de usuarios seleccionados. Como ya se ha comentado, los valores más bajos indican un mejor comportamiento del sistema.

	Acc			GYR			
EER(%)	**Color**	**BN**	**Lines**	**Color**	**BN**	**Lines**	**Media**
resnet18	24.6	18.6	21.9	23.1	20.4	19.4	**21.3**
resnet50	26	26.2	16.3	23.2	23.6	19.1	22.4
efficientnet	24.2	28.1	**16.2**	19.3	24.6	21.8	22.4
mobilenet	25.2	26.4	16.6	23.9	27.7	19.8	23.3
custom	30.9	29.2	30.7	30.8	30.4	30.9	30.5
Media	26.2	25.7	**20.4**	24.1	25.4	22.2	24

Tabla 8.1: Resultados preliminares: EER por red, sensor y tipo de representación visual. En negrita se indican los mejores resultados.

8.2.4. Discusión

Como se puede apreciar en la Figura 8.1, los resultados muestran que algunas combinaciones de red y representación ofrecen una ventaja significativa respecto a otras.

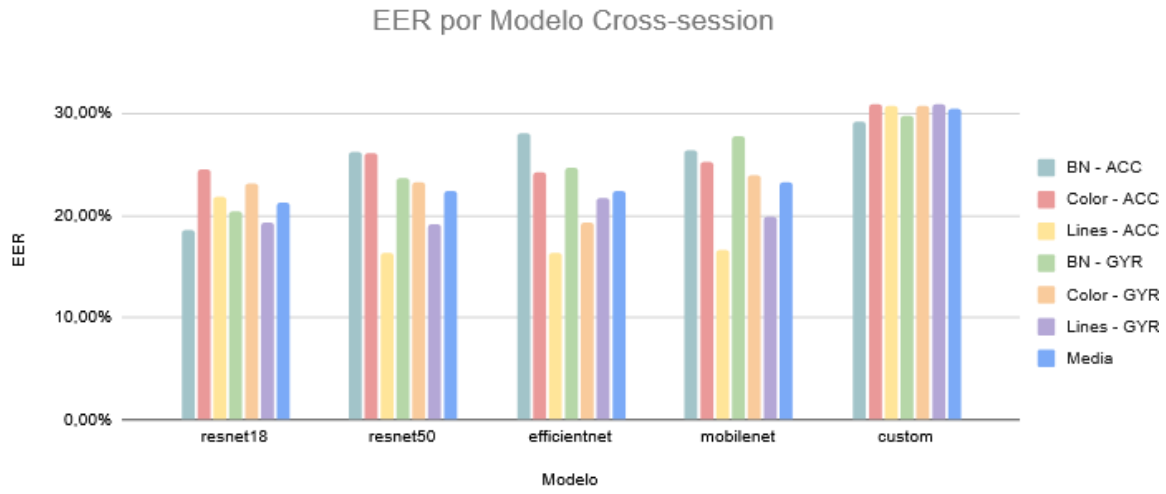


Figura 8.1: EER por arquitectura para cada combinación de sensor y tipo de imágenes

Se pueden extraer algunas conclusiones de estos datos. Por ejemplo, para casi todos los casos, las imágenes de tipo gráfico de líneas obtienen resultados significativamente mejores que los otros dos, obteniendo una media total de 21,3 % frente al 24,6 % que obtienen las imágenes en blanco y negro y al 25,2 % que se logra con los mapas en color.

El giroscopio es de media mejor que el acelerómetro (23,9 % frente a 24,1 %), sin embargo la diferencia es muy pequeña. La red que mejor resultado promedio obtiene es, con diferencia, resnet18. La arquitectura que peores resultados obtiene es, consistentemente, la CNN custom.

8.3. Experimento 2: Estudio comparativo

Una vez completado el estudio preliminar, se procede a identificar las dos configuraciones más prometedoras y a evaluarlas a gran escala sobre el conjunto completo de usuarios de la base de datos. Esta fase tiene como objetivo validar la robustez y generalización de las configuraciones seleccionadas, así como confirmar que los patrones observados en el subconjunto inicial se mantienen cuando se consideran todos los sujetos. Además, permitirá seleccionar la mejor combinación de datos y red.

8.3.1. Criterios de selección

De acuerdo con los resultados mostrados en la Tabla 8.1, la configuración que alcanzó el mejor rendimiento corresponde al modelo EfficientNet, utilizando como entrada imágenes de tipo gráfico de líneas generadas a partir del sensor acelerómetro (ACC). Esta combinación obtuvo un EER medio de 16,2 %, el valor más bajo entre todas las evaluadas.

Adicionalmente se identificaron las combinaciones con mejor rendimiento medio por cada dimensión del espacio experimental:

- **Red con mejor media**: ResNet18 (*EER media global: 21,3 %*)
- **Sensor con mejor media**: Giróscopo (GYR) (*EER media: 23,8 %*)
- **Tipo de imagen con mejor media**: Gráfico de Líneas (*EER media: 21,3 %*)

Estos datos motivan la evaluación de dos configuraciones clave: la mejor combinación puntual, EfficientNet + ACC + Líneas; y la combinación formada por los mejores elementos individuales, ResNet18 + GYR + Líneas.

8.3.2. Resultados

Para evaluar esta configuración óptima, se ha entrenado un modelo para cada usuario de la base de datos, siguiendo el protocolo de verificación empleado anteriormente. Además, se incluyen los resultados de la evaluación *mono-session*, simulando la situación en la que se toma una sola muestra del usuario y se autentica en la misma sesión con una segunda muestra. La Tabla 8.2 reúne los resultados de EER obtenidos por cada configuración por usuario.

8.3.3. Discusión

Los resultados recogidos en la Figura 8.2 permiten extraer algunas comparativas entre las dos arquitecturas.

A nivel agregado, EfficientNet obtiene un EER medio de 16.8 % en protocolo *mono-sesion* y 26.2 % en protocolo *cross-sesion*, mientras que ResNet18 alcanza 18.5 % y 26.0 % respectivamente. Estas cifras indican que, en el contexto *mono-sesion*, EfficientNet supera en rendimiento a ResNet18 con una mejora absoluta de 1.7 puntos porcentuales, lo que representa una mejora relativa del 9.2 %. Por otro lado, en el protocolo *cross-sesion*, las diferencias son más reducidas, siendo ligeramente favorable a ResNet18 por apenas 0.2 puntos porcentuales, lo que no resulta significativo dada la variabilidad interusuario.

Esta diferencia sugiere que EfficientNet presenta mejor capacidad de adaptación y generalización cuando el entorno de prueba es más controlado, mientras que en entornos más realistas, ambas redes muestran comportamientos comparables.

La variabilidad del EER por usuario es considerable para ambas arquitecturas y protocolos, con valores que oscilan entre el 7.3 % y el 52.4 %. Esta dispersión podría atribuirse a diversos factores, como diferencias individuales en el patrón de marcha.

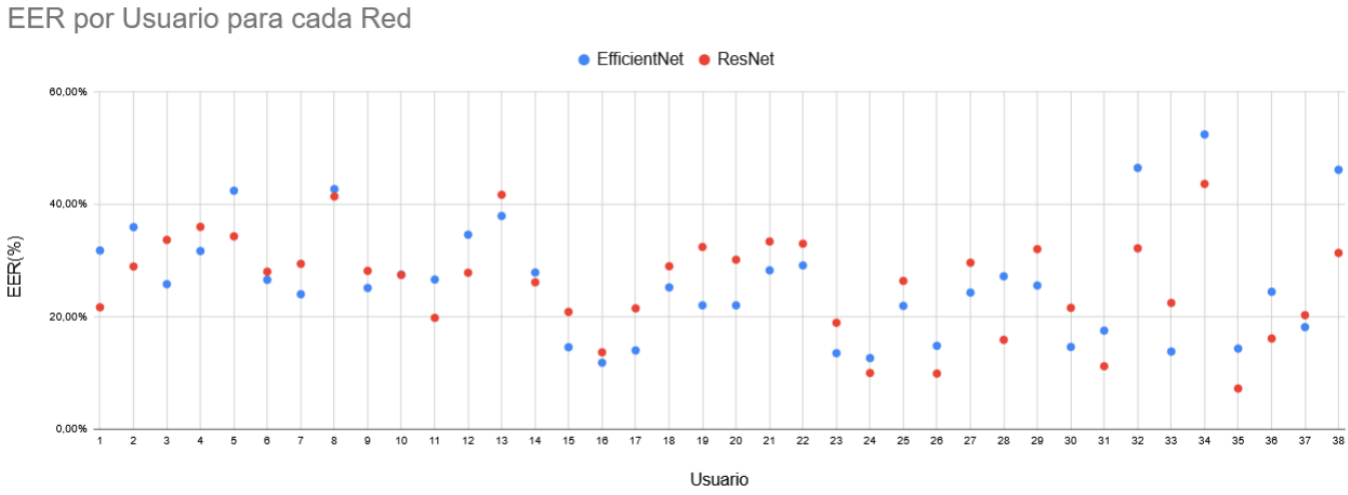


Figura 8.2: EER por usuario para cada red

Pese a esta variabilidad, se observa que EfficientNet mantiene un comportamiento más robusto frente a valores extremos de EER en *mono-sesion*, mientras que ResNet18 presenta una dispersión más elevada.

Tras analizar los resultados recogidos , se ha determinado que la configuración más adecuada para el estudio final es el uso de EfficientNet en combinación con el acelerómetro. Aunque los resultados de la dinámica *mono-sesion* resulten más prometedores, las pruebas finales se realizarán empleando el escenario de sesión cruzada por ser más realista.

Por tanto, se adopta esta configuración como punto de partida para la siguiente fase del proyecto, en la que se procederá a la optimización exhaustiva de hiperparámetros, con el objetivo de refinar aún más el rendimiento del sistema.

Usuario	**Cross Session**		**Mono Session**	
EER (%)	**EfficientNet**	**ResNet18**	**EfficientNet**	**ResNet18**
1	31.8	21.7	10.9	13.6
2	36.0	29.0	18.3	18.1
3	25.8	33.7	11.8	19.4
4	31.7	35.9	9.9	7.6
5	42.4	34.3	20.9	14.2
6	26.6	28.0	11.6	20.5
7	24.0	29.4	26.9	25.0
8	42.7	41.4	16.2	21.3
9	25.1	28.2	15.7	19.5
10	27.5	27.5	12.8	12.1
11	26.6	19.8	15.6	15.1
12	34.6	27.8	22.9	16.7
13	37.9	41.7	14.8	10.8
14	27.9	26.1	20.5	25.4
15	14.6	20.9	12.3	20.2
16	11.8	13.7	13.0	12.7
17	14.0	21.5	8.6	11.8
18	25.2	29.0	32.7	25.4
19	17.8	32.4	17.9	24.6
20	22.1	30.1	18.0	19.7
21	28.3	33.4	16.9	8.8
22	29.1	33.0	12.5	24.9
23	13.5	18.9	10.6	10.6
24	12.7	10.0	—	—
25	22.0	26.9	—	—
26	14.4	9.9	—	—
27	24.3	29.6	22.5	20.9
28	27.2	15.9	14.7	19.3
29	25.6	32.0	15.8	19.9
30	14.6	21.6	14.1	10.2
31	17.6	11.2	11.3	10.6
32	46.5	32.2	23.5	24.2
33	13.8	22.5	9.6	22.7
34	52.4	43.6	37.4	5.7
35	14.3	7.3	—	—
36	24.5	16.2	16.7	16.7
37	18.2	20.3	—	—
38	46.1	31.3	—	—
Media	**26.2**	**26**	**16.8**	**18.5**

Tabla 8.2: Evaluación de EER por usuario. Los valores faltantes corresponden a usuarios con una sola muestra para la primera sesión.

8.4. Experimento 3: Estudio de hiperparámetros

Para este último experimento se ha realizado un estudio sistemático evaluando el rendimiento de la arquitectura seleccionada bajo distintas configuraciones de número de capas del clasificador, tamaño de la primera capa oculta, tamaño del lote y tasa de aprendizaje.

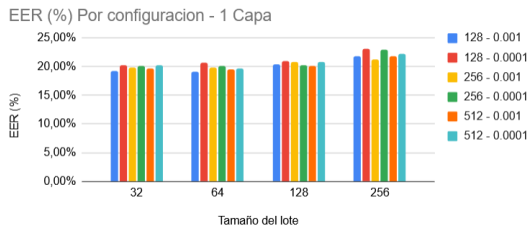
Esto da lugar a un total de 48 configuraciones únicas, evaluadas en 10 usuarios seleccionados aleatoriamente bajo protocolo *cross-session* con el sensor de acelerómetro y gráficos de líneas. Los resultados más relevantes se resumen en la tabla 8.3

Capas	Neuronas 1ª capa	Lote	LR	EER Medio (%)
2	512	32	0.001	19.1
1	128	64	0.001	19.2
2	256	32	0.001	19.3
1	128	32	0.001	19.3
2	128	32	0.001	19.4
2	256	64	0.001	19.4

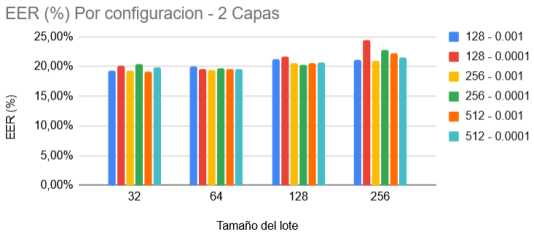
Tabla 8.3: Configuraciones óptimas según EER medio.

8.4.1. Discusión

Se incluyen las Figuras 8.3a y 8.3b que presentan el EER promedio obtenido en las pruebas con 1 y 2 capas ocultas, respectivamente, agrupados por tamaño del lote. Cada barra corresponde a una configuración de tamaño de la primera capa oculta y tasa de aprendizaje. Los resultados con más capas ocultas no se incluyen por no suponer una mejora sobre estos.



(a) EER por configuración con 1 capa oculta



(b) EER por configuración con 2 capas ocultas

Figura 8.3: EER promedio para cada configuración probada

Finalmente se ha confeccionado un conjunto de gráficos de caja y bigotes (Figuras 8.4, 8.5 y 8.6) que permiten comparar las distribuciones de los EER obtenidos por cada clasificador para cada número de capas ocultas, tamaño de la primera capa oculta y tamaño de lote.

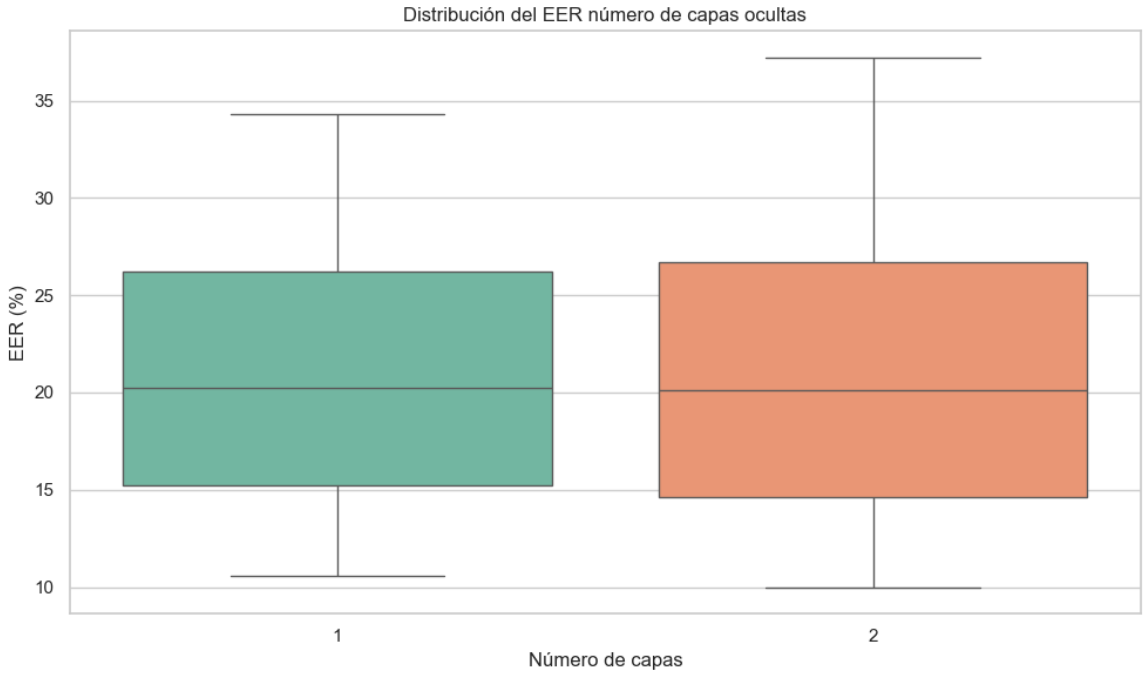


Figura 8.4: Distribución del EER por número de capas ocultas

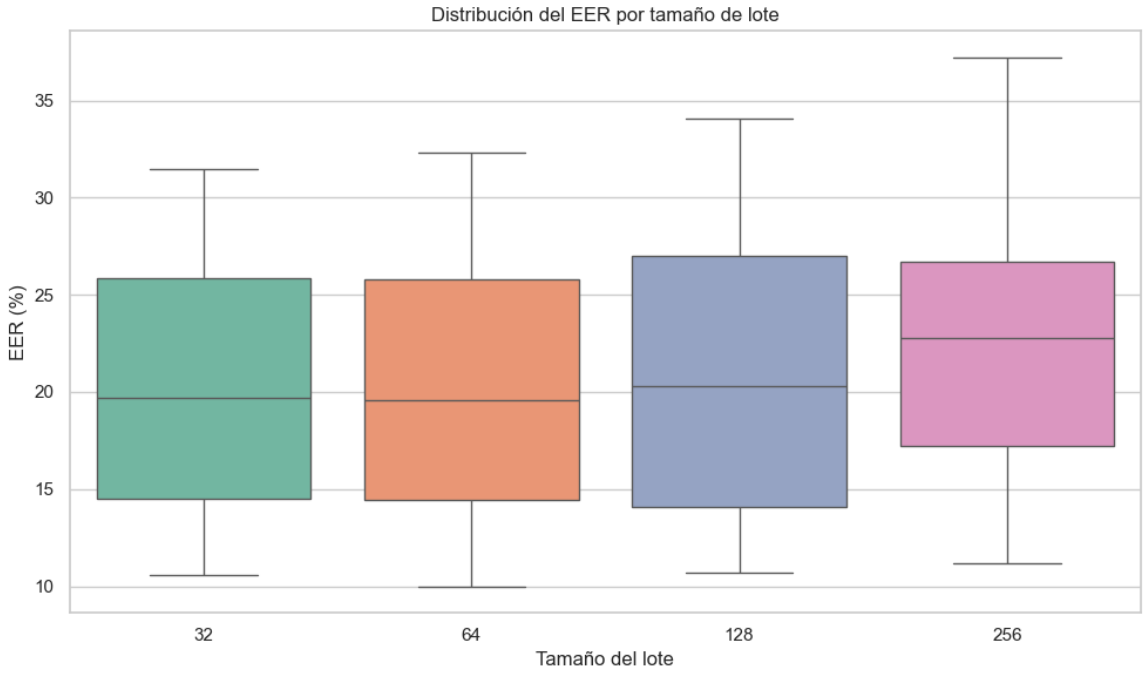


Figura 8.6: Distribución del EER por tamaño de lote

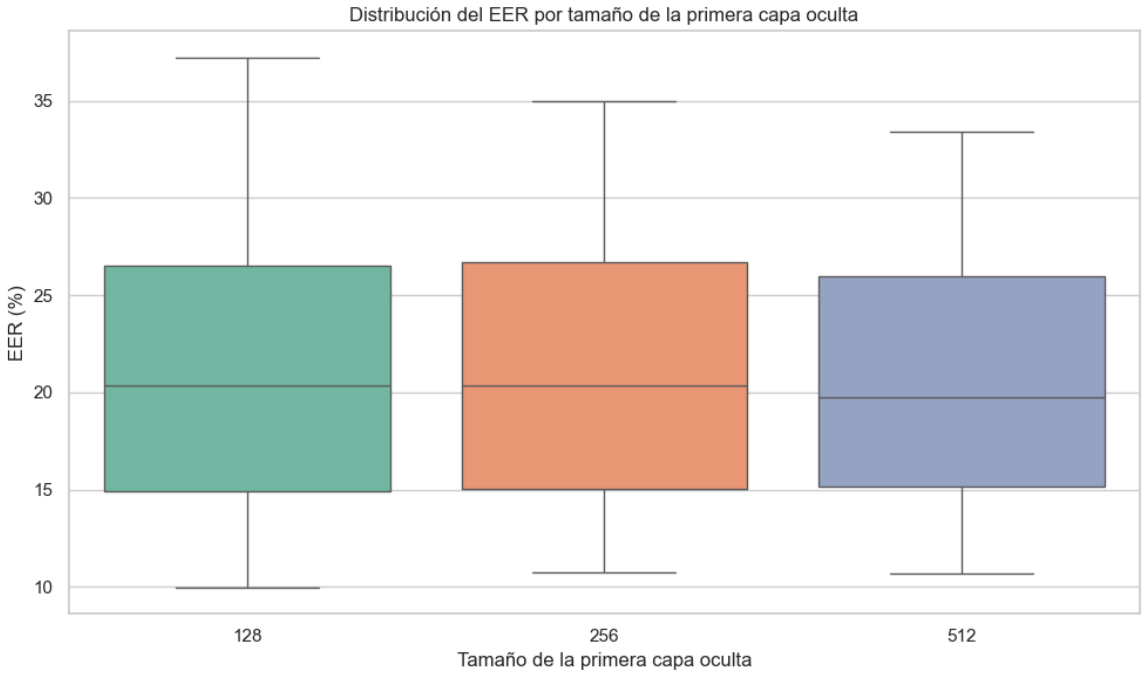


Figura 8.5: Distribución del EER por tamaño de la primera capa oculta

Analizando todos los resultados, queda bastante claro que las variaciones que se producen conforme cambian los hiperparámetros son muy ligeras. No obstante, existen algunas tendencias que se discutirán a continuación.

Emplear dos capas ocultas para el clasificador obtiene un EER promedio de 20,7 % en este experimento, mientras que añadir una capa más mejora ligeramente hasta un 20,5 % (Figura 8.4). Las pruebas realizadas con tres capas ocultas revelan que no existe una mejora significativa que justifique seguir complicando la arquitectura del clasificador. Cabe destacar que, al tratarse de diagramas de caja y bigotes, se puede observar que la variabilidad del EER entre usuarios es mayor al emplear dos capas ocultas.

La Figura 8.5 muestra cómo se distribuye el EER promedio por usuario para cada tamaño de la primera capa oculta implementada en el clasificador. El tamaño de las sucesivas capas se calcula en función de la primera. Se aprecia una disminución del valor medio del EER conforme se aumenta el tamaño de la primera capa oculta (20,8 %, 20,6 % y 20,4 % respectivamente). También se puede ver que la variabilidad entre usuarios se reduce significativamente, por lo que 512 parece la mejor alternativa.

El efecto del tamaño del lote en el EER promedio (Figura 8.6) es bastante limitado, e incluso perjudicial. Se aprecia una ligera mejora al emplear 64 muestras por lote (19,7 %) frente a 32 (19,8 %), aunque también presenta una mayor variabilidad entre usuarios en el segundo caso. No obstante, seguir aumentándolo a 128 o 256 degrada considerablemente el rendimiento (20,7 % y 22,2 %, respectivamente).

Para la tasa de aprendizaje, como se aprecia en las figuras 8.3a y 8.3b, emplear 0,001 obtiene mejores resultados frente a emplear 0,0001 en todas las situaciones excepto en cuatro, es decir, en 20 de las 24 configuraciones con 0,001 se obtiene una mejora frente a la alternativa. El promedio del EER total es de 20,2 % para 0,001 y de 20,9 % para 0,0001.

Finalmente, se seleccionan los hiperparámetros que obtienen mejores resultados para realizar una última prueba con el conjunto completo de datos. En este caso, para el clasificador se emplearán dos capas ocultas, la primera de ellas con 512 neuronas. Se configura un tamaño de lote de 32 muestras, ya que la mejora con 64 es escasa y se empleará una tasa de aprendizaje de 0,001.

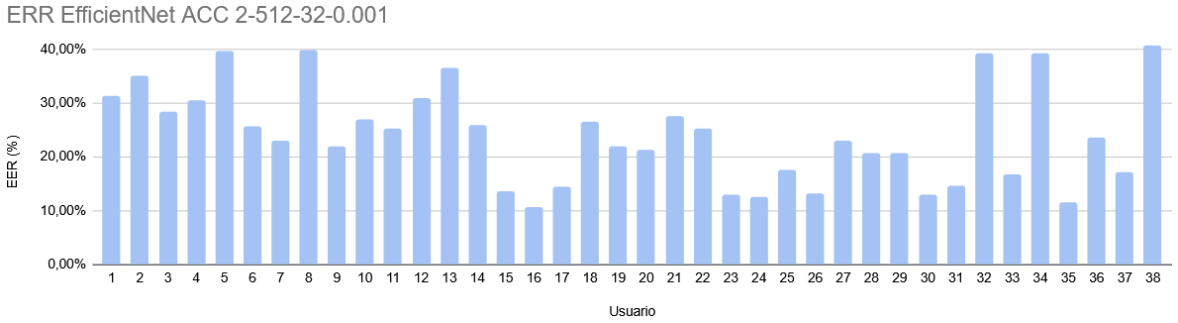


Figura 8.7: EER por usuario para la configuración final.

En la Figura 8.7 se presenta el EER (%) para cada usuario obtenido por la configuración final seleccionada tras el estudio de hiperparámetros. El EER promedio obtenido es del 24,1 %, una mejora del 2,1 % frente a la versión previa de la arquitectura empleada en el estudio comparativo.

8.5. Discusión Final y Comparación de Resultados

El mejor resultado obtenido tras la validación completa sobre los 38 usuarios del conjunto de datos alcanza un valor medio de 26,2 % de Equal Error Rate (EER). Posteriormente, la arquitectura fue ajustada para minimizar este valor y se exploraron distintas combinaciones de hiperparámetros; los resultados han mejorado hasta el 24,1 %.

Este resultado no es competitivo frente a otros estudios relevantes que se han llevado a cabo sobre este problema. Sin embargo, en dichos estudios se han empleado bases de datos diferentes a la aquí utilizada. Por tanto, la comparativa con el estado del arte no es del todo objetiva, ya que la base de datos recopilada por el grupo puede suponer un mayor reto debido a su tamaño y a su naturaleza (se emplean dispositivos comerciales en condiciones de adquisición lo más realistas posibles y en varias sesiones).

Para poder establecer una conclusión objetiva, es necesario comparar estos resultados con aquellos obtenidos sobre los mismos datos. En este caso, en el grupo se cuenta con los datos obtenidos por la alumna Silvia Nieto Casado en su TFG "Estudio del efecto de entrenar y probar bajo condiciones diferentes en reconocimiento biométrico por la forma de andar", que se presentará en la convocatoria de julio.

En este trabajo, en la modalidad de sesión cruzada y empleando el sensor acelerómetro, se obtienen resultados de 23 % de EER con Random Forest y 30,6 % usando máquinas de vectores soporte con núcleo radial. Por tanto, los resultados son peores para SVM y ligeramente mejores cuando se clasifica con Random Forest. Se puede concluir entonces que el presente trabajo sí supone una vía interesante de investigación, ya que logra resultados competitivos.

Capítulo 9

Conclusiones

El objetivo principal de este TFG ha sido evaluar la viabilidad del uso de redes neuronales convolucionales para abordar el problema de la verificación biométrica mediante el análisis de la marcha, a partir de datos capturados con dispositivos ponibles de tipo comercial. Se ha planteado una metodología experimental estructurada en varias fases y diseñada para comparar arquitecturas, representaciones visuales y sensores, que ha culminado en una exploración sistemática de los hiperparámetros del modelo seleccionado.

También se ha llevado a cabo un estudio exhaustivo del estado del arte, se han empleado los datos del conjunto UVa procesándolos y convirtiéndolos en imágenes, se han diseñado, evaluado y comparado diversas estructuras de CNN y se han elaborado una serie de discusiones y conclusiones. Con esto se dan por cumplidos todos los objetivos específicos planteados durante la fase de planificación, lo que ha garantizado que se cumpla el objetivo principal.

El sistema propuesto ha sido capaz de aprender representaciones discriminativas a partir de datos inerciales obtenidos de acelerómetros y giroscopios y transformados en imágenes. En una primera fase, se ha demostrado que la combinación más prometedora en este entorno corresponde al uso del sensor acelerómetro, las representaciones gráficas de tipo línea y la arquitectura EfficientNet.

En definitiva, el trabajo ha cumplido con éxito su propósito de exploración tecnológica, permitiendo analizar en profundidad las posibilidades y limitaciones de las tecnologías empleadas, y proporcionando un marco experimental extensible y bien fundamentado para estudios futuros.

9.1. Líneas de trabajo futuras

El presente trabajo abre múltiples posibilidades de exploración y mejora que podrían ser abordadas en investigaciones futuras. A continuación, se enumeran algunas de esas posibilidades que permitirían superar las limitaciones observadas:

- **Transformación al dominio de la frecuencia**: Actualmente, las imágenes empleadas se basan en la representación directa de los datos temporales. Una línea futura interesante consiste en aplicar transformaciones como la Transformada de Fourier o *Wavelet*, generando espectrogramas o mapas de energía que capturen patrones rítmicos y frecuencia-específicos de la forma de andar, que podrían ser especialmente útiles para tareas de verificación.
- **Exploración de otras arquitecturas**: Otros tipos de redes podrían garantizar mejores rendimientos. Por ejemplo, las redes siamesas ampliamente utilizadas en problemas de verificación biométrica permiten comparar directamente pares de entradas y aprender una función de similitud. Su uso podría ser especialmente adecuado en este contexto, dado que el objetivo no es la identificación, sino la verificación.
- **Aumento de datos**: La limitación de datos genuinos por usuario puede paliarse mediante técnicas de *data augmentation*. La incorporación de bases de datos públicas y reconocidas, como *WISDM* o *ZJU-GaitAcc*, permitiría ampliar la diversidad de los usuarios, mejorando la capacidad de generalización del modelo.
- **Fusión multimodal**: Otra vía de desarrollo consiste en combinar los datos de múltiples sensores (acelerómetro, giroscopio, magnetómetro, etc.) o distintas vistas (por ejemplo, señales crudas y transformadas), a través de técnicas de *early* o *late fusion*. Esta estrategia puede enriquecer la representación del patrón de marcha y mejorar la discriminación entre usuarios.
- **Estudio de la estabilidad temporal**: Finalmente, un aspecto crucial para la biometría es la permanencia del patrón en el tiempo. Sería de interés estudiar cómo varía el rendimiento cuando se entrena en una sesión y se prueba en otras más alejadas temporalmente, evaluando así la estabilidad longitudinal del sistema.

Estas propuestas pueden permitir mejorar la precisión y robustez del sistema, así como también extender el alcance del trabajo hacia aplicaciones reales, manteniendo el enfoque en soluciones accesibles y eficientes.

Bibliografía

[1] Gunawan Ariyanto and Mark S. Nixon. Marionette mass-spring model for 3d gait biometrics. In *2012 5th IAPR International Conference on Biometrics (ICB)*, pages 354–359, 2012.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Lavanya Banga and Samaya Pillai. Impact of behavioural biometrics on mobile banking system. *Journal of Physics: Conference Series*, 1964:062109, 07 2021.

[4] Asif Ali Banka, Dr Ajaz, and Hussain Mir. Human identification based on gait, 05 2010.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, October 2018.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.

[8] Hanqing Chao, Yiwei He, Junping Zhang, and Jianfeng Feng. Gaitset: Regarding gait as a set for cross-view gait recognition, 2018.

[9] Pete Chapman, Julian Clinton, Thomas Khabaza, Thomas Reinartz, Colin Shearer, and Rüdiger Wirth. Crisp-dm 1.0: Step-by-step data mining guide, 1999.

[10] Google Cloud. Precios de gpu - google cloud, 2025. Disponible en: https://cloud.google.com/compute/gpus-pricing?hl=es-419 Consultado en enero de 2025.

[11] Sruti Das Choudhury and Tardi Tjahjadi. Gait recognition based on shape and motion analysis of silhouette contours. *Computer Vision and Image Understanding*, 117:1770–1785, 12 2013.

[12] Paula Delgado-Santos, Ruben Tolosana, Richard Guest, Farzin Deravi, and Ruben Vera-Rodriguez. Exploring transformers for behavioural biometrics: A case study

in gait recognition. In *Proceedings of the IEEE International Joint Conference on Biometrics (IJCB)*, 2022.

[13] Paula Delgado-Santos, Ruben Tolosana, Richard Guest, Ruben Vera-Rodriguez, and Julian Fierrez. M-gaitformer: Mobile biometric gait verification using transformers. *Pattern Recognition*, 135, 2023.

[14] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):5962–5979, October 2022.

[15] Mohammad Omar Derawi, Claudia Nickel, Patrick Bours, and Christoph Busch. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *2010 Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 306–311, 2010.

[16] George T. Doran. There's a s.m.a.r.t. way to write management's goals and objectives. *Management Review*, 70(11), 1981.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[18] Javier Galbally, Sébastien Marcel, and Julian Fierrez. Biometric antispoofing methods: A survey in face recognition. *IEEE Access*, 2:1530–1552, 01 2014.

[19] Abhishek Gangwar and Akanksha Joshi. Deepirisnet: Deep iris representation with applications in iris recognition and cross-sensor iris recognition. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2301–2305, 2016.

[20] Glassdoor. Sueldo: Jefe proyecto en españa, 2024. Disponible en: https://www.glassdoor.es/Sueldos/españa-jefe-de-proyecto-sueldo-SRCH_IL.0,6_IN219_KO7,23.htm Consultado en enero de 2025.

[21] Glassdoor. Sueldo: Tester de software en españa, 2024. Disponible en: https://www.glassdoor.es/Sueldos/tester-de-software-sueldo-SRCH_KO0,18.htm Consultado en enero de 2025.

[22] Xavier Glorot, Antoine Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15, 01 2010.

[23] Ana González-Muñiz. *Aplicación de técnicas de aprendizaje profundo (deep learning) al análisis y mejora de la eficiencia en sistemas de ingeniería*. PhD thesis, University of Oviedo, 02 2023.

[24] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services and Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

[25] Luiz G. Hafemann, Robert Sabourin, and Luiz S. Oliveira. Offline handwritten signature verification — literature review. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–8, 2017.

[26] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9), 2009.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[29] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[30] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[31] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[32] Minyoung Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning?, 2016.

[33] IMMUNE Technology Institute. Inteligencia artificial: salarios y oportunidades laborales, 2024. Disponible en: https://immune.institute/blog/inteligencia-artificial-salarios/ Consultado en enero de 2025.

[34] Project Management Institute. *Guía de los Fundamentos para la Dirección de Proyectos (Guía del PMBOK)*. Project Management Institute, sexta edición edition, 2017.

[35] International Organization for Standardization. ISO/IEC 19795-1:2021: Information technology – biometric performance testing and reporting – part 1: Principles and framework, 2021. Disponible en: `https://www.iso.org/standard/73515.html` Consultado en enero de 2025.

[36] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[37] A.K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 14(1):4–20, 2004.

[38] A.K. Jain, A.A. Ross, and K. Nandakumar. *Introduction to Biometrics*. SpringerLink : Bücher. Springer US, 2011.

[39] Zhengshuai Jiang, Haoyang Li, Xinyi Sui, Yutian Cai, Guoxin Yu, and Wei Zhang. Deep learning in biometric recognition: Applications and challenges. In *2024 IEEE 2nd International Conference on Sensors, Electronics and Computer Engineering (ICSECE)*, pages 352–358, 2024.

[40] Jobted.es. ¿cuánto cobra un programador? (sueldo 2025), 2025. Disponible en: https://www.jobted.es/salario/programador Consultado en enero de 2025.

[41] P. Kartik, R. Prasad, and S.R. Prasanna. Noise robust multimodal biometric person authentication system using face, speech and signature features, 01 2009.

[42] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[43] Maryam Lafkih, Mounia Mikram, Sanaa Ghouzali, and Mohammed El Haziti. Evaluation of the impact of noise on biometric authentication systems. In *Proceedings of the 3rd International Conference on Advances in Artificial Intelligence*, ICAAI '19, page 188–192, New York, NY, USA, 2020. Association for Computing Machinery.

[44] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553), 2015.

[45] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[46] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization, 2017.

[47] Fei Luo, Salabat Khan, Yandao Huang, and Kaishun Wu. Activity-based person identification using multimodal wearable sensor data. *Journal of Biomedical Informatics*, 108, 2020.

[48] Davide Maltoni, Dario Maio, Anil Jain, and Salil Prabhakar. *Handbook of Fingerprint Recognition*. Springer Science & Business Media, 2003.

[49] Mahmoud Maqableh, Huda Karajeh, and Ra'Ed Masa'deh. Job scheduling for cloud computing using neural networks. *Communications and Network*, 6:191–200, 08 2014.

[50] Sébastien Marcel and Stan Li. *Handbook of Biometric Anti-Spoofing: Trusted Biometrics under Spoofing Attacks*. Springer, 01 2014.

[51] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[52] Sakorn Mekruksavanich and Anuchit Jitpattanakul. A residual deep learning network for smartwatch-based user identification using activity patterns in daily living. *Computers and Electrical Engineering*, 121:109883, 2025.

[53] Microsoft. Project professional 2024. Disponible en: https://www.microsoft.com/es-es/microsoft-365/p/project-professional-2024/cfq7ttc0ph40?activetab=pivot%3aoverviewtab Consultado en enero de 2025.

[54] Shervin Minaee, Amirali Abdolrashidi, Hang Su, Mohammed Bennamoun, and David Zhang. Biometrics recognition using deep learning: A survey, 2021.

[55] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Voxceleb: A large-scale speaker identification dataset. In *Interspeech 2017*. ISCA, 2017.

[56] Asif Nawaz, Sofian Saidi, Tha'Er Sweidan, Nagy Osman, and Nguyen Hai. A novel methodology for patient prescreening using wireless body area networks (wbans). *Journal of Medical Artificial Intelligence*, 05 2024.

[57] Irene Salvador Ortega. Investigación y desarrollo de un sistema de reconocimiento biométrico mediante dispositivos ponibles (wearables). Master's thesis, Universidad de Valladolid, 2020. Trabajo Fin de Máster, Inteligencia de Negocios y Big Data en Entornos Seguros.

[58] Salil Prabhakar, S. Pankanti, and Anil Jain. Biometric recognition: Security and privacy concerns. *Security and Privacy IEEE*, 1:33 – 42, 04 2003.

[59] Avi Deb Raha, Apurba Adhikary, Gain Mrityunjoy, Yu Qiao, and Choong Seon Hong. Boosting federated domain generalization: The role of advanced pre-trained architectures, 2024.

[60] Rumeth Randombage and Nuwan Jayawardene. Smartwatch-based gait authentication using siamese lstm networks. In *2024 9th International Conference on Information Technology Research (ICITR)*, pages 1–5, 2024.

[61] Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. Deep ppg: Large-scale heart rate estimation with convolutional neural networks. *Sensors*, 19(14), 2019.

[62] RentIt. Renting de ordenadores, 2025. Disponible en: https://rentingdeordenadores.com.

[63] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386 – 408, 1958.

[64] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2009.

[65] Sherif Said, Samer Al Kork, Vishnu Nair, Itta Gowthami, Taha Beyrouthy, Xavier Savatier, and Fayek Abdrabbo. Experimental investigation of human gait recognition database using wearable sensors. *Advances in Science, Technology and Engineering Systems Journal*, 3, 08 2018.

[66] Irene Salvador-Ortega, Carlos Vivaracho-Pascual, and Arancha Simon-Hurtado. A new post-processing proposal for improving biometric gait recognition using wearable devices. *Sensors*, 23(3), 2023.

[67] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[68] S. Sarkar, P.J. Phillips, Z. Liu, I.R. Vega, P. Grother, and K.W. Bowyer. The humanid gait challenge problem: data sets, performance, and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(2):162–177, 2005.

[69] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815–823. IEEE, June 2015.

[70] Colin Shearer. The crisp-dm model: the new blueprint for data mining. *Journal of Data Warehousing*, 5(4), 2000.

[71] Kritpawit Soongswang and Chantana Chantrapornchai. Accelerating automatic model finding with layer replications case study of mobilenetv2. *PLOS ONE*, 19, 08 2024.

[72] Fangmin Sun, Chenfei Mao, Xiaomao Fan, and Ye Li. Accelerometer-based speed-adaptive gait authentication method for wearable iot devices. *IEEE Internet of Things Journal*, 6(1):820–830, 2019.

[73] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[74] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

[75] Yao Tang, Fei Gao, Jufu Feng, and Yuhang Liu. Fingernet: An unified deep network for fingerprint minutiae extraction, 2017.

[76] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation, 2017.

[77] Changsheng Wan, Li Wang, and Vir V. Phoha. A survey on gait recognition. *ACM Computing Surveys*, 51(5), August 2018.

[78] James Wayman, Anil Jain, Davide Maltoni, and Dario Maio. Biometric systems: Technology, design and performance evaluation, 01 2005.

[79] James L. Wayman. Biometric performance testing and reporting. In *Handbook of biometrics*, pages 321–335. Springer, 2005.

[80] Jee weon Jung, Hee-Soo Heo, Ju ho Kim, Hye jin Shim, and Ha-Jin Yu. Rawnet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification, 2019.

[81] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014.

[82] Bonan Zhang, Chao Chen, Ickjai Lee, Kyungmi Lee, and Kok-Leong Ong. A survey on security and privacy issues in wearable health monitoring devices. *Computers and Security*, 155:104453, 2025.