Escuela Técnica Superior de Ingeniería
Universidad de Huelva

Grado en Ingeniería Informática

Trabajo Fin de Grado

Clasificación de arritmias en escalogramas de señales ECGs mediante redes neuronales convolucionales

María Luisa Silva Mejías

Julio de 2025

Índice de contenidos

Resumen

El diagnóstico automático de arritmias cardíacas a partir de señales de electrocardiograma (ECG) representa un desafío relevante en el ámbito de la salud digital, debido a la complejidad de las señales y la necesidad de revisar grandes volúmenes de datos por parte del personal médico especializado.

Este Trabajo Fin de Grado presenta el diseño e implementación de un sistema de clasificación de arritmias basado en redes neuronales convolucionales (CNN) aplicadas a imágenes de escalogramas RGB generadas a partir de señales ECG.

La propuesta metodológica se fudamenta en el uso de la arquitectura *Inception-v3*, junto con técnicas de *data augmentation* y una fase de *fine-tuning*. Asimismo, se evaluó una estrategia de votación multicanal entre modelos entrenados individualmente con distintas derivaciones del ECG.

La experimentación se llevó a cabo sobre un conjunto de más de 10.000 señales preprocesadas, extraídas del dataset propuesto por Yoon y Kang (2023), empleando exclusivamente escalogramas RGB. Los datos estaban previamente divididos en conjuntos de entrenamiento, validación y test siguiendo una proporción 64-16-20.

El modelo final, basado en votación multicanal, obtuvo una *accuracy* del 92,81 % y una *AUC* de 0,99 en el conjunto de test, con mejoras notables en métricas como sensibilidad y *F1-score* respecto a los modelos individuales. Estos resultados son competitivos frente a modelos del estado del arte más complejos y con arquitecturas bimodales.

En conclusión, el sistema propuesto ha demostrado ser robusto y eficaz, con potencial para integrarse como herramienta de apoyo al diagnóstico clínico de arritmias a partir de ECG, contribuyendo a reducir la carga de trabajo del personal sanitario especializado.

Palabras claves

Aprendizaje profundo, aprendizaje automático, redes neuronales convolucionales, electrocardiograma (ECG), escalogramas RGB, clasificación de arritmias, *Inception-v3*, aumento de datos, votación multicanal.

Abstract

The automatic diagnosis of cardiac arrhythmias from electrocardiogram (ECG) signals represents a significant challenge in the field of digital health, due to the complexity of the signals and the need for specialized medical personnel to review large volumes of data.

This Bachelor's Thesis presents the design and implementation of an arrhythmia classification system based on convolutional neural networks (CNNs) applied to RGB scalogram images generated from ECG signals.

The proposed methodology relies on the Inception-v3 architecture, combined with data augmentation techniques and a fine-tuning phase. Additionally, a multichannel voting strategy was evaluated, involving individually trained models using different ECG leads.

The experiments were conducted on a dataset of over 10,000 preprocessed signals extracted from the dataset proposed by Yoon and Kang (2023), using only RGB scalograms. The data were previously split into training, validation, and test sets following a 64-16-20 ratio.

The final model, based on multichannel voting, achieved an accuracy of 92.81% and an AUC of 0.99 on the test set, showing notable improvements in sensitivity and F1-score compared to individual models. These results are competitive with more complex state-of-the-art bimodal architectures.

In conclusion, the proposed system has proven to be robust and effective, with the potential to be integrated as a clinical decision support tool for arrhythmia diagnosis from ECGs, helping to reduce the workload of specialized healthcare professionals.

Keywords

Deep learning, machine learning, convolutional neural networks, electrocardiogram (ECG), RGB scalograms, arrhythmia classification, Inception-v3, data augmentation, multichannel voting.

1 Propuesta del trabajo

1.1 Motivación

Este trabajo surge de una doble motivación. En primer lugar, responde al interés personal por profundizar en técnicas avanzadas de Aprendizaje Automático, concretamente en el uso de redes neuronales convolucionales, cuyo estudio no se contempla en detalle dentro del Plan de Estudios del Grado en Ingeniería Informática. Estas técnicas han demostrado un gran potencial en múltiples áreas de aplicación, especialmente en aquellas relacionadas con el procesamiento y análisis de datos visuales.

En segundo lugar, se plantea la aplicación práctica de dichas técnicas al ámbito de la salud, en concreto, a la clasificación de arritmias cardíacas a partir de escalogramas generados a partir de señales de electrocardiogramas (ECG). Esta temática resulta de especial interés debido a su relevancia clínica, ya que una detección temprana y precisa de las arritmias puede contribuir significativamente al diagnóstico y tratamiento de enfermedades cardiovasculares. El uso de herramientas automáticas para el análisis de señales médicas representa un avance importante hacia el desarrollo de sistemas de apoyo a la decisión médica más eficientes y accesibles.

1.2 Objetivos

El objetivo general de este trabajo es diseñar, implementar y evaluar un sistema de clasificación de arritmias cardíacas basado en redes neuronales convolucionales, aplicadas a imágenes de escalogramas generadas a partir de señales ECG. Esta propuesta permite, por un lado, adquirir una formación sólida en técnicas avanzadas de aprendizaje profundo, no contempladas en detalle en el plan de estudios; y por otro, contribuir al desarrollo de herramientas de apoyo al diagnóstico médico, con especial interés en la detección temprana de alteraciones cardíacas.

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

- Formación teórico/práctica en Aprendizaje Profundo (*Deep Learning, DL*):
    - Estudiar los fundamentos teóricos de las redes neuronales convolucionales y su aplicabilidad en tareas de clasificación de imágenes.
    - Analizar diferentes arquitecturas y técnicas de entrenamiento utilizadas en modelos de DL.
    - Adquirir experiencia práctica en el uso de bibliotecas y entornos de desarrollo para DL (como *TensorFlow* y *Keras*).
    - Explorar métodos de preprocesamiento y aumento de datos aplicables a imágenes médicas.
- Aplicación a un caso práctico de interés:

- Utilizar un conjunto de datos de imágenes de escalogramas obtenidos a partir de señales ECG etiquetadas con distintos tipos de arritmias.
- Implementar y entrenar modelos basados en redes neuronales convolucionales para la clasificación de dichas imágenes.
- Evaluar y comparar el rendimiento de los modelos mediante métricas estándar.
- Analizar los resultados obtenidos y discutir sus implicaciones clínicas.

1.3 Competencias

El Trabajo Fin de Grado ha permitido desarrollar y aplicar una serie de competencias adquiridas a lo largo del Grado en Ingeniería Informática. La principal competencia trabajada ha sido la siguiente:

- **CE7-C**: Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

Esta competencia ha sido central en todas las fases del proyecto, desde el análisis y preprocesamiento de señales biomédicas hasta la implementación, entrenamiento y evaluación de modelos de redes neuronales convolucionales (CNN) para la clasificación de arritmias.

Además, este trabajo ha favorecido el desarrollo de otras competencias complementarias del plan de estudios, como:

- **CG03**: Capacidad para la resolución de problemas.
- **CG04**: Capacidad para tomar decisiones basadas en criterios objetivos.

Estas competencias también han permitido consolidar un aprendizaje significativo en el ámbito de la inteligencia artificial aplicada a la salud.

1.4 Hardware y software

El desarrollo del proyecto se ha llevado a cabo utilizando un equipo de alto rendimiento del Laboratorio de Robótica y Visión Artificial del Departamento de Ingeniería Electrónica, Sistemas Informáticos y Automática de la Escuela Técnica Superior de Ingeniería de la Universidad de Huelva. Este equipo contaba con los recursos necesarios para llevar a cabo las tareas de procesamiento, entrenamiento y evaluación de modelos de forma eficiente.

Por otro lado, se ha utilizado un equipo personal para la redacción y edición de la memoria del proyecto.

1.4.1 Hardware

- **Equipo del DIESIA**:
  - **Procesador**: Intel Core i9-9900K CPU @ 3.60GHz
  - **Memoria RAM**: 32 GB
  - **Tarjeta gráfica**: NVIDIA GeForce RTX 2080 Ti
  - **Sistema operativo**: Microsoft Windows 10 Pro (64 bits)
  - **Almacenamiento**: 1.28 TB
- **Equipo personal**:
  - **Procesador**: Intel Core i7-1165G7 (11ª generación) @ 2.80GHz
  - **Memoria RAM**: 16 GB
  - **Tarjeta gráfica**: Intel Iris Xe Graphics
  - **Almacenamiento**: SSD de 512 GB
  - **Sistema operativo**: Windows 10 Home, versión 22H2

1.4.2 Software

- **Lenguaje de programación**: Python 3.10.11
- **Entornos de desarrollo**: Jupyter Notebook y Visual Studio Code
- **Librerías y frameworks**:
  - *TensorFlow* y *Keras* (para la implementación y entrenamiento de redes neuronales)
  - NumPy y Pandas (para el tratamiento de datos)
  - Matplotlib y Seaborn (para la visualización de resultados)
  - Scikit-learn (para la evaluación del modelo)

1.5 Organización de la memoria

Además, de este primer capítulo, en el que se presenta la propuesta del trabajo incluyendo la motivación, los objetivos, las competencias adquiridas, y los recursos utilizados, y del último capítulo dedicado a la bibliografía, esta memoria se ha estructurado en los siguientes capítulos:

- *Capítulo 2. Introducción y estado del arte*: se expone el contexto del problema abordado y se realiza una revisión de los trabajos más relevantes en la literatura científica relacionados con la clasificación de imágenes médicas mediante redes neuronales convolucionales.
- *Capítulo 3. Materiales*: se describen los datos utilizados, la organización del conjunto de datos en subconjuntos de entrenamiento, validación y prueba, así como las métricas que se emplearán para evaluar el rendimiento del modelo.
- *Capítulo 4. Metodología*: se detallan las arquitecturas de red utilizadas, las técnicas de aumento de datos aplicadas, los principales hiperparámetros definidos para el entrenamiento, y los criterios empleados para la selección del modelo final.

- *Capítulo 5. Resultados y discusión*: se presentan los resultados obtenidos por los modelos entrenados sobre el conjunto de prueba y se comparan con resultados de referencia del estado del arte. Se analiza también el comportamiento de las redes seleccionadas.
- *Capítulo 6. Conclusiones y trabajos futuros*: se resumen las principales conclusiones técnicas derivadas del proyecto, se proponen posibles líneas de trabajo a desarrollar en el futuro y se incluye una valoración personal sobre la experiencia.

Además, se han incluido tres anexos con contenido teórico de apoyo.

- El *Anexo I* contiene una introducción al DL y el funcionamiento de las redes neuronales y sus arquitecturas más relevantes.
- El *Anexo II* aborda aspectos fundamentales del electrocardiograma y su relación con los escalogramas empleados.
- El *Anexo III* explica la generación de escalogramas a partir de señales temporales mediante la Transformada Continua de Wavelet (CWT), detallando los fundamentos teóricos y la metodología empleada para su obtención.

2 Introducción y estado del arte

Las enfermedades cardiovasculares (ECV) constituyen una de las principales causas de mortalidad en el mundo (World Health Organization, 2021). La electrocardiografía (ECG) estándar de 12 derivaciones es una herramienta fundamental en su diagnóstico clínico, debido a su naturaleza no invasiva, bajo coste y amplio uso en entornos hospitalarios (Kligfield et al., 2007). Sin embargo, la interpretación manual de los ECG requiere una alta especialización y supone una carga significativa de trabajo para los profesionales sanitarios (Clifford et al., 2017). En este contexto, los métodos automáticos de análisis y clasificación de señales ECG basados en técnicas de aprendizaje automático y profundo se presentan como soluciones prometedoras para asistir a los cardiólogos en la detección precoz y precisa de diversas arritmias (Acharya et al., 2017).

En los últimos años, el uso de redes neuronales convolucionales (CNN) ha permitido mejorar significativamente los resultados en tareas de clasificación de señales ECG (LeCun et al., 1998; Hannun et al., 2019). Tradicionalmente, estas redes se han aplicado a representaciones unidimensionales de las señales, aunque enfoques recientes han explorado su conversión a imágenes bidimensionales, como gráficos de ECG renderizados en escala de grises (Jun et al., 2018; Li et al., 2021), o escalogramas generados mediante transformadas wavelets (Madan et al., 2022). Estas representaciones permiten capturar características relevantes tanto en el dominio temporal como en el frecuencial, facilitando la extracción automática de patrones discriminativos por parte de las redes neuronales (Madan et al., 2022).

En este Trabajo Fin de Grado hemos tenido como referencia el artículo de Yoon y Kang (2023). En este artículo se propone un enfoque bimodal basado en CNN que combina imágenes en escala de grises y escalogramas como entradas simultáneas a una arquitectura con dos ramas idénticas de *Inception-v3*, con el objetivo de mejorar la clasificación de cuatro clases de ritmos cardíacos a partir de señales ECG de 12 derivaciones. En su estudio, se utilizaron datos del conjunto Chapman University and Shaoxing People's Hospital, con más de 10.000 registros de pacientes. Su modelo bimodal demostró un rendimiento superior al de enfoques tradicionales basados en modelos estadísticos, RNN, LSTM, y CNNs aplicadas por separado a imágenes en escala de grises o escalogramas, alcanzando una precisión del 95,74% y un *AUC* de 0,994 (Yoon & Kang, 2023).

Otros trabajos relevantes en la literatura han explorado enfoques similares. Por ejemplo, Madan et al. (2022) propusieron un modelo híbrido CNN-LSTM basado en escalogramas para la clasificación de arritmias, mientras que Jeong y Lim (2021) desarrollaron una red CNN sobre mapas de características tiempo-frecuencia derivados del ECG. Sin embargo, el enfoque bimodal de Yoon y Kang fue pionero en combinar simultáneamente imágenes en escala de grises y escalogramas como entradas, con resultados destacados.

A pesar de estos avances, la mayoría de los estudios previos se centran en arquitecturas bimodales, y no en analizar el comportamiento de modelos convolucionales clásicos entrenados exclusivamente sobre una única representación (como los escalogramas RGB), o

sobre combinaciones específicas de derivaciones seleccionadas por su rendimiento individual. Este tipo de exploraciones más concretas pueden aportar información valiosa sobre la utilidad real de estas representaciones y sobre la diversidad de soluciones entrenadas en distintos canales del ECG (Hannun et al., 2019).

Contribución de este trabajo

Este trabajo se centra en la clasificación de arritmias cardíacas a partir de imágenes de escalogramas RGB generadas desde señales ECG de 12 derivaciones. Frente a propuestas bimodales como la de Yoon y Kang (2023), que combinan imágenes en escala de grises y escalogramas en un modelo conjunto, este trabajo opta por un enfoque más simple y modular, basado exclusivamente en escalogramas y en el entrenamiento de modelos independientes, con el objetivo de estudiar diferentes configuraciones y arquitecturas de red, técnicas de aumento de datos y derivaciones del ECG.

En esta propuesta se han empleado algunos hiperparámetros inspirados en los descritos por Yoon y Kang (2023), así como la arquitectura *Inception-v3* preentrenada sobre *ImageNet*, que se ha utilizado como extractor de características para entrenar modelos sobre imágenes de escalogramas RGB. A partir de esta arquitectura base, se ha evaluado el comportamiento de distintos modelos en la clasificación multiclase de ritmos cardíacos, considerando variantes tanto en la complejidad de la parte densa de clasificación como en la estrategia de aumento de datos.

La innovación de este trabajo radica en un análisis sistemático del impacto de cada derivación, junto con el estudio de diversas configuraciones arquitectónicas y estrategias de aumento de datos, así como en proponer una solución de votación multicanal simple y efectiva. Todo ello proporciona una base experimental útil para futuras investigaciones en clasificación de señales ECG mediante redes neuronales convolucionales aplicadas a imágenes tiempo-frecuencia.

3 Materiales

En este capítulo se describen los recursos utilizados para el desarrollo experimental del proyecto. En primer lugar, se presenta la base de datos empleada, detallando su origen, estructura y las clases en las que se organiza. A continuación, se explica el proceso seguido para dividir dicha base de datos en los conjuntos de entrenamiento, validación y prueba. Finalmente, se enumeran las métricas utilizadas para evaluar el rendimiento del modelo.

3.1 Base de datos

La base de datos empleada se deriva de un extenso conjunto de datos de ECG de 12 derivaciones. Esta base de datos fue recopilada conjuntamente por *Chapman University* y el Hospital Popular de Shaoxing, que forma parte de la Facultad de Medicina de la Universidad de Zhejiang (Zheng et al., 2020).

El conjunto de datos original contiene 10.588 registros de ECG preprocesados en los 12 canales: *Lead I*, *Lead II*, *Lead III*, *aVR*, *aVL*, *aVF*, *V1*, *V2*, *V3*, *V4*, *V5* y *V6* (ver *Anexo II* para una descripción detallada del significado de las derivaciones del electrocardiograma). Estos registros fueron obtenidos de 10.646 pacientes y están asociados a 11 ritmos cardíacos diferentes, los cuales fueron etiquetados por médicos especialistas. Cada registro de *ECG* se muestreó a una frecuencia de 500 Hz durante un periodo de 10 segundos, por lo que por cada canal se registran 5000 valores de tensión.

Para el articulo tomado como referencia, los registros de ECG unidimensionales se convirtieron en imágenes bidimensionales, concretamente en imágenes de escala de grises y escalogramas. Sin embargo, en este proyecto, únicamente se utilizaron las imágenes de escalogramas en formato RGB generadas a partir de esta base de datos. La generación de un escalograma a partir de una señal temporal se describe de forma teórica en el *Anexo III* de esta memoria.

Es importante señalar que, en el estudio original (Yoon & Kang, 2023), los 11 ritmos cardíacos se agruparon jerárquicamente en cuatro categorías principales: *AFIB* (*Fibrilación Auricular*), *GSVT* (*Taquicardia Supraventricular Generalizada*), *SB* (*Bradicardia Sinusal*) y *SR* (*Ritmo Sinusal*). Esta agrupación se realizó para abordar posibles problemas de desequilibrio de datos debido al reducido número de muestras en algunos de los 11 ritmos originales. La distribución del número de instancias por cada uno de estos cuatro grupos se detalla en la Tabla 3.1.

Grupo ECG (Clase)	Ritmos ECG Originales	Número de instancias
AFIB	Fibrilación Auricular (*AFIB*), *Aleteo Auricular* (*AF*), *Taquicardia Auricular* (*AT*)	2218
GSVT	*Taquicardia Reentrante del Nodo AV* (*AVNRT*), *Taquicardia Reentrante Auriculoventricular* (*AVRT*), *Ritmo Auricular Errante* (*SAAWR*), *Taquicardia Sinusal* (*ST*), *Taquicardia Supraventricular* (*SVT*)	2260
SB	Bradicardia Sinusal (*SB*)	3888
SR	Ritmo Sinusal (*SR*), *Irregularidad Sinusal* (*SI*)	2222
Total		**10588**

Tabla 3.1: Distribución de la base de datos por grupo ECG fusionado.

Para una mejor comprensión de las características visuales de los datos empleados en este proyecto, se incluyen en la Figura 3.1 ejemplos de imágenes de escalogramas RGB para cada una de las cuatro clases de ritmos cardíacos: *AFIB*, *GSVT*, *SB* y *SR*.

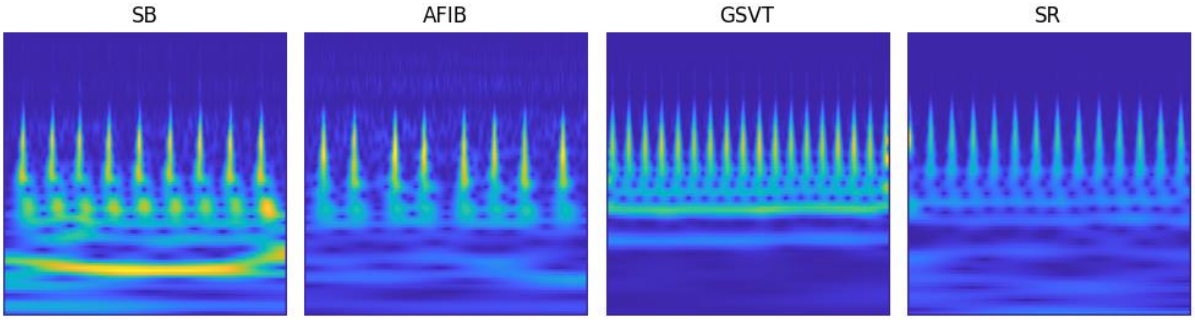

Figura 3.1: Ejemplos de escalogramas RGB: (a) AFIB, (b) GSVT, (c) SB, (d) SR.
Fuente: Elaboración propia.

3.2 Conjuntos de datos: entrenamiento, validación y prueba

Con el fin de entrenar, validar y evaluar el modelo desarrollado, el conjunto total de 10.588 imágenes de escalogramas RGB se dividió en tres subconjuntos mutuamente excluyente. Es

importante subrayar que estos subconjuntos mantienen la proporción de las instancias de cada clase de nuestro problema existente en el conjunto completo.

- **Conjunto de entrenamiento**: Este conjunto se destinó al entrenamiento del modelo. Se han utilizado 6780 imágenes de escalogramas RGB. Esto representa aproximadamente el 64% del total de los datos disponibles. La distribución del número de instancias por clase en el conjunto de entrenamiento se presenta en la Figura 3.2.

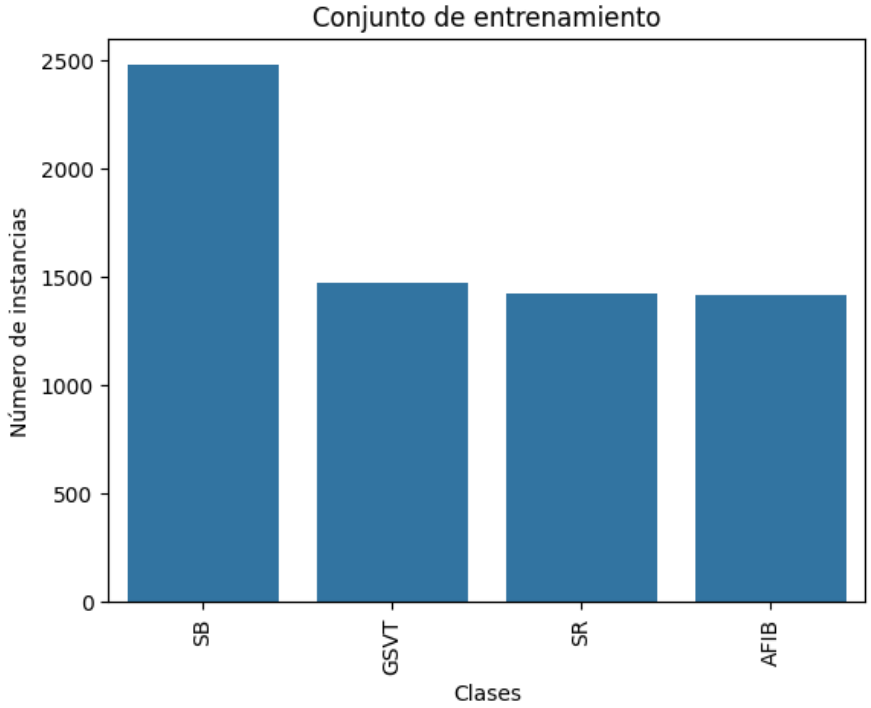

Figura 3.2: Distribución del número de instancias por clase en el conjunto de entrenamiento.
Fuente: Elaboración propia.

- **Conjunto de validación**: Este conjunto se empleó para monitorear el rendimiento del modelo durante la fase de entrenamiento, con el objetivo de evitar el sobreajuste. Se han utilizado 1694 imágenes de escalogramas RGB, lo que equivale a alrededor del 16% del conjunto de datos total. La Figura 3.3 ilustra el recuento de instancias para cada clase en el conjunto de validación.

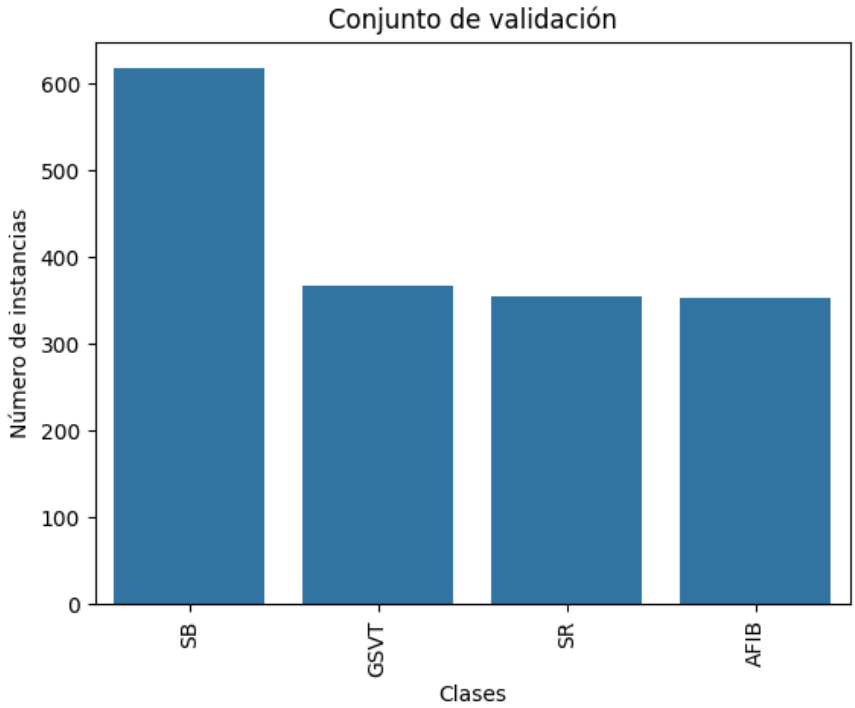

Figura 3.3: Distribución del número de instancias por clase en el conjunto de validación.
Fuente: Elaboración propia.

- **Conjunto de prueba**: Finalmente, se utilizó un conjunto de datos independiente, no empleado en las fases previas, para evaluar el rendimiento final del modelo entrenado y validado. Este conjunto estuvo compuesto por 2114 imágenes de escalogramas RGB, representando aproximadamente el 20% de la base de datos completa. La distribución de las clases en el conjunto de prueba se muestra en la Figura 3.4.

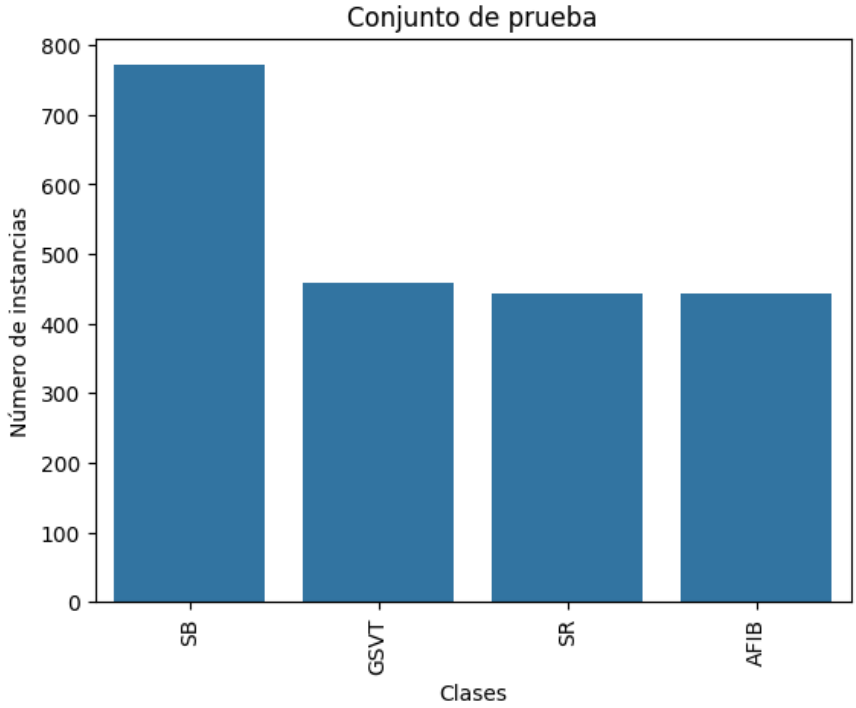

Figura 3.4: Distribución del número de instancias por clase en el conjunto de prueba.
Fuente: Elaboración propia.

3.3 Métricas de evaluación

Para evaluar el rendimiento del modelo de clasificación de enfermedades cardiovasculares basado en imágenes de escalogramas RGB, se emplearán las siguientes métricas de evaluación:

- **Exactitud (*Accuracy*)**: Proporción de predicciones correctas realizadas por el modelo sobre el total de las instancias evaluadas.
- **Precisión (*Precision*)**: En un contexto multiclase, se calcula de forma individual para cada clase tratándola como la clase positiva y agrupando las demás como negativas (estrategia "uno contra el resto"). Indica la proporción de instancias clasificadas como pertenecientes a una clase que realmente lo son.
- **Sensibilidad (*Recall*)**: Evalúa la capacidad del modelo para detectar correctamente las instancias reales de cada clase. En contextos multiclase, se calcula individualmente para cada clase, tomando como referencia el número total de verdaderos positivos frente a los positivos reales.
- **Puntuación F1 (*F1-score*)**: Media armónica entre la precisión y la sensibilidad. Puede agregarse como promedio macro (media aritmética entre clases) o promedio ponderado (media ponderada por el número de instancias por clase).
- **Área bajo la curva ROC (*AUC*)**: En problemas multiclase, se puede calcular un *AUC* por cada clase utilizando el enfoque "uno contra el resto" y luego obtener un promedio macro o ponderado para valorar el desempeño global del modelo distinguiendo entre las diferentes clases.

En todos los casos, una clase se considera “positiva” cuando se evalúa su rendimiento individual mediante la estrategia de “uno contra el resto”, es decir, tratándola como la clase de interés frente al resto del conjunto de clases.

Las definiciones detalladas y las fórmulas matemáticas de estas métricas de evaluación se presentan en el *Anexo I* (apartado 8.2.2.5) de esta memoria.

Esta selección de métricas permitirá obtener una visión completa y detallada del rendimiento del modelo. Además, al ser las mismas métricas empleadas en el artículo científico de referencia, será posible comparar de forma adecuada los resultados obtenidos.

4 Metodología

Este capítulo describe en detalle el enfoque metodológico seguido para abordar la clasificación de arritmias a partir de imágenes de escalogramas RGB. En primer lugar, se presentan las arquitecturas de red utilizadas, indicando las adaptaciones realizadas en la parte de clasificación para ajustarlas a las clases del problema. A continuación, se explican las técnicas de aumento de datos aplicadas sobre las imágenes, con el fin de mejorar la capacidad de generalización del modelo. También se especifican los principales hiperparámetros empleados en el entrenamiento, incluyendo el uso de *transfer learning* con pesos preentrenados en *ImageNet*, la función de pérdida, el algoritmo de optimización y los criterios de parada. Por último, se detalla el proceso de entrenamiento y validación de los modelos, acompañando el análisis con gráficas de evolución de métricas y comparativas entre configuraciones, lo que permite seleccionar la red con mejor rendimiento.

4.1 Arquitecturas de la red

La arquitectura principal utilizada en este trabajo se basa en la red *Inception-v3* (Szegedy et al., 2016), empleada como extractor de características. Esta red es una evolución de la red *GoogLeNet* (Szegedy et al., 2015) —también conocida como *Inception-v1*—, y se caracteriza por una mayor profundidad, eficiencia computacional y capacidad de extracción de características a múltiples escalas. A diferencia de su versión original, *Inception-v3* introduce mejoras como la factorización de convoluciones grandes (por ejemplo, 5×5) en secuencias de convoluciones más pequeñas (por ejemplo, 3×3), el uso de *batch normalization* de forma sistemática y una arquitectura más profunda con bloques optimizados, lo que reduce el coste computacional y mejora el rendimiento en tareas de clasificación. En la Figura 4.1 se puede observar su arquitectura.

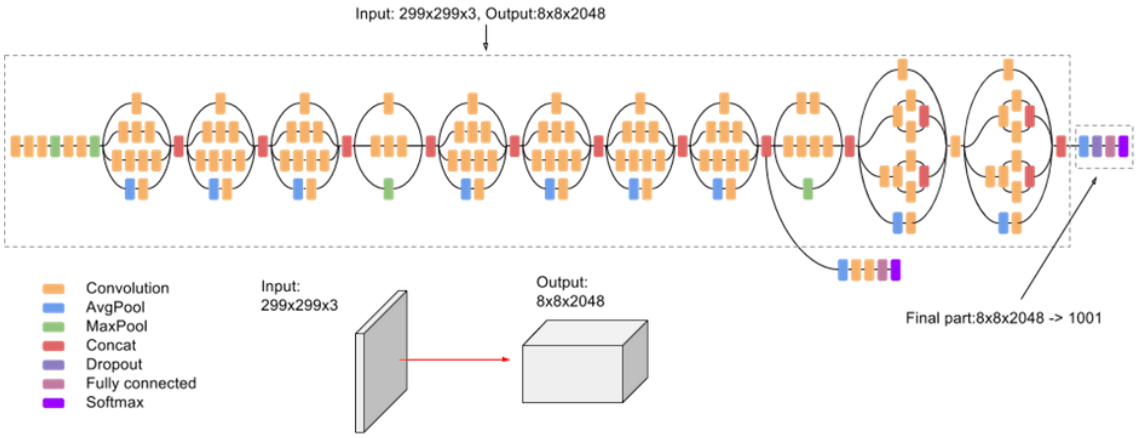

Figura 4.1: Arquitectura de la red Inception-v3.
Fuente: https://paperswithcode.com/method/inception-v3

La elección de esta arquitectura se basa en nuestro artículo de referencia de Yoon y Kang (2023), en el que *Inception-v3* demostró un rendimiento notable en tareas de clasificación de arritmias utilizando imágenes derivadas de señales ECG.

Esta arquitectura se adaptó para trabajar con el número de clases de salida de nuestro problema. En concreto, se ha tomado como salida la capa convolucional Mixed_7c (última capa del bloque convolucional en Inception-v3), que genera un mapa de características de tamaño 8 x 8 x 2048. Sobre este mapa se ha aplicado una capa de *average pooling* global para reducir las dimensiones a un vector 1 x 1 x 2048, que se emplea como entrada para la fase de clasificación.

Las imágenes de entrada utilizadas en este trabajo tienen un tamaño de 299x299 píxeles y tres canales de color. Este tamaño coincide con la dimensión estándar de entrada para la arquitectura *Inception-v3*, lo que permite aprovechar directamente las capas convolucionales preentrenadas sin necesidad de redimensionar ni modificar la estructura inicial de la red. De esta forma, se garantiza que las características espaciales y cromáticas de las imágenes sean adecuadamente procesadas, favoreciendo la extracción de características para la clasificación de arritmias.

La parte densa de clasificación se ha diseñado en dos variantes principales. La primera y más compleja consta de tres capas completamente conectadas (*dense block*) con activación *ReLU*. Tras las dos primeras capas se han añadido capas de *batch normalization* y *dropout* con el objetivo de estabilizar el aprendizaje y reducir el sobreajuste. La última capa de salida es también densa, con activación *softmax*, ajustada al número de clases del problema (cuatro tipos de ritmos cardíacos). En la Figura 4.2 se muestra cómo quedaría la parte de clasificación de esta variante, que sustituye la parte final de la arquitectura original de *Inception-v3*.

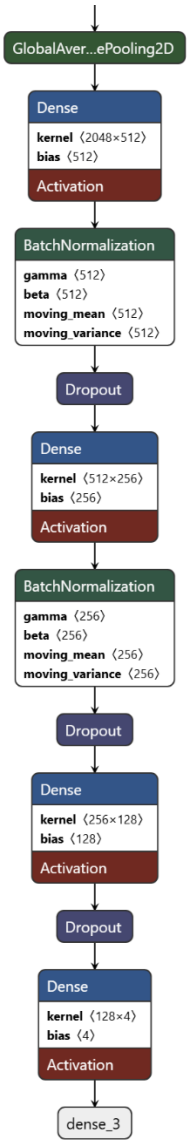

Figura 4.2: Parte de clasificación con tres capas completamente conectadas.
Fuente: Elaboración propia.

La segunda variante evaluada consiste en una única capa densa con activación *ReLU*, seguida de *dropout*, y la correspondiente capa de salida *softmax*. Esta configuración más sencilla se utilizó con el objetivo de comparar su eficiencia frente a la arquitectura más profunda, como se puede observar en la Figura 4.3.

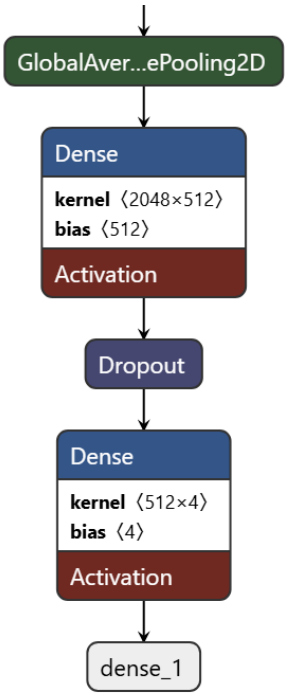

Figura 4.3: Parte de clasificación con una capa completamente conectada.
Fuente: Elaboración propia.

El entrenamiento de estas propuestas de arquitectura se ha llevado a cabo en dos fases: en una primera fase se ha entrenado con los pesos ajustados sobre su entrenamiento en el conjunto de datos *ImageNet* (Deng et al., 2009) mediante la técnica de *transfer learning*, congelando las capas del modelo base y ajustando únicamente la parte de clasificación final; en una segunda fase, se realizó un entrenamiento *fine-tuning* en el que se descongelaron los pesos para su ajuste del 10 % final de las capas de *Inception-v3*.

4.2 Data Augmentation

Para mejorar la capacidad de generalización del modelo y evitar el sobreajuste, se ha aplicado aumento de datos (*data augmentation*) a las imágenes de escalogramas utilizadas como entrada. Esta técnica consiste en realizar transformaciones geométricas y sobre los niveles de gris de los canales RGB sobre las imágenes originales del conjunto de entrenamiento, generando variaciones que simulan nuevas muestras sin necesidad de recopilar datos adicionales. De este modo, se mejora la capacidad de generalización de los modelos.

El aumento de datos empleado en esta experimentación se ha realizado mediante la clase *ImageDataGenerator* de *Keras*, utilizando dos estrategias principales: una versión de aumento suave y otra más agresiva. La versión suave incluye transformaciones geométricas básicas como rotaciones, desplazamientos, cambios de escala o zoom y variaciones leves de brillo, con el fin de preservar las características esenciales de las señales ECG representadas en las imágenes. En concreto, los parámetros definidos para esta versión suave son:

- Rotación máxima de ±10 grados.

- Desplazamientos horizontales y verticales máximos del 10 % del tamaño de la imagen.
- Zoom máximo del 10 %.
- Variaciones de brillo en un rango entre 0,8 y 1,2.
- Reescalado de píxeles para normalizar los valores entre 0 y 1.
- Relleno de píxeles con el valor más cercano para evitar introducir información irrelevante.

A continuación, se muestran ejemplos de imágenes originales y aumentadas con esta versión suave (ver Figura 4.4).

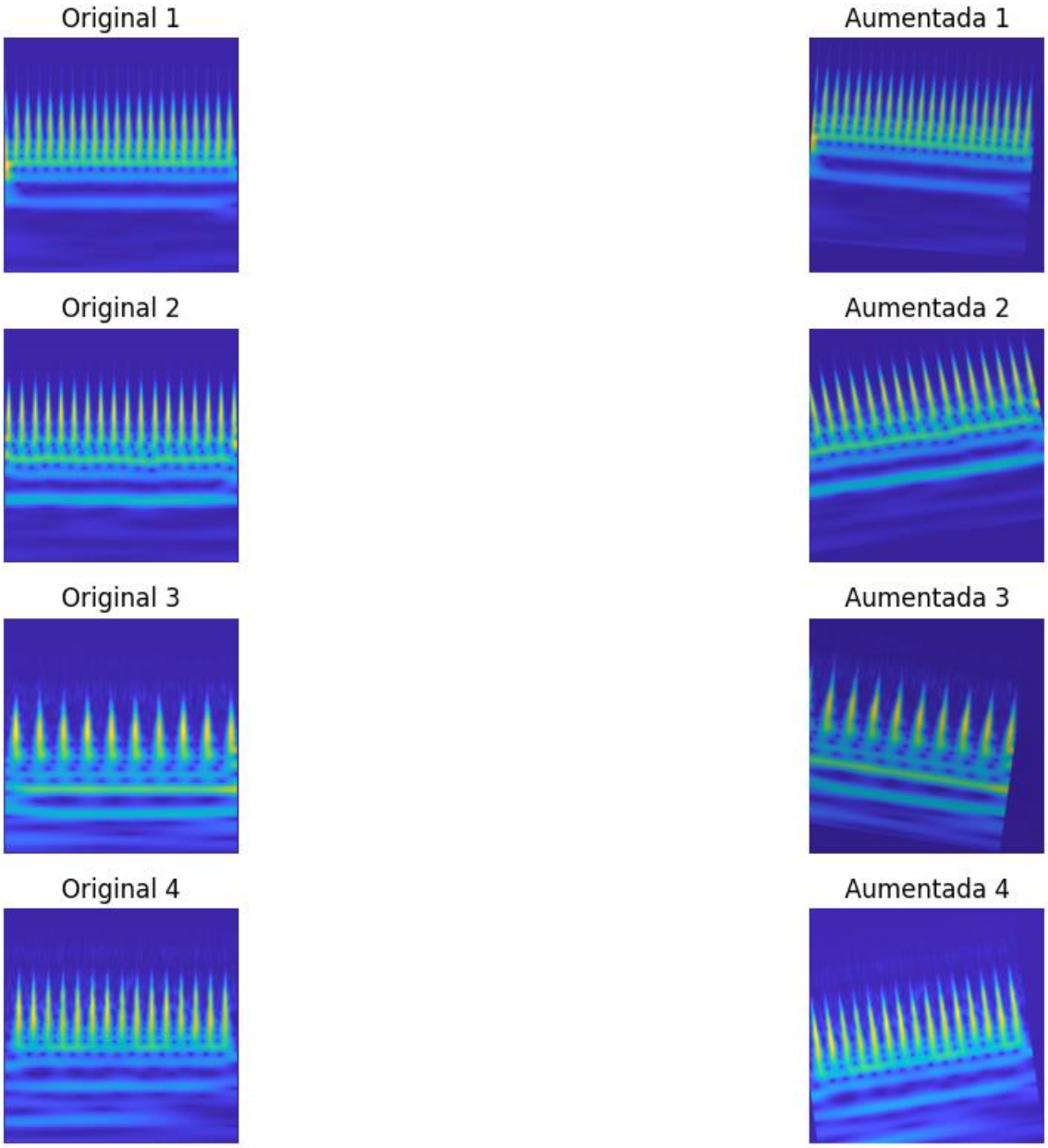

Figura 4.4: Ejemplos de imágenes originales y aumentadas con aumento de datos suave.
Fuente: Elaboración propia.

Es importante destacar que no se ha aplicado volteo horizontal debido a que la orientación espacial en las imágenes ECG es fundamental para su correcta interpretación, y modificarla podría alterar la información relevante de las señales.

Por otro lado, la versión más agresiva de aumento de datos incluye, además de las transformaciones geométricas descritas, modificaciones en la intensidad de los canales RGB, adición de ruido gaussiano con desviación estándar de 3, y ajustes aleatorios en el contraste y la saturación de las imágenes. Esta técnica se ilustra en la Figura 4.5.

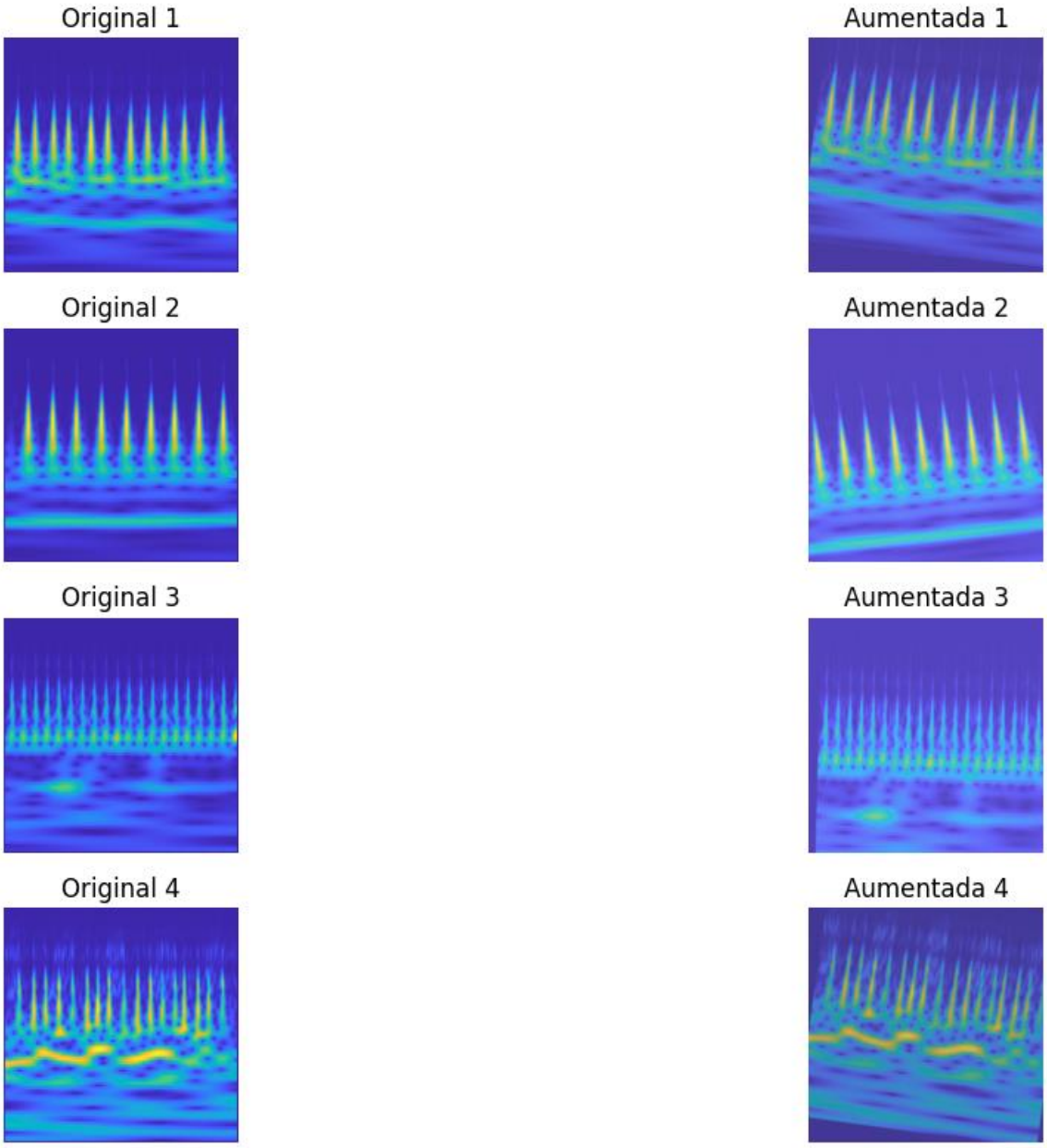

Figura 4.5: Ejemplos de imágenes originales y aumentadas con aumento de datos agresivo.
Fuente: Elaboración propia.

La elección de los parámetros utilizados en ambas versiones de aumento de datos responde a la necesidad de preservar la morfología esencial de las señales ECG representadas en los

escalogramas, al tiempo que se mejora la capacidad del modelo para generalizar ante variaciones. En la versión suave, se han aplicado transformaciones geométricas moderadas (rotaciones, desplazamientos, zoom y variaciones leves de brillo) con el fin de simular ligeras alteraciones que pueden darse en entornos reales sin alterar los patrones característicos como los complejos QRS, ondas P y T. En la versión agresiva, se han incorporado transformaciones adicionales como modificaciones en los canales de color, contraste, saturación y la adición de ruido gaussiano, con el objetivo de aumentar la robustez del modelo ante imágenes que puedan presentar distintas condiciones de calidad o iluminación. Estas decisiones se fundamentan en observaciones visuales, principios biomédicos y estrategias empleadas en trabajos previos que aplican técnicas de *data augmentation* a imágenes derivadas de señales fisiológicas.

4.3 Hiperparámetros

En este apartado se describen los principales hiperparámetros configurados para el entrenamiento de los modelos propuestos, así como los criterios de parada y selección aplicados durante todo el proceso.

La configuración de estos hiperparámetros resultó fundamental para lograr un entrenamiento eficiente y evitar el sobreajuste, especialmente dada la naturaleza del conjunto de datos utilizado.

4.3.1 Transfer learning y pesos iniciales

Se ha utilizado la técnica de *transfer learning* mediante el modelo *Inception-v3* preentrenado con los pesos de *ImageNet*. Todas las capas convolucionales del modelo base se congelaron inicialmente para preservar las características generales aprendidas, y solo se entrenaron las capas densas añadidas. Posteriormente, en la fase de ajuste fino (*fine-tuning*), se descongelaron las últimas 30 capas del modelo base (aproximadamente el 10 % de las capas) para permitir una adaptación parcial a las características específicas del conjunto de datos.

4.3.2 Función de pérdida y optimización

La función de pérdida utilizada fue la entropía cruzada categórica (*categorical crossentropy*), adecuada para problemas de clasificación multiclase. El algoritmo de optimización empleado fue **Adam**, configurado con una tasa de aprendizaje (*learning rate*) de 10^{-4}, y los parámetros $\beta_1 = 0{,}9$ y $\beta_2 = 0{,}999$, tal como se especifica en el artículo de referencia (Yoon & Kang, 2023), en el que también se basa la arquitectura empleada. Tanto la función de pérdida como el algoritmo de optimización utilizados se explican en el *Anexo I* (apartado 8.2.2.2) de esta memoria.

4.3.3 Métricas de control

Durante el entrenamiento, se registraron diversas métricas con el fin de supervisar el rendimiento del modelo en cada época. En particular, se monitorizaron las métricas *accuracy*, *precision*, *recall*, *AUC* y *loss*[1], tanto para el conjunto de entrenamiento como para el conjunto de validación. Estas métricas permiten evaluar el comportamiento del modelo durante el proceso de aprendizaje y detectar posibles síntomas de sobreajuste o infraajuste.

Además, se realizó un seguimiento visual mediante gráficas de evolución de la *accuracy* y la *loss* en ambos conjuntos, con el objetivo de detectar posibles signos de sobreajuste o estancamiento durante el proceso de entrenamiento.

4.3.4 Criterios de parada y selección del modelo

Se aplicó la técnica de parada temprana (*early stopping*) para evitar el sobreajuste y mejorar la generalización. En la primera fase, se utilizó una paciencia de 20 épocas, mientras que en la segunda fase, al implicar la actualización de un mayor número de parámetros, se aumentó la paciencia a 40 épocas. En ambos casos, se restauraron los pesos del modelo que proporcionaron la mejor pérdida de validación.

4.3.5 Tamaño de lote y épocas

Se definieron tamaños de lote distintos para cada subconjunto: 8 para el conjunto de entrenamiento, y 128 para los conjuntos de validación y prueba. El tamaño de lote para el entrenamiento también se ha tomado del artículo de referencia, con el fin de mantener condiciones experimentales comparables. El número máximo de épocas se estableció en 500 para ambas fases, aunque el entrenamiento se detuvo antes gracias al criterio de *early stopping*.

4.3.6 Inicialización de pesos y capas densas

Las capas densas añadidas en la parte de clasificación se inicializaron con esquemas adecuados a sus funciones de activación: *He Normal* para capas con *ReLU* y *Xavier Normal* para la capa de salida con función *softmax* (inicializaciones descritas en el *Anexo I*, apartado 8.2.2.2.2, de esta memoria).

4.4 Entrenamiento

En este apartado se describen los experimentos realizados para entrenar y seleccionar el modelo final propuesto. El objetivo principal ha sido analizar el impacto de diferentes

[1] La métrica *loss* indica la función de pérdida del modelo, que mide la discrepancia entre las predicciones y las etiquetas reales. A menor pérdida, mejor ajuste del modelo.

configuraciones arquitectónicas, estrategias de aumento de datos y canales del ECG sobre el rendimiento del sistema.

Todos los modelos se han entrenado empleando la arquitectura ***Inception-v3*** como base para la extracción de características, utilizando pesos preentrenados en ***ImageNet*** mediante la técnica de *transfer learning*. El proceso de entrenamiento se ha estructurado en dos fases:

- **Primera fase (entrenamiento inicial):** se congelan todas las capas convolucionales del modelo base y se entrena únicamente la parte superior de la red, es decir, las capas densas añadidas para la tarea de clasificación.

- **Segunda fase (*fine-tuning*):** se descongela el 10 % final de las capas de Inception-v3 y se reentrena el modelo completo, permitiendo un ajuste fino de los pesos preentrenados a la tarea específica.

Cada experimento incluye una visualización conjunta de las curvas de *accuracy* y pérdida durante ambas fases de entrenamiento, lo que permite analizar la evolución del modelo en su conjunto y detectar fenómenos como la convergencia temprana o el sobreajuste. Los resultados cuantitativos de cada modelo se resumen al final de cada fase mediante tablas comparativas, facilitando el análisis entre distintas configuraciones.

4.4.1 Primera fase de experimentación: arquitectura con tres capas densas

En esta primera fase se entrenaron tres modelos con la misma arquitectura de tres capas densas en la parte de clasificación, diferenciándose únicamente en la estrategia de aumento de datos aplicada durante el entrenamiento: sin aumento de datos, con aumento de datos suave y con aumento de datos agresivo.

Todos los modelos se entrenaron utilizando imágenes generadas a partir del canal *Lead II* del ECG, ya que en el artículo de referencia (Yoon & Kang, 2023) este canal mostró un rendimiento especialmente destacado en el conjunto de prueba.

1. Modelo sin aumento de datos

El primer modelo se centró en un entrenamiento sin aplicar ninguna técnica de aumento de datos.

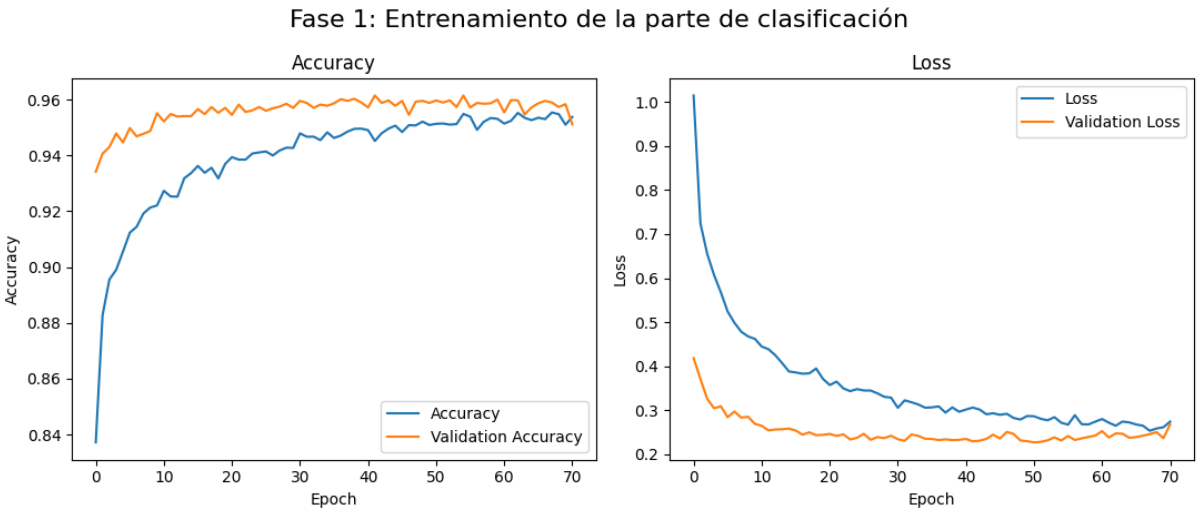

Figura 4.6: Curvas de accuracy y pérdida durante el entrenamiento inicial del modelo con arquitectura de tres capas densas, sin aumento de datos.

Fuente: Elaboración propia.

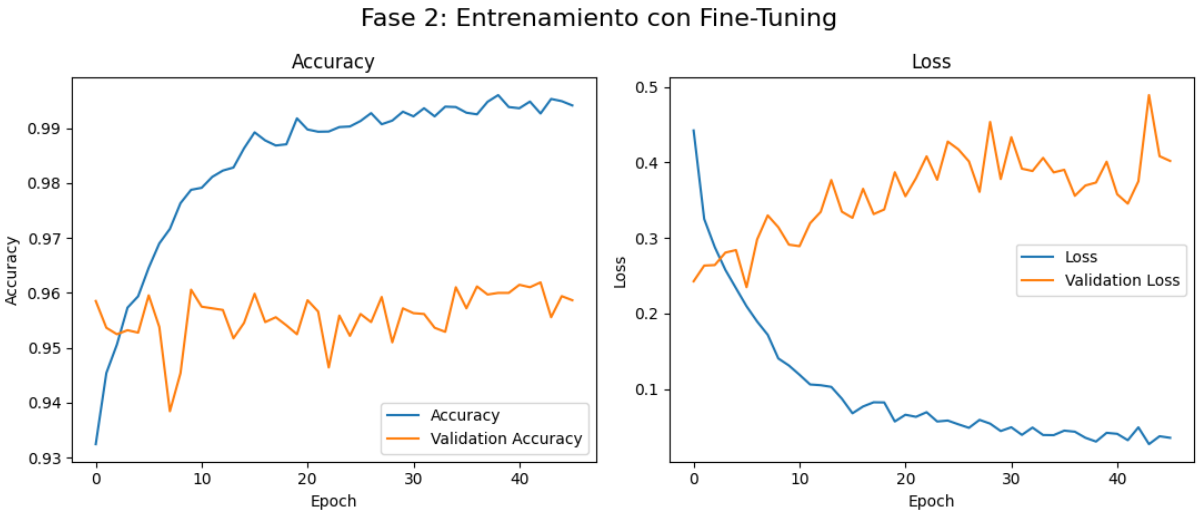

Figura 4.7: Curvas de accuracy y pérdida durante la fase de fine-tuning del modelo con arquitectura de tres capas densas, sin aumento de datos.

Fuente: Elaboración propia.

Como se puede observar en las Figuras 4.6 y 4.7, durante el entrenamiento este modelo mostró una rápida convergencia inicial, alcanzando buenos resultados de validación. No obstante, en las últimas épocas se observaron indicios de sobreajuste, como la divergencia entre la pérdida de entrenamiento y la de validación.

2. Modelo con aumento de datos suave

El segundo modelo utilizó un aumento de datos suave, aplicando transformaciones geométricas básicas, reescalado de píxeles y ajustes leves de brillo.

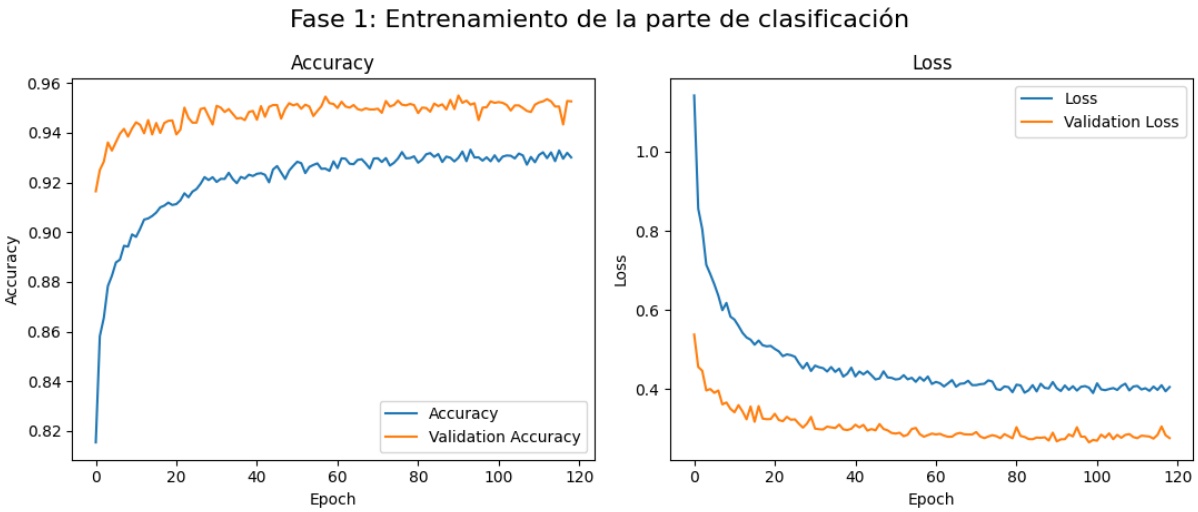

Figura 4.8: Curvas de accuracy y pérdida durante el entrenamiento inicial del modelo con arquitectura de tres capas densas, con aumento de datos suave.

Fuente: Elaboración propia.

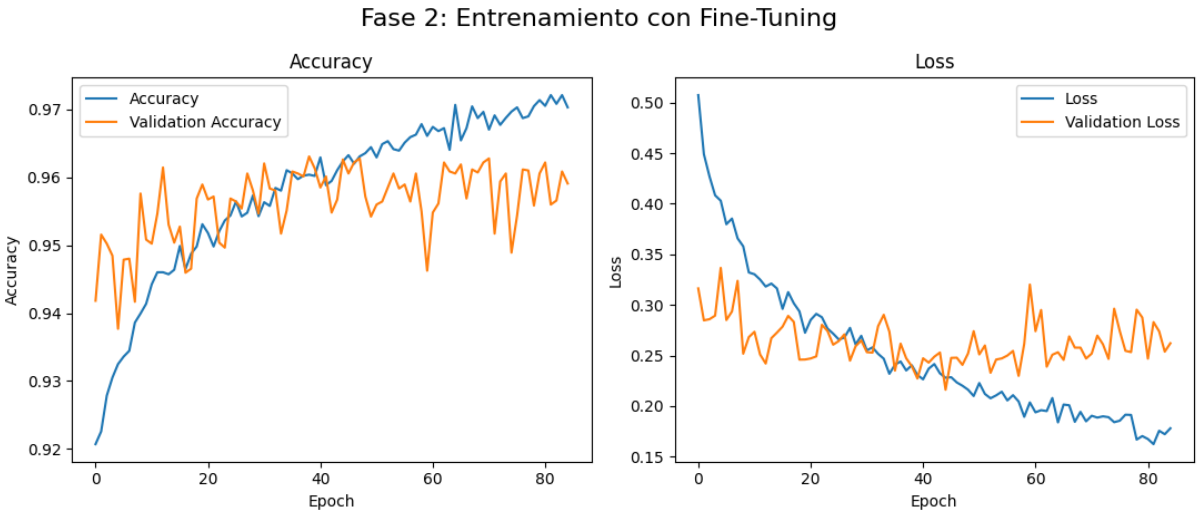

Figura 4.9: Curvas de accuracy y pérdida durante la fase de fine-tuning del modelo con arquitectura de tres capas densas, con aumento de datos suave.

Fuente: Elaboración propia.

Las curvas de *accuracy* y pérdida mostradas en la Figura 4.8 indican una convergencia más estable y un mejor comportamiento en las últimas épocas, sin señales claras de sobreajuste. No obstante, al igual que en los casos anteriores, las curvas correspondientes al ajuste fino (Figura 4.9), evidencian un claro sobreaprendizaje, en este caso, a partir de la época 40.

3. Modelo con aumento de datos agresivo

El tercer modelo incorporó un aumento de datos más agresivo, incluyendo variaciones en intensidad de canales RGB, ruido gaussiano y ajustes de contraste y saturación, además de las transformaciones usadas en el aumento suave.

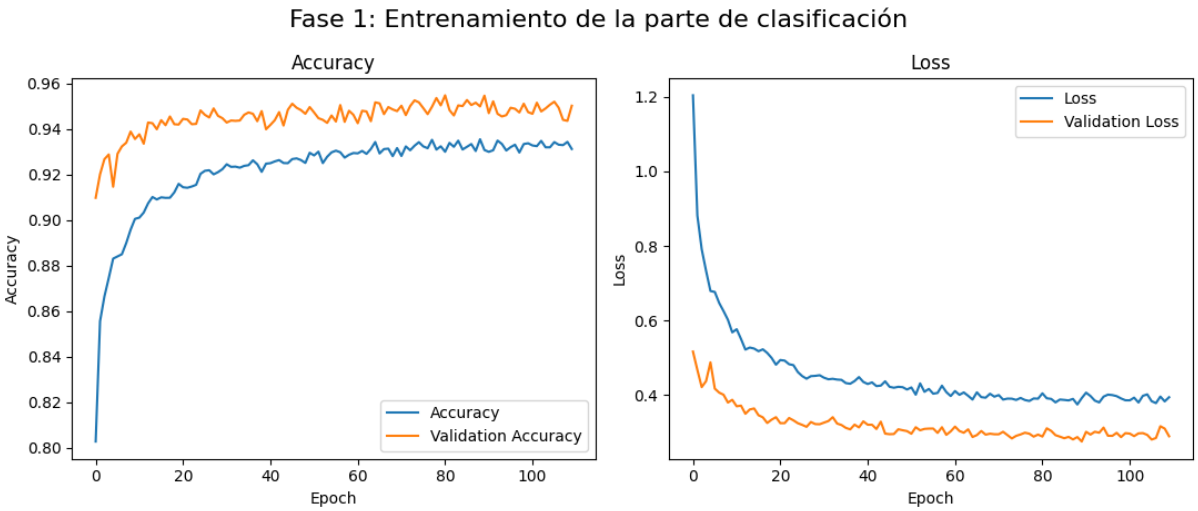

Figura 4.10: Curvas de accuracy y pérdida durante el entrenamiento inicial del modelo con arquitectura de tres capas densas, con aumento de datos agresivo.

Fuente: Elaboración propia.

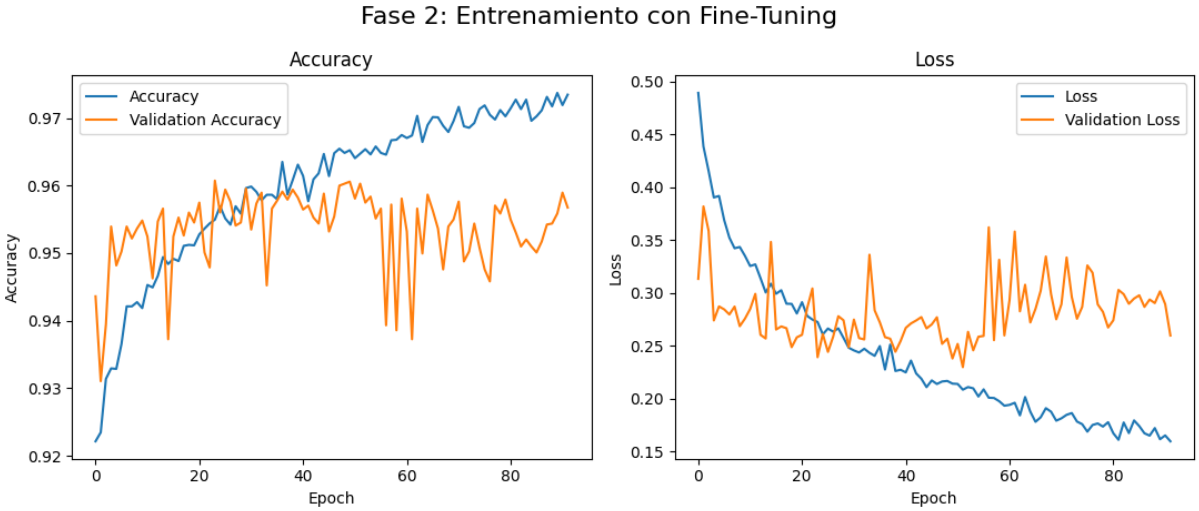

Figura 4.11: Curvas de accuracy y pérdida durante la fase del fine-tuning del modelo con arquitectura de tres capas densas, con aumento de datos agresivo.

Fuente: Elaboración propia.

Las Figuras 4.10 y 4.11 muestran que la calidad de las curvas de entrenamiento y validación es similar a la del aumento suave. Sin embargo, el tiempo de entrenamiento fue considerablemente mayor.

Modelo	Conjunto	*Loss*	*Accuracy*	Precisión	Sensibilidad	*AUC*	*F1-score*
Sin AD	Entrenamiento	0,128	0,978	0,957	0,954	0,996	0,956
	Validación	0,233	0,959	0,920	0,916	0,988	0,918
Con AD suave	Entrenamiento	0,152	0,974	0,949	0,946	0,995	0,947
	Validación	0,216	0,963	0,928	0,922	0,989	0,925
Con AD agresivo	Entrenamiento	0,168	0,969	0,939	0,935	0,994	0,937
	Validación	0,230	0,960	0,923	0,917	0,988	0,920

Tabla 4.1: Resultados de entrenamiento y validación para los modelos con tres capas densas utilizando diferentes estrategias de aumento de datos.

La Tabla 4.1 presenta una comparativa de las métricas obtenidas durante el entrenamiento y la validación de los modelos con arquitectura de tres capas densas, empleando distintas estrategias de aumento de datos. Se observa que la aplicación de aumento de datos mejora ligeramente el rendimiento en el conjunto de validación en la mayoría de las métricas, especialmente en términos de precisión, sensibilidad y F1-score. Aunque el modelo sin aumento alcanza una elevada precisión en el conjunto de entrenamiento, su rendimiento en validación es ligeramente inferior, lo que sugiere un mayor riesgo de sobreajuste. En cambio, los modelos con aumento de datos logran un mejor equilibrio entre ambos conjuntos, lo que indica una mayor capacidad de generalización. Además, el modelo con aumento de datos suave obtiene métricas de validación superiores al modelo con aumento agresivo, presentando también una mayor eficiencia en tiempo de entrenamiento. Por estos motivos, se selecciona el modelo con aumento de datos suave para las siguientes fases de la experimentación.

4.4.2 Segunda fase de experimentación: comparativa de arquitecturas densas (una capa vs tres capas con aumento de datos suave)

En esta segunda fase se entrenaron dos modelos que comparten la estrategia de aumento de datos suave, diferenciándose únicamente en la arquitectura de la parte densa de clasificación: uno con una **única capa densa** y otro con **tres capas densas**.

Al igual que en la fase anterior, ambos modelos se entrenaron exclusivamente con escalogramas generados a partir del canal *Lead II*, dado su buen desempeño en estudios previos.

El modelo con tres capas densas ha sido descrito previamente en el apartado 4.4.1, por lo que solo se ha entrenado un único modelo nuevo. Este incorpora una única capa densa con función de activación *ReLU* antes de la capa de salida, siendo una arquitectura más simple y menos propensa al sobreajuste.

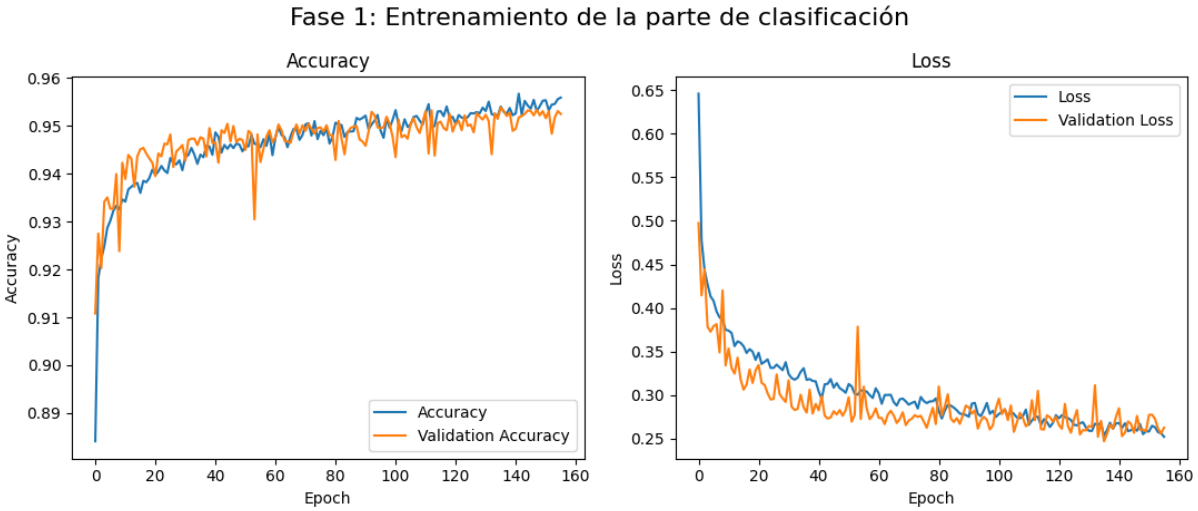

Figura 4.12: Curvas de accuracy y pérdida durante el entrenamiento inicial del modelo con arquitectura de una capa densa, con aumento de datos suave.

Fuente: Elaboración propia.

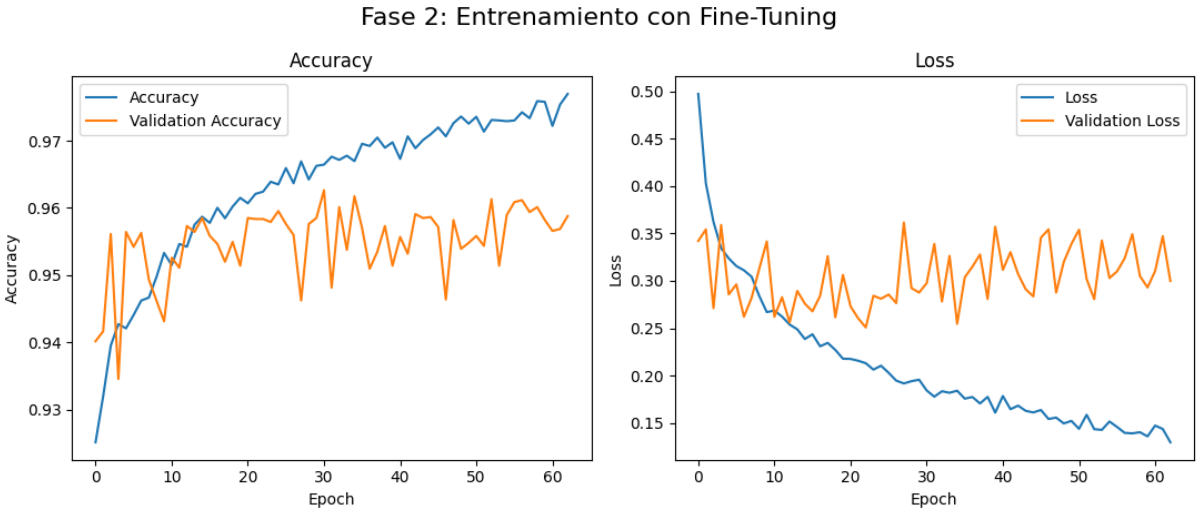

Figura 4.13: Curvas de accuracy y pérdida durante la fase de fine-tuning del modelo con arquitectura de una capa densa, con aumento de datos suave.

Fuente: Elaboración propia.

Como se puede observar en las Figuras 4.12 y 4.13, las gráficas del conjunto de entrenamiento y del de validación están muy alineadas, lo que indica que el modelo aprende bien, sin sobreajuste ni subajuste.

Modelo	Conjunto	*Loss*	*Accuracy*	Precisión	Sensibilidad	*AUC*	*F1-score*
1 capa densa	Entrenamiento	0,230	0,959	0,923	0,912	0,990	0,917
	Validación	0,255	0,952	0,909	0,897	0,987	0,903
3 capas densas	Entrenamiento	0,152	0,974	0,949	0,946	0,995	0,947
	Validación	0,216	0,963	0,928	0,922	0,989	0,925

Tabla 4.2: Resultados de entrenamiento y validación para los modelos con aumento de datos suave utilizando diferentes arquitecturas densas.

Aunque las gráficas mostraban un buen aprendizaje, los resultados de la Tabla 4.2 muestran que el modelo con tres capas densas obtiene mejores métricas tanto en el conjunto de entrenamiento como en el de validación. En particular, se observan mejoras en *accuracy*, *precisión*, *sensibilidad* y *f1-score*, lo que refleja un mayor poder de discriminación y generalización en comparación con el modelo más simple. Por ello, se confirma que la arquitectura con tres capas densas resulta más adecuada para las siguientes fases de la experimentación.

4.4.3 Tercera fase de experimentación: comparación de canales (V1, Lead II y Lead III)

En esta última fase de la experimentación se llevó a cabo una comparación del rendimiento del modelo con arquitectura óptima (*Inception-v3* seguida de tres capas densas y aumento de datos suave) al entrenarse con imágenes de escalogramas obtenidas a partir de distintos canales del ECG: *V1*, *Lead II* y *Lead III*. Estos canales se seleccionaron debido a que son los que mejores resultados de test obtuvieron en el artículo de referencia (Yoon & Kang, 2023), lo que motivó su análisis en el presente trabajo.

El modelo entrenado con el canal *Lead II* ya se describió previamente en la primera fase descrita en el apartado 4.4.1. Para completar la comparativa, se entrenaron dos nuevos modelos utilizando exclusivamente imágenes generadas a partir de los canales *V1* y *Lead III*, manteniéndose constantes el resto de las condiciones experimentales.

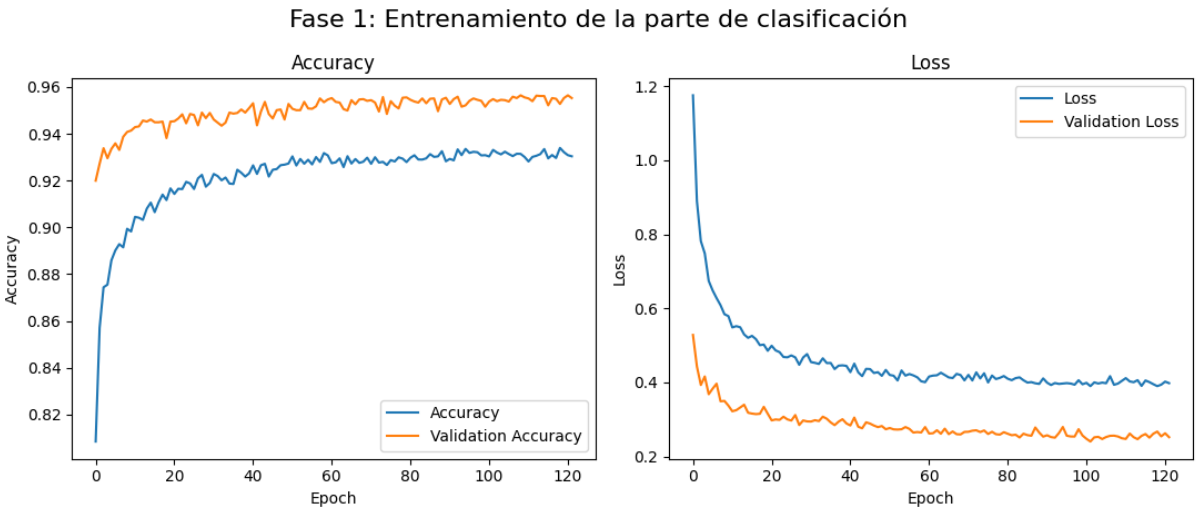

Figura 4.14: Curvas de accuracy y pérdida durante el entrenamiento inicial del modelo con arquitectura de tres capas densas y aumento de datos suave, utilizando el canal V1.

Fuente: Elaboración propia.

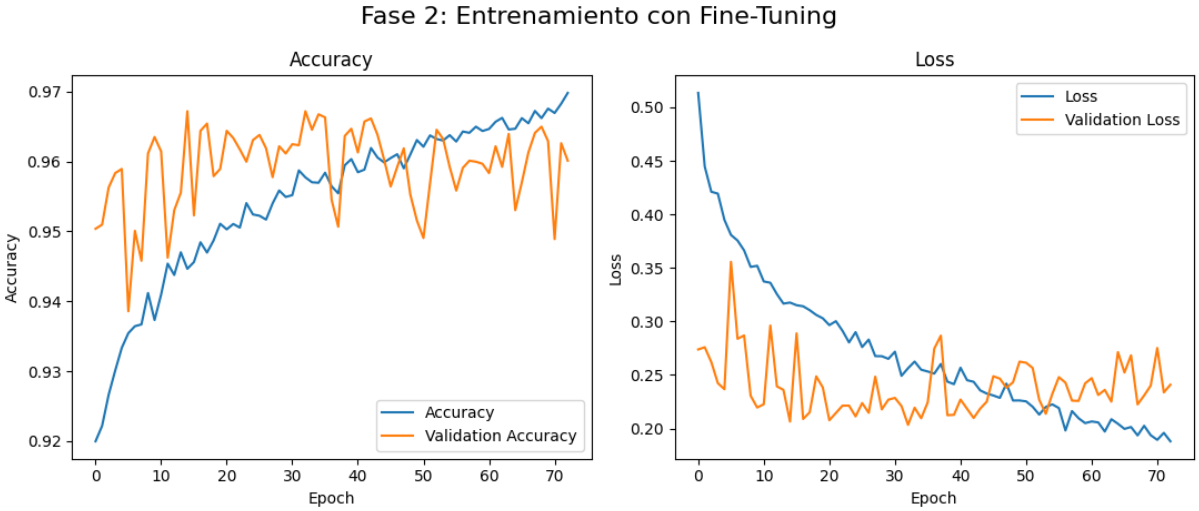

Figura 4.15: Curvas de accuracy y pérdida durante la fase de fine-tuning del modelo con arquitectura de tres capas densas y aumento de datos suave, utilizando el canal V1.

Fuente: Elaboración propia.

Las curvas obtenidas durante el entrenamiento del modelo con la derivación *V1*, representadas en las Figuras 4.14 y 4.15, muestran un comportamiento muy positivo, tanto en *accuracy* como en función de pérdida. A lo largo del proceso, se observa una mejora progresiva en ambas métricas, alcanzando valores cercanos al 97 % de *accuracy* y descendiendo hasta aproximadamente 0,2 en la pérdida del conjunto de validación. Estas gráficas reflejan una evolución estable del aprendizaje, sin signos de sobreajuste ni oscilaciones abruptas, lo cual sugiere que el modelo ha logrado una buena generalización al conjunto de validación.

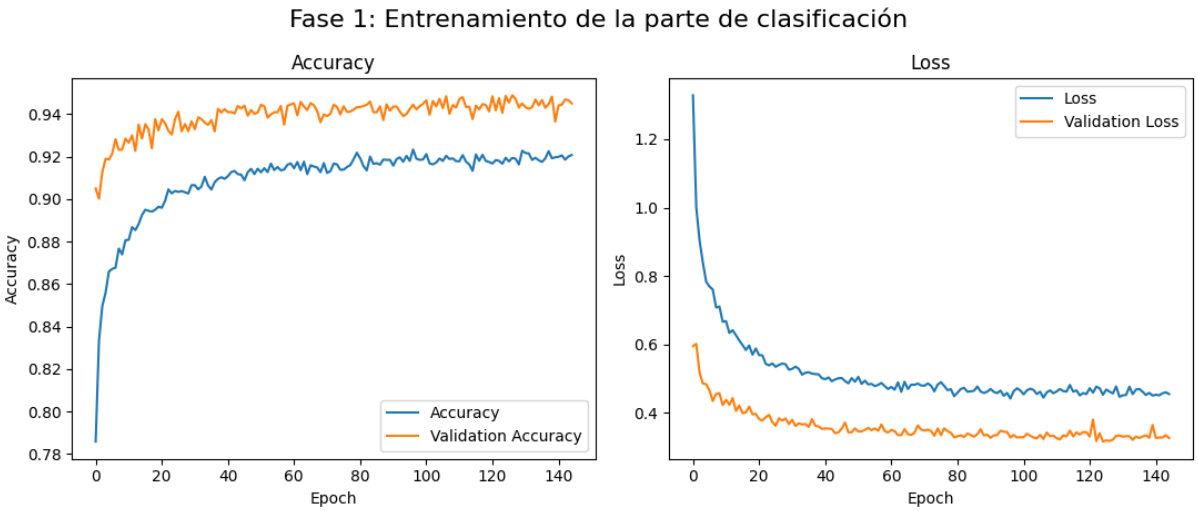

Figura 4.16: Curvas de accuracy y pérdida durante el entrenamiento inicial del modelo con arquitectura de tres capas densas y aumento de datos suave, utilizando el canal Lead III.

Fuente: Elaboración propia.

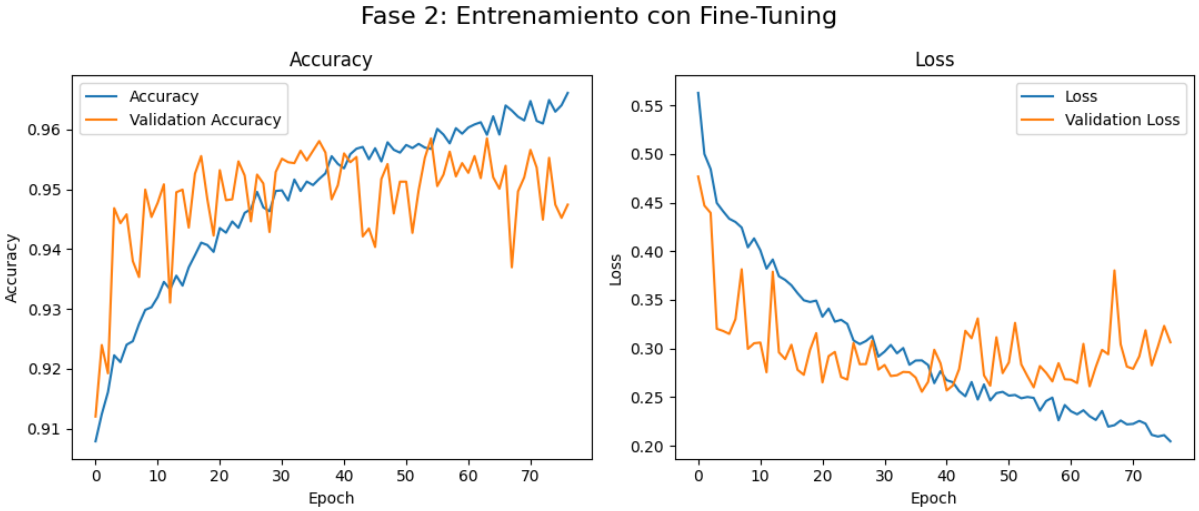

Figura 4.17: Curvas de accuracy y pérdida durante la fase de fine-tuning del modelo con arquitectura de tres capas densas y aumento de datos suave, utilizando el canal Lead III.

Fuente: Elaboración propia.

En el caso del modelo entrenado con la derivación *Lead III*, cuyas curvas se muestran en las Figuras 4.16 y 4.17, el rendimiento es algo inferior. Aunque la curva de *accuracy* también mejora de forma progresiva, no llega a superar el 96 % en el conjunto de validación. Asimismo, la función de pérdida se estabiliza en torno a valores ligeramente superiores a 0,25. El entrenamiento completo evidencia un aprendizaje correcto, pero con una capacidad de generalización más limitada en comparación con la obtenida mediante la derivación *V1*.

Modelo / canal	Conjunto	*Loss*	*Accuracy*	Precisión	Sensibilidad	*AUC*	*F1-score*
Lead II	Entrenamiento	0,152	0,974	0,949	0,946	0,995	0,947
	Validación	0,216	0,963	0,928	0,922	0,989	0,925
V1	Entrenamiento	0,177	0,969	0,941	0,936	0,994	0,938
	Validación	0,203	0,967	0,938	0,931	0,991	0,934
Lead III	Entrenamiento	0,189	0,967	0,938	0,931	0,993	0,935
	Validación	0,256	0,958	0,919	0,913	0,986	0,916

Tabla 4.3: Resultados de entrenamiento y validación para los modelos con tres capas densas y aumento de datos suave utilizando diferentes derivaciones.

La Tabla 4.3 muestra los resultados de entrenamiento y validación obtenidos por los modelos con arquitectura de tres capas densas y aumento de datos suave, utilizando distintas derivaciones (*Lead II*, *V1* y *Lead III*). Se observa que el modelo entrenado con el canal *V1* es el que obtiene el mejor rendimiento general, especialmente en el conjunto de validación, donde alcanza una *accuracy* del 96,7 % y una pérdida de 0,203, además de las mejores puntuaciones en precisión, sensibilidad, *AUC* y *f1-score*. El modelo con el canal *Lead II*, que se utilizó como referencia en las fases anteriores de la experimentación, muestra un rendimiento ligeramente inferior, aunque también muy competitivo. En cambio, el modelo entrenado con la derivación *Lead III* obtiene los peores resultados de los tres, destacando una mayor pérdida (0,256) y menor *accuracy* (95,8 %) en el conjunto de validación, así como las métricas más bajas en las demás categorías. Estos resultados confirman que el canal *V1* proporciona mayor capacidad discriminativa en esta tarea de clasificación, al menos bajo las condiciones del presente experimento.

5 Resultados y discusión

5.1 Resultados en el conjunto de test

En este apartado se presentan los resultados obtenidos en el conjunto de test por los modelos seleccionados tras cada una de las fases de entrenamiento descritas en el capítulo anterior. Estos resultados permiten evaluar la capacidad de generalización de los modelos frente a datos no vistos durante el entrenamiento ni la validación.

5.1.1 Resultados de las fases 1 y 2: aumento de datos y arquitectura

En las dos primeras fases se evaluaron distintas estrategias de aumento de datos y configuraciones arquitectónicas. El modelo que obtuvo el mejor rendimiento global fue el que combinaba la arquitectura *Inception-v3* con tres capas densas y una estrategia de aumento de datos suave. Este modelo se seleccionó como base para las siguientes fases. La Tabla 5.1 recoge sus métricas de clasificación sobre el conjunto de test.

Modelo	***AUC***	***Accuracy* (%)**	**Sensibilidad**	**Precisión**	***F1-score***
***Inception-v3* + 3 capas densas + aumento suave**	0,985	95,674	0,91	0,916	0,913

Tabla 5.1: Resultados en test del modelo seleccionado tras las fases de aumento de datos y arquitectura (Inception-v3 + 3 capas densas + aumento suave).

Como se puede apreciar en la Tabla 5.1, este modelo obtuvo un rendimiento notable en el conjunto de test. Con una *accuracy* superior al 95 % y una *AUC* de 0,985, el sistema demuestra una alta capacidad de generalización y una gran eficacia en la discriminación entre clases. Además, el equilibrio entre precisión, sensibilidad y *F1-score* confirma que el modelo mantiene un rendimiento sólido tanto en la detección como en la clasificación de las arritmias.

Además de estas métricas, se han generado las curvas ROC por clase y la matriz de confusión correspondiente al modelo, disponibles en la Figura 5.1. Estas visualizaciones permiten analizar con mayor detalle el comportamiento del modelo frente a cada clase individual y los patrones de error más frecuentes.

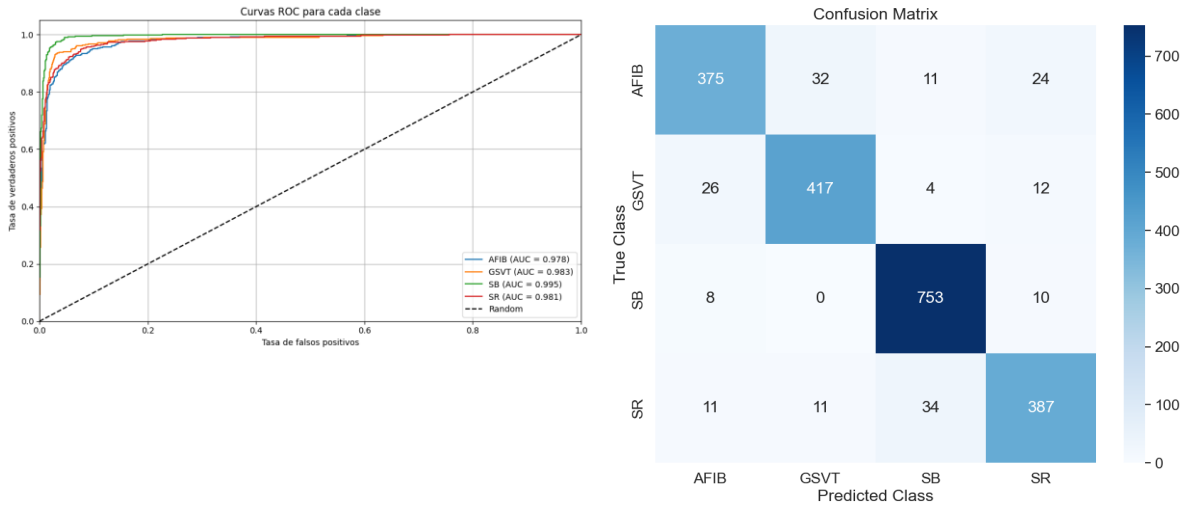

Figura 5.1: Curvas ROC de cada clase y matriz de confusión para el modelo Inception-v3 con aumento de datos suave (canal Lead II).

Fuente: Elaboración propia.

5.1.2 Resultados de la tercera fase: comparación entre canales

En esta fase se analizaron los resultados obtenidos por modelos entrenados exclusivamente con imágenes procedentes de tres canales individuales del ECG: *Lead II*, *V1* y *Lead III*. El objetivo era determinar cuál de ellos proporcionaba un mayor rendimiento clasificando escalogramas de dicho canal por separado.

Cabe destacar que el modelo entrenado con el canal *Lead II* corresponde al modelo seleccionado en la fase anterior (5.1.1), ya que se mantuvo la arquitectura y estrategia de aumento de datos, variando únicamente el canal de entrada.

Las métricas obtenidas por cada uno de estos modelos sobre el conjunto de test se muestran en la Tabla 5.2.

Modelo / canal	*AUC*	*Accuracy* (%)	Sensibilidad	Precisión	*F1-score*
Lead II	0,985	95,674	0,91	0,916	0,913
V1	0,986	95,827	0,914	0,919	0,916
Lead III	0,981	94,704	0,887	0,899	0,893

Tabla 5.2: Resultados en test de los modelos individuales entrenados con canales Lead II, V1 y Lead III.

Como se observa, los resultados obtenidos en el conjunto de test reflejan la misma tendencia que en las métricas del conjunto de validación, con el canal *V1* alcanzando el mejor

rendimiento, seguido por *Lead II* y, finalmente, *Lead III*. Esta jerarquía se mantiene de manera consistente en todas las métricas evaluadas.

En esta fase, las curvas ROC por clase y las matrices de confusión para los modelos entrenados con cada canal (Figuras 5.1 a 5.3) permiten visualizar con detalle las diferencias en rendimiento entre los tres canales. Aunque las curvas ROC del modelo entrenado con el canal *Lead II* muestran un rendimiento parecido e incluso ligeramente superior al del canal *V1*, la matriz de confusión muestra una mayor concentración de errores en las clases minoritarias. En cambio, el modelo basado en *V1* presenta un comportamiento más equilibrado, con una mejor capacidad para discriminar entre clases, evitando en mayor medida la clasificación trivial centrada en la clase mayoritaria. Por ello, se considera que *V1* ofrece un rendimiento más robusto.

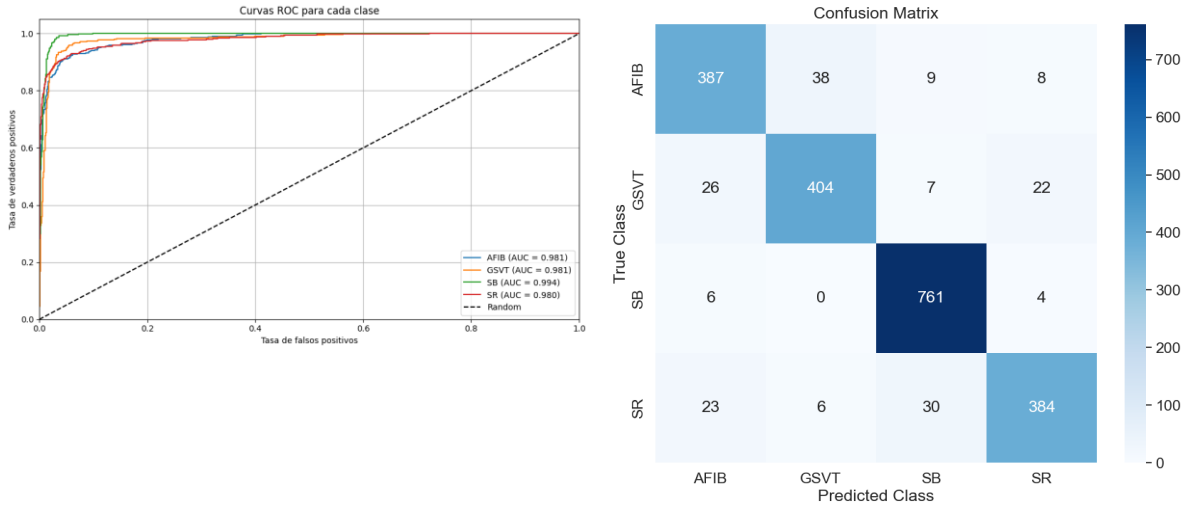

Figura 5.2: Curvas ROC por clase y matriz de confusión para el modelo Inception-v3 con aumento de datos suave (canal V1).

Fuente: Elaboración propia.

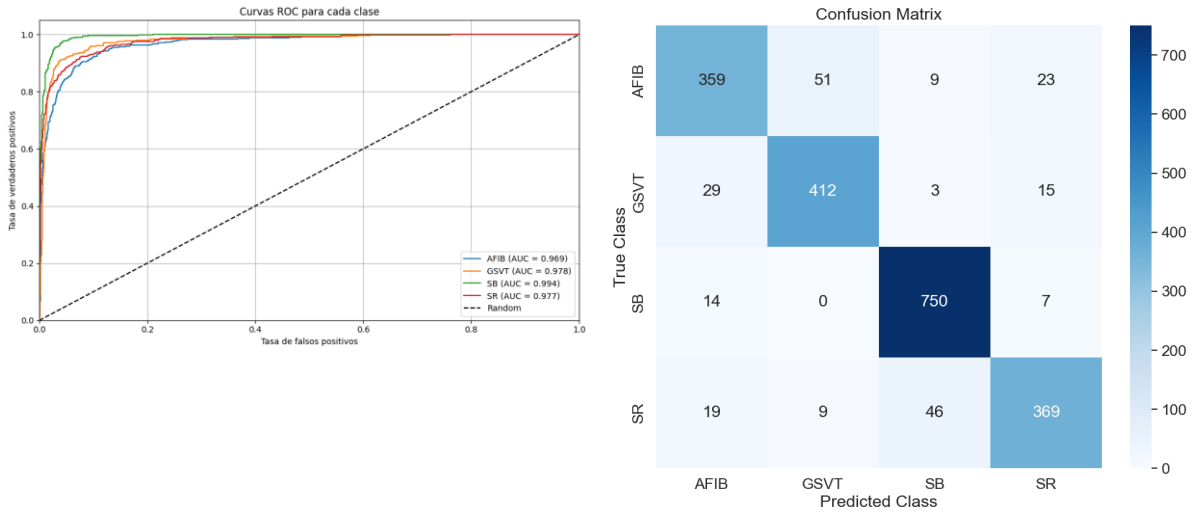

Figura 5.3: Curvas ROC de cada clase y matriz de confusión para el modelo Inception-v3 con aumento de datos suave (canal Lead III).
Fuente: Elaboración propia.

5.1.3 Resultados de la cuarta fase: votación multicanal

En esta última fase se evalúa una estrategia de votación por mayoría utilizando los tres modelos entrenados con los canales individuales *V1*, *Lead II* y *Lead III*. El objetivo es analizar si la combinación de sus predicciones permite mejorar la capacidad de clasificación del sistema frente al uso de modelos individuales.

Cada modelo, entrenado con un único canal, genera una predicción sobre la clase de cada muestra del conjunto de test. A continuación, se aplica una votación entre las tres predicciones para determinar la clase final asignada a cada muestra. En caso de empate, se considera la predicción del modelo con mayor rendimiento en el conjunto de validación (en este caso, el modelo entrenado con *V1*).

La Tabla 5.3 compara los resultados obtenidos en el conjunto de test por la estrategia de votación con respecto al modelo individual que obtuvo el mejor rendimiento (*V1*).

Modelo	*AUC*	*Accuracy* (%)	Sensibilidad	Precisión	*F1-score*
V1	0,986	95,827	0,914	0,919	0,916
Votación (*V1*, *Lead II, III*)	0,990	92,813	0,928	0,928	0,927

Tabla 5.3: Resultados en test del modelo individual V1 y de la estrategia de votación multicanal.

Aunque el *accuracy* de la estrategia de votación es inferior al del modelo *V1*, se observa una mejora en el resto de las métricas, especialmente en la sensibilidad y el F1-score. Esto indica

que la estrategia multicanal es capaz de identificar un mayor número de verdaderos positivos, aunque a costa de un ligero descenso en la exactitud global.

Este comportamiento puede interpretarse de forma positiva, ya que una *accuracy* elevada acompañada de métricas más bajas podría indicar que el modelo tiende a favorecer la clase mayoritaria. En este caso, la mejora en las métricas que evalúan el rendimiento por clase indica una clasificación más equilibrada y eficaz, y demuestra una mayor capacidad del modelo para discriminar correctamente entre todas las clases, evitando una clasificación trivial centrada en la clase más frecuente.

Las Figuras 5.2 y 5.4 permiten comparar visualmente el comportamiento del modelo V1 y la estrategia de votación multicanal. En las curvas ROC de la estrategia de votación se observa una mejora en todas las clases, con curvas más cercanas al vértice superior izquierdo, lo que indica una mayor capacidad de discriminación. Además, la matriz de confusión refleja una distribución de errores más equilibrada, con una reducción en las confusiones asociadas a clases minoritarias. Esta mejora cualitativa respalda los resultados métricos presentados en la Tabla 5.3, y refuerza la idea de que la combinación de predicciones multicanal contribuye a una mejor clasificación entre las distintas clases.

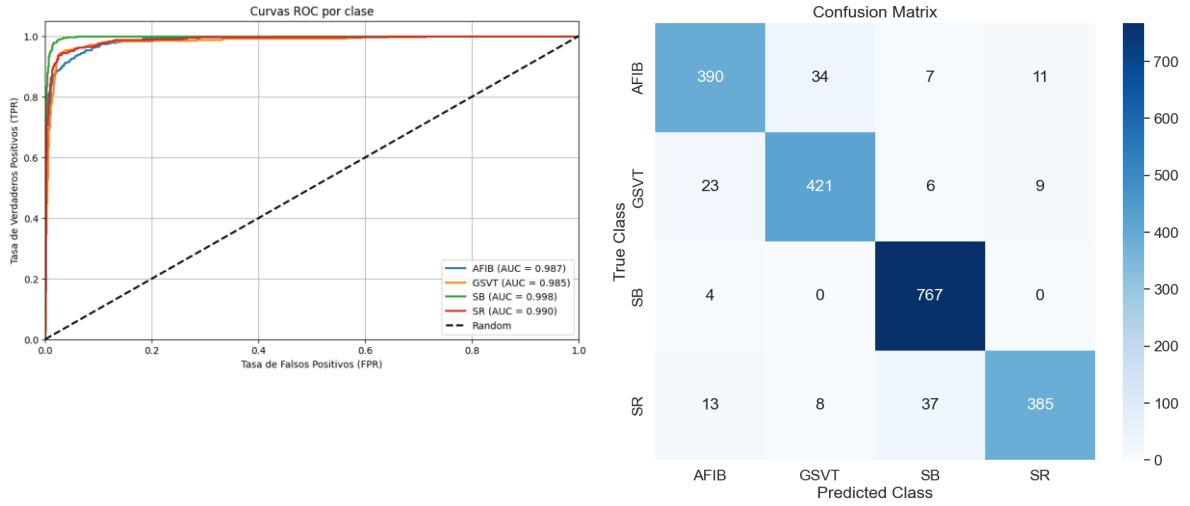

Figura 5.4: Curvas ROC por clase y matriz de confusión para la estrategia de votación multicanal.
Fuente: Elaboración propia.

5.2 Análisis y discusión

Con el objetivo de valorar el rendimiento del sistema propuesto en este trabajo, se ha realizado una comparación con los modelos más relevantes del estado del arte, tomando como referencia principal el estudio de Yoon y Kang (2023). Se han identificado varios puntos de comparación posibles, que se han organizado en tres tablas diferenciadas según el tipo de modelo.

1. Comparación de modelos unicanal basados en Lead II

En la Tabla 5.4 se compara el rendimiento en test de los modelos unicanal entrenados con escalogramas RGB generados a partir del canal *Lead II*, tanto en el presente trabajo como en el artículo de referencia.

Modelo	*AUC*	*Accuracy* (%)	Sensibilidad	Precisión	*F1-score*
CNN unicanal (*Lead II*, Yoon & Kang, 2023)	0,990	94,09	0,931	0,935	0,932
CNN unicanal (*Lead II*, este trabajo)	0,985	95,674	0,91	0,916	0,913

Tabla 5.4: Resultados en test de los modelos unicanal entrenados con escalogramas del canal Lead II.

Como se observa, el modelo desarrollado en este trabajo logra una *accuracy* superior, aunque el modelo de Yoon y Kang presenta valores ligeramente mejores en *AUC*, sensibilidad, precisión y *F1-score*. Este comportamiento sugiere que, si bien el sistema propuesto acierta más en términos globales, el modelo de referencia es algo más equilibrado en la clasificación por clase, especialmente en la detección de verdaderos positivos.

2. Comparación entre los mejores modelos Unicanal

En este trabajo, el modelo unicanal que ofreció mejores resultados fue el entrenado con escalogramas del canal *V1*. Por su parte, el mejor modelo unicanal de Yoon y Kang (2023) fue una red convolucional bimodal, que combinaba escalogramas y grises del canal *Lead II*. La Tabla 5.5 compara ambos enfoques.

Modelo	*AUC*	*Accuracy* (%)	Sensibilidad	Precisión	*F1-score*
CNN bimodal (*Lead II*, Yoon & Kang, 2023)	0,992	95,08	0,942	0,946	0,944
CNN unicanal (*V1*, este trabajo)	0,986	95,827	0,914	0,919	0,916

Tabla 5.5: Comparación entre los mejores modelos unicanal de cada trabajo.

El modelo unicanal con *V1* supera al bimodal de referencia únicamente en *accuracy*, lo que indica un mayor número de aciertos globales. Sin embargo, el modelo de Yoon y Kang muestra un mejor comportamiento en la clasificación por clase. Aun así, los resultados obtenidos por el modelo propuesto son destacables, ya que logra competir en rendimiento sin recurrir a una arquitectura multimodal.

3. Comparación de estrategias multicanal

Por último, se comparan los modelos que combinan información de varios canales. El modelo de referencia utiliza un *ensemble* bimodal que combina los 12 canales del ECG, mientras que en este trabajo se propone una votación multicanal entre tres modelos unicanal entrenados con *V1*, *Lead II* y *Lead III*. En la tabla 5.6 se recogen los resultados de ambas estrategias.

Modelo	*AUC*	*Accuracy* (%)	Sensibilidad	Precisión	*F1-score*
***Ensemble* bimodal (Yoon & Kang, 2023)**	0,994	95,74	0,950	0,953	0,952
Votación multicanal (este trabajo)	0,990	92,813	0,928	0,928	0,927

Tabla 5.6: Comparación entre los modelos multicanal.

Aunque la estrategia multicanal propuesta obtiene valores ligeramente inferiores en todas las métricas, los resultados siguen siendo competitivos si se considera que no se ha empleado un enfoque bimodal ni se han utilizado los 12 canales del ECG, sino exclusivamente escalogramas RGB generados a partir de 3 canales.

En conjunto, se puede concluir que el sistema desarrollado en este trabajo ofrece un rendimiento sólido y generalizable, logrando resultados competitivos frente a propuestas más complejas. Esto demuestra que una arquitectura bien optimizada, combinada con una estrategia de votación multicanal, puede alcanzar un rendimiento equilibrado y eficaz sin necesidad de incrementar significativamente la complejidad del modelo.

6 Conclusiones y trabajos futuros

6.1 Conclusiones técnicas

El desarrollo de este Trabajo Fin de Grado ha permitido diseñar y evaluar un sistema de clasificación automática de arritmias cardíacas a partir de imágenes de escalogramas RGB generadas desde señales ECG multicanal. A nivel técnico, se han alcanzado los objetivos propuestos, destacando los siguientes logros:

- **Arquitectura eficiente y robusta**: El modelo basado en la arquitectura *Inception-v3* junto con tres capas densas y una estrategia de aumento de datos suave ha demostrado un alto rendimiento en la tarea de clasificación, logrando una *accuracy* del 95,67 % y una *AUC* de 0,985 en test, con un buen equilibrio entre *precision*, *recall* y *F1-score*.

- **Influencia del canal ECG**: El análisis individual de las derivaciones *V1*, *Lead II* y *Lead III* ha evidenciado diferencias notables en el rendimiento. El canal *V1* se ha identificado como el más relevante, obteniendo mejores métricas globales y una mayor capacidad para discriminar entre clases.

- **Estrategia multicanal efectiva**: La combinación de predicciones mediante una estrategia de votación mayoritaria entre los modelos unicanal ha permitido mejorar métricas como la sensibilidad y el *F1-score*, a pesar de una ligera caída en la *accuracy*. Esto indica una clasificación más equilibrada y una menor inclinación hacia la clase mayoritaria, lo que reduce el riesgo de comportamientos triviales.

- **Comparativa con el estado del arte**: Aunque no se ha empleado un enfoque bimodal ni el uso completo de los 12 canales del ECG, el sistema propuesto ha logrado resultados competitivos frente a modelos más complejos, como los de Yoon y Kang (2023). En particular, se han alcanzado métricas similares utilizando una arquitectura más simple, lo que refuerza la eficacia del planteamiento.

En conjunto, el trabajo ha demostrado que es posible desarrollar un sistema de clasificación robusto y eficiente utilizando únicamente escalogramas RGB generados desde un número reducido de derivaciones, aplicando técnicas bien fundamentadas de *transfer learning*, *data augmentation* y votación multicanal.

6.2 Trabajos futuros

A partir de los resultados obtenidos y las decisiones metodológicas adoptadas durante el desarrollo de este trabajo, se identifican diversas líneas de mejora y ampliación que podrían explorarse en futuras investigaciones:

- **Uso de arquitecturas más avanzadas**: Aunque *Inception-v3* ha ofrecido un rendimiento muy competitivo, podrían evaluarse otras arquitecturas más recientes como *EfficientNet* o *MobileNetV2*.

- **Incorporación de un enfoque bimodal**: Siguiendo la línea del trabajo de referencia (Yoon & Kang, 2023), sería interesante combinar escalogramas RGB con representaciones en escala de grises, señales sin procesar o imágenes extraídas mediante otros métodos, lo cual podría enriquecer la información utilizada para la clasificación.

- **Optimización del sistema de votación multicanal**: La estrategia de votación mayoritaria podría mejorarse incorporando pesos distintos para cada canal según su rendimiento individual, o bien mediante enfoques de ensamblado más complejos.

- **Ampliación del conjunto de datos y clases**: Incluir más clases de arritmias o nuevos registros provenientes de otras bases de datos públicas podría mejorar la capacidad generalizadora del sistema. Asimismo, podrían explorarse técnicas de balanceo de clases o generación sintética de datos para mejorar el rendimiento en clases minoritarias.

- **Despliegue en un entorno clínico o aplicación práctica**: Como paso final hacia su aplicabilidad real, podría desarrollarse una interfaz interactiva que permita cargar señales ECG reales, convertirlas automáticamente en escalogramas y mostrar la predicción en tiempo real. Esto facilitaría la integración del sistema como herramienta de apoyo al diagnóstico clínico.

En definitiva, el trabajo sienta las bases para futuras mejoras tanto a nivel técnico como funcional, manteniendo siempre como prioridad que el sistema sea robusto, eficiente y útil en situaciones reales de aplicación clínica.

6.3 Valoración personal

La realización de este Trabajo Fin de Grado ha supuesto un importante reto académico y personal, que me ha permitido consolidar y aplicar de forma práctica los conocimientos adquiridos a lo largo de la carrera. A través del diseño, implementación y evaluación de un sistema de clasificación de arritmias basado en imágenes de escalogramas y redes neuronales convolucionales, he podido profundizar en áreas clave como el DL, el procesamiento de señales biomédicas y la evaluación de modelos.

Uno de los aspectos que más me ha aportado en este proyecto ha sido tener que tomar decisiones técnicas con criterio: desde elegir la arquitectura de red y las técnicas de aumento de datos, hasta comparar distintos canales del ECG y combinar los resultados con métodos de ensamblado. Esto me ha permitido comprender con mayor profundidad la importancia de

experimentar de forma ordenada y analizar los resultados con sentido crítico para desarrollar soluciones que funcionen bien y se puedan aplicar en distintos casos.

Además, este trabajo me ha permitido adquirir experiencia en el uso de herramientas profesionales, como *TensorFlow*, *Keras*, y bibliotecas para el tratamiento y visualización de datos, así como en la redacción de documentación técnica rigurosa.

En resumen, este TFG ha sido una oportunidad para integrar múltiples competencias, superar dificultades técnicas y reafirmar mi interés por el ámbito de la inteligencia artificial y el DL. Considero que la experiencia ha sido muy valiosa tanto a nivel académico como personal, y me deja motivada para seguir aprendiendo y desarrollándome en este campo.

7 Bibliografía

- Acharya, U. R., Oh, S. L., Hagiwara, Y., Tan, J. H., & Adam, M. (2017). A deep convolutional neural network model to classify heartbeats. *Computers in Biology and Medicine*, *89*, 389–396. https://doi.org/10.1016/j.compbiomed.2017.08.022
- Addison, P. S. (2005). *Wavelet transforms and the ECG: A review*. Physiological Measurement, 26(5), R155–R199. https://doi.org/10.1088/0967-3334/26/5/R01
- Al-Ani, M. S. (2018). ECG waveform classification based on P-QRS-T wave recognition. *UHD Journal of Science and Technology, 2*(2), 7–14. https://doi.org/10.21928/uhdjst.v2n2y2018.pp7-14
- Clifford, G. D., Liu, C. Y., Moody, B., Lehman, L. H., Silva, I., Li, Q., Johnson, A. E. W., & Mark, R. G. (2017). AF classification from a short single lead ECG recording: The PhysioNet/Computing in Cardiology Challenge 2017. *Computing in Cardiology Conference*, *44*, 1–4. https://doi.org/10.22489/CinC.2017.065-469
- Clopton, E., & Hyrkäs, E. K. (2024). Assessing the Accuracy of ECG Chest Electrode Placement by EMS and Clinical Personnel Using Two Evaluation Methods. *International Journal of Paramedicine, 6*, 29–47. https://doi.org/10.56068/JGDQ2473
- Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). *ImageNet: A Large-Scale Hierarchical Image Database*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 248–255. https://doi.org/10.1109/CVPR.2009.5206848
- Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics, 36*(4), 193-202. https://doi.org/10.1007/BF00344251
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS 2014), 27*, 2672–2680. https://proceedings.neurips.cc/paper_files/paper/2014/hash/f033ed80deb0234979a61f95710dbe25-Abstract.html
- Hannun, A. Y., Rajpurkar, P., Haghpanahi, M., Tison, G. H., Bourn, C., Turakhia, M. P., & Ng, A. Y. (2019). Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, *25*(1), 65–69. https://doi.org/10.1038/s41591-018-0268-3
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770-778. https://doi.org/10.1109/CVPR.2016.90
- Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation, 18*(7), 1527-1554. https://doi.org/10.1162/neco.2006.18.7.1527
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

- Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700-4708. https://doi.org/10.1109/CVPR.2017.243
- Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology, 148*(3), 574-591. https://doi.org/10.1113/jphysiol.1959.sp006308
- Jeong, E. Y., & Lim, Y. G. (2021). Convolutional neural network for classification of eight types of arrhythmia using 2D time–frequency feature map from standard 12-lead electrocardiogram. *Scientific Reports, 11*, 20351. https://doi.org/10.1038/s41598-021-99975-6
- Jun, T. J., Nguyen, H. T., Kang, D., & Kim, D. (2018). ECG arrhythmia classification using a 2-D convolutional neural network. *arXiv preprint arXiv:1804.06812*. https://arxiv.org/abs/1804.06812
- Kligfield, P., Gettes, L. S., Bailey, J. J., Childers, R., Deal, B. J., Hancock, E. W., van Herpen, G., Kors, J. A., Macfarlane, P., Mirvis, D. M., Pahlm, O., Rautaharju, P. M., & Wagner, G. S. (2007). Recommendations for the standardization and interpretation of the electrocardiogram. *Journal of the American College of Cardiology*, *49*(10), 1109–1127. https://doi.org/10.1016/j.jacc.2007.01.024
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems, 25*, 1097-1105. https://doi.org/10.1145/3065386
- Lara Prado, J. I. (2016). El electrocardiograma: una oportunidad de aprendizaje. *Revista de la Facultad de Medicina (México)*, *59*(6), 39-45. https://www.scielo.org.mx/pdf/facmed/v59n6/2448-4865-facmed-59-06-39.pdf
- LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324. https://doi.org/10.1109/5.726791
- Li, C. L., Hong, H. T., Pan, C. Y., & Li, T. S. (2021). DeepECG: Image-based electrocardiogram interpretation with deep convolutional neural networks. *Biomedical Signal Processing and Control, 69*, 102824. https://doi.org/10.1016/j.bspc.2021.102824
- Madan, P., Krishnan, M., & Muthukumaran, V. (2022). A hybrid deep learning approach for ECG-based arrhythmia classification. *Bioengineering, 9*(4), 152. https://doi.org/10.3390/bioengineering9040152
- Olanrewaju, R. F., Ibrahim, S. N., Asnawi, A. L., & Altaf, H. (2021). Classification of ECG signals for detection of arrhythmia and congestive heart failure based on continuous wavelet transform and deep neural networks. *Indonesian Journal of Electrical Engineering and Computer Science*, *22*(3), 1520–1528. https://doi.org/10.11591/ijeecs.v22.i3.pp1520-1528
- Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345-1359. https://doi.org/10.1109/TKDE.2009.191
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536. https://doi.org/10.1038/323533a0

- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). *Going deeper with convolutions*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2015)* (pp. 1–9). https://doi.org/10.1109/CVPR.2015.7298594
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). *Rethinking the Inception Architecture for Computer Vision*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2818–2826. https://doi.org/10.1109/CVPR.2016.308
- World Health Organization. (2021). *Cardiovascular diseases (CVDs)*. https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)
- Yoon, T., & Kang, D. (2023). Bimodal CNN for cardiovascular disease classification by co-training ECG grayscale images and scalograms. *Scientific Reports, 13*(1), 4789. https://doi.org/10.1038/s41598-023-30208-8
- Zheng, J., Zhang, J., Danioko, S., Selles, M., Pérez Alday, E. A., Li, Q., & Clifford, G. D. (2020). A 12-lead ECG database for arrhythmia research covering more than 10,000 patients. *Scientific Data, 7*(1), 48. https://doi.org/10.1038/s41597-020-0386-x

8 Anexo I. Fundamentos y conceptos básicos de Deep Learning y redes neuronales

8.1 Introducción al deep learning

El *deep learning* (DL), aprendizaje profundo, es una subdisciplina del aprendizaje automático (*machine learning*) y de la inteligencia artificial.

8.1.1 Historia

El desarrollo del DL ha sido influenciado por varios hitos históricos:

- **Décadas de 1940 y 1950**: Los primeros modelos matemáticos de neuronas fueron propuestos por Warren McCulloch y Walter Pitts en 1943, conocidos como la neurona de McCulloch-Pitts. En 1958, Frank Rosenblatt presentó el perceptrón, un modelo de red neuronal capaz de aprender.
- **Década de 1980**: Kunihiko Fukushima diseñó por primera vez, en 1980, las redes neuronales convolucionales para tareas de reconocimiento de patrones en imágenes, desarrollando una red neuronal artificial llamada *Neocognitron* (Fukushima, 1980). La popularización del algoritmo *Backpropagation* por David E. Rumelhart, Geoffrey E. Hinton y Ronald J. Williams en 1986 permitió el entrenamiento de redes neuronales con más de una capa, conocidas como redes neuronales profundas (Rumelhart et al., 1986). En 1989, Yann LeCun presentó la primera red neuronal convolucional de la historia entrenada mediante el algoritmo Backpropagation, llamada *LeNet-5* (LeCun et al., 1998).
- **Década 1990**: La inteligencia artificial IBM Deep Blue venció a Garry Kaspárov, campeón mundial de ajedrez.
- **Década 2000**: Geoffrey E. Hinton, Simon Osindero y Yee-Whye Teh publicaron en 2006 un trabajo que permitió entrenar redes con un número mayor de capas, lo que motivó a otros investigadores a explorar redes más profundas (Hinton et al., 2006). En 2008 se comienzan a crear las primeras GPUs para el entrenamiento de la red neuronal.
- **Año 2012**: La red neuronal convolucional *AlexNet* (Krizhevsky et al., 2012), creada por Geoffrey Hinton y su alumno Alex Krizhevsky, ganó la competición *ImageNet*, reduciendo significativamente el error de clasificación de imágenes y popularizando el uso de redes neuronales convolucionales y DL.
- **Año 2014**: Ian Goodfellow crea las redes neuronales generativas adversarias, que compiten entre sí en un juego para aprender a generar nuevos datos con las mismas estadísticas que el conjunto de entrenamiento (Goodfellow et al., 2014).
- **Años 2015-2016**: Redes como ResNet (He et al., 2016) y DenseNet (Huang et al., 2017) introdujeron innovaciones como las *skip connections* y *dense blocks*, mejorando aún más los resultados en competiciones como *ImageNet*. En 2016, AlphaGo,

desarrollado por DeepMind, venció al campeón mundial de Go, demostrando el potencial del DL en juegos complejos (Silver et al., 2016).

- **Año 2018**: Yann LeCun, Geoffrey Hinton y Yoshua Bengio fueron galardonados con el Premio Turing por sus contribuciones al campo del DL.
- **Año 2022**: Google y DeepMind lanzan sus dos modelos de IA (IMAGEN y *Dall-E mini*) que pueden crear imágenes originales a partir de textos proporcionados por los usuarios.

El DL ha revolucionado campos como la visión por computador, el procesamiento del lenguaje natural y la conducción autónoma, gracias a su capacidad para extraer características automáticamente de los datos y mejorar continuamente con más datos y mayor capacidad de cómputo.

8.1.2 Tipos de tecnologías deep learning actuales más aceptadas

8.1.2.1 Redes neuronales

Las **redes neuronales** imitan, desde un punto de vista matemático, el comportamiento del cerebro humano.

Los componentes básicos de las redes neuronales son:

1. **Neurona Artificial**: La unidad básica de una red neuronal. Recibe entradas, las transforma a través de una función de activación y genera una salida.
2. **Capas**: Como podemos ver en la Figura 8.1, las neuronas se organizan en capas:
    - **Capa de entrada**.
    - **Capas ocultas**.
    - **Capa de salida**.

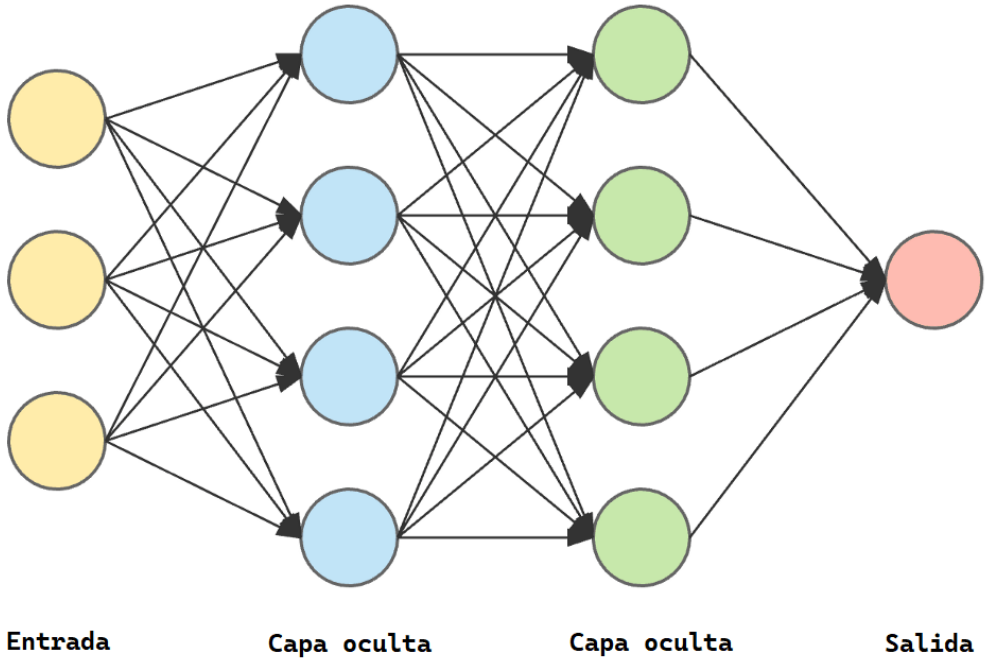

Figura 8.1: Arquitectura de una red neuronal.
Fuente: https://openwebinars.net/blog/que-son-las-redes-neuronales-y-sus-aplicaciones/

Las funciones de activación incorporan características no lineales en la red que le permiten manejar relaciones más sofisticadas.

El proceso de entrenamiento de una red neuronal consiste en modificar los pesos entre las neuronas para reducir al máximo el error en las predicciones.

Las redes neuronales se utilizan en una gran variedad de aplicaciones, incluyendo:

- **Visión por computador**.
- **Procesamiento del lenguaje natural**.
- **Conducción autónoma**.

Las redes neuronales han demostrado ser herramientas poderosas para resolver problemas complejos en diversas áreas, gracias a su capacidad para aprender y generalizar a partir de grandes cantidades de datos.

8.1.2.2 Redes neuronales convolucionales

Las **redes neuronales convolucionales** (CNN) son un tipo de red neuronal inspirada en la corteza visual del cerebro, que procesa la información visual de manera jerárquica. Estas redes son especialmente efectivas para el procesamiento de imágenes y han revolucionado el campo del DL.

La arquitectura fundamental de las CNN se compone de varios tipos de capas:

- **Capa de convolución**.
- **Capa de pooling**.

En la Figura 8.2 podemos observar las diferentes capas.

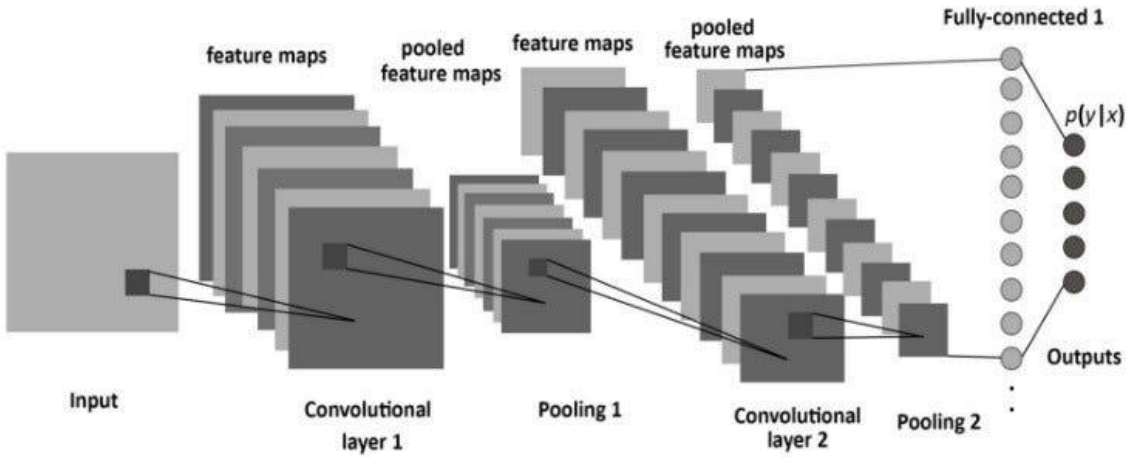

Figura 8.2: Arquitectura de una red neuronal convolucional.
Fuente: https://data-universe.org/redes-neuronales-convolucionales-aplicaciones-en-procesamiento-de-imagenes/

Las CNN se utilizan en una amplia variedad de aplicaciones, incluyendo:

- **Clasificación de imágenes**.
- **Detección de objetos**.
- **Segmentación de imágenes**.
- **Procesamiento de audio y texto**.

Estas redes han demostrado ser extremadamente efectivas en tareas de visión por computador y se han adaptado para una variedad de campos, gracias a su capacidad para extraer características de manera automática y eficiente.

8.1.2.3 Redes recurrentes y LSTM

Las **redes** neuronales **recurrentes** (RNN) son un tipo de red neuronal diseñada para procesar datos secuenciales, como series temporales, texto y audio. A diferencia de las redes neuronales tradicionales, las RNN tienen conexiones que permiten la retroalimentación, lo que les da una memoria (o estado oculto) para recordar información de pasos anteriores en la secuencia.

Las RNN se componen de neuronas que procesan una entrada en cada paso de tiempo y mantienen un estado oculto que se actualiza en cada iteración. En la Figura 8.3 podemos ver una representación visual de este tipo de redes.

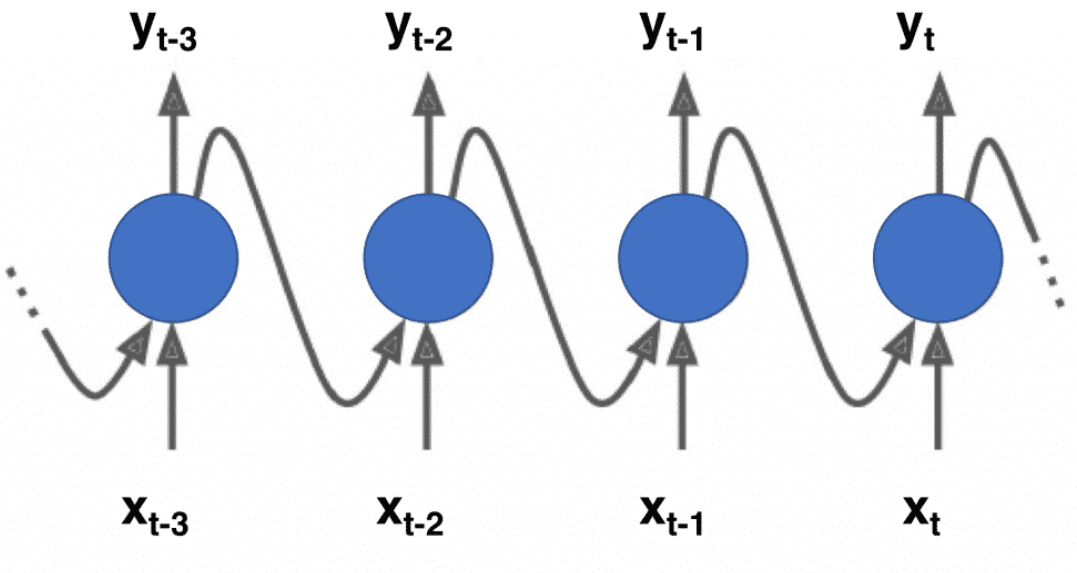

Figura 8.3: Representación de una red recurrente.
Fuente: https://torres.ai/redes-neuronales-recurrentes/

Las redes de memorias de corto y largo plazo (**LSTM**) son una variante de las RNN diseñadas para abordar problemas como el decaimiento y la explosión del gradiente (memoria de corto plazo) (Hochreiter & Schmidhuber, 1997). Las LSTM introducen un mecanismo de puertas que controlan el flujo de información a través de la red, permitiendo que la red recuerde información durante períodos de tiempo más largos. En la Figura 8.4 se puede ver una representación de todo esto.

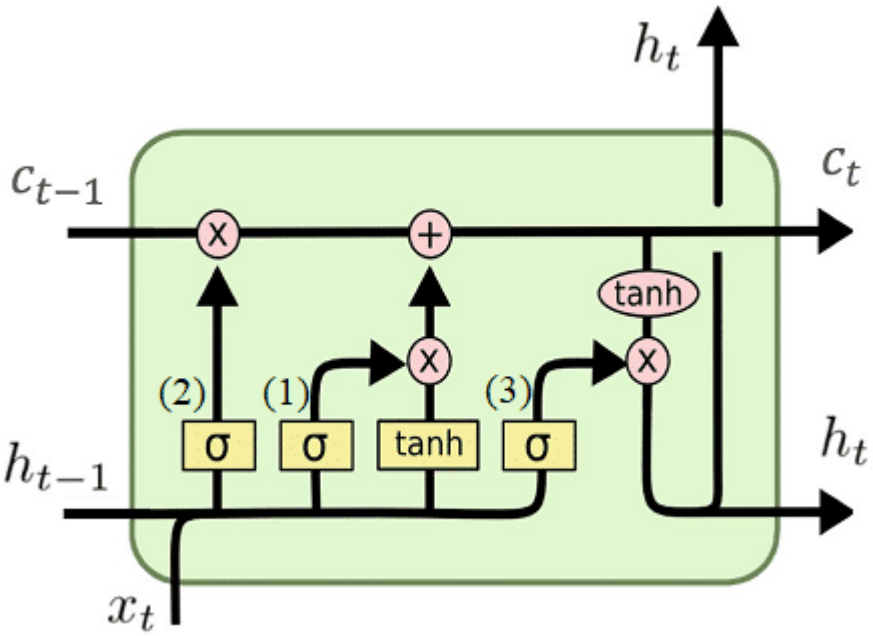

Figura 8.4: Diagrama de una célula de memora LSTM.
Fuente: https://datascientest.com/es/memoria-a-largo-plazo-a-corto-plazo-lstm

Las RNN y LSTM son herramientas poderosas para el procesamiento de datos secuenciales y han demostrado ser efectivas en una diversidad de aplicaciones, desde la traducción automática hasta el reconocimiento de voz y el análisis de sentimientos.

8.1.2.4 Transformers

Estos modelos son capaces de traducir texto y voz prácticamente en tiempo real, aunque también se emplean en diversas tareas relacionadas con la visión por ordenador.

Los transformadores procesan las secuencias de entrada de forma paralela, lo que los hace altamente eficientes. A diferencia de otras tecnologías, como las redes recurrentes (RNN) o las LSTM, su eficiencia no depende únicamente de agregar más GPU. Además, requieren menos tiempo de entrenamiento en comparación con estas arquitecturas más antiguas.

En la Figura 8.5 se puede observar la arquitectura de un Transformer.

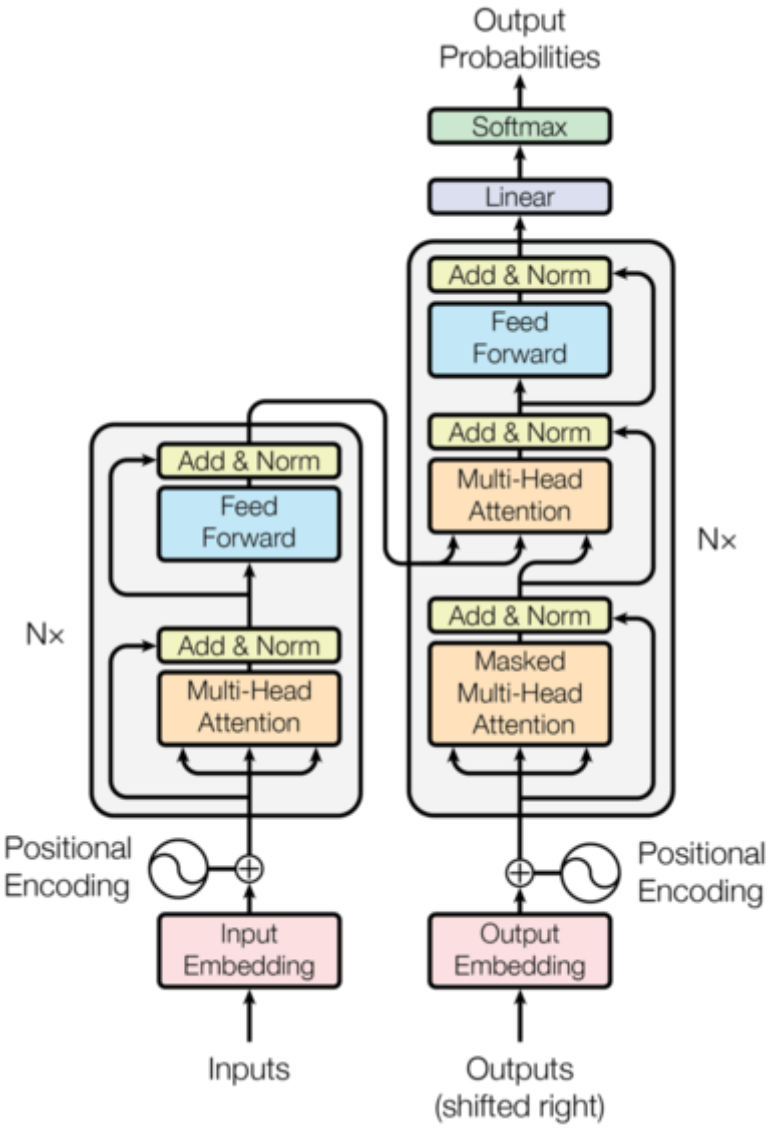

Figura 8.5: Arquitectura de un Transformer.

Fuente: https://www.aprendemachinelearning.com/como-funcionan-los-transformers-espanol-nlp-gpt-bert/

8.1.3 Objetivos de aplicación sobre imágenes

8.1.3.1 Clasificación

La **clasificación** de imágenes consiste en asignar una etiqueta o categoría a una imagen completa basándose en el contenido visual presente en ella. En este proceso, el modelo analiza la imagen en su totalidad para identificar qué clase representa, sin enfocarse en la localización o el conteo de los objetos. Un ejemplo sencillo sería un modelo entrenado para clasificar imágenes de animales como “perro” o “gato”.

8.1.3.2 Detección

La **detección** de objetos combina la localización de objetos y la clasificación de imágenes, generando rectángulos delimitadores de los objetos. En el ejemplo anterior de clasificación entre “perro” o “gato”, con la detección, podríamos localizar en que parte de la foto se encuentra el animal. En la Figura 8.6 se observa la diferencia entre clasificación y detección.

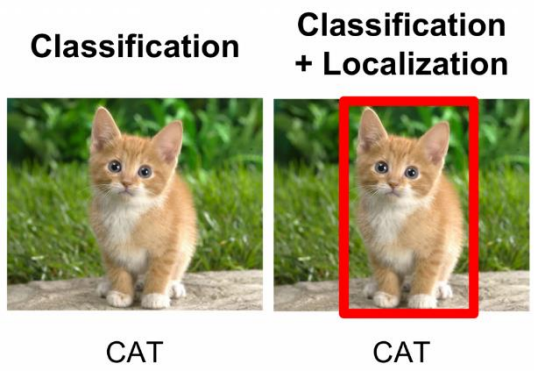

Figura 8.6: Diferencia entre clasificación y detección.

Fuente: https://colab.research.google.com/github/juansensio/blog/blob/master/045_cv_tareas/cv_tareas.ipynb

8.1.3.3 Segmentación

La **segmentación** de imágenes es una técnica que se encarga de agrupar píxeles en grupos diferentes según una serie de características visuales, como pueden ser el nivel de intensidad de los píxeles, el brillo, el color... Con esto se logra identificar los límites de los objetos y las regiones de fondo, como podemos ver en la Figura 8.7, donde las ovejas se diferencian del perro.

8.1.3.4 Segmentación por instancias

La **segmentación por instancias** consiste en clasificar los píxeles de una imagen, no solo según su clase, sino también distinguiendo a qué objeto o individuo pertenecen. Ejemplo: si tenemos una imagen con varias sillas y varias mesas, se tendrán las etiquetas “mesa 1”, “mesa 2”, “mesa 3”, “silla 1”, “silla 2”, “silla 3” … Para ello, se realizará, primero, un proceso de detección y, después, una etapa de segmentación del objeto o individuo principal de cada detección. En la Figura 8.8 se puede apreciar que, a diferencia de la Figura 8.7, en esta imagen se distinguen claramente las ovejas entre sí.

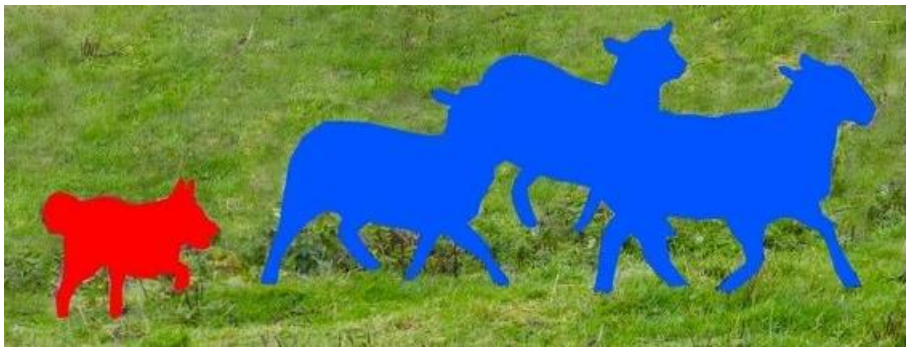

Figura 8.7: Ejemplo de segmentación de animales.

Fuente: https://www.reddit.com/r/learnmachinelearning/comments/kt0hov/difference_in_image_classification_semantic/?tl=es-es

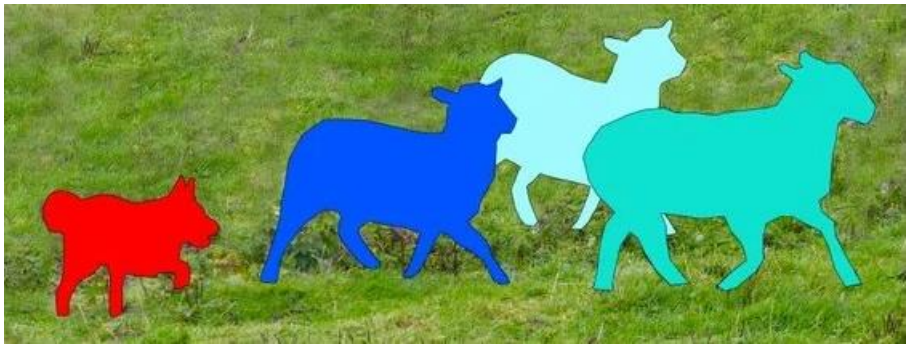

Figura 8.8: Ejemplo de segmentación por instancias de animales.

Fuente: https://www.reddit.com/r/learnmachinelearning/comments/kt0hov/difference_in_image_classification_semantic/?tl=es-es

8.2 Redes neuronales

8.2.1 Fundamentos

8.2.1.1 Perceptrón

El **perceptrón** es uno de los modelos más básicos y fundamentales en el ámbito de las redes neuronales. Inspirado en el funcionamiento de las neuronas biológicas, como se puede

observar en la Figura 8.9, su diseño matemático busca imitar cómo las dendritas, el soma y el axón procesan información.

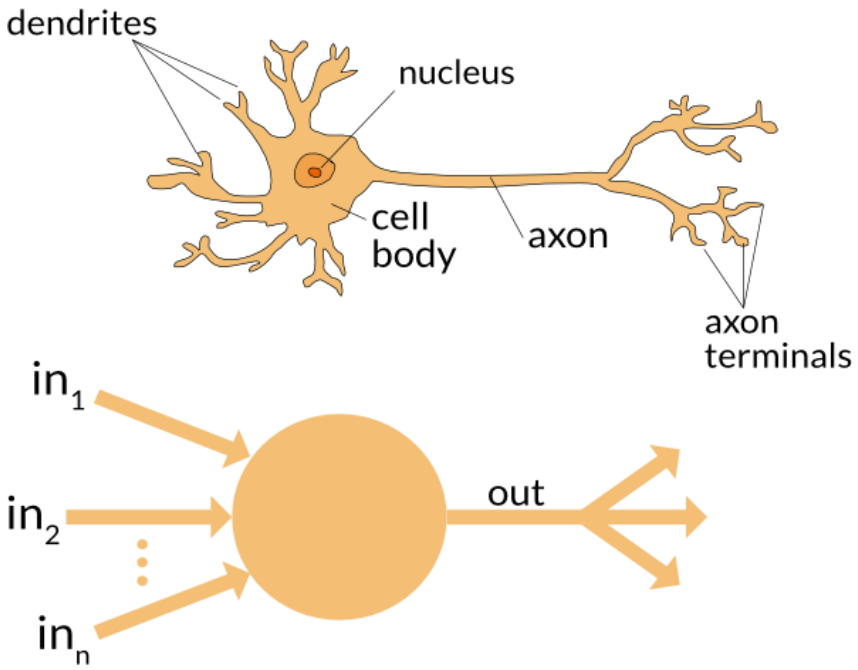

Figura 8.9: Semejanza de una neurona artificial a una neurona biológica.
Fuente: https://appliedgo.net/perceptron/

El perceptrón recibe un conjunto de entradas (x_1, x_2, x_3, …, x_n), cada una asociada a un peso (w_1, w_2, w_3, …, w_n) que representa la importancia de dicha entrada en el modelo. A partir de estas entradas y sus pesos, el perceptrón calcula una suma ponderada:

$$y = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b \tag{8.1}$$

Donde b es el término de sesgo que permite ajustar el hiperplano de decisión. La salida del perceptrón se obtiene al aplicar una función de activación, como la función escalón o sigmoide, que determina si la neurona se activa (1) o no (0) dependiendo de si la suma ponderada supera un umbral.

Inicialmente, el perceptrón se utilizaba para resolver problemas de clasificación binaria, como distinguir entre dos categorías de datos. Sin embargo, su capacidad es limitada a problemas linealmente separables. Un caso conocido que refleja esta limitación se puede ver en la Figura 8.10, donde el perceptrón es incapaz de resolver la función XOR, ya que no es linealmente separable.

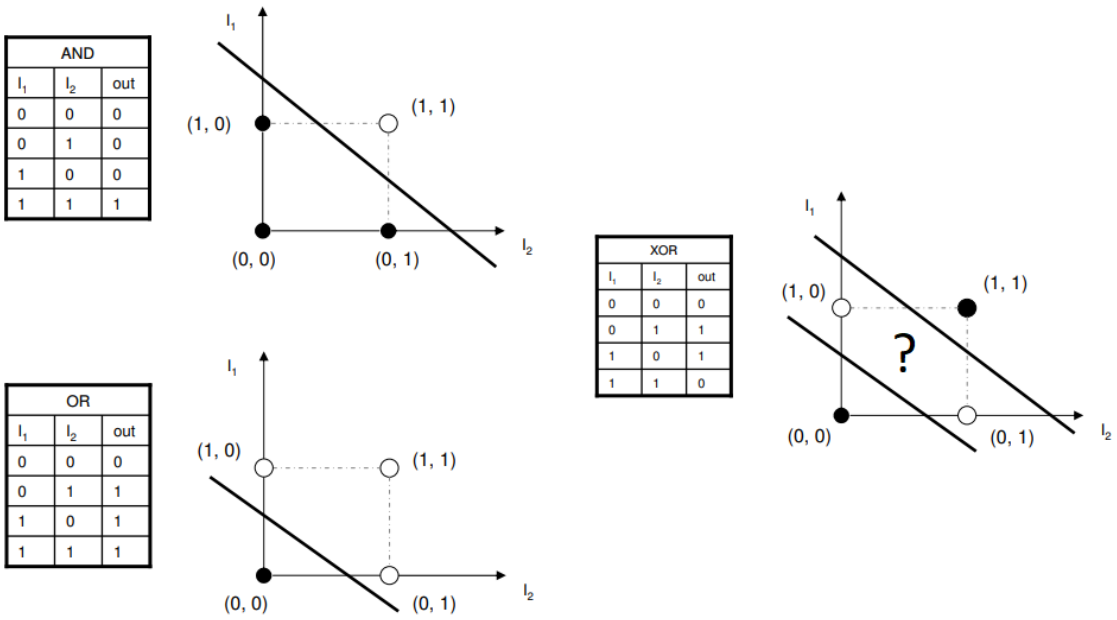

Figura 8.10: Representación gráfica de funciones lógicas y la no linealidad del XOR.
Fuente: https://medium.com/@lucaspereira0612/solving-xor-with-a-single-perceptron-34539f395182

8.2.1.2 Funciones de activación

Las **funciones de activación** son un componente fundamental en las redes neuronales. Permiten que las redes puedan aprender patrones complejos en los datos y resolver problemas que no son linealmente separables. Las funciones de activación actúan transformando la salida de las neuronas, sin modificar el hiperplano del perceptrón, lo que determina si estas se activan o no, y modulan cómo se transmiten las señales entre capas.

Las funciones de activación más utilizadas son:

- **Identidad**:

$$f(x) = x \tag{8.2}$$

  Devuelve la entrada x como salida, como podemos ver en la Figura 8.11.

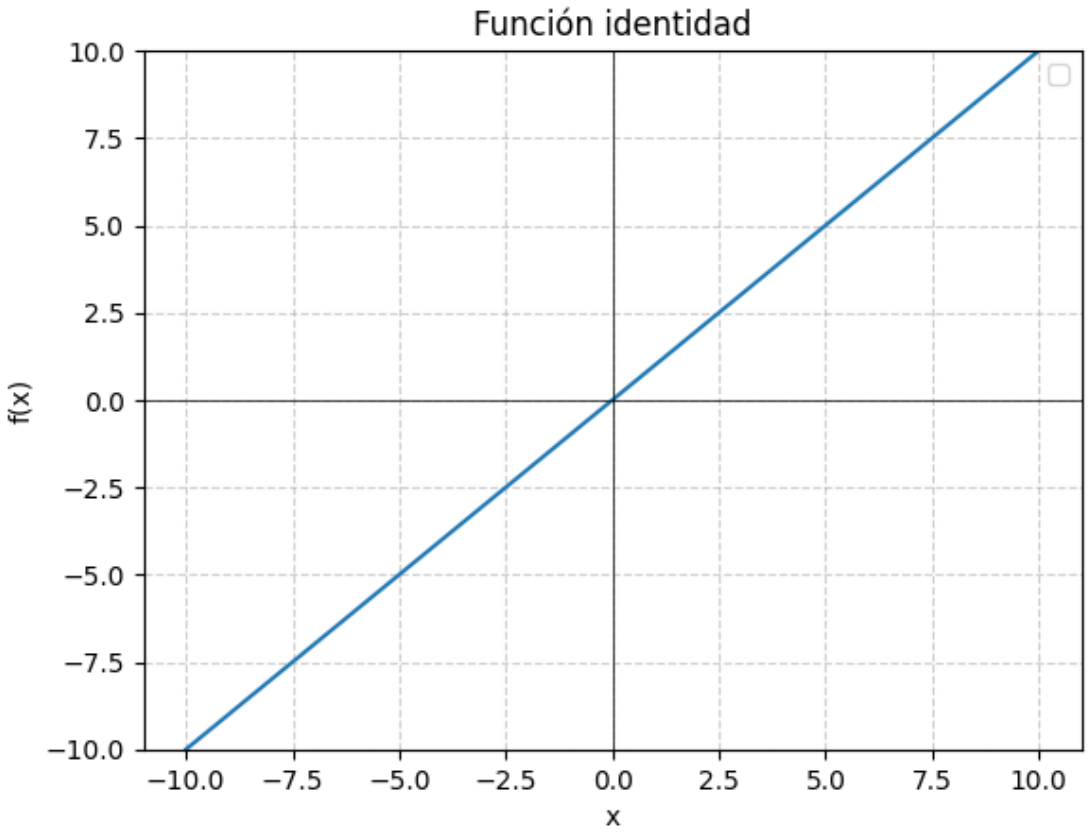

Figura 8.11: Gráfica de la función de activación identidad.

Fuente: Elaboración propia.

- **Función umbral o escalonada**: define una activación binaria (la neurona se activa (1) o no se activa (0), dependiendo de si la entrada supera un umbral). Este tipo de función tiene limitaciones, ya que es discontinua y no permite calcular gradientes, lo que dificulta el entrenamiento de modelos más avanzados. En la Figura 8.12 podemos ver la gráfica de esta función.

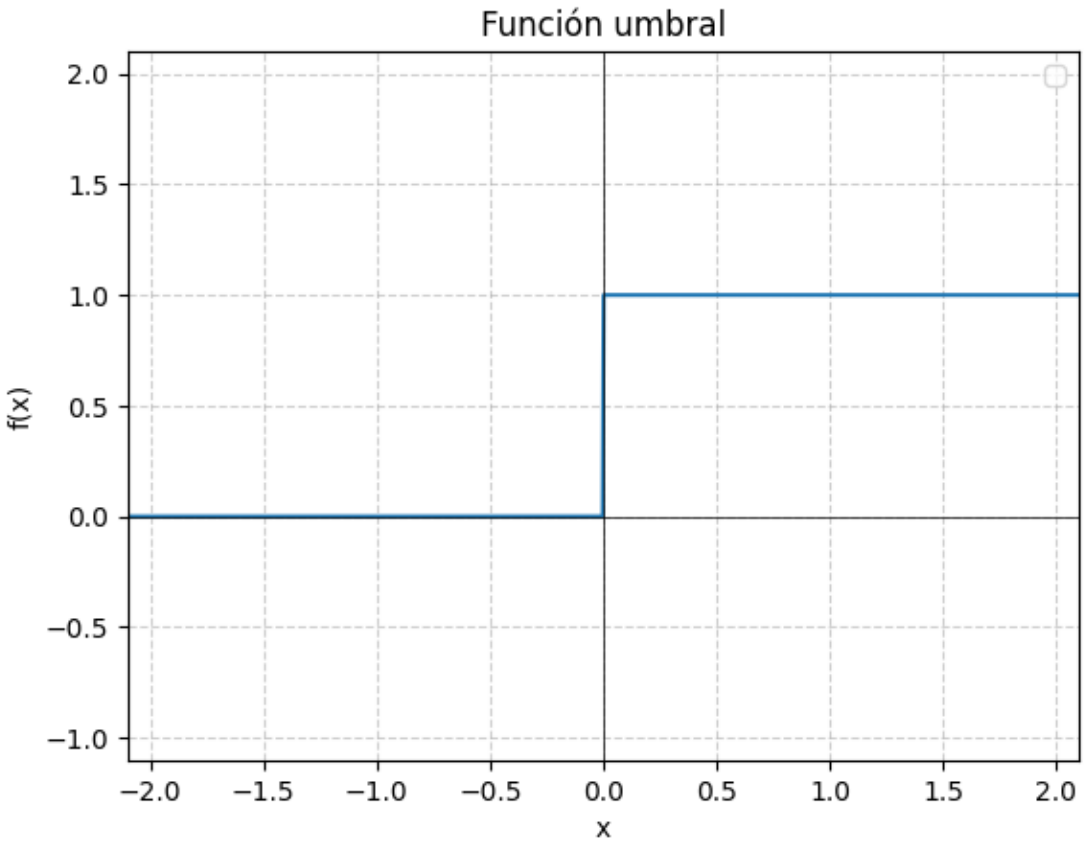

Figura 8.12: Gráfica de la función de activación umbral.
Fuente: Elaboración propia.

- **Sigmoide o logística**:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{8.3}$$

Como podemos observar en la Figura 8.13, su salida es un valor entre 0 y 1, ideal para representar probabilidades. Cuando la entrada es un valor negativo, la salida tiende a 0, y en caso contrario, la salida tiende a 1. La desventaja que tiene esta función es que puede saturarse (gradiente cercano a cero) y afectar al aprendizaje.

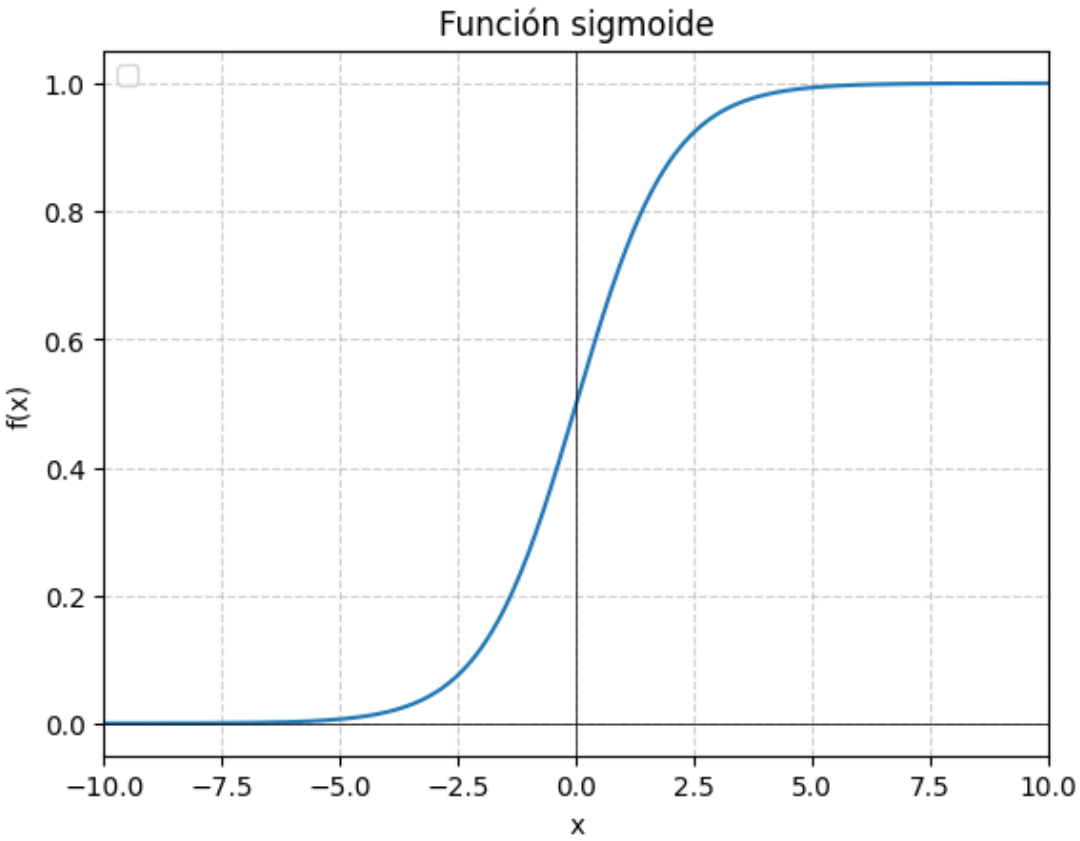

Figura 8.13: Gráfica de la función de activación sigmoide.

Fuente: Elaboración propia.

- **Tangente hiperbólica (*tanh*)**:

$$f(x) = \tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{8.4}$$

Su rango es entre -1 y 1, lo que centra las salidas alrededor de 0 como podemos comprobar en la Figura 8.14 Su ventaja con respecto a la función sigmoide es que, con este rango se tiene gradientes más fuertes y esto puede ayudar a que el entrenamiento converja más rápidamente. Aun así, se sigue teniendo el problema de desvanecimiento de gradiente.

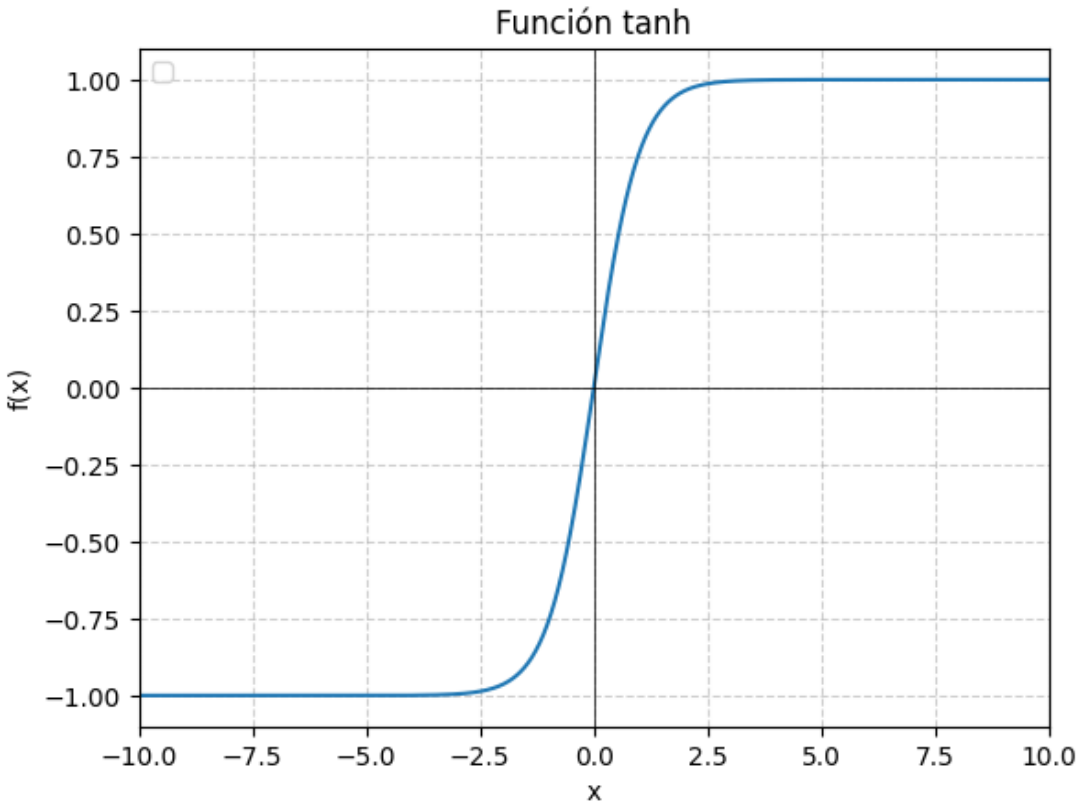

Figura 8.14: Gráfica de la función de activación hiperbólica.

Fuente: Elaboración propia.

- ***ReLU* (Unidad Lineal Rectificada)**:

$$f(x) = \max(0, x) \tag{8.5}$$

Con esta función, las entradas positivas no serán modificadas y, por tanto, el gradiente tampoco. Esto hace que se reduzca el problema de desvanecimiento de gradiente. Sin embargo, puede sufrir de “neuronas muertas” (gradiente cero para valores negativos). En la Figura 8.15 se puede observar la gráfica de esta función.

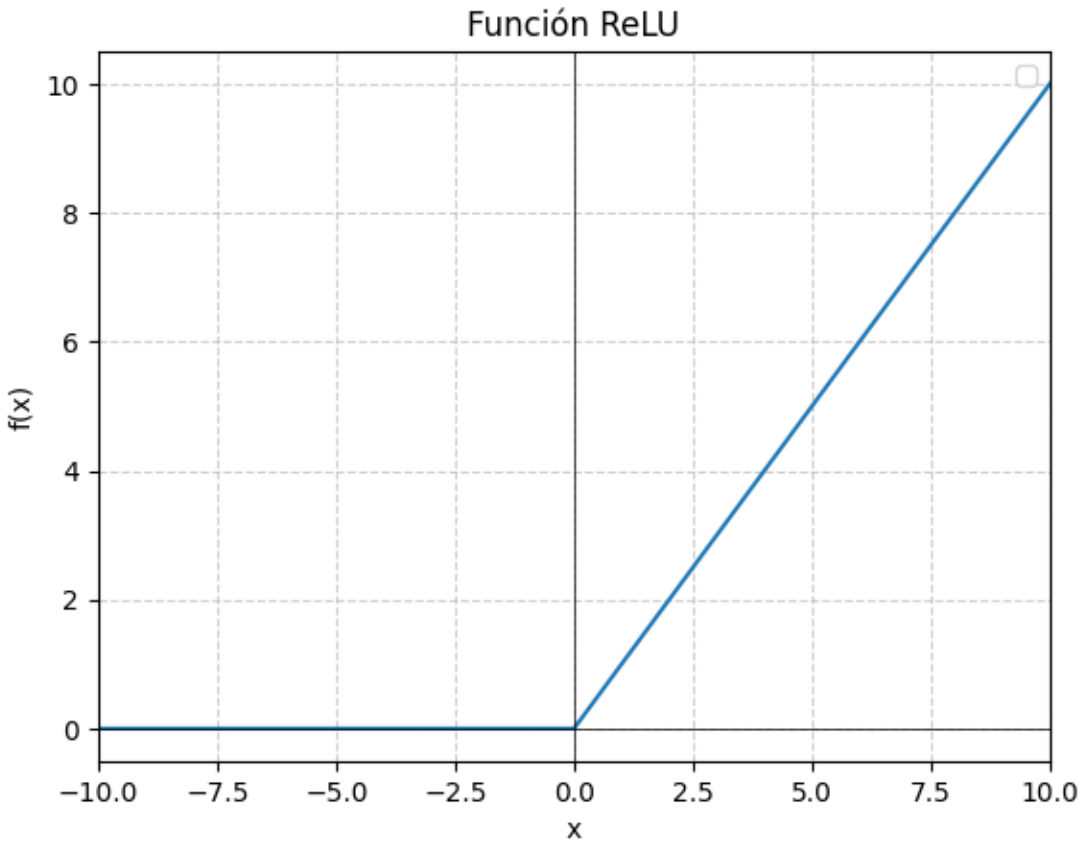

Figura 8.15: Gráfica de la función de activación ReLU.
Fuente: Elaboración propia.

- ***Softmax***:

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{8.6}$$

Es utilizada en la capa de salida de redes para clasificación multiclase. Genera una distribución de probabilidad sobre las clases posibles. La gráfica de esta función se puede ver en la Figura 8.16.

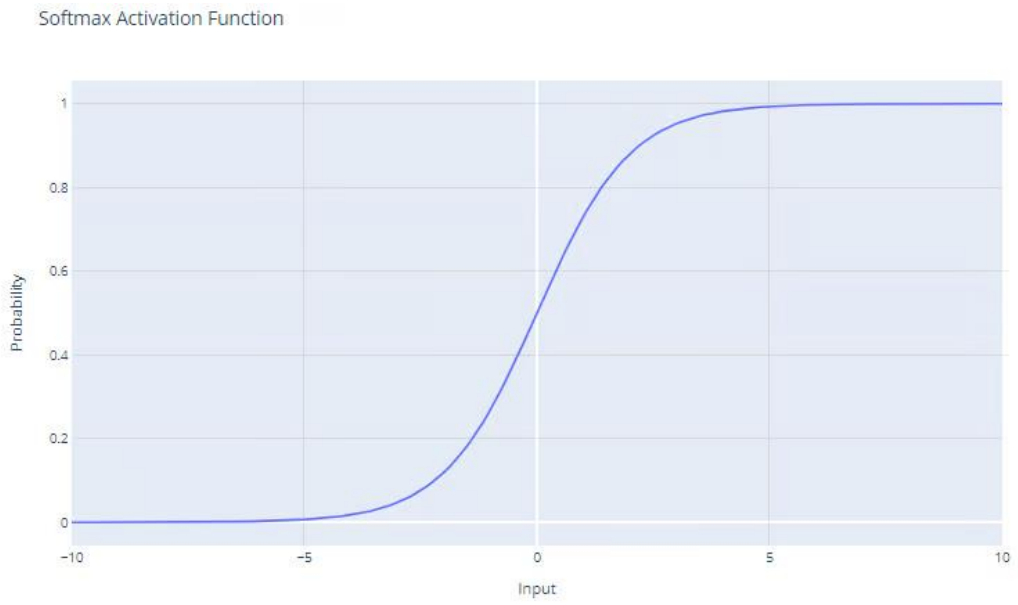

Figura 8.16: Gráfica de la función de activación softmax.
Fuente: https://www.datacamp.com/es/tutorial/introduction-to-activation-functions-in-neural-networks

8.2.1.3 Arquitectura (Capas y Redes neuronales)

La **arquitectura** de una red neuronal define la estructura y disposición de las neuronas en diferentes **capas**, así como las conexiones entre ellas. Dentro de las **redes neuronales**, el primer componente distintivo es la capa, que agrupa neuronas que reciben las mismas entradas.

Los tipos de capa que hay son:

- **Capa de entrada**: Es la primera capa de la red y se encarga de recibir las características del conjunto de datos de entrada. Cada neurona de esta capa representa una característica del conjunto de datos.
- **Capas ocultas**: Estas capas están compuestas por neuronas que realizan cálculos sobre las entradas recibidas, pasan el resultado por una función de activación y lo pasan a la siguiente capa. Una red neuronal puede tener una o más capas ocultas.
- **Capa de salida**: Es la última capa de la red y genera la predicción final de la red utilizando el mismo método que emplean las capas ocultas para calcular sus salidas. Puede tener una o varias neuronas, dependiendo de si tenemos un problema de clasificación binaria, donde la salida es 0 o 1, o de clasificación multiclase.

Las redes neuronales con una sola capa oculta, como la de la Figura 8.17, se denominan redes neuronales simples, mientras que las redes neuronales con más de una capa oculta se denominan redes neuronales profundas.

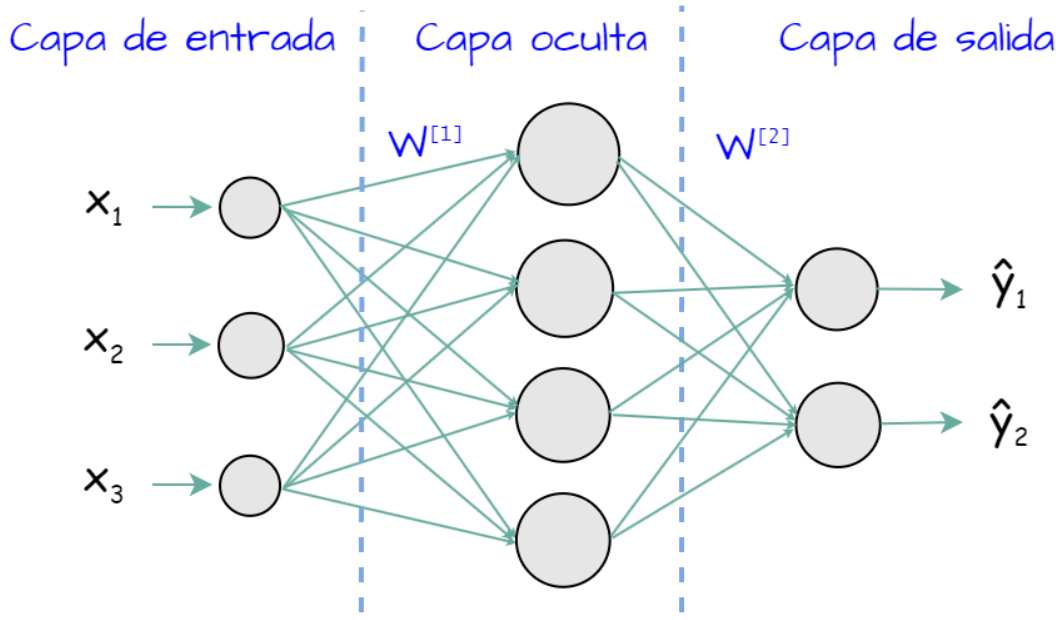

Figura 8.17: Arquitectura de una red neuronal simple.
Fuente: https://playbyte.es/articulos/ia/20240528_redes-neuronales4/mlp.html

8.2.1.4 Aplicación a problemas de clasificación: softmax y codificación one-hot

La función ***softmax*** es una función de activación crucial en la capa de salida de una red neuronal diseñada para problemas de clasificación multiclase.

Esta función toma un vector de números reales como entrada y lo convierte en un vector de probabilidades que suman 1. Cada elemento del vector de salida representa la probabilidad de que la entrada pertenezca a una determinada clase.

Se utiliza porque permite interpretar los resultados generados por la red como una estimación probabilística de pertenencia a cada clase.

La **codificación *one-hot*** es una técnica que convierte el vector de probabilidades anterior en un vector binario. En este vector, solo una posición tendrá el valor 1 (indicando la clase a la que pertenece la muestra), mientras que el resto serán 0. La clase con la probabilidad más alta se selecciona como la predicción final (clasificación).

Ejemplo:

$$y = \begin{bmatrix} 2 \\ 1 \\ 0{,}1 \end{bmatrix} \tag{8.7}$$

Softmax

$$S(y) = \begin{bmatrix} 0{,}7 \\ 0{,}2 \\ 0{,}1 \end{bmatrix} \tag{8.8}$$

One-hot

$$C = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{8.9}$$

8.2.1.5 Aplicación a imágenes: flattening

El ***flattening*** es un proceso que se usa para poder procesar los datos que tengan dimensiones espaciales, como las **imágenes**, debido a que las redes neuronales tienen como entrada y como salida un vector.

Esta técnica lo que hace es convertir los datos en una representación vectorial, descomponiendo las dimensiones espaciales de los datos de entrada para formar un vector. En el ejemplo de la Figura 8.18 se tiene una imagen de 3x3 (9 píxeles) que se convierte en un vector de 9 elementos, cada uno con el valor de cada píxel de la imagen.

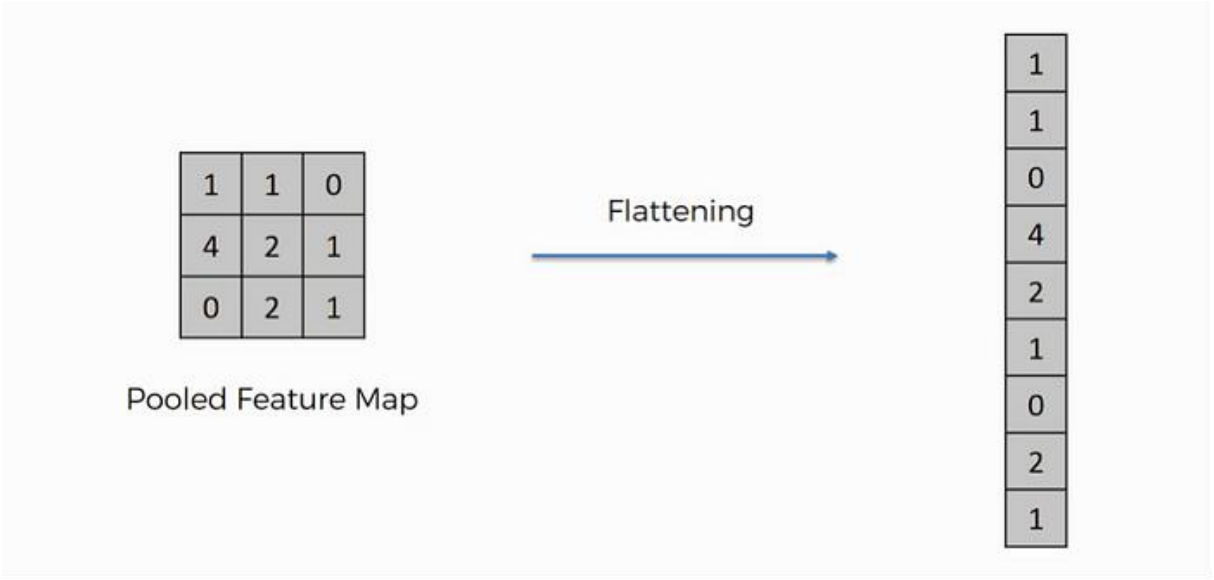

Figura 8.18: Ejemplo de flattening.
Fuente: https://bootcamp9.medium.com/redes-neuronales-convolucionales-5e0ce960caf8

8.2.2 Desarrollo de redes neuronales

8.2.2.1 Etapas de desarrollo

En las **etapas de desarrollo** de las redes neuronales podemos distinguir la de entrenamiento, la de validación y la de test o prueba.

1. Entrenamiento

El objetivo de esta etapa es ajustar los pesos de la red para que sea capaz de aproximar correctamente la función objetivo definida por los datos del conjunto de entrenamiento.

Durante esta etapa, la red neuronal utiliza un conjunto de datos llamado "conjunto de entrenamiento" para ajustar los pesos de sus neuronas mediante un proceso iterativo de optimización. Este conjunto contiene ejemplos de entrada junto con las salidas esperadas, lo que permite a la red aprender una función que relacione ambas. El entrenamiento suele dividirse en épocas, donde cada época utiliza todos los datos de entrenamiento disponibles.

2. Validación

El objetivo de la validación es verificar la capacidad de generalización del modelo con datos no vistos previamente, evitar problemas como el sobreajuste, asegurando que el modelo no dependa únicamente de los datos de entrenamiento, y ayudar a determinar cuándo detener el entrenamiento.

En esta etapa se utiliza un conjunto de datos independiente llamado "conjunto de validación". Este conjunto no se emplea para actualizar los pesos de la red, sino para monitorear el progreso y tomar decisiones, como ajustar hiperparámetros o detener el entrenamiento si el modelo deja de mejorar.

3. Test o prueba

El objetivo de la etapa de prueba es el mismo que el de la etapa de validación, pero con la diferencia de que aquí se usa otro conjunto de datos diferente al finalizar el entrenamiento. Además, se hace una estimación definitiva del rendimiento del modelo y se evalúan métricas clave para determinar si la red cumple con los requisitos del problema.

Esta etapa evalúa el rendimiento final de la red neuronal utilizando un conjunto de datos conocido como "conjunto de prueba", que es independiente tanto del entrenamiento como de la validación. La red no tiene acceso a estos datos durante las etapas anteriores.

8.2.2.2 Etapa de entrenamiento: aspectos generales

8.2.2.2.1 Estimación del error: funciones de pérdida

La **estimación del error** en el entrenamiento de redes neuronales depende de las **funciones de pérdida**, que son herramientas fundamentales para medir la discrepancia entre la salida de la red y el valor esperado. Estas funciones permiten guiar el ajuste de los pesos de la red para disminuir progresivamente el error.

Las principales funciones de pérdida son:

- **Error cuadrático medio (ECM)**: Esta función es comúnmente utilizada en problemas de regresión. Consiste en calcular la media de las diferencias al cuadrado entre las salidas esperadas y las salidas generadas por la red. La fórmula general para un conjunto de datos de tamaño *n* y con *m* dimensiones es:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(SalidaRed^{i}{}_{j} - SalidaEsperada^{i}{}_{j}\right)^{2} \tag{8.10}$$

  Como se puede observar en la Figura 8.19, el ECM penaliza fuertemente las diferencias grandes debido a la operación de elevar al cuadrado, lo cual puede ser desfavorable en conjuntos de datos con outliers.

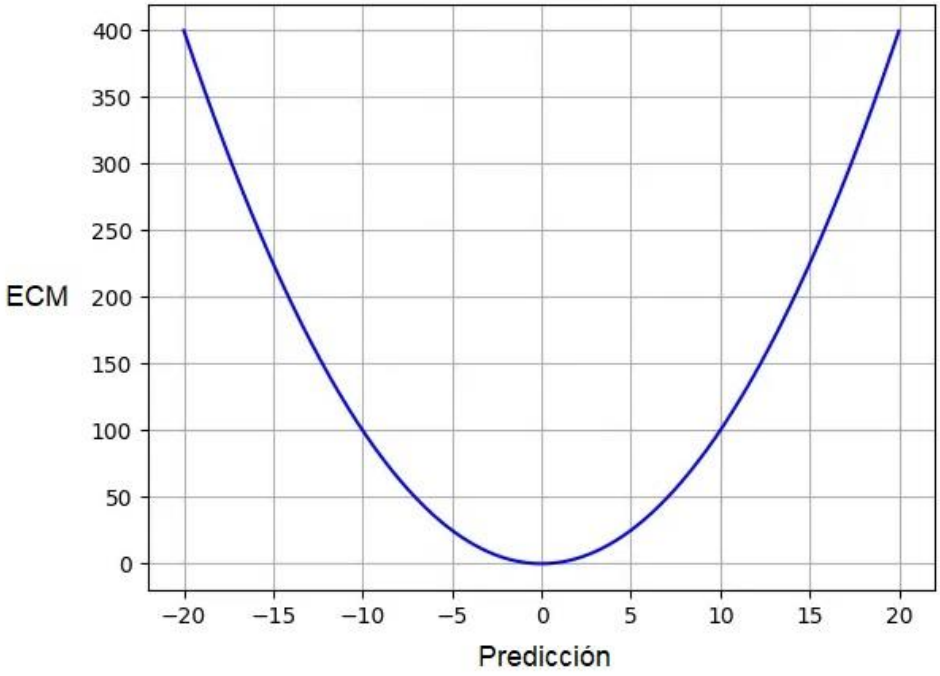

Figura 8.19: Gráfica de la función del error cuadrático medio.
Fuente: https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

- **Cross-entropy (CE)**: Se usa en problemas de clasificación en los que las salidas son distribuciones de probabilidad. La fórmula para calcularla con un conjunto de datos de tamaño *n* y *m* neuronas de salida es:

$$CE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} -\ln\left(SalidaReal^{i}{}_{j}\right) * SalidaEsperada^{i}{}_{j} \tag{8.11}$$

  Como la salida esperada es un vector binario donde todos los elementos están a 0 menos el de clase positiva que está a 1, la fórmula al final se quedará en la negación del logaritmo neperiano de la probabilidad dada para la clase positiva entre el número de ejemplos *n* del conjunto de datos.
  Como se puede apreciar en la Figura 8.20, esta función alcanza su valor mínimo (cero) cuando la predicción coincide exactamente con la salida esperada.

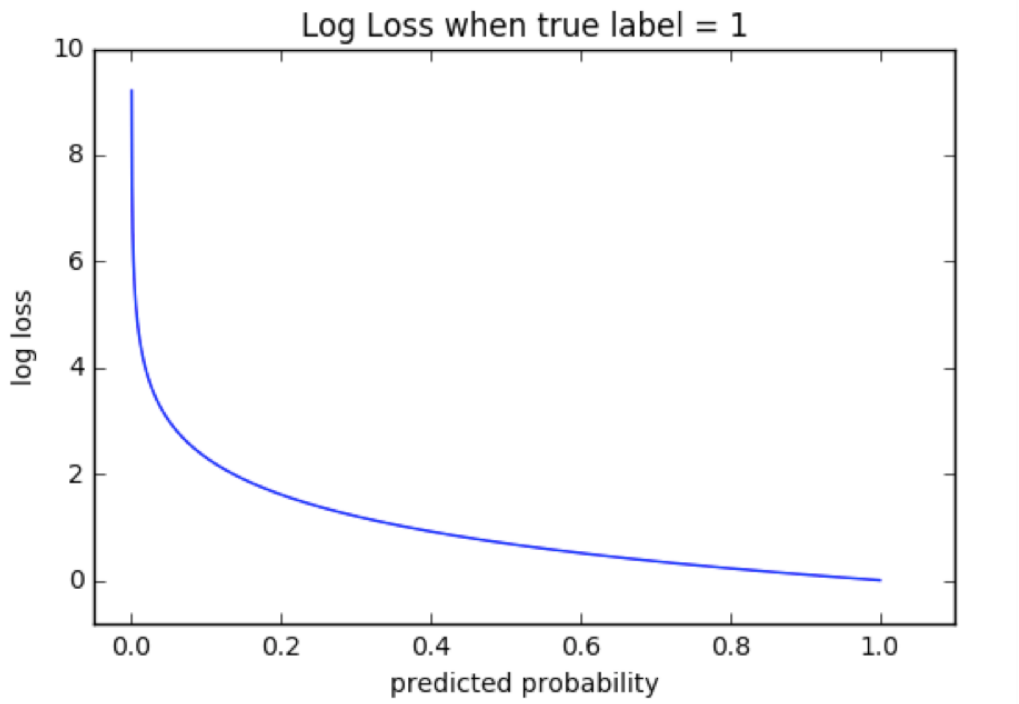

Figura 8.20: Gráfica de la función de Cross-Entropy.
Fuente: https://datascience.eu/es/matematica-y-estadistica/funciones-de-perdida/

Las funciones de pérdida deben seleccionarse cuidadosamente dependiendo del problema. Por ejemplo, el ECM es más adecuado para regresión, mientras que CE lo es para clasificación.

En algunos casos, se pueden diseñar funciones personalizadas o modificar las existentes para adaptarse a la naturaleza específica del problema, como en escenarios con outliers o distribuciones de datos desbalanceadas.

El propósito principal de estas funciones es proporcionar una métrica directa del error cometido por la red neuronal durante el entrenamiento. Esto permite ajustar los pesos para minimizar la pérdida y mejorar la capacidad de la red para generalizar en datos no vistos.

8.2.2.2.2 Inicialización de pesos: Xavier, He, *transfer learning*

La **inicialización de pesos** en las redes neuronales es una etapa crítica para garantizar un entrenamiento efectivo y evitar problemas como el desvanecimiento o explosión del gradiente. Dependiendo del tipo de red y de las funciones de activación utilizadas, se han desarrollado técnicas específicas para optimizar este proceso:

- **Xavier**: Esta inicialización busca abordar el problema de la saturación de las funciones de activación cuando los pesos son inicializados de manera puramente aleatoria. Consiste en asegurar que la varianza de los pesos de una neurona sea igual a 1, evitando que las salidas saturen las funciones de activación. Matemáticamente, los pesos w_i se inicializan con valores extraídos de una distribución normal:

$$w_i^j \sim N\left(0, \sqrt{\frac{1}{n}}\right) \tag{8.12}$$

donde n representa el número de entradas de la neurona *j*.

- **He**: La inicialización He, propuesta para redes neuronales profundas con funciones de activación *ReLU*, mejora la convergencia de redes profundas. Al igual que la inicialización Xavier, He utiliza una distribución normal para generar los pesos iniciales:

$$w_i^j \sim N\left(0, \sqrt{\frac{2}{n}}\right) \tag{8.13}$$

- ***Transfer learning***: La transferencia de aprendizaje se basa en reutilizar los pesos obtenidos por una red ya entrenada para resolver un nuevo problema. Esto permite iniciar el entrenamiento desde un punto más próximo al mínimo global, en lugar de empezar desde cero con una inicialización aleatoria. Es especialmente útil en problemas donde se dispone de conjuntos de datos pequeños o cuando el nuevo problema guarda similitudes con el problema original. De esta forma, se acelera el entrenamiento y se mejora el rendimiento de la red (Pan & Yang, 2010).

8.2.2.2.3 Algoritmos de optimización: descenso por gradiente, descenso por gradiente con momento, Adam

Para lograr la minimización de la función de pérdida, se emplean **algoritmos de optimización** basados en gradiente, que permiten actualizar iterativamente los parámetros en la dirección que reduce el error de predicción. Algunos de estos algoritmos son:

- **Descenso por gradiente**: El descenso por gradiente es un método iterativo que ajusta los pesos de la red en sentido contrario a la pendiente de la función de pérdida. Este gradiente indica la dirección de mayor incremento, por lo que moverse en sentido contrario reduce el valor de la función de pérdida. La regla de actualización de este método es:

$$W_{t+1} = W_t - \alpha \nabla_W \mathcal{L}_{W_t} \tag{8.14}$$

donde α es la tasa de aprendizaje (*learning rate*), un parámetro que controla cuánto se ajustan los pesos en cada paso, ∇_W es el gradiente y L_{Wt} es la función de pérdida. Este método puede necesitar muchas iteraciones para alcanzar la convergencia.

- **Descenso por gradiente con momento**: Este algoritmo mejora el descenso por gradiente al incorporar información sobre las iteraciones previas. Este método utiliza

una media móvil exponencial de los gradientes anteriores, es decir, los gradientes más recientes tendrán más importancia que los anteriores, para acelerar la convergencia en la dirección correcta. La modificación de los pesos se lleva a cabo mediante la fórmula:

$$v_t = \beta v_{t-1} + \alpha \nabla_W \mathcal{L}_{W_t} \tag{8.15}$$

$$W_{t+1} = W_t - v_t \tag{8.16}$$

Aquí, β es el coeficiente de momento que determina cuánto influyen los gradientes pasados en la actualización actual. Un valor típico de β es 0,9. Este enfoque acelera el descenso en direcciones coherentes.

- **Adam** (*Adaptive Moment Estimation*): Combina las ventajas del descenso por gradiente con momento y una tasa de aprendizaje adaptativa, es decir, que se adapta en función de la variación del gradiente. Un algoritmo que utiliza esto es RMSprop, que calcula estimaciones de primer y segundo momento del gradiente utilizando medias móviles exponenciales:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_W \mathcal{L}_{W_t} \tag{8.17}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) [\nabla_W \mathcal{L}_{W_t}]^2 \tag{8.18}$$

Para corregir los sesgos iniciales, se normalizan las estimaciones:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{8.19}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{8.20}$$

La actualización de los pesos se realiza como:

$$W_{t+1} = W_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{8.21}$$

donde ϵ es un pequeño valor para evitar divisiones por cero, y los valores típicos para β_1 y β_2 son 0.9 y 0.999, respectivamente.

8.2.2.2.4 Tipos de optimización: lotes, estocástica y mini-lotes

Según cómo se procesan los datos durante el cálculo del gradiente y la actualización de los parámetros, los métodos de **optimización** pueden clasificarse en tres enfoques principales:

por **lotes, estocástica y** por **mini-lotes**. A continuación, se describe cada uno de estos métodos:

- **Optimización por lotes**: La optimización por lotes consiste en utilizar la totalidad del conjunto de datos de entrenamiento para calcular el gradiente de la función de pérdida en relación con los parámetros de la red. Esto implica que, en cada iteración, se procesan todos los ejemplos del conjunto de datos para actualizar los pesos. Este método garantiza unos resultados óptimos, pero puede requerir un alto coste computacional y resultar lento cuando se trabaja con grandes volúmenes de datos y requiere gran cantidad de memoria para procesar todos los ejemplos simultáneamente.
- **Optimización estocástica**: En el método estocástico, el gradiente se calcula y los parámetros se actualizan para cada ejemplo individual del conjunto de datos. Este enfoque introduce un grado de aleatoriedad en el proceso de optimización, ya que utiliza un solo ejemplo para cada actualización. Es eficiente en términos de memoria, puede escapar de mínimos locales y es adecuado para grandes conjuntos de datos, pero los resultados pueden ser inconsistentes o menos precisos.
- **Optimización por mini-lotes**: La optimización por mini-lotes combina características de los dos métodos anteriores. En este enfoque, el conjunto de datos de entrenamiento se divide en pequeños lotes (mini-lotes), y cada lote se utiliza para calcular el gradiente y actualizar los parámetros. Esto permite un equilibrio entre estabilidad y eficiencia.

En la práctica, el método de mini-lotes es el más utilizado, ya que equilibra la estabilidad del gradiente y la eficiencia computacional.

8.2.2.2.5 Backpropagation

El algoritmo *Backpropagation* o propagación hacia atrás es una técnica esencial en el entrenamiento de redes neuronales artificiales, especialmente cuando estas tienen múltiples capas. Su introducción resolvió una de las limitaciones principales del perceptrón multicapa: la dificultad de ajustar los pesos de las capas ocultas debido a la falta de una salida esperada conocida para estas capas.

Backpropagation emplea la regla de la cadena, una técnica matemática que permite calcular la derivada de una función compuesta. Este método se utiliza para determinar el gradiente de la función de pérdida respecto a los pesos de las capas ocultas.

El proceso de este algoritmo se divide en dos etapas principales:

1. ***Forward pass***: Se realiza una propagación de los datos de entrada a través de la red hasta la capa de salida, calculando la salida final y el valor de la función de pérdida.
2. ***Backward pass***: Una vez obtenida la pérdida, el error se retropropaga desde la capa de salida hacia las capas anteriores, ajustando los pesos en base al gradiente calculado. Durante esta etapa, se optimizan los pesos de cada capa siguiendo un

enfoque iterativo, utilizando algoritmos de optimización como el descenso por gradiente.

El cálculo del gradiente en Backpropagation puede representarse a través de un grafo computacional, donde cada nodo simboliza una operación de la red. Este enfoque permite calcular las derivadas de manera eficiente y simbólica.

8.2.2.3 Etapa de entrenamiento: hiperparámetros

El **entrenamiento** de una red neuronal involucra la selección de varios **hiperparámetros**, que son parámetros ajustables que afectan directamente el rendimiento del modelo y su capacidad de generalización. Algunos de los principales hiperparámetros incluyen:

- **Arquitectura de la red**: Determina su capacidad para modelar la información. Se deben definir aspectos como el número de capas, la cantidad de neuronas por capa, y la función de activación. Una estrategia común es utilizar arquitecturas que han sido efectivas en problemas similares, aunque no hay garantía de que funcionen igual de bien en otros casos. Es importante ajustar la arquitectura al nivel de complejidad del problema para evitar el sobreajuste.
- **Inicialización de pesos**: La forma en la que se inicializan los pesos afecta significativamente la rapidez y estabilidad del entrenamiento. Cuando sea posible, se intentará utilizar la técnica de *transfer learning*, ya que se prefieren pesos preentrenados de modelos previamente ajustados, debido a que acelera la convergencia.
- **Número de épocas**: Un número demasiado bajo puede impedir que el modelo aprenda lo suficiente, mientras que un número excesivo puede provocar sobreajuste.
- **Tamaño de lotes**: El tamaño del lote afecta el cálculo del gradiente y la eficiencia computacional. Se suelen emplear potencias de dos (16, 32, 64 y 128) debido a su compatibilidad con la memoria del hardware. Los tamaños grandes requieren más memoria.
- **Elección del algoritmo de optimización**: El algoritmo de optimización define cómo se ajustan los pesos de la red en función del gradiente. Existen numerosos algoritmos de optimización, cada uno con sus propias ventajas y desventajas. Por ejemplo, el descenso por gradiente es más rápido de calcular que Adam, lo que lo hace mejor en ciertos casos. Una ventaja clave del descenso por gradiente es que tiende a encontrar mínimos aplanados en la función de pérdida en lugar de mínimos muy pronunciados. Esto es importante porque un mínimo aplanado implica que, aunque los pesos cambien ligeramente, la red seguirá en una zona de error bajo. En cambio, en un mínimo pronunciado, una pequeña variación en los pesos puede hacer que el error aumente significativamente.
- **Parámetros del algoritmo de optimización**: Dentro de los algoritmos de optimización, existen parámetros clave como la tasa de aprendizaje (*learning rate*). Este es uno de los parámetros más críticos, ya que afecta directamente a la capacidad del modelo para alcanzar un buen mínimo de la función de pérdida. El *learning rate* no se elige de manera aislada, sino que está relacionado con el tamaño del lote utilizado

en el entrenamiento. Con lotes grandes, el gradiente es más estable y se puede usar un *learning rate* mayor, ya que la dirección calculada es más fiable. Pero con lotes pequeños, el gradiente es más ruidoso y menos preciso, por lo que se debe reducir el *learning rate* para evitar grandes variaciones.

- **Función de pérdida**: El CE es utilizado para problemas de clasificación con múltiples clases y el ECM es más común para tareas de regresión. Sin embargo, estas funciones pueden modificarse para adaptarse mejor a las características del problema en cuestión. Por ejemplo, si el conjunto de datos contiene *outliers*, el ECM puede producir errores muy grandes. Para reducir este efecto, se puede ajustar el exponente de la función en lugar de utilizar el cuadrado. No hay una restricción específica sobre qué función de pérdida se puede utilizar, siempre que sea derivable, ya que esto es esencial para el algoritmo de Backpropagation.

8.2.2.4 Etapa de entrenamiento: control y seguimiento. Selección de la red.

Durante el **entrenamiento** de una red neuronal es fundamental llevar un **control y seguimiento** del proceso para garantizar un aprendizaje adecuado y prevenir problemas como sobreajuste o infraajuste. Para ello, se emplean diversas estrategias:

- **Almacenamiento de pesos**: Es crucial para utilizar la red entrenada en la fase de inferencia. Lo ideal es guardar los pesos cada vez que se obtiene una mejora en la función de pérdida o en las métricas del conjunto de validación.
- ***Early stopping***: Se trata de una estrategia que interrumpe el entrenamiento cuando no se observa mejora en la función de pérdida del conjunto de validación durante un número determinado de épocas. Esto ayuda a evitar el sobreajuste, ya que impide que la red siga ajustándose excesivamente a los datos de entrenamiento.
- **Política de actualización del *learning rate***: En los algoritmos donde el *learning rate* no se ajusta de manera adaptativa, se puede implementar una política de actualización. Una estrategia común es disminuir el *learning rate* en función del número de épocas o cuando la mejora en la función de pérdida es mínima.
- **Regularización**: Busca reducir la complejidad del modelo mediante la penalización de valores altos en los pesos de la red. Se fundamenta en la idea de que modelos más simples tienen valores menores en los pesos. Una técnica común es la regularización l_2, que añade un componente adicional en la función de pérdida que castiga la magnitud de los pesos.
- ***Batch normalization***: La normalización por lotes mejora la estabilidad del entrenamiento al normalizar las entradas de las neuronas. Esto ayuda a reducir problemas derivados de distribuciones de datos muy variables a lo largo de las capas de la red y acelerando el entrenamiento.
- ***Dropout***: Es una técnica que consiste en desactivar aleatoriamente ciertas neuronas, tal como se muestra en la Figura 8.21, durante la propagación hacia delante. Esto previene el sobreajuste.

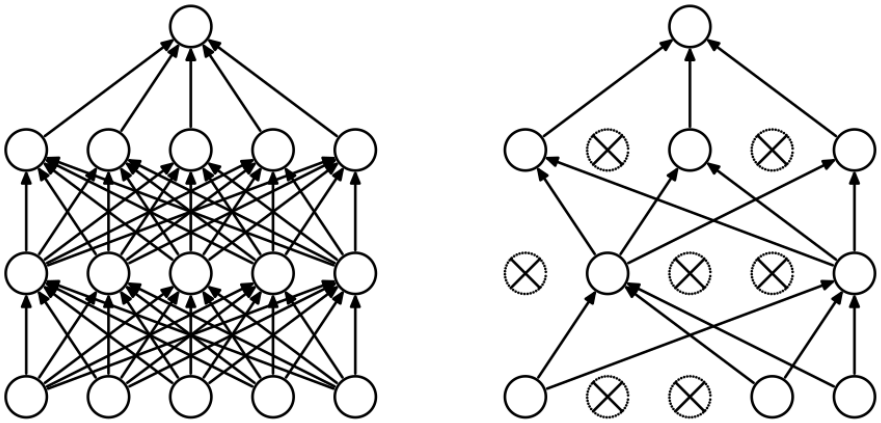

Figura 8.21: Efecto de dropout en una red neuronal.
Fuente: https://medium.com/@nicolas_araque/regularizando-nuestra-red-dropout-da84975593b

- **Aumento de datos**: Consiste en aplicar transformaciones a los datos de entrada, como rotaciones, cambios de escala, desplazamientos o modificaciones en la iluminación, con el objetivo de ampliar la variedad dentro del conjunto de entrenamiento y mejorar la capacidad de generalización del modelo.

8.2.2.5 Etapa de inferencia: aplicación y evaluación de la red. Métricas y curvas de rendimiento de clasificación.

La **etapa de inferencia** en redes neuronales consiste en utilizar el modelo previamente entrenado para realizar predicciones sobre nuevos datos sin modificar sus pesos.

Durante esta fase, se evalúa el rendimiento del modelo aplicando diferentes **métricas** de evaluación, que permiten cuantificar su precisión y capacidad de generalización. Estas métricas son fundamentales para determinar el desempeño del modelo en problemas de clasificación. Para ello, se comparan las predicciones generadas con las etiquetas reales del conjunto de prueba. Algunas de las métricas más utilizadas incluyen:

- **Exactitud (*Accuracy*)**: Mide la proporción de predicciones correctas respecto al total de predicciones.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8.22}$$

- **Precisión (*Precision*)**: Indica la tasa de elementos clasificados como positivos que realmente lo son, en relación con todos los ejemplos clasificados como positivos.

$$Precision = \frac{TP}{TP + FP} \tag{8.23}$$

- **Exhaustividad o Sensibilidad (*Recall*)**: Representa la capacidad del modelo para identificar correctamente los elementos de la clase positiva. Evalúa la proporción de ejemplos clasificados correctamente como positivos en relación con todos los ejemplos etiquetados como positivos.

$$Recall = \frac{TP}{TP + FN} \tag{8.24}$$

- **Especificidad**: Mide la proporción de elementos de la clase negativa que han sido correctamente identificados, lo cual es equivalente al *recall*, pero para la clase negativa.

$$Specificity = \frac{TN}{TN + FP} \tag{8.25}$$

- **Puntuación F1 (*F1-score*)**: Es una buena métrica para problemas con clases desbalanceadas. Se define como la media armónica entre la *precisión* y la *sensibilidad* o *recall*, combinando ambas en una única medida. Su valor oscila entre 0 y 1, donde 1 indica un rendimiento perfecto.

$$f1\_score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{8.26}$$

Cuando se trabaja con modelos de clasificación, es común evaluar su desempeño mediante curvas que permiten visualizar la relación entre distintas métricas, como, por ejemplo, la **curva ROC**. Esta curva representa los valores de sensibilidad frente a los de especificidad. Su calidad se mide a través del ***AUC* (Área Bajo la Curva)**, donde, como podemos observar en la Figura 8.22, los valores cercanos a 1 indican un modelo con buen rendimiento.

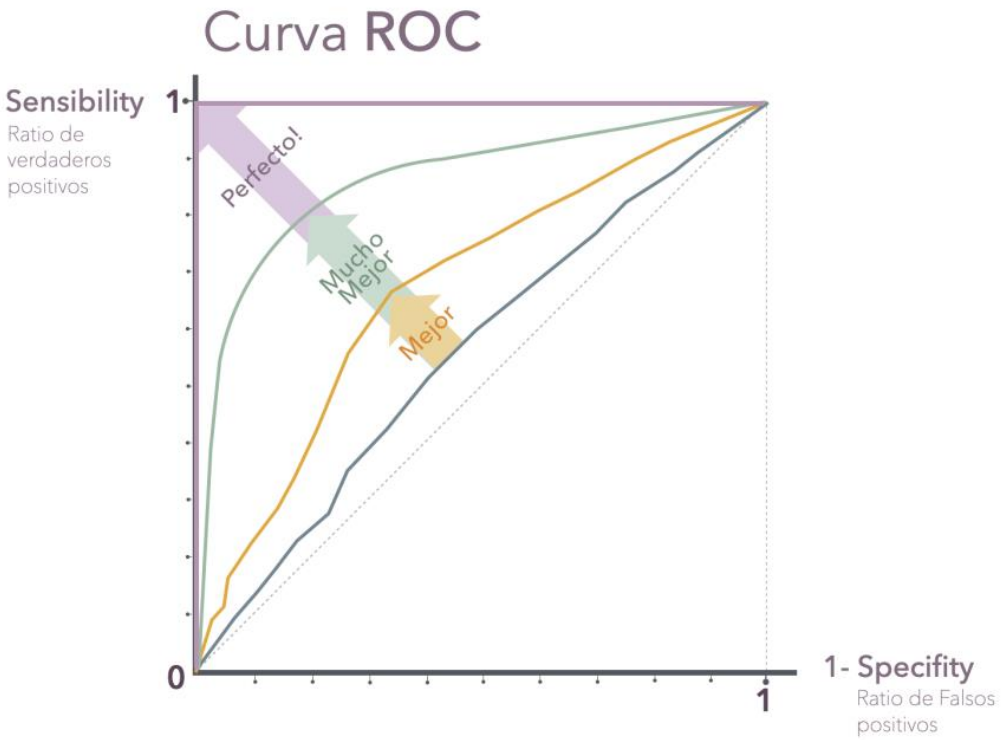

Figura 8.22: Gráfico con diferentes curvas ROC de varios modelos.
Fuente: https://mamel.es/modelos-predictivos-y-la-curva-roc/

8.3 Redes neuronales convolucionales

8.3.1 Introducción

Las redes neuronales convolucionales (*CNN*, por sus siglas en inglés) son un modelo de red neuronal inspirado en la forma en que la corteza visual del cerebro humano procesa la información. Se han convertido en una de las herramientas más poderosas en el campo del aprendizaje profundo.

El desarrollo de estas redes tiene sus raíces en los experimentos de los neurocientíficos David Hubel y Torsten Wiesel en 1959 (Hubel & Wiesel, 1959). En sus estudios sobre el cerebro de los gatos, descubrieron que ciertas neuronas de la corteza visual se activaban ante la presencia de patrones específicos, como líneas y bordes de diferentes orientaciones. Este hallazgo sirvió de inspiración para el diseño de las redes neuronales convolucionales, las cuales procesan imágenes de manera jerárquica, extrayendo características de menor a mayor nivel.

A diferencia de las redes neuronales tradicionales, que tratan los datos en forma de vectores planos, las CNN mantienen la estructura espacial de la información, permitiendo detectar patrones en distintas posiciones de una imagen.

Desde su aparición, las redes neuronales convolucionales han logrado avances significativos en múltiples aplicaciones, como el reconocimiento de objetos, la conducción autónoma y la segmentación de imágenes. Un hito importante en su desarrollo fue la creación de *AlexNet* en 2012, una red que revolucionó la clasificación de imágenes en la competición *ImageNet*, reduciendo drásticamente la tasa de error en comparación con modelos anteriores.

En la actualidad, las CNN siguen evolucionando con arquitecturas más profundas y eficientes, permitiendo su aplicación en diversos campos más allá del procesamiento de imágenes, como la medicina y la automatización industrial.

8.3.2 Tipos de capas

Las redes neuronales convolucionales (CNN) están compuestas por diferentes **tipos de capas**, cada una con una función específica en el procesamiento de los datos. Si nos centramos en la estructura fundamental se distinguen dos tipos de capas:

- **Capa de convolución**: Es fundamental en las CNN. Su función principal es extraer características de los datos de entrada aplicando filtros (*kernels*) a través de la operación matemática de convolución. Esta operación permite detectar patrones como bordes, texturas y formas en las imágenes.
  En el contexto del DL, la operación utilizada en las CNN no es exactamente una convolución matemática tradicional, sino una correlación cruzada. En la Figura 8.23 se muestra un ejemplo de esta operación. El resultado de cada filtro genera un mapa de características.

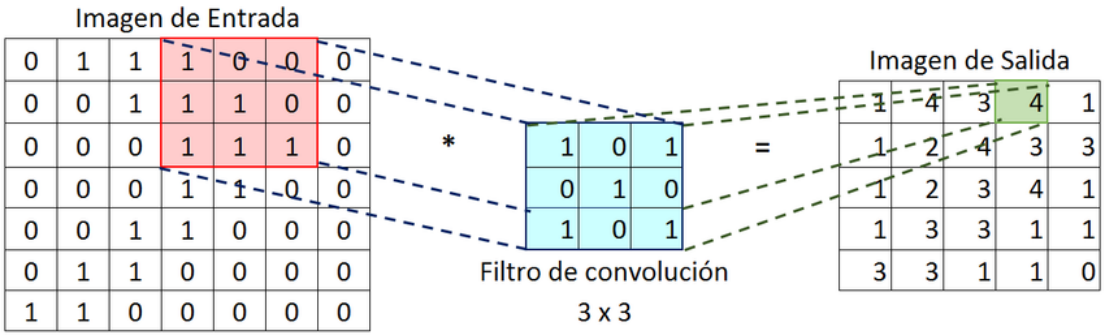

Figura 8.23: Ejemplo de convolución en una red neuronal convolucional.
Fuente: https://www.researchgate.net/figure/Figura-4-Ejemplo-del-proceso-de-convolucion_fig1_354519890

  Los principales hiperparámetros de una capa de convolución son:
  - **Tamaño del *kernel***: Define el tamaño de la ventana de la convolución. En el ejemplo de la Figura el tamaño sería de 3x3.
  - ***Stride* (paso)**: Determina el desplazamiento del filtro sobre la entrada.
  - ***Padding* (relleno)**: Se usa para ajustar el tamaño de la salida agregando ceros alrededor de la entrada.
- **Capa de pooling**: Se emplean para disminuir las dimensiones de los mapas de características, lo que disminuye la cantidad de parámetros y aumenta el campo receptivo de las capas posteriores. Además, ayudan a proporcionar invariancia a pequeñas variaciones en la entrada.

La operación de pooling es similar a la convolución en términos de parámetros, ya que cuenta con los mismos parámetros (tamaño de ventana, paso y relleno).
Las dos formas más comunes de pooling son:

- ***Max-Pooling***: Selecciona el valor máximo en una ventana determinada, como se ilustra en la Figura 8.24.

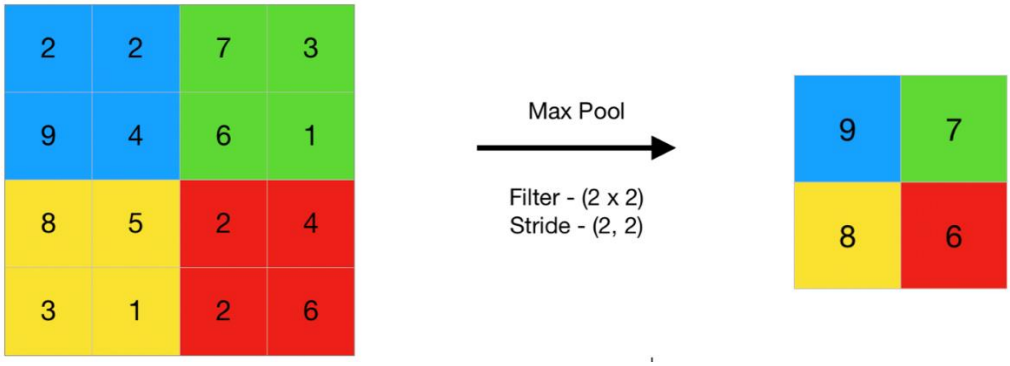

Figura 8.24: Ejemplo de cálculo de max-pooling.
Fuente: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

- ***Average pooling***: Calcula el promedio de los valores dentro de la ventana, tal y como se puede observar en la Figura 8.25.

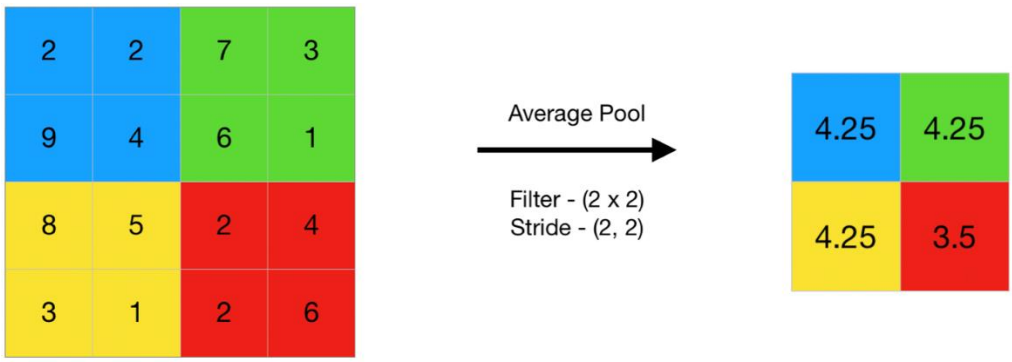

Figura 8.25: Ejemplo de cálculo de average pooling.
Fuente: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/

Las conexiones ente capas pueden variar para mejorar el rendimiento de la red. Algunos tipos de conexiones son:

- **Secuencial**: La conexión tradicional, donde se repite una secuencia de capas de convolución y pooling, como se puede apreciar en la Figura 8.26.

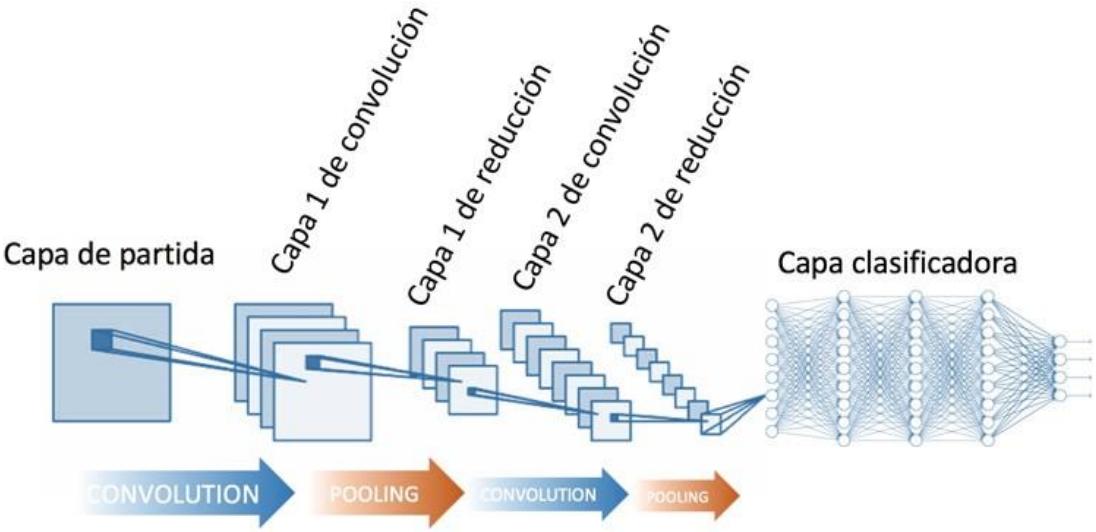

Figura 8.26: Estructura de una red neuronal convolucional con conexión secuencial.
Fuente: https://www.bravent.net/noticias/redes-neuronales-convolucionales-en-el-reconocimiento-de-imagenes/

- ***Inception block***: Como se puede apreciar en la Figura 8.27, procesa el mismo mapa de características con diferentes operaciones y combina los resultados.

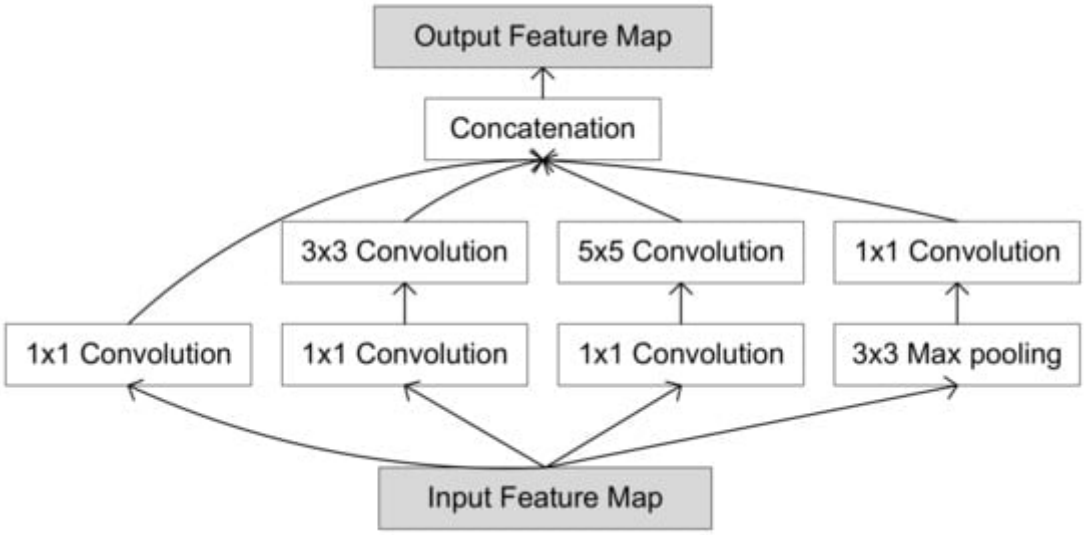

Figura 8.27: Ejemplo de bloque inception.
Fuente: https://www.mdpi.com/1999-4893/13/5/125

- ***Skip connection***: Introducida por ResNet, suma la salida de una capa con la entrada de una capa anterior. La Figura 8.28 muestra un esquema representativo del funcionamiento.

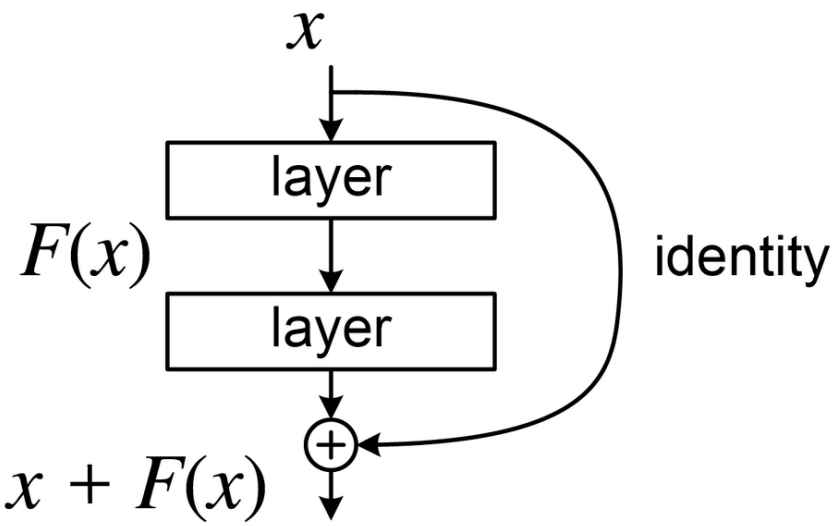

Figura 8.28: Ejemplo de skip connection.
Fuente: https://es.wikipedia.org/wiki/Red_neuronal_residual

- ***Dense block***: Concatenan mapas de características de capas anteriores para preservar la información, como se puede ver en el esquema de la Figura 8.29.

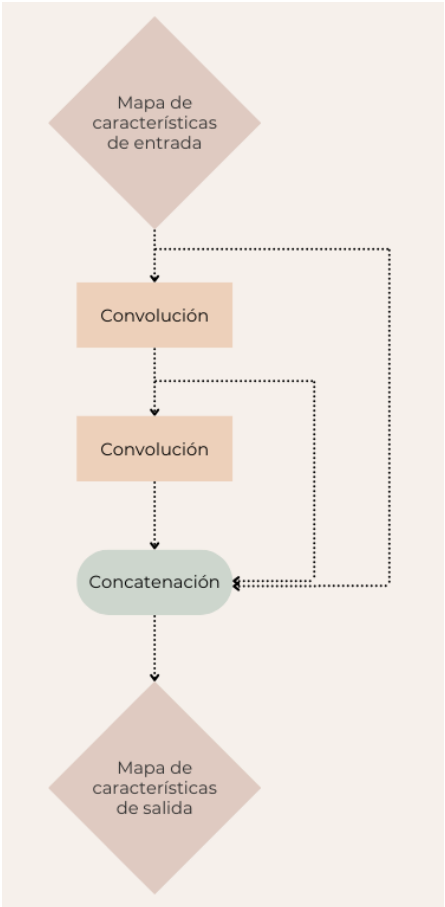

Figura 8.29: Ejemplo de dense block.
Fuente: Elaboración propia.

8.3.3 Arquitecturas populares

En este apartado, exploraremos algunas de las arquitecturas más influyentes en la historia reciente del DL, destacando su impacto y sus innovaciones clave.

1. LeNet-5

LeNet-5 fue una de las primeras arquitecturas exitosas en el reconocimiento de dígitos escritos a mano (LeCun et al., 1998). Como podemos observar en la Figura 8.30, esta red consta de capas convolucionales y de pooling alternadas, seguidas de capas totalmente conectadas.

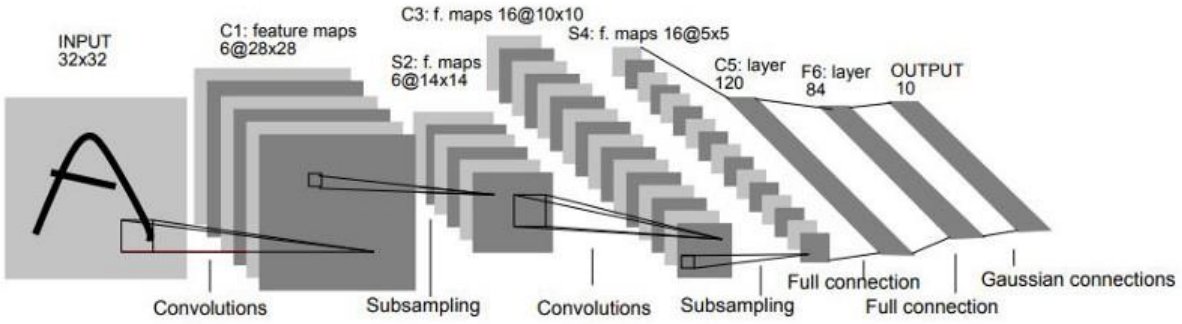

Figura 8.30: Arquitectura de la red LeNet-5.
Fuente: https://www.datasciencecentral.com/lenet-5-a-classic-cnn-architecture/

2. AlexNet

AlexNet (Krizhevsky et al., 2012) introdujo el uso de GPUs para entrenar redes neuronales y marcó un antes y un después en el campo de las redes neuronales convolucionales al ser la primera en ganar la competición ***ImageNet*** en 2012. Logró una significativa reducción del error en comparación con modelos previos. Además, contribuyó significativamente a la popularización del DL en tareas de procesamiento de imágenes.

Esta red tiene una estructura considerablemente más grande que su predecesora, *LeNet-5*, y como podemos ver en la Figura 8.31, está formada por cinco capas de convolución, tres capas de pooling y una red neuronal de cuatro capas. La red contiene más de 60 millones de parámetros.

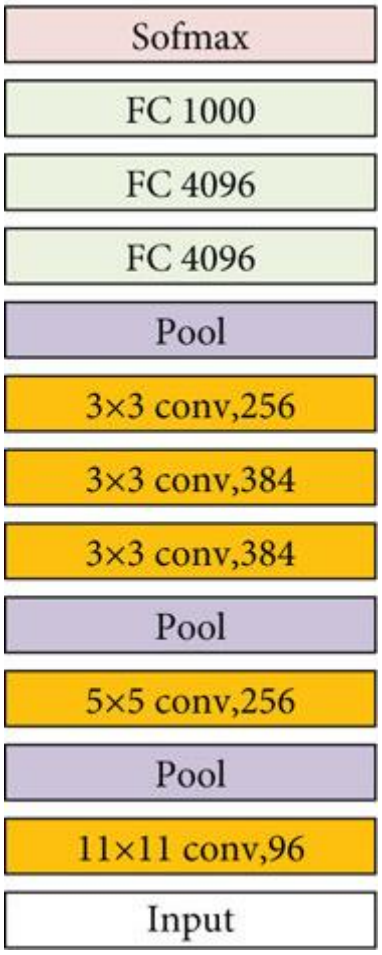

Figura 8.31: Arquitectura de la red AlexNet.
Fuente: https://www.researchgate.net/figure/The-network-structure-of-AlexNet-VGGNet16-and-VGGNet19_fig5_352984400

Esta red introdujo varias innovaciones, como el uso de unidades *ReLU* como función de activación y el entrenamiento en GPUs, que permitió una reducción considerable en el tiempo de entrenamiento.

3. GoogLeNet

GoogLeNet (Szegedy et al., 2015) ganó ***ImageNet*** 2014 e introdujo los bloques *inception*. Estos bloques se diseñaron para aumentar la profundidad de la red neuronal convolucional, pero de una manera computacionalmente eficiente. El objetivo principal era reducir el número de parámetros mediante el uso de convoluciones 1x1 como compresores.

GoogLeNet presenta una arquitectura de 22 capas, que se puede ver en la Figura 8.32.

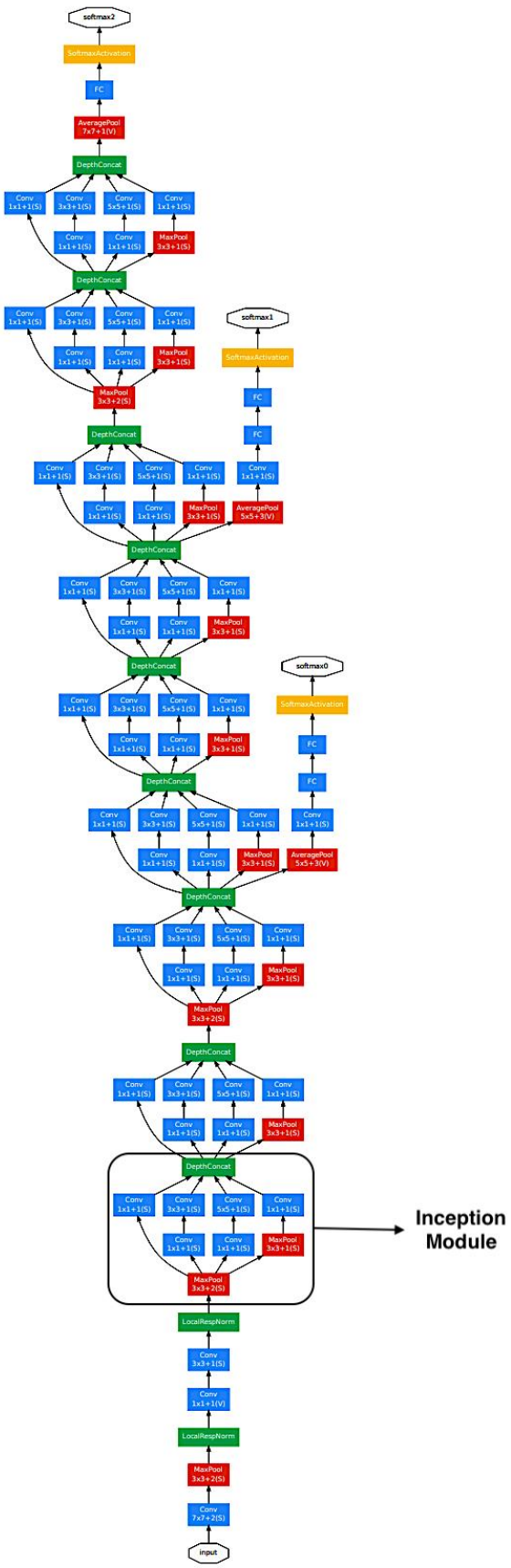

Figura 8.32: Arquitectura de la red de GoogLeNet.
Fuente: https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597

4. ResNet

ResNet (He et al., 2016) lideró ***ImageNet*** en 2015.

La innovación fundamental de *ResNet* reside en la introducción de las *skip connections*. Además, se propuso el uso de *average pooling* global antes de la capa de clasificación.

Las arquitecturas *ResNet* se caracterizan por su profundidad extrema. Entre los ejemplos más destacados se encuentran *ResNet-50*, *ResNet-101* y *ResNet-152*, donde el número indica la cantidad total de capas en la red. Las arquitecturas de estas redes se pueden ver en la Figura 8.33.

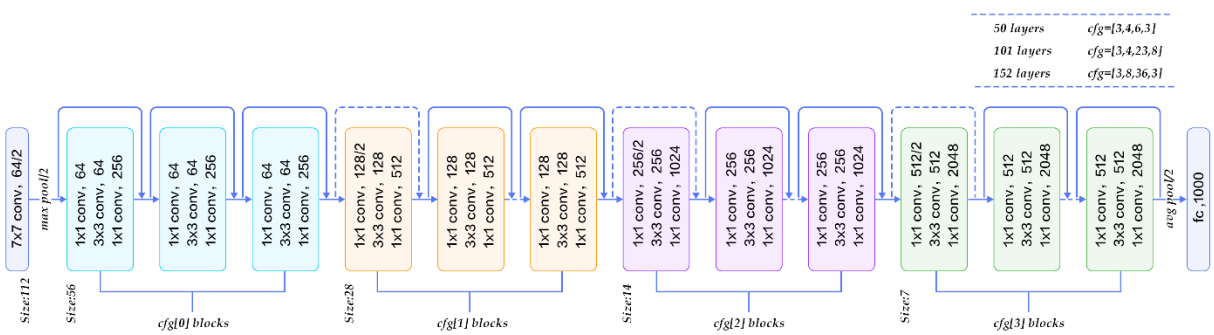

Figura 8.33: Arquitectura de las redes ResNet-50, ResNet-101 y ResNet-152.
Fuente: https://www.jesusutrera.com/articles/article07.html

5. DenseNet

DenseNet llevó la idea de las *skip connections* un paso más allá al conectar directamente cada capa con todas las anteriores (*dense blocks*), lo que ayuda a mejorar la eficiencia del entrenamiento (Huang et al., 2017). Esta arquitectura fue la ganadora de la competición ***ImageNet*** en 2016.

A pesar de su gran profundidad, *DenseNet-169* utiliza solo 12 millones de parámetros para 169 capas. Su arquitectura se muestra en la Figura 8.34.

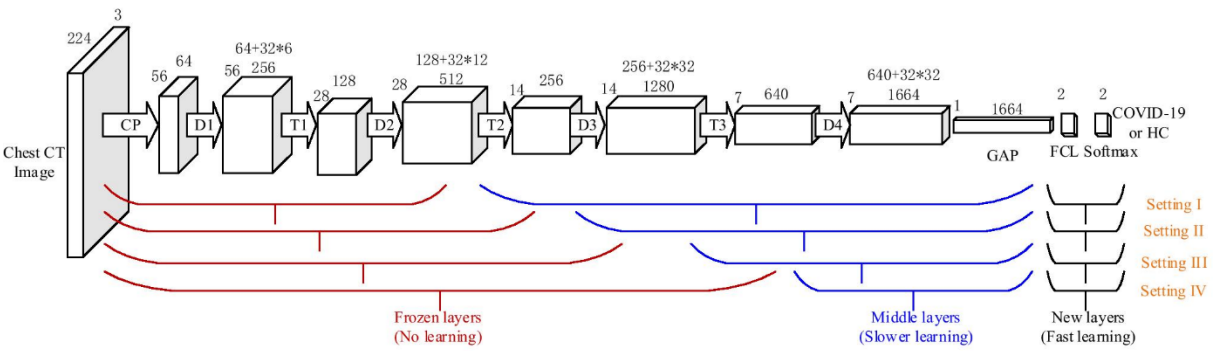

Figura 8.34: Arquitectura de la red DenseNet-169.
Fuente: https://link.springer.com/article/10.1007/s12559-020-09776-8

Aunque *DenseNet* requiere menos parámetros, realiza muchas más operaciones que otras arquitecturas, lo que requiere más capacidad de procesamiento.

9 Anexo II. Fundamentos y configuración del electrocardiograma de 12 derivaciones

El electrocardiograma (ECG) es una prueba diagnóstica fundamental en cardiología, que permite registrar la actividad eléctrica del corazón. Es una herramienta crucial para la evaluación de la función cardíaca y la detección de diversas patologías, incluyendo arritmias, infartos y otras condiciones cardiovasculares. El ECG clásico de 12 derivaciones ofrece una visión detallada de la actividad eléctrica del corazón desde múltiples ángulos, lo que facilita una interpretación más precisa. Esta tecnología fue desarrollada a principios del siglo XX por Willem Einthoven, quien revolucionó la cardiología con su invención (Lara Prado, 2016).

Las derivaciones del ECG se dividen en dos grandes grupos: las derivaciones frontales y las derivaciones precordiales (Lara Prado, 2016):

- **Derivaciones frontales**: Estas incluyen las derivaciones estándar (I, II, III) y amplificadas (aVR, aVL y aVF). Cada una de estas derivaciones ofrece una representación de la actividad eléctrica desde diferentes perspectivas del plano frontal (Lara Prado, 2016).
- **Derivaciones precordiales**: Estas derivaciones (V1 a V6) se colocan sobre el tórax, permitiendo observar la actividad eléctrica en el plano horizontal del corazón (Lara Prado, 2016).

Es fundamental colocar los electrodos correctamente para obtener un ECG preciso (véase Figura 9.1). Además, la relación entre los electrodos y las derivaciones se ha establecido de manera clara en estudios previos (Clopton & Hyrkäs, 2024).

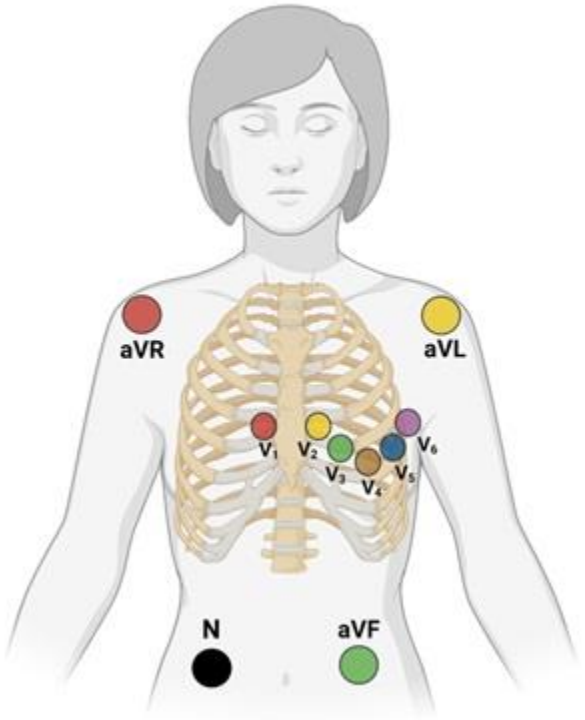

Figura 9.1: Distribución de los electrodos en un ECG de 12 derivaciones.
Fuente: https://fisiologia.facmed.unam.mx/index.php/taller-de-interpretacion-del-electrocardiograma/

En un ECG, las ondas principales son fundamentales para interpretar la actividad eléctrica del corazón. La onda P refleja la despolarización de las aurículas, el complejo QRS representa la despolarización de los ventrículos, y la onda T corresponde a la repolarización ventricular (Al-Ani, 2018).

Las 12 derivaciones del ECG permiten una evaluación detallada y precisa de las arritmias, como la fibrilación auricular, las taquicardias y las arritmias ventriculares. Por ejemplo, la fibrilación auricular se caracteriza por una falta de ondas P claras y un ritmo irregular, mientras que en la taquicardia ventricular se observa un complejo QRS ancho y rápido. Cada derivación del ECG mira el corazón desde un ángulo distinto. Al analizarlas todas juntas, los médicos pueden detectar patrones eléctricos que indican enfermedades cardíacas específicas (Al-Ani, 2018).

10 Anexo III. Generación de escalogramas a partir de señales temporales

La generación de escalogramas a partir de señales unidimensionales, como los registros de electrocardiogramas (ECG), se basa en el uso de herramientas de análisis tiempo-frecuencia, entre las cuales destaca la Transformada Continua de Wavelet (CWT, por sus siglas en inglés). Esta técnica permite descomponer una señal en componentes de frecuencia a lo largo del tiempo, lo que la hace especialmente útil para señales no estacionarias como las fisiológicas (Addison, 2005).

Una señal de ECG es una serie temporal que representa la actividad eléctrica del corazón, muestreada a lo largo del tiempo. Para transformarla en una imagen bidimensional que conserve tanto la información temporal como frecuencial, se aplica la CWT, cuya fórmula general se expresa como:

$$CWT_x(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t)\ \psi^* \left(\frac{t-b}{a}\right) dt \tag{10.1}$$

donde *x(t)* es la señal original, ***a*** es el parámetro de escala (relacionado inversamente con la frecuencia), *b* es el parámetro de traslación (tiempo), y *ψ* es la función wavelet madre (Addison, 2005).

La CWT genera una matriz de coeficientes que describe cómo varían las frecuencias a lo largo del tiempo. Esta matriz se puede visualizar mediante un escalograma, que representa gráficamente la energía de la señal en función del tiempo y la escala. Los valores de los coeficientes son escalares y se codifican mediante una paleta de colores (*colormap*) que asigna a cada valor un color en formato RGB. Esto significa que, aunque la matriz inicial es bidimensional con valores numéricos, al aplicar esta función de mapeo se genera una imagen con tres canales (rojo, verde y azul), donde cada píxel tiene un color que refleja la magnitud del coeficiente en esa posición. De esta manera, se obtiene una imagen RGB que proporciona una forma intuitiva y visualmente descriptiva de observar las variaciones de frecuencia en distintos instantes temporales (Olanrewaju et al., 2021).

En este proyecto, los escalogramas se han generado en formato RGB de 300×300 píxeles, con tres canales de color, permitiendo su procesamiento mediante redes neuronales convolucionales bidimensionales (2D-CNN). Para generar la matriz de coeficientes mencionada anteriormente, se ha utilizado la función *cwt.m* de la **Wavelet Toolbox** de Matlab 2020a, que permite aplicar la transformada con distintas funciones wavelet madre. En el caso del análisis de señales ECG, una de las más empleadas es la **wavelet de Daubechies**, especialmente en su versión *db4*, por su capacidad para detectar transiciones rápidas como el complejo QRS y su buena localización temporal y frecuencial (Addison, 2005).

La representación espectral obtenida es más adecuada para redes convolucionales profundas que las señales 1D originales, ya que captura patrones espacio-frecuenciales que pueden estar asociados a determinadas condiciones cardíacas, como se ha demostrado en estudios recientes centrados en redes profundas aplicadas a escalogramas de señales cardíacas (Yoon & Kang, 2023).

Índice de figuras

Índice de tablas