

---

**Predicción de Riesgo Cardiovascular Explicable  
mediante *Deep Learning***  
**Explainable Cardiovascular Risk Prediction  
using Deep Learning**

---



**Trabajo de Fin de Grado**  
**Curso 2024–2025**

**Autor**  
**Noelia Barranco Godoy**

**Directores**  
**Belén Díaz Agudo**  
**Juan A. Recio García**  
**María Ángeles Díaz Vicente**

**Doble Grado en Ingeniería Informática y Matemáticas**  
**Facultad de Informática**  
**Universidad Complutense de Madrid**



Predicción de Riesgo Cardiovascular  
Explicable mediante *Deep Learning*  
Explainable Cardiovascular Risk  
Prediction using Deep Learning

Trabajo de Fin de Grado en Ingeniería Informática

**Autor**  
**Noelia Barranco Godoy**

**Director**  
**Belén Díaz Agudo**  
**Juan A. Recio García**  
**María Ángeles Díaz Vicente**

**Convocatoria:** Febrero 2025

Doble Grado en Ingeniería Informática y Matemáticas  
Facultad de Informática  
Universidad Complutense de Madrid

20 de enero de 2025



# Dedicatoria

*A mi pareja, Ian, por ser mi refugio en cada instante de agobio y recordarme, con su presencia y cariño, que ninguna meta es inalcanzable cuando no se afronta sola.*



# Agradecimientos

Deseo expresar mi más sincero reconocimiento a los directores de mi TFG. A Juan Antonio por sus consejos a lo largo del desarrollo del trabajo, a Belén, por su constante apoyo y por haberme presentado a Marian, cuya presencia, más allá de los aspectos estrictamente académicos, fue esencial a nivel profesional y personal, ofreciéndome una dedicación, cercanía y orientación que marcaron profundamente mi proceso formativo. Estas atenciones, que superaron con creces el ámbito académico, han dejado una huella imborrable en mi trayectoria.

Asimismo, quiero agradecer también al profesor Mauro Buelga, del departamento de Cardiología del Hospital Universitario Ramón y Cajal y profesor de Enfermería de la Universidad de Alcalá de Henares, así como al doctor Gonzalo Alonso, del departamento de Cardiología del Hospital Universitario de Navarra, por haber revisado y valorado mi trabajo, brindando una valiosa validación experta. Sus observaciones han sido fundamentales para mejorar y afianzar la calidad de este proyecto.

También dentro del ámbito académico, no puedo no mencionar a los profesores del departamento de arquitectura de hardware de nuestra facultad, Joaquín Recas y Luis Piñel, que me han concedido acceso a una máquina de la facultad, y me han ayudado con la solución de problemas técnicos cuando ha sido necesario.

Por último, agradezco profundamente a Sergio González Cabeza y Mario Sanz Guerrero, cuyos trabajos relacionados y sus consejos han sido de gran ayuda en este trabajo.



# Resumen

## Predicción de Riesgo Cardiovascular Explicable mediante *Deep Learning*

Este trabajo se centra en el uso de técnicas de Inteligencia Artificial para la clasificación de señales de electrocardiogramas (ECGs), un área de gran relevancia en la medicina para el diagnóstico de enfermedades cardíacas. Este trabajo tiene dos objetivos principales: explorar y evaluar diferentes enfoques de clasificación de ECG y aplicar explicabilidad a algunos de los modelos entrenados.

Para el primer objetivo, utilizamos bases de datos abiertas como PTB-XL, y aplicar transformaciones de señales como la Transformada de Fourier de Tiempo Reducido (STFT) o la Transformada de Onda Continua (CWT). Con esto entrenaremos y compararemos modelos clásicos y modificados para trabajar con estas transformadas, y se analiza su rendimiento usando métricas.

La explicabilidad de los modelos es también un aspecto clave del estudio, ya puede emplearse tanto para docencia como para justificar las decisiones tomadas por el modelo. Al final de este trabajo validamos los resultados con un experto médico para asegurar la aplicabilidad clínica y docente de los resultados obtenidos.

Las conclusiones obtenidas subrayan la importancia de la colaboración entre la IA y los profesionales de la salud. Por último se proponen direcciones futuras para la mejora y expansión del enfoque propuesto.

## Palabras clave

ECG, predicción de anomalías, STFT, CWT, explicabilidad en la IA, redes neuronales profundas, clasificación multietiqueta, riesgo cardiovascular.



# Abstract

## Explainable Cardiovascular Risk Prediction using Deep Learning

This work focuses on the use of Artificial Intelligence techniques for the classification of electrocardiogram (ECG) signals, a highly relevant area in medicine for the diagnosis of cardiac diseases. The study has two main objectives: to explore and evaluate different approaches for ECG classification and to apply explainability to some of the trained models.

For the first objective, we use open databases such as PTB-XL and apply signal transformations like the Short-Time Fourier Transform (STFT) and Continuous Wavelet Transform (CWT). These transformations are used to train and compare classic models and modified models designed to work with these transformations, analyzing their performance using metrics.

Model explainability is also a key aspect of the study, as it can be employed both for educational purposes and to justify decisions made by the model. At the end of this work, the results are validated by a medical expert to ensure their clinical and educational applicability.

The findings highlight the importance of collaboration between AI and healthcare professionals. Finally, future directions are proposed for improving and expanding the proposed approach.

## Keywords

ECG, anomaly detection, STFT, CWT, AI explainability, deep neural networks, multilabel classification, cardiovascular risk.



# Capítulo 1

## Introducción

**RESUMEN:** En este capítulo abordaremos la relevancia de la IA en el ámbito médico, describiremos los objetivos y el alcance del trabajo, y presentaremos la estructura general del documento.

### 1.1. Fundamentos del electrocardiograma

Un electrocardiograma (ECG) es una prueba diagnóstica no invasiva que registra la actividad eléctrica del corazón a lo largo del tiempo. Mediante la colocación de electrodos en puntos específicos del cuerpo (usualmente diez), se captan variaciones en los potenciales eléctricos producidos por el músculo cardíaco (Ribeiro et al., 2020). Estas variaciones se reflejan en la forma de ondas y complejos (P, QRS, T), cuya interpretación permite identificar diversas patologías. Aunque estas mediciones pueden hacerse de diversas formas, lo habitual es medir la diferencia de potencial entre doce pares de electrodos. Cada una de estas mediciones genera una señal, que recibe el nombre de derivación.

Dado que no es posible recoger una cantidad continua de mediciones en el tiempo, se toma una cantidad de mediciones finita a una frecuencia específica (usualmente de 100Hz a 500Hz, dependiendo de la precisión del aparato medidor). Por tanto un ECG puede almacenarse en doce vectores de igual tamaño (variable dependiendo de la duración y frecuencia de la prueba). En la Figura 1.1 podemos ver un ejemplo de electrocardiograma (extraido de Wagner et al. (2022) y dibujado mediante la librería de python *ecg\_plot*).

La frecuencia a la que se mide un ECG es muy importante. Si tiene una mayor frecuencia, la onda se puede ver con una mayor resolución, lo que permite detectar mejor las posibles anomalías. No obstante, además de que un aparato con capacidad de medir a más resolución es más complejo y caro, hay otras desventajas de tener una frecuencia muy alta. Una mayor frecuencia implica que para almacenar un ECG de la misma duración necesitamos más memoria, y será computacionalmente más costoso cualquier tipo de procesamiento o análisis que queramos realizar sobre este.

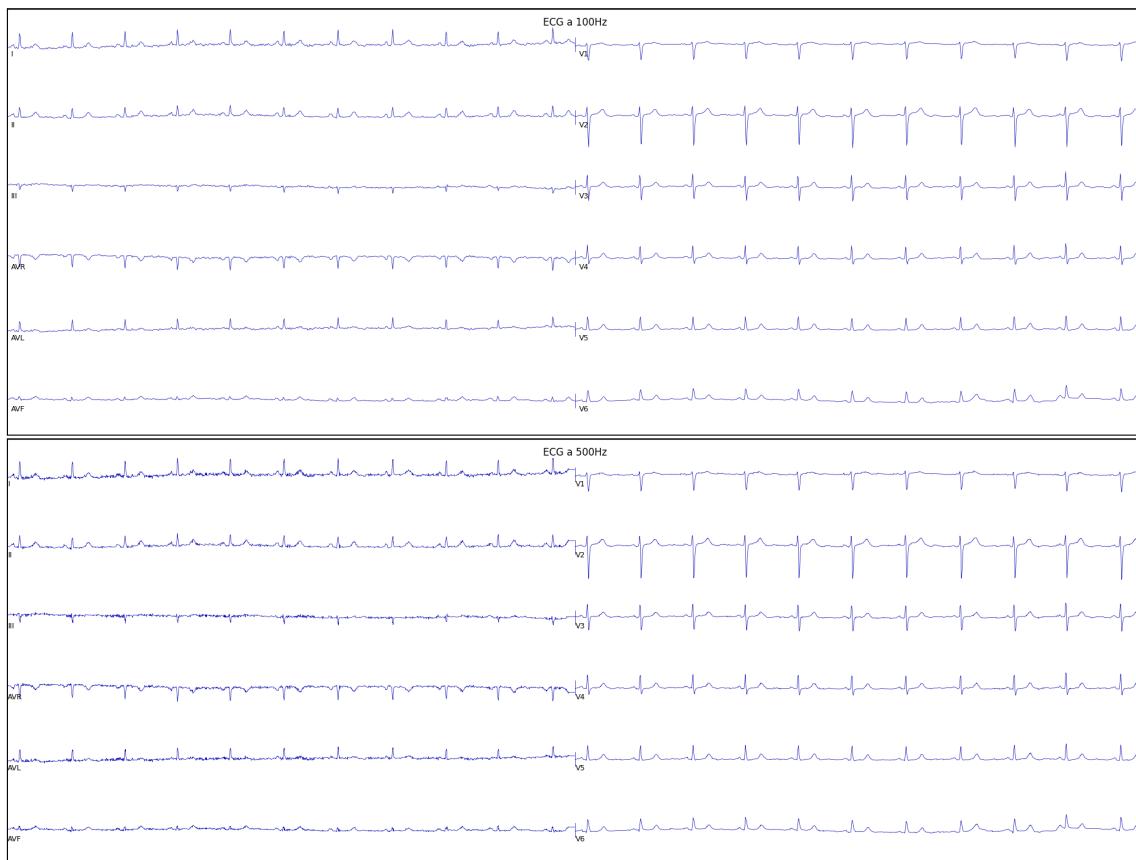


Figura 1.1: ECG de 12 derivaciones tomado a 100Hz (arriba) y 500Hz (abajo).

La relevancia clínica del ECG es enorme, además de ser una prueba de bajo coste y gran disponibilidad, el electrocardiograma constituye el primer paso para la detección de anomalías cardíacas en servicios de urgencias, consultas de atención primaria y especializadas. De acuerdo con la Organización Mundial de la Salud (OMS), las enfermedades cardiovasculares representan una de las principales causas de mortalidad a nivel global (World Health Organization, 2021); por ello, optimizar su diagnóstico temprano a través de métodos automatizados de análisis y clasificación puede tener un impacto significativo en la mejora de la salud pública.

En este contexto, se han desarrollado modelos de *Deep Learning* muy precisos para el diagnóstico interpretable de anomalías cardíacas (Lu et al., 2024). A lo largo de este capítulo se profundizará en aspectos esenciales de la señal, se expondrán los fundamentos de las ondas principales y se describirán las bases de datos de ECGs más utilizadas para la investigación.

### 1.1.1. Ondas y segmentos principales

En la Figura 1.2 se muestra un trazado típico de un ECG con etiquetas para cada onda y segmento. A grandes rasgos se pueden distinguir:

- **Onda P:** la primera elevación relativamente pequeña.

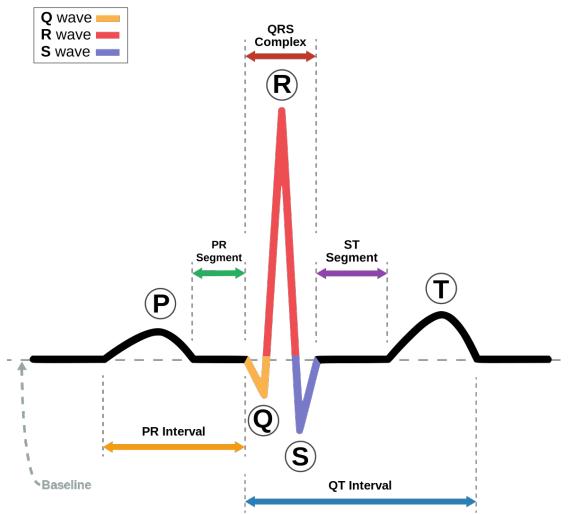


Figura 1.2: Ejemplo de ECG con etiquetas en P, QRS, T, ST y PR. Fuente: Wikimedia Commons (2023)

- **Complejo QRS:** El conjunto de picos y valles centrales, normalmente lo más destacado.
- **Onda T:** La elevación posterior que aparece después del complejo QRS.
- **Segmento ST:** La sección entre el final del complejo QRS y el inicio de la onda T.
- **Intervalo PR:** El espacio desde el inicio de la onda P hasta el comienzo del complejo QRS.
- **Intervalo QT:** El espacio desde el inicio de la onda Q hasta el final de la onda T.
- **Segmento PR:** La sección entre el final de la onda P y el comienzo del complejo QRS.

Aunque cada uno de estos componentes tiene implicaciones fisiológicas y diagnósticas, en este trabajo no entraremos en ellos, ya que el conocimiento profundo de los procesos cardíacos que generan cada onda no es necesario para las técnicas de análisis y clasificación que utilizaremos. Para una descripción en detalle de las partes de un ECG, así como de la relación con los movimientos del músculo cardíaco, el lector puede referirse a **Manual de Electrocardiografía Básica** Macfarlane et al. (2010).

### 1.1.2. Principales anomalías en un ECG

Existen numerosas anomalías en un ECG que pueden reflejar alteraciones en la función cardíaca. En este trabajo, agruparemos las anomalías de la misma forma en

la que están agrupadas en la base de datos elegida, que podemos ver en la Sección 2.1.2.

## 1.2. Motivación

La Inteligencia Artificial (IA) ha transformado numerosos campos. En particular, la medicina ha experimentado avances significativos mediante la aplicación de la inteligencia artificial al diagnóstico y análisis de datos. Uno de los campos más prometedores es el de la interpretación automatizada de señales en un ECG, fundamentales para la detección temprana de enfermedades cardíacas. Las enfermedades cardiovasculares son una de las principales causas de muerte a nivel global (World Health Organization, 2021), lo que subraya la necesidad de métodos rápidos y precisos para su diagnóstico.

El análisis de ECGs ha sido históricamente una tarea realizada por profesionales de la salud, como cardiólogos, debido a la complejidad y variabilidad de las señales y de la interpretación de estas. Sin embargo, este proceso es lento, costoso y depende en gran medida de la experiencia del especialista. Con la adopción de modelos de *Deep Learning*, como las redes neuronales convolucionales (CNN), surge la oportunidad de automatizar y mejorar este análisis, proporcionando resultados rápidos, y en muchos casos, comparables a los obtenidos por expertos humanos (Hannun et al., 2019). Este enfoque no solo promete aumentar la eficiencia del diagnóstico, sino también mejorar la precisión y reducir la carga de trabajo del personal médico.

Sin embargo, los médicos se han mostrado reacios a utilizar estas tecnologías debido a que en muchas ocasiones, por ser modelos predictivos que no proporcionan explicaciones (modelos de caja negra), no pueden comprender su funcionamiento. Para que este tipo de herramientas genere confianza en entornos clínicos y cumpla con los estándares de calidad y ética, es fundamental que sean explicables (Molnar, 2019). La explicabilidad de los modelos de IA hace posible comprender y justificar las decisiones tomadas durante el análisis de la señal cardíaca, un factor crítico en aplicaciones de alto impacto como la medicina, donde un error puede afectar directamente a la salud de los pacientes (Goodman y Flaxman, 2017).

Si bien se ha realizado una cantidad significativa de investigación en el campo de análisis y predicción de riesgos a partir de ECGs (por ejemplo ?), no son tantos los trabajos que hay que ponen el foco en tener modelos explicables, lo que es muy importante (como señalábamos en el párrafo anterior) para que estos modelos sean realmente utilizados. Creemos que, si los modelos transformados obtienen resultados similares (o mejores) que el modelo original, estos podrían dar mejores resultados a la hora de aplicar técnicas de explicación.

COMENTARIO: Esto lo he añadido para resaltar el motivo de utilizar transformadas, ver si dan mejores explicaciones. pero claro, entre que el modelo no da buenos resultados con las transformadas y que hay limitaciones técnicas que descubrimos más adelante, no hemos llegado a probar explicar las transformadas. Esta es la última revisión del trabajo, así que creo que o dejo este último párrafo o lo quito, pero no quiero estar haciendo cambios de redacción mayores (salvo erratas o meteduras de pata similares) a estas alturas.

A lo largo de este trabajo, exploraremos diversas arquitecturas de redes neuronales y técnicas de transformación de señales para clasificar y detectar anomalías en ECGs, con el objetivo de desarrollar y mejorar un modelo clasificador de ECGs en cuatro grupos de anomalías cardíacas. Además, incorporaremos métodos de explicabilidad para que los resultados del modelo puedan ser interpretados y validados por profesionales de la salud, lo que es esencial para la adopción clínica de estas tecnologías.

## 1.3. Objetivos

El principal objetivo de este trabajo es modificar, entrenar y evaluar el modelo propuesto originalmente por Ribeiro et al. (2020) (en adelante, '*gold standard*'), aplicándolo sobre una base de datos pública. Para lograrlo, introduciremos una modificación en la capa de entrada del modelo que permita la inclusión de transformadas (por ejemplo STFT o CWT) a fin de explorar si la conversión de la señal en distintas representaciones puede mejorar el rendimiento. No obstante, no se realizan cambios en la arquitectura interna del modelo.

Con base en lo anterior, los objetivos específicos son:

1. **Entrenar el *gold standard*** con una base de datos pública.
2. **Modificar el *gold standard*** para que admita transformaciones de la señal, posibilitando el entrenamiento de versiones alternativas del modelo con datos transformados.
3. **Comparar el rendimiento** de cada variante con el *gold standard* mediante métricas como F1-Score, analizando si las transformaciones resultan o no beneficiosas.
4. **Aplicar un método de explicabilidad** para que un especialista pueda comprender qué partes de la señal de un ECG influyen en la predicción.

## 1.4. Estructura del documento

El presente documento se organiza en seis capítulos, cuyo contenido se describe a continuación:

- **Capítulo 2: Estado del arte**

Se exponen los conceptos básicos sobre electrocardiogramas, resaltando la importancia de sus ondas (P, QRS, T) y las anomalías más comunes que se suelen detectar. Asimismo, se revisa la literatura relacionada con el uso de redes neuronales en el análisis de ECGs y se describen bases de datos relevantes (como PTB-XL).

- **Capítulo 3: Metodología y preparación de datos**

Se detalla el proceso de tratamiento de los ECGs, incluyendo la división en conjuntos de entrenamiento, validación y prueba. Además, se describen las métricas utilizadas para evaluar los modelos y las transformaciones empleadas.

- **Capítulo 4: Entrenamiento y resultados**

Se explican las condiciones de entrenamiento de cada modelo (incluyendo las modificaciones del *gold standard*) y las librerías utilizadas y se presentan y comparan los resultados cuantitativos obtenidos por las métricas.

- **Capítulo 5: Explicabilidad**

Se describe el método de explicabilidad basado en *saliency maps* de gradientes aplicado al modelo de Ribeiro. Asimismo, se incluye la perspectiva de un médico especialista que analiza las explicaciones generadas y se reflexiona sobre el método y posibles mejoras.

- **Capítulo 6: Conclusiones y trabajo futuro**

Se integran las conclusiones derivadas de los resultados, valorando en qué medida se han cumplido los objetivos planteados. Además, se señalan las principales contribuciones de este trabajo y se proponen líneas de investigación futuras (como la inclusión de capas Conv2D).

Por último, se incluye un apartado de bibliografía, donde se recogen todas las fuentes consultadas a lo largo del documento, así como un anexo con el código utilizado<sup>1</sup> y otro con la totalidad de librerías instaladas en el entorno de *Python* sobre el que se ejecutó el código.

---

<sup>1</sup>Todo el contenido de este trabajo puede encontrarse en este repositorio de github.

# Capítulo 2

## Estado del Arte

**RESUMEN:** En este capítulo revisaremos las principales bases de datos disponibles y examinaremos los métodos de *Deep Learning* más relevantes para su análisis, destacando el modelo de Ribeiro et al. (2020) como punto de referencia en el estado del arte. Por último, revisaremos el significado de explicabilidad en el ámbito de la IA y mencionaremos algunos trabajos que tratan temas relacionados con este.

### 2.1. Bases de datos

Para entrenar modelos que permitan analizar automáticamente las señales de un ECG es esencial la disponibilidad de una gran cantidad de ECGs. En esta sección se revisan brevemente algunas colecciones de ECGs ampliamente utilizadas en el ámbito de la investigación y se presentan en detalle las características de PTB-XL, la base de datos elegida para la realización de este trabajo.

#### 2.1.1. Bases de datos disponibles

Como queremos comparar el rendimiento de varias modificaciones de un modelo ya existente, lo ideal sería poder trabajar con la misma base de datos con la que se entrenó el modelo de Ribeiro. Esta base de datos es CODE (Ribeiro et al., 2021b), pero no es pública, por lo que esto no es una opción.

Existe una versión pública reducida de esta base de datos llamada CODE-15 (Ribeiro et al., 2021a), que tiene alrededor de 350000 casos. No obstante, no podemos comparar un modelo entrenado con los datos de CODE-15 con otro entrenado con los de CODE, porque la ventaja en el conjunto de datos de entrenamiento haría imposible comparar la eficiencia de manera equitativa. Por ello, vamos a volver a entrenar el *gold standard* con una base de datos pública, PTB-XL (Wagner et al., 2022). Esta base de datos cuenta con unos 22000 casos de prueba, pero todos estos

casos están con el mismo formato, por lo que el procesamiento de los mismos será más sencillo. Adicionalmente, dado que el objetivo es comparar distintas modificaciones del modelo para evaluar si mejora el rendimiento, es mucho más importante emplear bases de datos uniformes y poder realizar más entrenamientos que usar una base de datos más grande, lo que incrementaría significativamente el tiempo de entrenamiento.

### 2.1.2. PTB-XL

La base de datos PTB-XL es un conjunto de registros de ECGs de 12 derivaciones que reune información de miles de pacientes, con edades y condiciones de salud variadas (Wagner et al., 2020). Se caracteriza por:

#### Número de muestras

La base de datos incluye 21799 registros. Cada registro tiene una duración de 10 segundos y está disponible a una frecuencia de 100Hz y 500Hz. El hecho de que todos los registros sean uniformes facilita en gran medida el preprocesamiento de los datos.

#### Etiquetas clínicas

Se proporciona un conjunto de anotaciones diagnósticas que abarcan una amplia serie de condiciones. Adicionalmente, se incluye un archivo en formato *.csv* con información sobre cada condición.

Todas las posibles anomalías de un ECG se agrupan en cuatro clases diferentes, con una clase adicional para los ECGs categorizados como normales. Las etiquetas son las siguientes:

- **Normal (NORM):** Esta etiqueta significa que el ECG se categoriza como normal, y por tanto no tiene ninguna anomalía<sup>1</sup>
- **Infartos de miocardio (MI):** Las anomalías en esta categoría presentan cambios similares a las STTC, pero su causa clínica es más específica.
- **Anomalías del segmento ST y onda T (STTC):** Esta categoría incluye elevaciones o descensos anómalos del segmento ST, así como cambios en la morfología de la onda T, siempre que no puedan ser clasificados como infartos de miocardio.

<sup>1</sup>Esto no significa que un ECG con esta etiqueta no pueda tener también otras etiquetas. Por como funciona el modelo, se decide para cada una de las 5 etiquetas la probabilidad de que pertenezca a esta, y luego (mediante un umbral de 0.5), se binariza la pertenencia a etiquetas. Esto hace que el modelo pueda etiquetar un mismo ECG como normal y con una anomalía, lo cuál no es intuitivo. Una posible línea de trabajo futuro es modificar esta etiqueta para que sea exclusiva, es decir, que solo se aplique si no coincide con ninguna anomalía.

- **Anomalías de la conducción (CD):** Esta categoría se caracteriza por ensanchamientos en el complejo QRS o modificaciones de la progresión de las ondas en el trazado.
- **Hipertrofia (HYP):** Esta categoría (que es la menos común) presenta alteraciones en la amplitud del complejo QRS o cambios en las ondas ST/T.

En la Figura 2.1 podemos ver ECGs que presentan cada tipo de anomalía. Para una descripción detallada y exhaustiva de las anomalías cardíacas y su clasificación, el lector puede referirse a **Braunwald's Heart Disease: A Textbook of Cardiovascular Medicine** Libby et al. (2021).

### Formato de los datos

Los ECGs se almacenan en formato WFDB (*Waveform Database*), un estándar ampliamente utilizado en el ámbito médico para almacenar datos de señales anotados (Xie et al., 2023).

### Información de pacientes

Cada ECG está asociado a información demográfica (edad, sexo, etc.)

En este trabajo únicamente tendremos en cuenta las doce derivaciones del ECG, así como las anotaciones que permiten clasificarlo en una de las cinco etiquetas contempladas en la Sección 2.1.2.

## 2.2. Modelos para la clasificación de ECG

### 2.2.1. Enfoques basados en técnicas tradicionales de aprendizaje automático

El desarrollo de clasificadores ha sido crucial en el campo de la IA enfocada a la detección de anomalías en señales de ECGs. Entre estos métodos, los árboles de decisión y el método k-NN han mostrado una eficiencia notable gracias a su sencillez y capacidad para modelar comportamientos complejos (Guo et al., 2023).

No obstante, estos enfoques presentan un problema, pues requieren seleccionar de manera manual una serie de características en la señal (como amplitud de la onda R, duración del complejo QRS, etc.), cosa que resulta extremadamente difícil ya que el análisis de electrocardiogramas es muy complejo. Aunque estos métodos resultan efectivos en escenarios con un volumen de datos moderado, su rendimiento depende en gran medida de las características anteriormente mencionadas (Acharya et al., 2017).

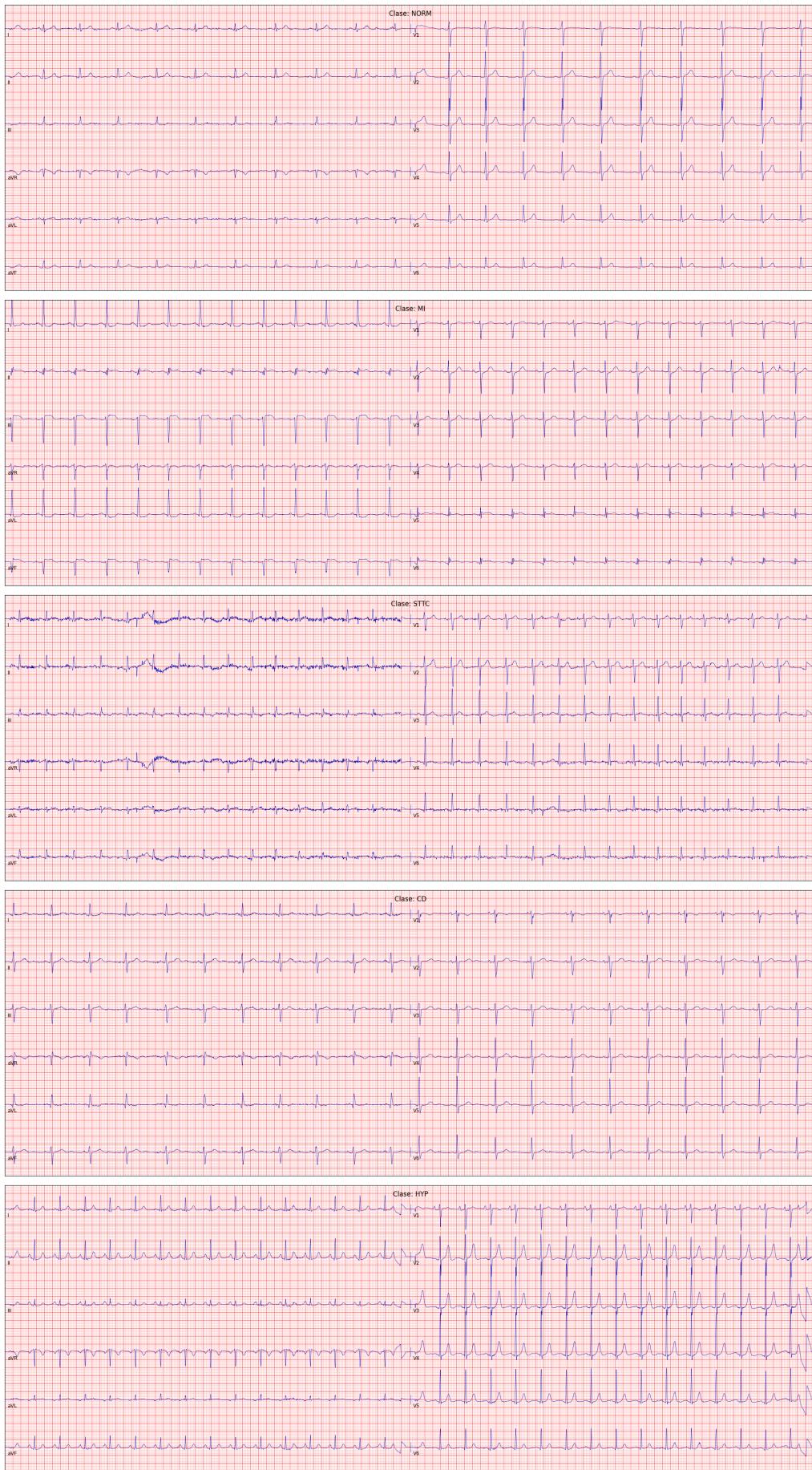


Figura 2.1: ECGs de todos los tipos de anomalías.

### 2.2.2. Modelos basados en redes neuronales

Las redes neuronales son un modelo computacional inspirado en las neuronas biológicas. Están formadas por capas de neuronas artificiales conectadas entre sí que aprenden a transformar los datos de las entradas en las salidas deseadas. Este aprendizaje se realiza ajustando los pesos de cada conexión, comúnmente mediante el algoritmo de *backpropagation* (Rumelhart et al., 1986), para minimizar un error definido (p.ej., la diferencia entre la salida prevista y la real). De esta forma, en lugar de depender de técnicas de extracción de características diseñadas manualmente, las redes neuronales aprenden internamente los rasgos más relevantes a partir de los datos.

En particular, las CNNs (*Convolutional Neural Networks*, Redes Neuronales Convolucionales) se han popularizado en tareas de visión por computador y análisis de señales al emplear operaciones de convolución (ver siguiente apartado).

Aunque nacieron enfocadas al reconocimiento de imágenes, se han adaptado con éxito a otros tipos de datos, como ondas, usando convoluciones unidimensionales en lugar de bidimensionales. Esto permite a la red detectar características relevantes de la señal (como la forma o duración de los intervalos) sin necesidad de ser elegidas manualmente por un experto, lo que le otorga una gran capacidad de generalización (Hannun et al., 2019).

## Convoluciones

Una convolución es una operación matemática que toma dos funciones o señales y produce una tercera, mostrando cómo una de ellas “filtra” a la otra. Por ejemplo, en el procesamiento de imágenes, la convolución se utiliza para aplicar filtros o detectores de bordes: se toma una matriz de píxeles (la imagen) y se combina con un kernel (también llamado filtro), multiplicando y sumando los valores de cada posición para generar un nuevo valor en la imagen resultante. Este proceso, repetido a lo largo de toda la matriz, ayuda a resaltar características como contornos o texturas específicas.

En redes neuronales, se utilizan convoluciones discretas (es decir, diseñadas para trabajar con una cantidad de datos discreta, lo que ocurre siempre ya que los datos son finitos). La convolución es la base para extraer características relevantes de datos de entrada como imágenes o señales (como es el caso de un ECG). Cada capa convolucional aprende pesos que responden a cierto patrón, de modo que, al aplicar esa capa sobre la entrada, se pueden detectar patrones específicos (por ejemplo, la presencia de bordes horizontales o verticales en una imagen). A través de varias capas de convolución, el modelo va construyendo representaciones cada vez más complejas, y eso permite un gran éxito en tareas de clasificación (como es el caso de los ECGs). Esta eficiencia y capacidad de extracción de características es lo que ha convertido a las convoluciones en una herramienta fundamental dentro de la informática moderna, especialmente en los campos de visión por computadora.

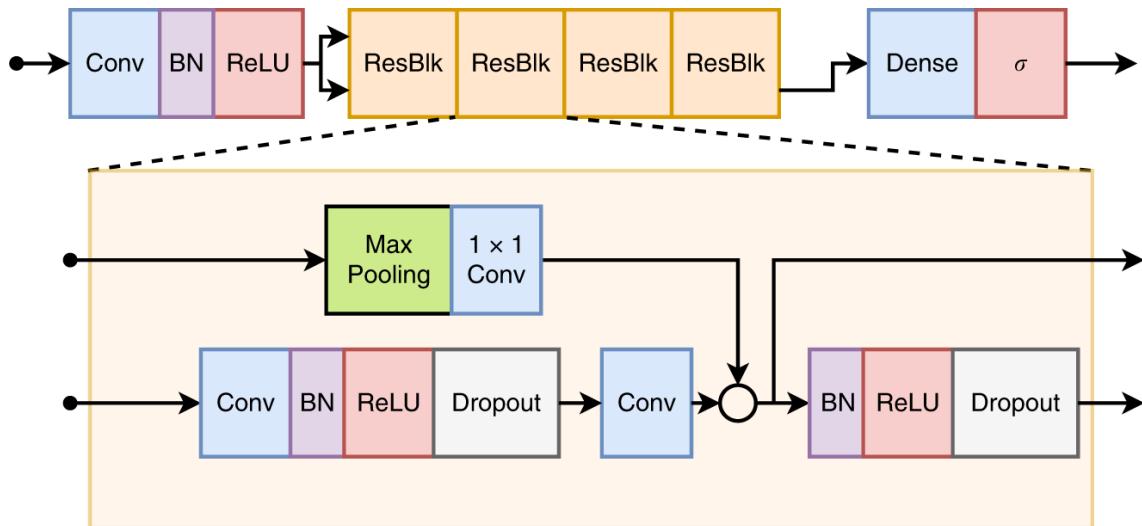


Figura 2.2: Arquitectura de la red neuronal que se emplea en el *gold standard*. Fuente: Ribeiro et al. (2020)

### 2.2.3. El modelo de Ribeiro

Dentro de los enfoques basados en redes convolucionales, destaca el modelo propuesto por Ribeiro et al. (2020), cuyo objetivo es la clasificación multiclase de ECGs de 12 derivaciones. La arquitectura (que podemos ver en la Figura 2.2) se caracteriza por una serie de capas convolucionales unidimensionales adaptadas a la naturaleza de la señal. Esto permite analizar el electrocardiograma como una señal, sin necesidad de representarla como una imagen.

Los distintos tipos de capas que utiliza el modelo son las siguientes (para ver más sobre los tipos de capas y su comportamiento completo, el lector puede referirse a Deep Learning Goodfellow et al. (2016)):

1. **Conv**: Las capas convolucionales mencionadas anteriormente.
2. **BN**: Son capas de *batch normalization*, que mejoran y aceleran el entrenamiento.
3. **ReLU**: Capas de activación, que devuelven 0 si el valor de la entrada es negativo.
4. **Dropout**: Una capa que desactiva un porcentaje aleatorio de las neuronas en cada entrenamiento, para evitar ciertos problemas como la sobrespecialización.
5. **Max Pooling**: Se utiliza para reducir la dimensionalidad de la entrada.
6. **Dense**: Esta capa se suele utilizar antes de una función de activación para adecuar el número de salidas del problema
7.  $\sigma$ : Capas sigmoides, transforman la salida de las neuronas en un valor real entre 0 y 1. Son útiles para acotar la salida.

Como puede verse en el artículo de Ribeiro et al. (2020), el modelo demostró un alto desempeño en la detección de diversas anomalías. En este trabajo, adaptaremos ligeramente la capa de entrada de este modelo para poder tomar como entrada matrices de cualquier tamaño (originalmente el modelo requería de matrices de dimensión 12xN) para poder entrenarlo con transformaciones de la señal original a imágenes. Sin embargo, no modificaremos la estructura interna de las capas convolucionales.

## 2.3. Explicabilidad en la IA

A pesar de que los modelos de *Deep Learning* han demostrado su eficacia en tareas de diagnóstico clínico, es necesario contar con métodos que ofrezcan explicaciones y permitan interpretar sus predicciones, ya que la actual legislación europea (*GDPR*) exige poder dar una explicación comprensible de cualquier decisión tomada por un algoritmo o modelo predictivo. Esto es particularmente relevante en el ámbito médico, donde cualquier decisión que tome un algoritmo debe poder justificarse ante profesionales de la salud, ya que la salud de una persona podría verse afectada por esta decisión.

Algunos de los métodos más notables de explicación en la IA son los siguientes:

### Métodos intrínsecos

Estos métodos consisten en analizar el código del modelo de IA para comprender qué hace exactamente un modelo por dentro. Estos métodos solo son aplicables cuando el modelo es intrínsecamente explicable, es decir, que hay una clara correlación entre el código del modelo y la salida del mismo, como es el caso de los árboles de decisión o regresiones lineales.

Las redes neuronales no entran dentro de este tipo de modelos, y por tanto no podemos aplicar este tipo de explicaciones en este trabajo.

### Métodos basados en visualización

Estos métodos consisten en representar gráficamente qué partes de la entrada son más relevantes a la hora de tomar una decisión. Por esto, están estrechamente relacionados con el significado de los datos de entrada, y no tanto con su forma. Supongamos que tenemos como entrada una matriz de dos dimensiones. Esto puede interpretarse (entre otras cosas) como una imagen o como una serie de vectores de mediciones distintas (esto último se conoce como serie temporal) en el tiempo:

- Si los datos representan imágenes, existen métodos como Grad-CAM para dibujar mapas de calor sobre la imagen original que señalan las partes más relevantes de la imagen a la hora de tomar decisiones.
- Si tenemos una serie temporal, para explicar qué zonas son más importantes lo que necesitamos es señalar, en cada una de las mediciones, qué franjas

temporales son más relevantes para la decisión. Esto es lo que hacen algunos métodos como *saliency maps*.

Si bien es cierto que en este trabajo tenemos tanto series temporales (los datos originales sin transformar) como imágenes (los datos transformados), en este trabajo utilizaremos exclusivamente *saliency maps*, por motivos que veremos en el Capítulo 5.

# Capítulo 3

## Metodología y preparación de datos

**RESUMEN:** En este capítulo describimos la metodología seguida para procesar y transformar las señales de ECG de la base de datos PTB-XL, detallamos la distribución de las clases, el preprocesamiento aplicado, las transformaciones realizadas y las métricas empleadas para evaluar los modelos.

### 3.1. Análisis de los datos y distribución por clases

La base de datos PTB-XL es un conjunto de registros formado por 21799 ECGs de 12 derivaciones, considerado uno de los mayores *datasets* públicos de ECGs disponibles en la actualidad (Wagner et al., 2020). Contiene muestras de ECGs de 10 segundos de duración anotadas con múltiples etiquetas diagnósticas. Para este trabajo, emplearemos la clasificación según las superclases definidas en la propia base de datos, que coinciden con las que presentamos en la Sección 2.1.2.

La Tabla 3.1 muestra la distribución de los datos de PTB-XL considerando estas cinco superclases. Se observa que la clase “NORM” es la más numerosa, mientras que la clase “HYP” tiene una cantidad considerablemente menor de datos. Este desbalanceo de clases puede causar varios problemas en el modelo, como por ejemplo:

1. **Sobreajuste hacia la clase mayoritaria:** Al haber bastantes más datos de entrenamiento de una clase (NORM) y menos de otra (HYP), el modelo puede aprender mejor los patrones que identifican las clases mayoritarias, haciendo que sepa distinguir peor las minoritarias, lo que en este caso podría reducir notablemente su capacidad de predicción de anomalías raras (He y Garcia, 2009).
2. **Métricas no representativas:** Las métricas más comunes, como la exactitud, pueden ser poco informativas cuando hay un desbalance en los datos de prueba, ya que un modelo que predice siempre la clase mayoritaria puede tener una

Número de registros	Superclase	Descripción	Porcentaje
9514	NORM	ECG Normal	43.64 %
5469	MI	Infarto de Miocardio	25.08 %
5235	STTC	Cambio ST/T	24.01 %
4898	CD	Transtorno de la conducción	22.46 %
2649	HYP	Hipertrofia	12.15 %

Tabla 3.1: Distribución de las superclases en PTB-XL

La información de esta tabla ha sido extraída directamente del repositorio de PTB-XL.

exactitud alta. Esto puede dificultar la evaluación real del rendimiento del modelo (Yanminsun et al., 2011).

3. **Dificultad en el entrenamiento:** Las redes neuronales profundas requieren de grandes cantidades de datos de entrenamiento para poder entender patrones complejos. Si una de las clases tiene muy pocos ejemplos, es muy probable que el modelo no sea capaz de predecirla correctamente (Leevy et al., 2018).

Para abordar estos problemas existen varias estrategias, como hacer *oversampling* o *undersampling*. El *oversampling* consiste en generar datos sintéticos a partir de los que ya tenemos para balancear las clases (He et al., 2008), pero esto no es una buena técnica cuándo los datos son complejos (como es el caso de un electrocardiograma), ya que no hay una técnica clara para crear datos sintéticos coherentes. Por otro lado, el *undersampling* hace que todas las clases se queden con el mismo número de candidatos que la clase minoritaria, lo que no es una técnica adecuada cuándo los datos de entrenamiento son reducidos desde un principio (Koziarski, 2019).

### 3.2. Preprocesamiento de datos

En cualquier desarrollo de IA, antes de poder utilizar los datos hay que hacer cierto procesamiento para asegurarnos que son adecuados. Lo primero que habría que hacer es quitar los datos repetidos, incompletos o corruptos, pero afortunadamente la base de datos que estamos utilizando ya ha sido revisada por sus creadores, por lo que podemos obviar este paso.

En procesamiento de señales biomédicas (especialmente en señales que son muy sensibles a determinadas perturbaciones, como es el caso de los ECGs) es muy importante aplicar determinados filtros antes de trabajar con las señales. En este trabajo utilizaremos los scripts que se utilizaron en el trabajo de González Cabeza (2024) (que nos han sido facilitados por el autor y están basados en el trabajo original de Ribeiro et al. (2020)). En concreto, los datos se preprocesan de la siguiente manera:

- Se reescalán todos para tener una frecuencia de 400Hz, que es con la que se entrenó al *gold standard*. Por tanto, tras hacer este procesamiento previo estaremos trabajando con vectores de 4096 elementos.

- Se elimina el desplazamiento de la línea base, que son las interferencias de baja frecuencia generadas por la respiración. Como podemos ver en Chouhan y Mehta (2007), es muy importante hacer esto antes de analizar un ECG.
- Se elimina la interferencia de la línea de alimentación, que es la interferencia generada por la corriente eléctrica del aparato medidor, lo que también es importante como podemos ver en González et al. (2005).

Por último, separamos los datos en tres conjuntos. Los autores de la base de datos presentan una estratificación (es decir, una división equilibrada de los datos) en diez clases (numeradas del uno al diez), y recomiendan utilizar el noveno estrato para validación, el décimo para pruebas y el resto para entrenamiento. Siguiendo esa clasificación, tendremos los siguientes conjuntos:

- **Entrenamiento (*train*):** El conjunto mayoritario (con un 80 % de los datos, es decir 17418 casos), que será usado para entrenar al modelo.
- **Validación (*validation*):** Este conjunto (que representa el 10 % de los datos, es decir 2183) se utilizará para ajustar los parámetros del modelo en el entrenamiento del mismo.
- **Pruebas (*test*):** Este conjunto (que está formado por el 10 % restante de los datos, es decir 2198 casos) es el que utilizaremos para obtener las diversas métricas de rendimiento del modelo.

### 3.3. Métricas de evaluación

En este apartado revisamos algunas de las métricas típicamente utilizadas y que usaremos para evaluar nuestros modelos.

#### 3.3.1. Métricas habituales

Entre las métricas más habituales podemos encontrar la *F-β Score*, *precision* y *recall*.

##### Precision (Precisión)

La precisión es la proporción de predicciones positivas que son realmente positivas, o más concretamente:

$$\text{Precision} = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos} + \text{Falsos positivos}}.$$

Un valor alto de esta métrica indica que el modelo es bueno minimizando falsos positivos, es decir, cuándo el modelo predice que un dato no pertenece a una clase, esa predicción es fiable.

### Recall (Sensibilidad)

La sensibilidad mide la proporción de casos positivos que el modelo predice correctamente, o más concretamente:

$$\text{Recall} = \frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos} + \text{Falsos negativos}}.$$

Un valor alto de esta métrica indica que el modelo es bueno minimizando falsos positivos, es decir, cuándo el modelo predice que un dato pertenece a una clase, esa predicción es fiable.

### F- $\beta$ Score

El F- $\beta$  Score es una media entre la precisión y el recall, la fórmula concreta es:

$$F_\beta = (1 + \beta^2) \times \frac{\text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}.$$

El valor más habitual para esto es  $\beta = 1$ , que nos da la media armónica y permite valorar tanto la fiabilidad del modelo cuando predice positivo como negativo.

Esto es adecuado cuándo, por la naturaleza de un problema, el coste de los falsos positivos es similar al de los falsos negativos, pero no es nuestro caso. En modelos aplicados a la salud, es mucho más importante predecir las anomalías correctamente (ya que de esto puede depender la salud de una persona) que predecir correctamente la ausencia de anomalías.

Los valores de  $\beta = 0,5$  y  $2$  hacen que tenga más peso la precisión y el recall respectivamente, por lo que la primera es más adecuada para cuándo los falsos positivos tienen un coste muy alto y la segunda para cuándo son los falsos negativos los que tienen el coste más alto.

### 3.3.2. Métricas en clasificadores multietiqueta

Todas las métricas que hemos listado anteriormente están definidas para clasificadores binarios, pero nuestro clasificador es multietiqueta, por lo que es necesario adaptarlas. Tres de los enfoques más habituales a este problema son el cálculo por clases, el promedio binario y el *micro average*.

#### Cálculo por clase

Este es el enfoque más sencillo de todos. Consideramos nuestro clasificador multietiqueta como uno binario para cada una de sus etiquetas, y calculamos las métricas para cada una de las clases.

Este enfoque permite ver el desempeño del modelo en cada una de sus clases, lo que permite entender mejor cuáles son sus debilidades y fortalezas. El principal problema que presenta este método es que no da un único valor para comparar modelos, por lo que puede ser difícil determinar qué modelo es el óptimo.

### Promedio binario

El promedio binario (también conocido como *one-vs-rest*) consiste en tratar cada clase de forma independiente frente a las demás, calcular las métricas para cada clase de manera binaria y, finalmente, promediarlas. De este modo, cada clase se considera positivamente etiquetada en un escenario (con sus correspondientes verdaderos positivos, falsos positivos y falsos negativos) mientras que el resto de las clases se consideran negativamente etiquetadas.

Este método permite obtener un valor global de la métrica (por ejemplo, F1) que resume el desempeño del modelo en todas las clases, pero puede enmascarar diferencias importantes en la distribución de datos (por ejemplo, cuando el número de instancias de cada clase es muy desigual). Sin embargo, sigue siendo un enfoque útil si se desea una única medida para comparar el rendimiento de diferentes clasificadores.

#### *Micro average*

El *micro average* se basa en sumar las predicciones correctas e incorrectas de todas las clases antes de calcular las métricas. En lugar de promediar los resultados clase por clase, este método reúne todos los verdaderos positivos, falsos positivos y falsos negativos de manera global. A continuación, se calcula la métrica global a partir de estos valores agregados.

Esta aproximación proporciona un único valor general de desempeño del modelo, especialmente útil cuando se desea obtener una medida global en problemas de múltiples clases o etiquetas. El micro average, al considerar todos los ejemplos de forma conjunta, puede ofrecer una visión más equilibrada del rendimiento global, aunque a costa de no mostrar el detalle de cómo se comporta el modelo en cada clase individual.

### 3.3.3. Métricas para nuestro problema

Tras realizar el análisis de las posibles métricas que implementar, hemos decidido calcular y mostrar varias métricas para cada modelo, y elegir una que consideramos mejor para afirmar qué modelo es el mejor. Para hacer métricas globales, ya que nuestros datos presentan un importante desbalanceo en una de sus clases, utilizaremos *micro average*. Las métricas que mostraremos son las siguientes:

- Para cada una de las clases:
  - Precisión.
  - Recall.
  - F-1 Score.
- Precisión global calculada como *micro average*.

- Recall global calculado como *micro average*.
- F-1 Score global calculado como *micro average*.
- F Score ajustada, una métrica personalizada que definiremos a continuación.

Todas las métricas que mostraremos, salvo la personalizada, tienen el objetivo de entender mejor cómo funciona el modelo, no obstante, es útil escoger una sola métrica para poder comparar estrictamente qué modelo consideramos *mejor*. Esta métrica será la F Score ajustada

Dado que nuestro modelo es un clasificador en el que las etiquetas no tienen el mismo significado, ya que una de ellas representa un ECG normal mientras que las demás representan diversas anomalías, el coste de los falsos positivos o negativos varía dependiendo de la etiqueta. Nuestro objetivo es que el modelo identifique lo mejor posible las anomalías (cuándo las haya), por lo que el coste de los falsos negativos en las etiquetas de anomalías es muy alto, mientras que el coste de los falsos positivos en la etiqueta normal es muy alto.

Por ello, definiremos la F Score ajustada como la media de la F-0.5 Score de la clase normal y las F-2 Score del resto de etiquetas. Esto nos permite tener una métrica que tiene en cuenta tanto reducir falsos negativos como falsos positivos en todas las etiquetas, pero dando más peso a los falsos negativos o positivos dependiendo de la etiqueta concreta.

La fórmula concreta de la métrica sería:

$$\text{F Score ajustada} = \frac{F - 0,5(\text{NORM}) + \sum_{i \neq \text{NORM}} F - 2(i)}{\text{Número de clases}}$$

## 3.4. Transformaciones

Pueden aplicarse gran cantidad de transformaciones a una señal antes de procesarla. En nuestro caso concreto, aplicamos una transformada STFT y otra CWT, ambas en sus versiones discretas. En las siguientes secciones explicaremos la idea general de estas transformadas. Para una definición formal de estas, así como un estudio más detallado de sus propiedades y aplicaciones, pueden consultarse Oppenheim y Schafer (2009); Allen y Rabiner (1977).

### 3.4.1. STFT

La Transformada de Fourier en Tiempo Corto (STFT, del inglés *Short-Time Fourier Transform*) es una herramienta matemática utilizada para analizar la evolución espectral de una señal a lo largo del tiempo. A diferencia de la Transformada de Fourier convencional, que proporciona información global sobre la distribución de las frecuencias de una señal sin preservar la dimensión temporal, la STFT incorpora una ventana que permite observar cómo las frecuencias van cambiando localmente en el tiempo (Allen y Rabiner, 1977; Oppenheim y Schafer, 2009).

En nuestro contexto, la STFT resulta particularmente útil porque permite estudiar señales no estacionarias, es decir, aquellas cuyas características cambian a lo largo del tiempo. El ECG es un ejemplo de este tipo de señal, ya que su frecuencia varía debido factores como cambios en el ritmo, arritmias u otras alteraciones.

La STFT divide la señal en pequeños segmentos de tiempo (llamados ventanas) y aplica la Transformada de Fourier a cada uno de ellos, lo que permite obtener una representación en el dominio tiempo-frecuencia. Esto significa que podemos visualizar cómo cambian las diferentes frecuencias de la señal en el tiempo.

Esta capacidad de analizar variaciones temporales de la frecuencia hace que la STFT sea ideal para identificar eventos transitorios en el ECG, como la aparición repentina de una arritmia, alteraciones momentáneas en la forma de las ondas, o cambios en la frecuencia cardíaca, que suelen indicar anomalías médicas como las que estamos intentando detectar.

A la hora de implementar la STFT, existen varios parámetros importantes a ajustar:

- **Ventana (*window*):** Se utiliza una ventana de tipo Hann. La ventana Hann es una función que reduce el efecto de discontinuidades en los bordes del segmento.
- **Frecuencia de muestreo:** La señal original se encuentra muestreada a 400 Hz. Esta frecuencia de muestreo determina la resolución temporal que se logra.
- **Longitud del segmento:** Se utilizan 256 muestras. El segmento de datos tomado para cada ventana afecta la resolución en frecuencia. Un mayor número de puntos mejora la resolución en frecuencia, pero empeora la resolución temporal.
- **Solapamiento:** Se emplea 128 muestras. Esto significa que cada ventana se solapa con la mitad (128 muestras) de la ventana anterior. Un solapamiento mayor permite detectar eventos transitorios con mayor detalle temporal, a costa de un mayor costo computacional.

La elección de estos parámetros busca un equilibrio entre la resolución temporal y frecuencial, adecuado para el análisis de señales cardíacas. El tamaño de 256 muestras por segmento, junto con la frecuencia de muestreo de 400 Hz, ofrece una resolución en frecuencia suficiente para el rango de interés cardiológico; y el solapamiento de 128 muestras asegura una adecuada continuidad y capacidad de capturar eventos transitorios.

En la Figura 3.1 se presenta un ejemplo de la transformada STFT aplicada a la primera derivación de un ECG de cada clase diagnóstica con los parámetros que acabamos de elegir. La representación muestra cómo las frecuencias varían en función del tiempo, lo que permite ver eventos específicos como arritmias o cambios en las características de las ondas. Esta visualización puede ayudar a comprender y analizar los patrones complejos presentes en las señales del ECG.

### 3.4.2. CWT

La Transformada de Onda Continua (CWT, del inglés *Continuous Wavelet Transform*) es una técnica matemática utilizada para analizar señales no estacionarias. A diferencia de la Transformada de Fourier, la CWT descompone la señal en funciones llamadas *wavelets*, que se representan en función de tiempo-frecuencia. Esto (al igual que con la STFT) permite una mayor precisión al estudiar eventos transitorios, ya que las *wavelets* pueden ajustarse para capturar detalles de diferentes escalas temporales o frecuenciales.

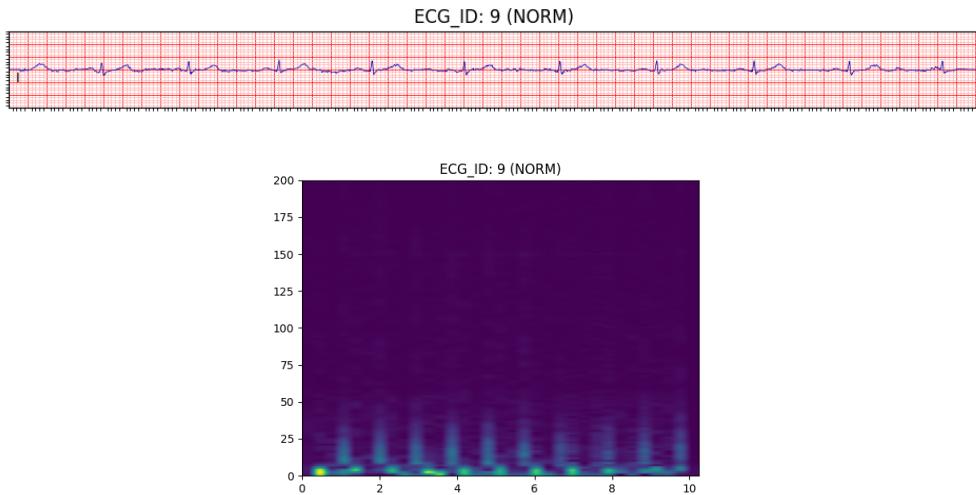
La CWT funciona mediante la correlación de la señal original con *wavelets* de diferentes escalas y ubicaciones temporales, lo que permite hacer una representación en el dominio tiempo-frecuencia. A diferencia de la STFT, donde las ventanas tienen un tamaño fijo, la CWT adapta la resolución automáticamente: se utilizan *wavelets* más cortas para capturar detalles de alta frecuencia, mientras que las más largas capturan las características de baja frecuencia. Esto permite un análisis detallado de las señales.

Al aplicar esta transformada, es necesario definir los siguientes parámetros:

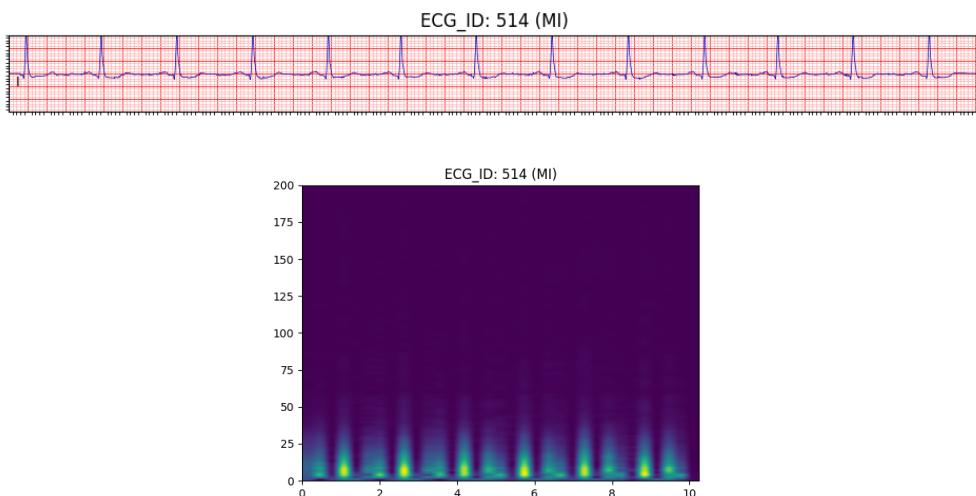
- **Wavelet:** Es la función base en la que se descompone la señal.
- **Escalas:** Son un conjunto de valores que determinan cómo se dilata (o compprime) la *wavelet* durante el análisis. Están relacionados con la frecuencia de la señal. Las escalas pequeñas corresponden a altas frecuencias mientras que las grandes corresponden a bajas frecuencias.

En este trabajo utilizaremos tanto la *wavelet* de Morlet como la de Ricker, ya que ofrecen un buen equilibrio entre localización temporal y frecuencial. Para ambas *wavelets* utilizamos cien valores de escalas, equiespaciadas desde 8 a 637 para la primera y desde 1 a 128 para la segunda.

En la Figura 3.2 se presentan dos ejemplos de transformadas con *wavelet* de Ricker y Morlet respectivamente, con los parámetros que acabamos de elegir. La representación muestra cómo las frecuencias varían a lo largo del tiempo, lo que permite ver eventos específicos como arritmias o cambios en las características de las ondas. Estas visualizaciones pueden ayudar a comprender y analizar los patrones complejos presentes en las señales del ECG con un nivel de detalle que no ofrece la Transformada de Fourier.

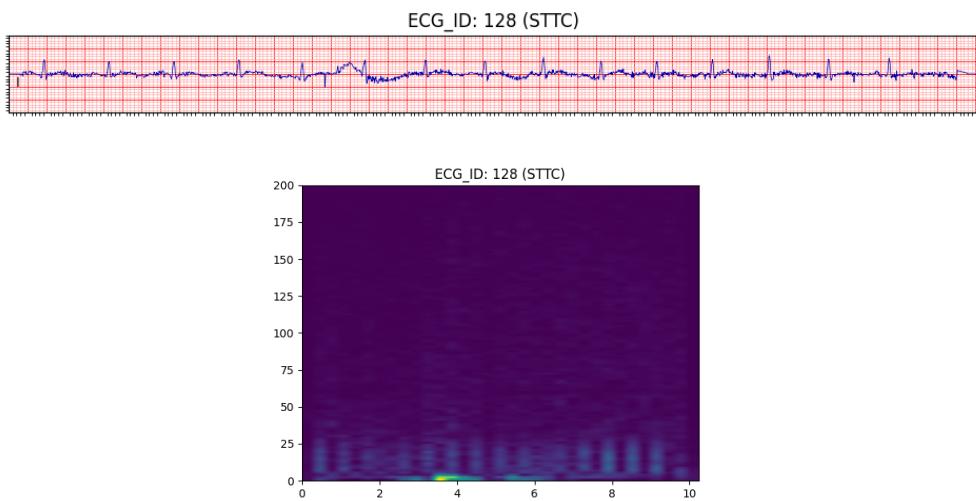


Primera derivación de un ECG normal (arriba) y su transformada STFT (abajo).

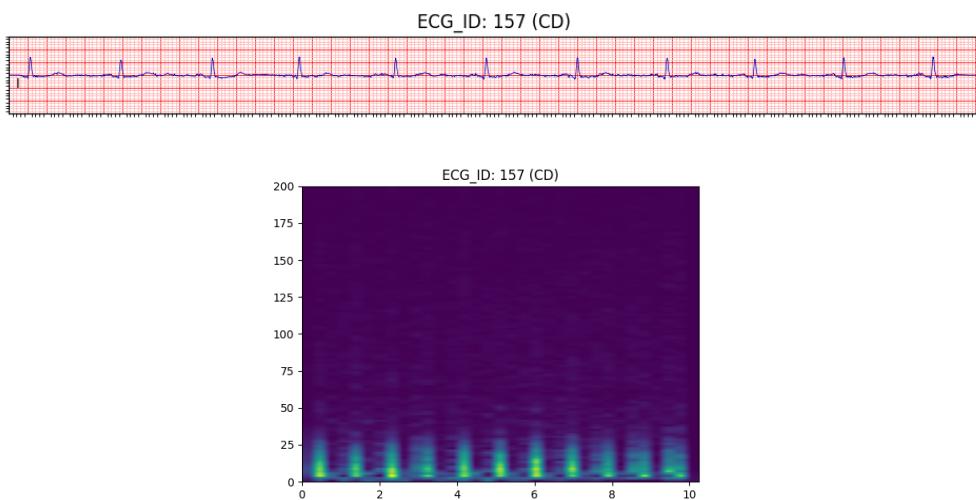


Primera derivación de un ECG de infarto de miocardio (arriba) y su transformada STFT (abajo).

Figura 3.1: Transformadas STFT de la primera derivación de ECGs para cada clase.

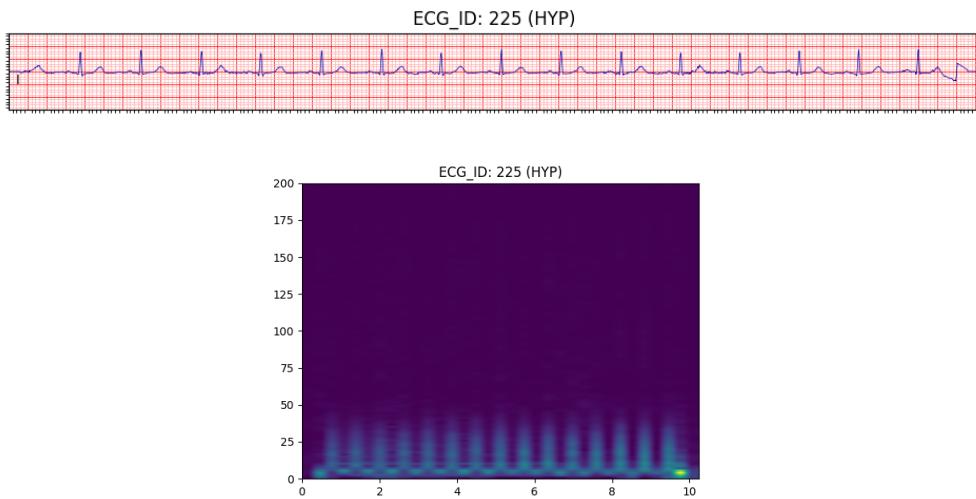


Primera derivación de un ECG de tipo STTC (arriba) y su transformada STFT (abajo).



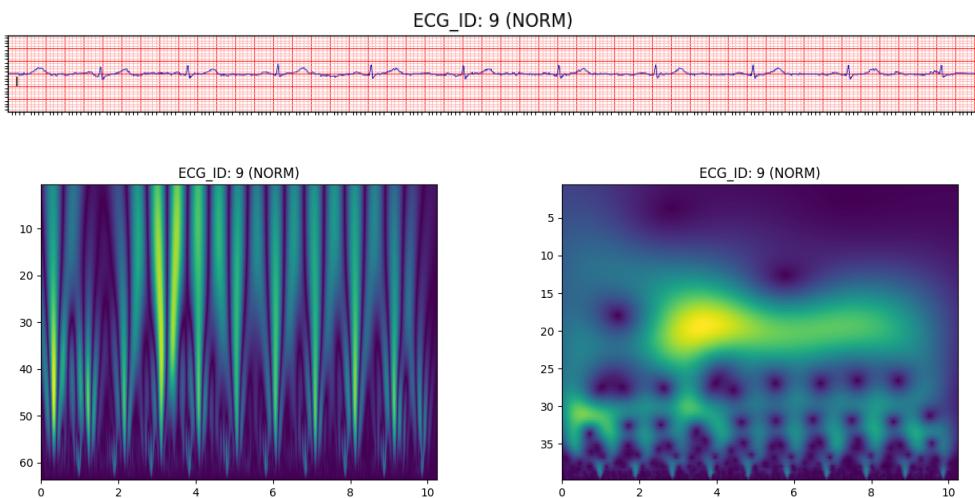
Primera derivación de un ECG de tipo CD (arriba) y su transformada STFT (abajo).

Figura 3.1: Transformadas STFT de la primera derivación de ECGs para cada clase.



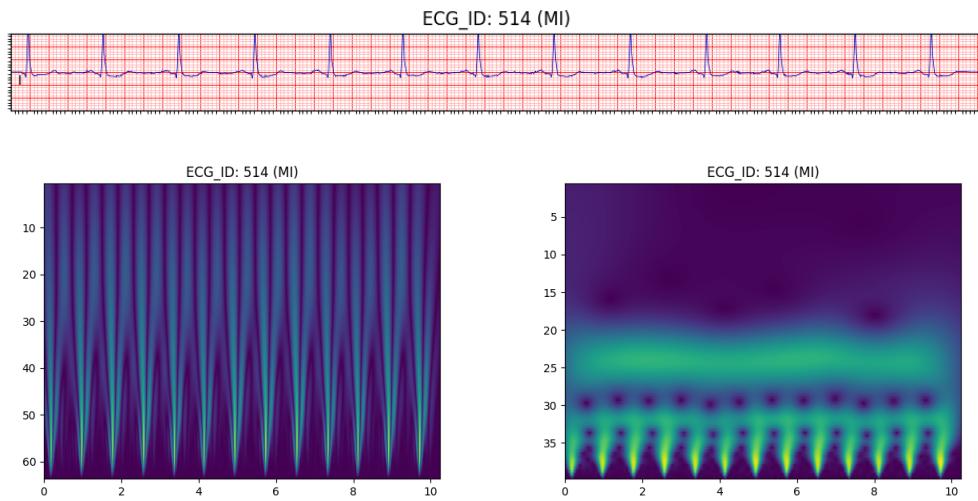
Primera derivación de un ECG con hipertrofia (arriba) y su transformada STFT (abajo).

Figura 3.1: Transformadas STFT de la primera derivación de ECGs para cada clase.

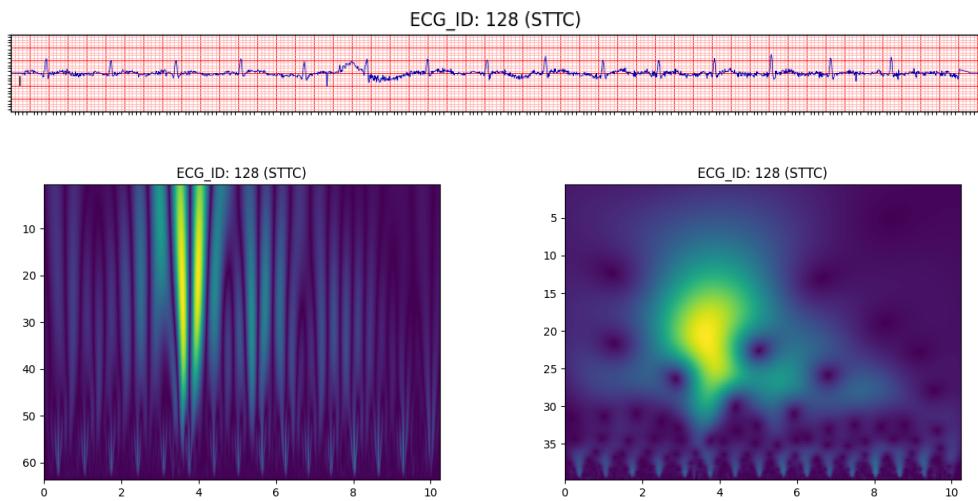


Primera derivación de un ECG normal (arriba) y su transformada CWT con *wavelet* de Ricker (izquierda) y Morlet (derecha).

Figura 3.2: Transformadas CWT de la primera derivación de ECGs para cada clase.

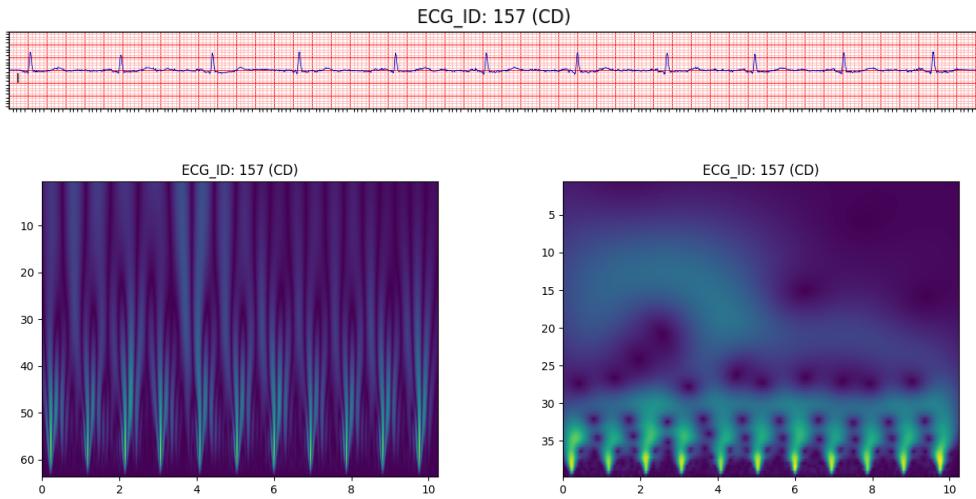


Primera derivación de un ECG de infarto de miocardio (arriba) y su transformada CWT con *wavelet* de Ricker (izquierda) y Morlet (derecha).

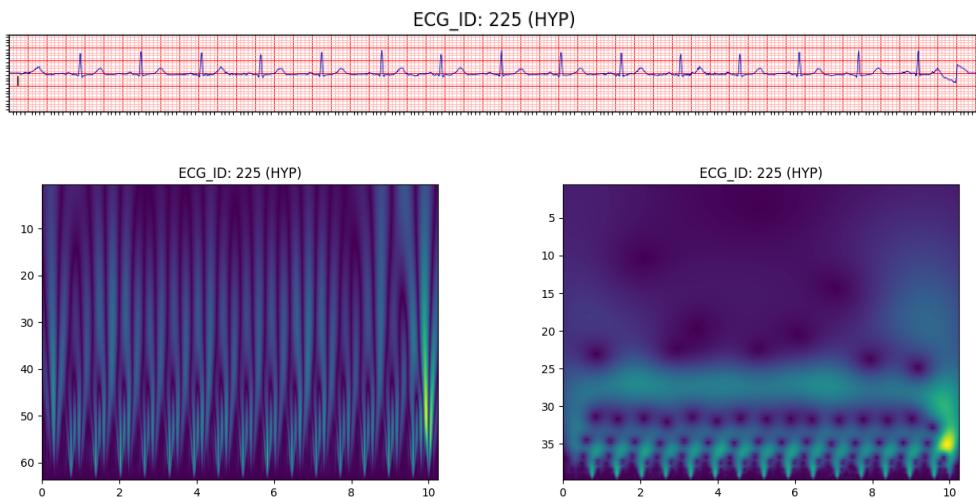


Primera derivación de un ECG de tipo STTC (arriba) y su transformada CWT con *wavelet* de Ricker (izquierda) y Morlet (derecha).

Figura 3.2: Transformadas CWT de la primera derivación de ECGs para cada clase.



Primera derivación de un ECG de tipo CD (arriba) y su transformada CWT con *wavelet* de Ricker (izquierda) y Morlet (derecha).



Primera derivación de un ECG con hipertrofia (arriba) y su transformada CWT con *wavelet* de Ricker (izquierda) y Morlet (derecha).

Figura 3.2: Transformadas CWT de la primera derivación de ECGs para cada clase.



# Capítulo 4

## Entrenamiento y resultados

**RESUMEN:** Este capítulo describe el proceso de entrenamiento de los modelos, detalla la configuración utilizada y presenta las métricas cuantitativas que permiten comparar su desempeño. Posteriormente se discuten los resultados obtenidos.

### 4.1. Modelos modificados entrenados

Modificamos la capa de entrada del *gold standard* para poder entrenar modelos con datos en matrices bidimensionales de cualquier tamaño, lo que nos permitió (con una modificación muy pequeña del *gold standard*) entrenar modelos con imágenes como datos de entrenamiento.

Al transformar cada una de las doce derivaciones obtenemos una matriz (de dimensiones dependientes exclusivamente del tamaño de la señal de entrada y de los parámetros de la transformada). Esto significa que al aplicar la misma transformada a las doce derivaciones de un ECG, obtenemos doce matrices de las mismas dimensiones, pero el modelo necesita solamente una matriz. Para abordar este problema, concatenamos las matrices en el eje de su altura, lo que nos permite utilizar el modelo modificado con los datos de las transformadas.

En concreto, los modelos modificados que entrenamos son los siguientes:

- **Modelo STFT:** Entrenado con los datos transformados utilizando los parámetros especificados en la subsección 3.4.1
- **Modelo CWT Ricker:** Entrenado con los datos transformados utilizando los parámetros especificados en la subsección 3.4.2 con el *wavelet* de Ricker.
- **Modelo CWT Morlet:** Entrenado con los datos transformados utilizando los parámetros especificados en la subsección 3.4.2 con el *wavelet* de Morlet.

## 4.2. Configuración del entrenamiento

### 4.2.1. *Hardware* y *software*

Todos los entrenamientos se han realizado en *bujaruelo*, una máquina del departamento de arquitectura de computadores y automática de la facultad de informática de la universidad.

#### *Hardware*

- **Procesador:** Intel(R) Xeon(R) CPU E5-2695 v3 @2.3GHz
- **GPUs:** El equipo tiene en total cuatro gráficas, dos de cada uno de los modelos siguientes:
  - NVIDIA GeForce GTX 1080
  - NVIDIA GeForce GTX 980
- **Discos:** El equipo cuenta con dos discos de 1TB, además del disco donde está montado el sistema operativo, de 74.5GB.

#### *Software*

En el Anexo ?? puede encontrarse una tabla con la totalidad de las librerías instaladas en el entorno de python que hemos utilizado (así como sus respectivas versiones), pero las más destacables para el trabajo son las siguientes:

- **Python:** Utilizamos la versión 3.11.7
- **TensorFlow:** Utilizamos la versión 2.15, por motivos de compatibilidad con el modelo de Ribeiro.
- **CUDA y cuDNN:** Estas librerías son las que utiliza TensorFlow para realizar los entrenamientos en las gráficas de NVIDIA. En nuestro caso utilizamos las versiones 12.2 y 8.9 respectivamente.

Además, utilizamos **Matplotlib** 3.8.2 y **Seaborn** 0.13.2 para generar los gráficos y **Pandas** 2.2.0 y **Scikit-learn** 1.3.0 para procesar y analizar los datos y generar métricas.

### 4.2.2. Parámetros de entrenamiento

Para entrenar a los modelos se ha utilizado el script de entrenamiento que aparece en el repositorio del *gold standard*, ligeramente modificado para que trabaje con los datos transformados, que tienen dimensiones diferentes a los datos originales.

## 4.3. Resultados cuantitativos

En la tabla 4.1 podemos ver los resultados de todos los entrenamientos.

Modelo	Métrica	NORM	MI	STTC	CD	HYP	Global
<i>Gold Standard</i>	F-1 Score	0.835	0.680	0.718	0.705	0.413	0.737
	Recall	0.849	0.603	0.665	0.647	0.284	0.687
	Precisión	0.822	0.778	0.780	0.773	0.759	0.796
	F Score ajustada	—	—	—	—	—	0.628
STFT	F-1 Score	0.826	0.536	0.687	0.672	0.466	0.706
	Recall	0.853	0.433	0.640	0.589	0.338	0.650
	Precisión	0.801	0.701	0.741	0.783	0.750	0.772
	F Score ajustada	—	—	—	—	—	0.587
CWT Morlet	F-1 Score	0.817	0.457	0.645	0.630	0.395	0.644
	Recall	0.857	0.342	0.636	0.548	0.270	0.623
	Precisión	0.780	0.688	0.654	0.741	0.732	0.734
	F Score ajustada	—	—	—	—	—	0.540
CWT Ricker	F-1 Score	0.775	0.316	0.629	0.587	0.426	0.628
	Recall	0.774	0.224	0.633	0.520	0.319	0.572
	Precisión	0.776	0.538	0.626	0.675	0.639	0.696
	F Score ajustada	—	—	—	—	—	0.512

Tabla 4.1: Resultados de los modelos (con tres dígitos decimales de precisión)

## 4.4. Análisis de los resultados

Los datos de la sección anterior muestran que al entrenar el *gold standard* con diversas transformadas arroja unos resultados destacablemente peores en casi todas las métricas. Esto puede deberse a:

- Una curva de aprendizaje más compleja para los modelos que utilizan transformadas, ya que generalizar patrones en estas podría ser más complejo para la red neuronal que hacerlo sobre las señales sin transformar. Esta hipótesis además se apoya en el hecho de que los modelos transformados hayan necesitado varios días para entrenarse, a diferencia del *gold standard* (que se pudo entrenar en tan solo unas horas).
- Una elección incorrecta de los parámetros para las transformadas, ya que, si bien estos se han elegido con el objetivo de cubrir los rangos de frecuencias donde aparecen las anomalías que estamos buscando, podrían elegirse una cantidad inmensa de configuraciones diferentes.
- Limitaciones de la arquitectura del *gold standard*, ya que utiliza redes Conv1D, que se especializan en detectar patrones en series temporales en lugar de en imágenes.

- Al transformar el modelo en un clasificador binario, se ha utilizado un *threshold* de 0.5 para decidir si el ECG entra dentro de cada una de las etiquetas. Es posible que utilizando diferentes valores se obtengan mejores predicciones.

# Capítulo 5

## Explicabilidad y validación clínica

**RESUMEN:** En este capítulo describimos cómo se generan y utilizan los *saliency maps* para explicar las decisiones del *gold standard*, detallamos la validación preliminar con un especialista y exponemos las mejoras introducidas a partir de su retroalimentación.

### 5.1. Método de explicabilidad utilizado

La única técnica de explicabilidad implementada en este trabajo son los *saliency maps* basados en gradientes, aplicados al *gold standard* exclusivamente. Esta decisión se debe a las siguientes razones:

1. **Desempeño superior del *gold standard*:** En el Capítulo 4 hemos visto que el *gold standard* obtiene mejores resultados en casi todas las métricas evaluadas.
2. **Imposibilidad de Grad-CAM en modelos transformados:** Como las redes transformadas mantienen la arquitectura con capas Conv1D, no se pueden aplicar métodos como Grad-CAM a la representación bidimensional. Además, los *saliency maps* no ofrecen una explicación clara sobre imágenes, ya que tratan cada una de las filas de la entrada como datos independientes.

### 5.2. Resultados de explicabilidad

Aplicando la librería de explicabilidad TSIinterpret<sup>1</sup>, hemos explicado cinco casos del conjunto de test para cada clase que cumplen las siguientes condiciones:

<sup>1</sup>Puede encontrarse su documentación en el siguiente enlace.

- El valor real de la etiqueta es solamente una condición, es decir, el valor esperado es 1.0 para una de las clases y 0.0 para el resto.
- La predicción es perfecta, es decir, la única etiqueta en la que el modelo clasifica el ECG es la que tiene un valor real de 1.0.
- La clase HYP no tiene ningún caso así, debido a que es la clase minoritaria, por lo que no hemos podido aplicar explicabilidad a esta clase.

Todas las explicaciones realizadas pueden encontrarse en la siguiente carpeta, pero en la Figura 5.1 podemos ver una imagen con la explicación de la primera derivación de un ECG de cada clase (excepto HYP).

En la Figura 5.2 puede verse una comparación entre nuestra explicación de un infarto de miocardio y la de otro artículo, y se puede comprobar que, en efecto, los modelos se fijan en las mismas zonas (en este caso concreto, después del valle más pronunciado).

### 5.3. Validación con los expertos médicos

Tras hacer las explicaciones, se las mostramos a varios médicos para obtener feedback de éstas. Su opinión general es que es una herramienta con potencial de ser muy útil, y que generalmente detecta correctamente zonas donde puede verse la anomalías, pero hizo los siguientes comentarios de cara a mejorarlo:

#### 1. Heterogeneidad de la explicación

El experto señaló que, salvo excepciones, las anomalías cardíacas se presentan en todos los latidos, no solo en algunos, por lo que es extraño que la explicación señale únicamente las anomalías en algunos latidos.

#### 2. Alternativa de presentación

El experto sugirió reducir las imágenes y mostrar únicamente dos latidos del corazón, ya que esto es una forma habitual de visualizar y explicar anomalías a los médicos en formación.

#### 3. Conformidad con la práctica clínica habitual

El experto señaló que la presentación que hemos hecho del ECG se separa bastante de la presentación habitual de estos. Indica que si las imágenes estuvieran estandarizadas según la práctica clínica habitual, podría ser una herramienta muy útil para la enseñanza, pero que en su estado actual podría llevar a confusión a los alumnos.

En general el experto tiene la sensación de que esta herramienta tiene el potencial de ser algo muy útil, principalmente en el ámbito de la enseñanza, pero que necesita implementar algunos cambios antes de ser utilizada.

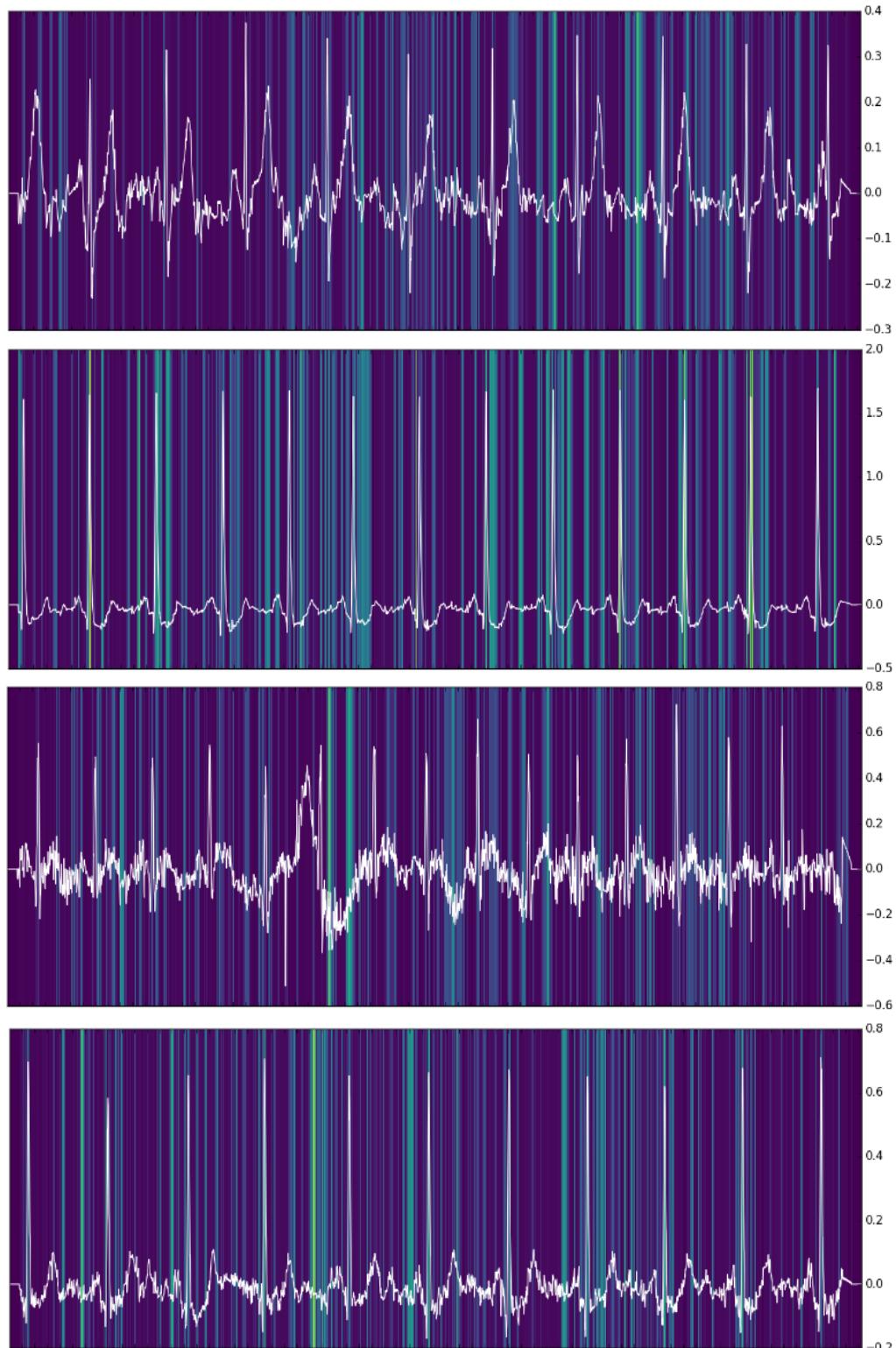


Figura 5.1: Explicaciones de la primera derivación de un ECG de las clases NORM, MI, STTC, CD (respectivamente)

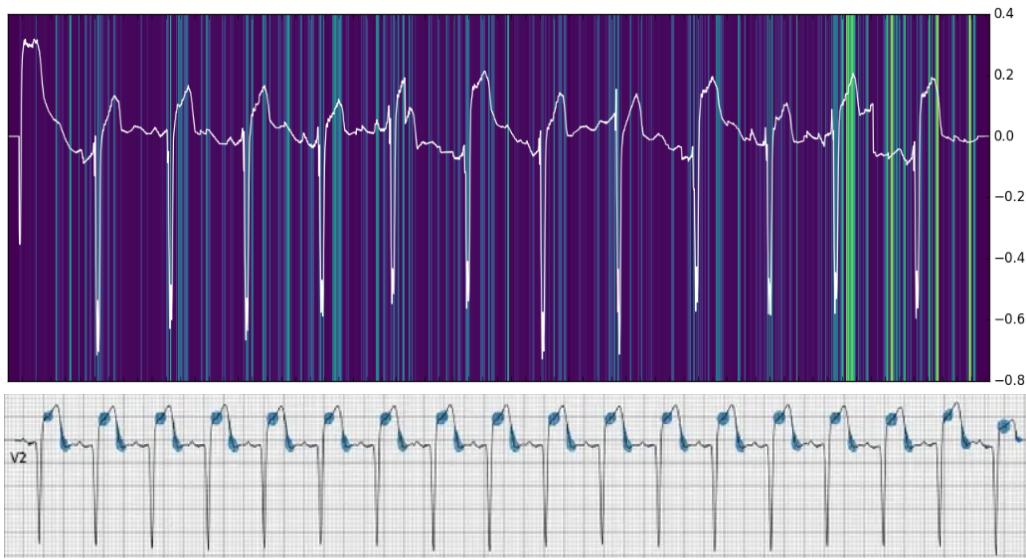


Figura 5.2: Explicación de un infarto de miocardio en nuestro modelo (arriba) y en un artículo de investigación (abajo). Fuente: Gustafsson et al. (2022).

## 5.4. Modificaciones a partir del feedback

Para poder hacer la explicación homogénea en cada latido o reducir la imagen a dos latidos representativos, necesitaríamos detectar donde empieza y acaba cada latido, así como las distintas partes de la señal. Si bien es cierto que esto es algo posible, se escapa del alcance de este trabajo, por lo que no lo implementaremos.

Respecto a la representación habitual del ECG, el motivo de que esté presentado de esta forma es que hemos utilizado la propia librería de explicabilidad para dibujar las gráficas, y ésta trabaja con series temporales en general. Para abordar este problema y mejorar la presentación, hemos modificado la imagen que genera librería *ecg\_plot*, de manera que también pueda plasmar los colores relativos a la explicación por encima. En la Figura 5.3 podemos ver una explicación mejorada para cada clase (excepto HYP).

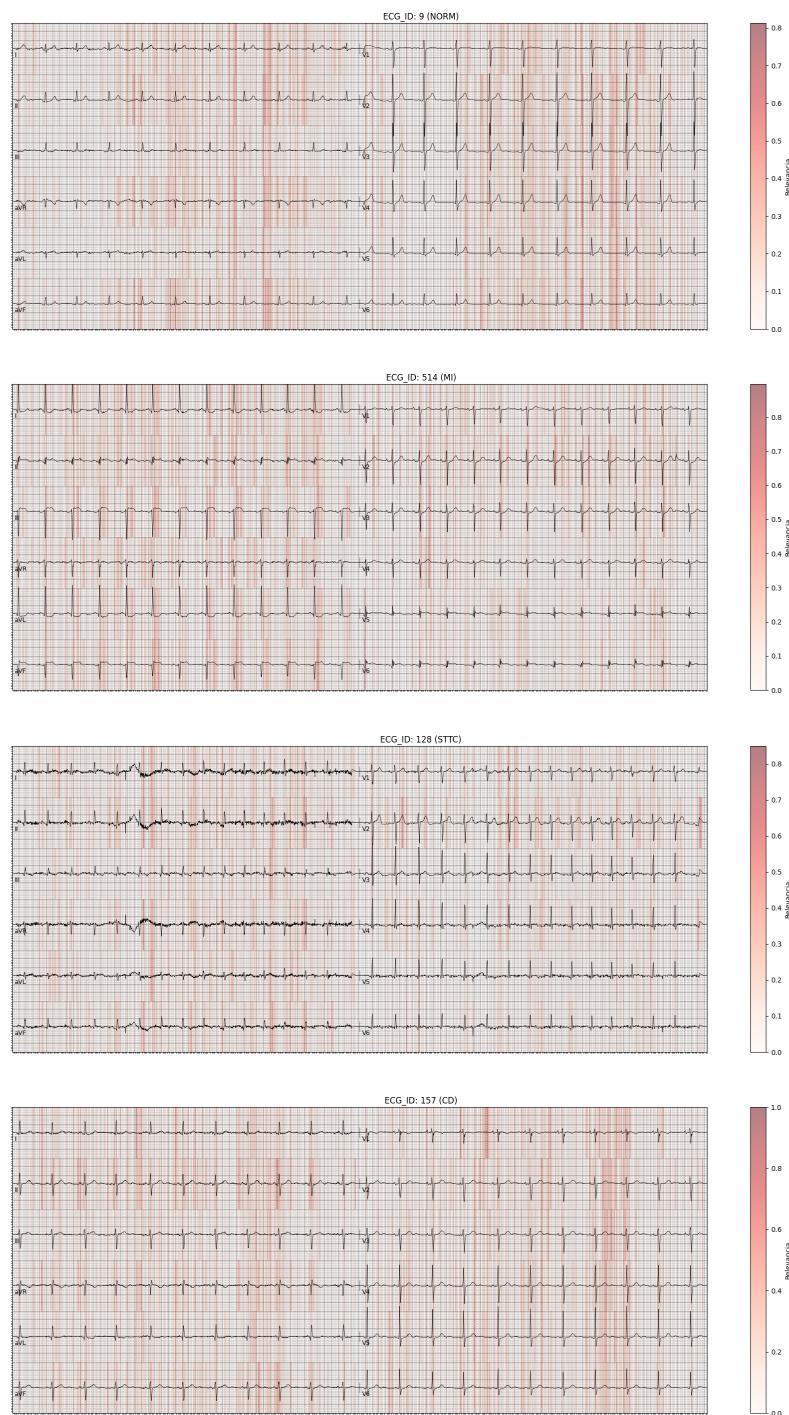


Figura 5.3: Explicación de un ECG de cada clase (excepto HYP) pintada sobre la librería *ecg\_plot*



# Capítulo 6

## Conclusiones y Trabajo Futuro

### 6.1. Conclusiones

A lo largo de este trabajo hemos demostrado que el modelo de Ribeiro funciona de manera adecuada aún entrenado con una base de datos más pequeña, lo que señala que es una arquitectura bastante sólida y prometedora.

No obstante, a la hora de introducir transformaciones, hemos descubierto que la arquitectura del modelo de Ribeiro no es adecuada para manejar imágenes, como es el caso de las transformadas. Esto no significa que el modelo de Ribeiro no pueda ser adaptado para trabajar con imágenes de manera adecuada, ya que una arquitectura con unos resultados tan buenos es muy prometedora.

Es importante señalar que el campo del *deep learning* aplicado a la medicina es relativamente nuevo, y se publican trabajos innovadores y avances muy frecuentemente, por lo que el uso de transformadas para el análisis de ECGs sigue siendo una posible dirección de avance.

En cuanto a la explicabilidad, los *saliency maps* basados en gradientes han mostrado ser un recurso prometedor para entender qué partes de la señal del ECG resultan relevantes para la red. Refinar estas explicaciones podría convertirlas en una herramienta de aprendizaje y educación útil tanto para estudiantes como para profesionales sanitarios que busquen familiarizarse con la IA. Además hemos visto como los profesionales reaccionaron muy positivamente a la existencia de herramientas que permitan explicar las decisiones de los modelos, y están convencidos de que, una vez se mejoren, van a ser un recurso muy importante en la educación.

### 6.2. Trabajo futuro

Dado el potencial del modelo y las limitaciones detectadas, se plantean varias líneas de trabajo para mejorar y ampliar los resultados:

1. **Modificar la arquitectura para admitir capas Conv2D**

Esta adaptación, si bien requeriría de un trabajo significativo, permitiría explotar mejor las ventajas de las transformaciones de la señal.

## 2. Explorar transformadas y parámetros adicionales

Siguiendo la línea anterior, una vez obtenida una arquitectura más adecuada para imágenes, podrían probarse otros parámetros a la hora de hacer las transformadas y estudiar su impacto.

## 3. Especializar el modelo en una anomalía

Transformar el modelo en un clasificador binario permitiría elegir los parámetros de las transformadas de manera más precisa, ya que distintas anomalías se manifiestan en distintas frecuencias.

## 4. Cambiar los umbrales de detección

En el trabajo de Ribeiro, en lugar de utilizar el umbral de 0.5 para clasificar las etiquetas, se calcularon los umbrales que optimizaban los resultados sobre el conjunto de validación. Esto es algo que, si bien en este trabajo no hemos abordado, podría mejorar los resultados de todos los modelos que hemos entrenado.

## 5. Delineación de señales en la explicación

Como señaló el experto, mediante delineación de señales podrían reducirse las explicaciones a dos latidos y asegurarse de que estas son homogéneas en ambos latidos. Estas mejoras harían viable el uso de estas explicaciones en el ámbito docente.

# Introduction

## 6.3. Motivation

AI has transformed numerous fields. In particular, the medical field has experienced significant advances through the application of artificial intelligence to diagnostics and data analysis. One of the most promising areas is the automated interpretation of signals in an ECG, which is essential for the early detection of heart diseases. Cardiovascular diseases are one of the leading causes of death globally (World Health Organization, 2021), highlighting the need for fast and accurate diagnostic methods.

The analysis of ECG's has historically been performed by healthcare professionals, such as cardiologists, due to the complexity and variability of the signals. However, this process is slow, costly, and highly dependent on the specialist's experience. By adopting *Deep Learning* models, such as convolutional neural networks (CNN), the goal is to automate and enhance this analysis, providing rapid results that, in many cases, are comparable to those obtained by human experts (Hannun et al., 2019). This approach not only promises to increase diagnostic efficiency but also improve accuracy and reduce the workload of medical staff.

Nevertheless, for these tools to gain trust in clinical settings and meet quality and ethical standards, they must be explainable (Molnar, 2019). The explainability of AI models permits understanding and justifying the decisions made during the analysis of cardiac signals, a critical factor in high-impact applications such as medicine, where an error can directly affect patient health (Goodman y Flaxman, 2017).

Throughout this work, we will explore various neural network architectures and signal transformation techniques to classify and detect abnormalities in ECGs, aiming to develop and refine an ECG classifier model for four groups of cardiac anomalies. In addition, we will incorporate explainability methods so that model results can be interpreted and validated by healthcare professionals, which is essential for clinical adoption of these technologies.

## 6.4. Objectives

The main objective of this work is to modify, train, and evaluate the model originally proposed by Ribeiro et al. (2020) (hereinafter, the “original model”) by applying it to the public PTB-XL database (Wagner et al., 2022). To achieve this, we will introduce a modification in the model’s input layer that allows the inclusion of transforms (e.g., STFT or CWT) to investigate whether converting the signal into different representations can improve performance. However, no changes are made to the model’s internal architecture.

Based on the above, the specific objectives are:

1. **Train the original model** using the PTB-XL database.
2. **Modify the original model** so that it accepts signal transformations, permitting the training of alternative versions of the model with transformed data.
3. **Compare the performance** of each variant with the original model using metrics such as F1-Score, analyzing whether the transformations are beneficial.
4. **Apply an explainability method** so that a specialist can understand which parts of the ECG signal influence the prediction. For reasons to be discussed later, this will only be applied to the original model.

## 6.5. Document structure

This document is organized into six chapters, whose contents are described below:

- **Chapter 2: Estado del arte**

Basic concepts related to electrocardiograms are presented, highlighting the importance of their waves (P, QRS, T) and the most common anomalies typically detected. Furthermore, the literature related to the use of neural networks in the analysis of ECG’s is reviewed, and relevant databases (such as PTB-XL) are described.

- **Chapter 3: Metodología y preparación de datos**

The process for handling ECG’s is detailed, including the division into training, validation, and test sets. In addition, the metrics used to evaluate the models and the transformations employed are described.

- **Chapter 4: Entrenamiento y resultados**

The training conditions for each model (including modifications to the original model) and the libraries used are explained, and the quantitative results obtained through the metrics are presented and compared.

- **Chapter 5: Explicabilidad**

The gradient-based *saliency maps* explainability method applied to Ribeiro's model is described. In addition, the perspective of a specialist physician analyzing the generated explanations is included, and reflections on the method and potential improvements are discussed.

- **Chapter 6: Conclusiones y trabajo futuro**

The conclusions derived from the results are integrated, assessing the extent to which the stated objectives have been met. Furthermore, the main contributions of this work are identified, and future research directions are proposed (such as the inclusion of Conv2D layers).

Finally, a bibliography section is included, containing all the sources consulted throughout the document, as well as an appendix with the code used<sup>1</sup> and another with the physician's comments transcribed.

---

<sup>1</sup>All the content of this work can be found in this GitHub repository.



# Conclusions and Future Work

## 6.6. Conclusions

Over the course of this work, the applicability of Ribeiro’s model for classifying ECG’s from the PTB-XL database has been demonstrated, achieving acceptable results when processing the original signal. However, when transformations (STFT, CWT, etc.) are introduced and the same one-dimensional convolution-based architecture is maintained, the results deteriorate significantly. This observation suggests that Ribeiro’s architecture, in its current form, is not suitable for handling images, thus requiring substantial modifications to fully exploit the time-frequency representations offered by the transforms.

Regarding explainability, gradient-based *saliency maps* have proven to be a promising resource for understanding which parts of the ECG signal are relevant to the network. Refining these explanations could turn them into a valuable learning and educational tool for both students and healthcare professionals seeking to familiarize themselves with AI.

## 6.7. Future Work

Given the model’s potential and the identified limitations, various lines of work are proposed to improve and extend the results:

1. **Modify the architecture to support Conv2D layers**

Although this adaptation would require significant effort, it would allow for better exploitation of the advantages of signal transformations.

2. **Explore additional transforms and parameters**

Following the previous line, once an architecture more suitable for images is obtained, other parameters could be tested when performing the transforms, and their impact could be studied.

3. **Specialize the model in one anomaly**

Transforming the model into a binary classifier would allow for more precise

parameter selection for the transforms, since different anomalies manifest in different frequency ranges.

#### 4. Signal delineation in explanations

As noted by the expert, through signal delineation it would be possible to reduce explanations to two heartbeats and ensure they are homogeneous in both. These improvements would make the use of these explanations viable in educational settings.

# Bibliografía

- ACHARYA, U. R., FUJITA, H., LIH, O. S., HAGIWARA, Y., TAN, J. H. y CHUA, C. Automated diagnosis of arrhythmia using different intervals of ECG segments with Convolutional Neural Network. *Information Sciences*, vol. 405, páginas 81–90, 2017.
- ALLEN, J. B. y RABINER, L. R. A unified approach to short-time fourier analysis and synthesis. *Proceedings of the IEEE*, vol. 65(11), páginas 1558–1564, 1977.
- CHOUHAN, V. y MEHTA, S. Total removal of baseline drift from ecg signal. En *2007 International Conference on Computing: Theory and Applications (ICCTA'07)*, páginas 512–515. 2007.
- GONZÁLEZ CABEZA, S. Are 12-lead ecgs necessary for detecting heart anomalies? a comparison between 12-lead and 3-lead ecg classification using deep learning, 2024. No Publicado.
- GONZÁLEZ, L. E. A., S., J. L. R. y CASTELLANOS, G. Métodos para la eliminación de interferencia ac en ecg. *Scientia et Technica*, vol. 3(29), páginas 49–53, 2005. ISSN 0122-1701. Accedido el 5 de diciembre de 2024.
- GOODFELLOW, I., BENGIO, Y. y COURVILLE, A. *Deep Learning*. MIT Press, 2016.
- GOODMAN, B. y FLAXMAN, S. European union regulations on algorithmic decision making and a “right to explanation”. *AI Magazine*, vol. 38(3), página 50–57, 2017. ISSN 2371-9621.
- GUO, C., AHMED, S. y ALOUINI, M.-S. Machine learning-based automatic cardiovascular disease diagnosis using two ecg leads. *arXiv preprint arXiv:2305.16055*, 2023.
- GUSTAFSSON, S., GEDON, D., LAMPA, E., RIBEIRO, A. H., HOLZMANN, M. J., SCHÖN, T. B. y SUNDSTRÖM, J. Development and validation of deep learning ecg-based prediction of myocardial infarction in emergency department patients. *Scientific Reports*, vol. 12, 2022.
- HANNUN, A. Y., RAJPURKAR, P., HAGHPANAH, M., TISON, G. H., BOURN, C., TURAKHIA, M. P. y NG, A. Y. Cardiologist-level arrhythmia detection

- and classification in ambulatory electrocardiograms using a deep neural network. *Nature Medicine*, vol. 25(1), páginas 65–69, 2019.
- HE, H., BAI, Y., GARCIA, E. A. y LI, S. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. En *Proceedings of the IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, páginas 1322–1328. IEEE, Hong Kong, China, 2008.
- HE, H. y GARCIA, E. A. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, vol. 21(9), páginas 1263–1284, 2009.
- KOZIARSKI, M. Radial-based undersampling for imbalanced data classification. *arXiv preprint arXiv:1906.00452*, 2019.
- LEEVY, J. L., KHOSHGOFTAAR, T. M., BAUDER, R. A. y SELIYA, N. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, vol. 5(1), página 42, 2018.
- LIBBY, P., BONOW, R., MANN, D., TOMASELLI, G., BHATT, D., SOLOMON, S. y BRAUNWALD, E. *Braunwald's Heart Disease: A Textbook of Cardiovascular Medicine*. Elsevier, 2021. ISBN 9780323824712.
- LU, L., ZHU, T., RIBEIRO, A. H. ET AL. Decoding 2.3 million ecgs: interpretable deep learning for advancing cardiovascular diagnosis and mortality risk stratification. *European Heart Journal - Digital Health*, vol. 5(3), páginas 247–259, 2024.
- MACFARLANE, P. W., VAN OOSTEROM, A., PAHLM, O., KLIGFIELD, P., JANSE, M. y CAMM, A. J. Comprehensive electrocardiology. *Springer*, vol. 1, páginas 1–1040, 2010.
- MOLNAR, C. *Interpretable Machine Learning*. Lulu.com, 2019. ISBN 9780244768522.
- OPPENHEIM, A. V. y SCHAFER, R. W. *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, USA, 3rd edición, 2009.
- RIBEIRO, A. H., PAIXAO, G. M., LIMA, E. M., HORTA RIBEIRO, M., PINTO FILHO, M. M., GOMES, P. R., OLIVEIRA, D. M., MEIRA JR, W., SCHON, T. B. y RIBEIRO, A. L. P. CODE-15 %: a large scale annotated dataset of 12-lead ECGs. <https://doi.org/10.5281/zenodo.4916206>, 2021a.
- RIBEIRO, A. H., RIBEIRO, M. H., PAIXÃO, G. M. M., OLIVEIRA, D. M., GOMES, P. R., CANAZART, J. A., FERREIRA, M. P. S., ANDERSSON, C. R., MACFARLANE, P. W., MEIRA JR., W., SCHÖN, T. B. y RIBEIRO, A. L. P. Automatic diagnosis of the 12-lead ECG using a deep neural network. *Nature Communications*, vol. 11(1), página 1760, 2020. Available at <https://doi.org/10.1038/s41467-020-15432-4>.

- RIBEIRO, A. H., RIBEIRO, M. H., PAIXÃO, G. M., OLIVEIRA, D. M., GOMES, P. R., CANAZART, J. A., FERREIRA, M. P., ANDERSSON, C. R., MACFARLANE, P. W., JR., W. M., SCHÖN, T. B. y RIBEIRO, A. L. P. CODE dataset. [https://figshare.scilifelab.se/articles/dataset/CODE\\_dataset/15169716](https://figshare.scilifelab.se/articles/dataset/CODE_dataset/15169716), 2021b.
- RUMELHART, D. E., HINTON, G. E. y WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, vol. 323(6088), páginas 533–536, 1986.
- SANZ GUERRERO, M. Peak attention u-net: attention-enhanced ecg delineation model for peak detection, 2024. No Publicado.
- WAGNER, P., STRODTHOFF, N., BOUSSELJOT, R.-D., SAMEK, W. y SCHAEFFTER, T. PTB-XL, a large publicy available electrocardiography dataset. <https://doi.org/10.1038/s41597-020-0495-6>, 2020. Último acceso: 28/09/2024.
- WAGNER, P., STRODTHOFF, N., BOUSSELJOT, R.-D., SAMEK, W. y SCHAEFFTER, T. PTB-XL, a large publicy available electrocardiography dataset. <https://doi.org/10.13026/kfzx-aw45>, 2022. (version 1.0.3). Último acceso: 28/09/2024.
- WIKIMEDIA COMMONS. Sinus rhythm labels. 2023. Accedido: 2024-12-26, licenced under CC BY-SA 3.0 URL: <https://commons.wikimedia.org/wiki/File:SinusRhythmLabels.svg>.
- WORLD HEALTH ORGANIZATION. Cardiovascular diseases (cvds). [https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)), 2021. [Online; accessed 2023-10-10].
- XIE, C., MCCULLUM, L., JOHNSON, A., POLLARD, T., GOW, B. y MOODY, B. Waveform database software package (wfdb) for python (version 4.1.0). PhysioNet, 2023.
- YANMIN SUN, WONG, A. y KAMEL, M. S. Classification of imbalanced data: a review. *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, 2011.



# Apéndice A

## Código

```
1 import json
2 from math import ceil
3 import os
4 import numpy as np
5 import h5py
6 import ecg_plot
7 import matplotlib.pyplot as plt
8
9 def plot_ecg_with_explanations(N_ECG, explanation_path):
10     # Cargar el vector de numpy
11     explicaciones = np.load(os.path.join(explanation_path, "explanation.npy")).T
12     with h5py.File("./data/test.hdf5") as f:
13         ecg = f["tracings"][N_ECG-2].T
14         #El título contiene el id del ECG y la etiqueta
15         ecg_plot.plot(ecg, sample_rate=400, title=f"ECG_ID: {explanation_path.split('/')[-1]} ({explanation_path.split('/')[-2]})", style="bw")
16     fig = plt.gcf()
17     fig.subplots_adjust(
18         left = 0.1,
19         right=0.95,
20         bottom=0.1,
21         top=0.95
22     )
23     #Quitamos las etiquetas de los ejes, que se concentran
24     #demasiado
25     ax = fig.axes[0]
26     ax.tick_params(
27         labelbottom=False,
28         labelleft=False,
29         bottom=False,
30         left=False,
31         length=0
32     )
33     #En esencia replicamos el código que hace la función plot (que
34     #calcula offsets para saber donde dibujar)
35     #para luego poner el heatmap en la misma posición
```

```

34     sample_rate = 400
35     secs = ecg.shape[1] / sample_rate
36
37     columns = 2
38     leads = ecg.shape[0] #Debería de ser doce
39     rows = int(ceil(leads / columns))
40     row_height = 6
41     lead_order = list(range(12))
42
43     for c in range(columns):
44         for i in range(rows):
45             #Eso es para cada derivación
46             idx = c * rows + i
47             if idx >= leads:
48                 break
49
50             t_lead = lead_order[idx]
51
52             # === Mismos offsets que en plot
53             x_offset = secs * c
54             y_offset = -(row_height / 2) * (i % rows)
55
56             # Preparamos la imagen que vamos a pintar po encima
57             exp_lead = explicaciones[t_lead]           # (4096,)
58             exp_lead_2d = exp_lead.reshape(1, -1)      # (1, 4096)
59
60             # === Definimos el rectángulo (extent) donde se pintará
61             # esa franja ===
62             local_x_min = x_offset
63             local_x_max = x_offset + secs
64
65             # Escogemos el ancho de la franja para que quede bien.
66             local_y_min = y_offset - 1.5
67             local_y_max = y_offset + 1.5
68
69             # Dibujamos el heatmap con imshow
70             # zorder=-1 para que quede POR DEBAJO de la onda ECG
71             # alpha=0.5 (p.ej.) para dejar ver la línea
72             heatmap = ax.imshow(
73                 exp_lead_2d,
74                 extent=(local_x_min, local_x_max, local_y_min,
75                         local_y_max),
76                 cmap='Reds',
77                 origin='lower',
78                 aspect='auto',
79                 alpha=0.5, #Para que no tape el resto de cosas
80                 zorder=-1 #Para que quede por debajo de la onda
81             )
82
83             # Añadimos una barra de color
84             cbar = fig.colorbar(heatmap, ax=ax, shrink=1)
85             cbar.set_label("Relevancia")
86             plt.savefig(os.path.join(explanation_path, "better_explanation.
87             png"))

```

Listing A.1: Código empleado para pintar las explicaciones sobre la imagen generada por *ecg\_plot*.

```

1 import json
2 import os
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import tensorflow as tf
6 import h5py
7 import pandas as pd
8 import sys
9
10 #Nombres de las etiquetas
11 label_names = ["CD", "HYP", "MI", "NORM", "STTC"]
12
13 # --- Carga de datos ---
14 # Cargamos los datos de pruebas
15 test_x = h5py.File('./data/test.hdf5', 'r')["tracings"]
16
17 # Cargamos las etiquetas desde el CSV
18 df = pd.read_csv('./data/test_db.csv')
19 test_y = df.drop(columns=['ecg_id']).to_numpy()
20
21 # --- Ejecución del modelo ---
22
23 # Obtenemos las predicciones mediante el script original de Ribeiro
24 os.system("python ./ribeiro/predict.py data/test.hdf5 final_models/
25           original_model.hdf5 --output_file ./tmp.npy --dataset_name
26           tracings")
27
28 # Las cargamos
29 predictions = np.load("./tmp.npy")
30
31 # Y las convertimos a booleanos
32 predictions = predictions > 0.5
33
34 # --- Comparación de resultados ---
35
36 # Comprobamos que la forma de las predicciones sea correcta
37 if predictions.shape[0] != test_y.shape[0]:
38     print("The number of test samples and predictions do not match"
39          )
40     sys.exit(1)
41 if predictions.shape[1] != test_y.shape[1] or predictions.shape[1]
42     != 5:
43     print("The number of classes is not 5")
44     sys.exit(1)
45
46 #Creamos un fichero out.txt donde se guardaran los resultados, si
47 # ya existe lo borramos
48 lista_de_indices = {}
49 with open("out.json", "w") as f:
50     # Para cada una de las etiquetas, generamos el vector que
51     # representa la predicción y valor real que necesita para ser

```

```

46     "adecuada" para explicarla
47     for label_idx in range(5):
48         #Desired_real es el vector que tiene 1.0 en la posicion
49         #label_idx y 0.0 en las demás
50         desired_real = np.zeros(5, dtype=np.float32)
51         desired_real[label_idx] = 1.0
52         desired_predictions = [False, False, False, False, False]
53         desired_predictions[label_idx] = True
54         desired_predictions = np.asarray(desired_predictions)
55         lista_de_indices[label_names[label_idx]] = []
56         for i in range(predictions.shape[0]):
57             # Para cada predicción, comprobamos si coincide con los
58             # arrays de valores deseados
59             if np.array_equal(test_y[i], desired_real) and np.
60                 array_equal(predictions[i], desired_predictions):
61                 # Si coincide, guardamos la línea del CSV donde est
62                 # á (que corresponde al índice en el dataset más
63                 # dos) y el id del ECG
64                 linea": i + 2, "ecg_id": int(df.iloc[i]['ecg_id'])
65             })
66         json.dump(lista_de_indices, f, indent=4)
67 os.remove("./tmp.npy")

```

Listing A.2: Código empleado para buscar los ECGs adecuados para explicaciones.

```

1  from scipy.signal import stft, cwt, ricker, morlet2
2  import numpy as np
3
4  FREQ = 400
5  #Creamos la clase para las transformaciones
6  class Transformaciones:
7      #El constructor recibe los datos, que son una cantidad de
8      #matrices de 4096x12
9      def __init__(self, ecgs):
10          self.ecgs = ecgs
11          self.n_cases = ecgs.shape[0]
12
13      def get_cwt_arrays(self, wavelet = ricker, scales = np.linspace
14          (1,128,100)):
15          #Calculamos las dimensiones de la matriz de salida usando
16          #la primera matriz de entrada
17          if wavelet == ricker:
18              f = FREQ / (2*np.pi*scales)
19          elif wavelet == morlet2:
20              f = (5 * FREQ) / (2 * np.pi * scales)
21          else:
22              raise NotImplementedError("Wavelet no reconocido")
23
24          t = np.arange(self.ecgs[0].shape[0]) / FREQ
25          return f,t
26
27      def cwt(self, wavelet = ricker, scales = np.linspace(1,128,
28          100)):
29          cwts = []

```

```

27     for i in range(self.n_cases):
28         ecg = self.ecgs[i]
29         cwt_i = np.zeros((len(scales) * ecg.shape[1], ecg.shape
30                           [0]))
31         for j in range(ecg.shape[1]): #Iteramos sobre las
32             derivaciones
33             Zxx = np.abs(cwt(ecg[:,j], wavelet=wavelet, widths=
34                           scales))
35             cwt_i[j * len(scales):(j + 1) * len(scales), :] =
36             Zxx
37             cwts.append(cwt_i)
38     return np.array(cwts)
39
40
41     def get_stft_arrays(self, nperseg=256, nooverlap=128):
42         #Calculamos las dimensiones de la matriz de salida usando
43         #la primera matriz de entrada
44         f, t, _ = stft(self.ecgs[0][:, 0], fs=FREQ, nperseg=nperseg,
45                         nooverlap=nooverlap)
46         return f, t
47
48     #Función que realiza la transformada STFT
49     def stft(self, nperseg=256, nooverlap=128):
50         #Calculamos las dimensiones de la matriz de salida usando
51         #la primera matriz de entrada
52         n_frequencies, n_times = stft(self.ecgs[0][:, 0], fs=FREQ,
53                                         nperseg=nperseg, nooverlap=nooverlap)[2].shape
54         #Creamos un array vacío para guardar los resultados
55         stfts = []
56         #Iteramos sobre los datos
57         for i in range(self.n_cases):
58             ecg = self.ecgs[i]
59             stft_i = np.zeros((n_frequencies * ecg.shape[1],
60                               n_times))
61             for j in range(ecg.shape[1]):
62                 _, _, Zxx = stft(ecg[:, j], fs=FREQ, nperseg=
63                                 nperseg, nooverlap=nooverlap)
64                 stft_i[j * n_frequencies:(j + 1) * n_frequencies,
65                         :] = np.abs(Zxx)
66             stfts.append(stft_i)
67     return np.array(stfts)

```

Listing A.3: Clase creada para hacer las transformaciones de los datos.

```

1 import numpy as np
2 import json
3 from sklearn.metrics import precision_recall_fscore_support,
4                             fbeta_score
5
6 class Metrics:
7     def __init__(self, y_true, y_pred, class_names=None):
8         """
9             Inicializa la clase Metrics con matrices de etiquetas
10            reales y predicciones.
11
12     Parameters:
13     y_true (np.ndarray): Array de valores reales (

```

```

    probabilidades entre 0 y 1).
12                               Dimensión: (n_samples, n_classes)
13 y_pred (np.ndarray): Array de predicciones (probabilidades
14                               entre 0 y 1).
15                               Dimensión: (n_samples, n_classes)
16 class_names (list, opcional): Lista con los nombres de las
17                               clases.
18                               Longitud: n_classes
19 """
20
21     self.y_true_prob = y_true
22     self.y_pred_prob = y_pred
23     self.class_names = class_names
24     self.n_classes = y_true.shape[1]
25
26
27     # Binarizar las etiquetas reales y predichas usando un
28     # umbral de 0.5
29     self.y_true = (self.y_true_prob >= 0.5).astype(int)
30     self.y_pred = (self.y_pred_prob >= 0.5).astype(int)
31
32
33     # Verificar si alguna clase no tiene representantes en
34     # y_true
35     self.classes_with_no_samples = []
36     for i in range(self.n_classes):
37         y_true_sum = np.sum(self.y_true[:, i])
38         if y_true_sum == 0:
39             if self.class_names:
40                 class_name = self.class_names[i]
41             else:
42                 class_name = f'Clase_{i}'
43             self.classes_with_no_samples.append(class_name)
44
45     if self.classes_with_no_samples:
46         print(f"Advertencia: Las siguientes clases no tienen
47               representantes en la muestra de test: {', '.join(
48               self.classes_with_no_samples)}")
49
50
51     def calculate_precision_recall_f1(self):
52         """
53             Calcula las métricas de precisión, recall y F1-score para
54             cada clase y de manera global.
55
56             Returns:
57             dict: Diccionario con las métricas calculadas.
58         """
59
60         precision_dict = {}
61         recall_dict = {}
62         f1_dict = {}
63
64         precision_list, recall_list, f1_list, _ =
65             precision_recall_fscore_support(self.y_true, self.y_pred,
66             , zero_division=np.nan)
67         for i in range(self.n_classes): #Para cada clase
68             if self.class_names:
69                 class_name = self.class_names[i]
70             else:
71                 class_name = f'Clase_{i}',
```

```

58     precision_dict[class_name] = precision_list[i]
59     recall_dict[class_name] = recall_list[i]
60     f1_dict[class_name] = f1_list[i]
61
62
63     global_precision, global_recall, global_f1, _ =
64         precision_recall_fscore_support(self.y_true, self.y_pred,
65                                         average='micro', zero_division=np.nan)
66
67     metrics = {
68         'precision': {
69             'by_class': precision_dict,
70             'global_average': global_precision
71         },
72         'recall': {
73             'by_class': recall_dict,
74             'global_average': global_recall
75         },
76         'f1_score': {
77             'by_class': f1_dict,
78             'global_average': global_f1
79         }
80     }
81
82     return metrics
83
84 def calculate_adjusted_f_score(self):
85     """
86     Calcula la métrica personalizada 'adjusted_f_score',
87     combinando F0.5 para 'NORM' y F2 para las demás clases.
88
89     Returns:
90     float or None: Valor de la métrica personalizada.
91     """
92
93     if self.class_names is None:
94         clase_especial = 3
95         print("Warning: No se han proporcionado los nombres de
96              las clases. Se asumirá que la clase NORM es la clase
97              3.")
98     else:
99         clase_especial = self.class_names.index('NORM')
100
101     metricas = []
102     for cls in range(self.n_classes):
103         if cls == clase_especial:
104             beta = 0.5
105         else:
106             beta = 2
107             f_score = fbeta_score(self.y_true[:,cls], self.y_pred
108                                   [:,cls], beta=beta)
109             metricas.append(f_score)
110
111     adjusted_f_score = np.mean(metricas)
112
113     return adjusted_f_score
114
115
116 def dump_to_json(self, path):
117     """

```

```
108     Escribe las métricas calculadas en un archivo JSON.
109
110     Parameters:
111     path (str): Ruta al archivo JSON de salida.
112     """
113     # Crear un diccionario ordenado para asegurar el orden de
114     # las métricas
115     metrics = {}
116
117     # Calcular la métrica personalizada y agregarla primero
118     adjusted_f_score = self.calculate_adjusted_f_score()
119     metrics['adjusted_f_score'] = adjusted_f_score
120
121     # Calcular las demás métricas
122     prf_metrics = self.calculate_precision_recall_f1()
123     metrics.update(prf_metrics)
124
125     # Escribir en el archivo JSON
126     with open(path, 'w') as json_file:
127         json.dump(metrics, json_file, indent=4)
```

Listing A.4: Clase creada para generar métricas y guardarlas en un json.

# Apéndice B

## Librerías utilizadas

Librería	Versión
_libgcc_mutex	0.1=main
_openmp_mutex	5.1=1_gnu
bzip2	1.0.8=h7b6447c_0
ca-certificates	2023.12.12=h06a4308_0
ld_impl_linux-64	2.38=h1181459_1
libffi	3.4.4=h6a678d5_0
libgcc-ng	11.2.0=h1234567_1
libgomp	11.2.0=h1234567_1
libstdcxx-ng	11.2.0=h1234567_1
libuuid	1.41.5=h5eee18b_0
ncurses	6.4=h6a678d5_0
openssl	3.0.13=h7f8727e_0
pip	23.3.1=py311h06a4308_0
python	3.11.7=h955ad1f_0
readline	8.2=h5eee18b_0
setuptools	68.2.2=py311h06a4308_0
sqlite	3.41.2=h5eee18b_0
tk	8.6.12=h1ccaba5_0
wheel	0.41.2=py311h06a4308_0
xz	5.4.5=h5eee18b_0
zlib	1.2.13=h5eee18b_0
absl-py	2.1.0
aiohttp	3.9.3
aiosignal	1.3.1
asttokens	2.4.1
astunparse	1.6.3
attrs	23.2.0
autograd	1.7.0
beautifulsoup4	4.12.3

Librería	Versión
bleach	6.1.0
cachetools	5.3.2
captum	0.7.0
certifi	2024.2.2
cffi	1.16.0
charset-normalizer	3.3.2
cloudpickle	3.1.0
comm	0.2.1
contourpy	1.2.0
cycler	0.12.1
deap	1.4.1
debugpy	1.8.0
decorator	5.1.1
defusedxml	0.7.1
deprecated	1.2.13
ecg-plot	0.2.8
et-xmlfile	1.1.0
executing	2.0.1
fastjsonschema	2.19.1
filelock	3.13.4
flatbuffers	23.5.26
fonttools	4.48.1
frozenlist	1.4.1
fsspec	2024.3.1
gast	0.5.4
google-auth	2.27.0
google-auth-oauthlib	1.2.0
google-pasta	0.2.0
grpcio	1.60.1
h5py	3.10.0
idna	3.6
ipykernel	6.29.2
ipython	8.21.0
jedi	0.19.1
jinja2	3.1.3
joblib	1.3.2
jsonschema	4.21.1
jsonschema-specifications	2023.12.1
jupyter-client	8.6.0
jupyter-core	5.7.1
jupyterlab-pygments	0.3.0
keras	2.15.0
kiwisolver	1.4.5
libclang	16.0.6

Librería	Versión
lightning-utilities	0.11.2
llvmlite	0.43.0
locket	1.0.0
markdown	3.3.4
markupsafe	2.1.5
matplotlib	3.8.2
matplotlib-inline	0.1.6
mistune	3.0.2
ml-dtypes	0.2.0
mpmath	1.3.0
multidict	6.0.5
nbclient	0.9.0
nbconvert	7.16.1
nbformat	5.9.2
nest-asyncio	1.6.0
networkx	3.3
numba	0.60.0
numpy	1.26.4
nvidia-cublas-cu12	12.1.3.1
nvidia-cuda-cupti-cu12	12.1.105
nvidia-cuda-nvcc-cu12	12.2.140
nvidia-cuda-nvrtc-cu12	12.1.105
nvidia-cuda-runtime-cu12	12.1.105
nvidia-cudnn-cu12	8.9.2.26
nvidia-cufft-cu12	11.0.2.54
nvidia-curand-cu12	10.3.2.106
nvidia-cusolver-cu12	11.4.5.107
nvidia-cusparse-cu12	12.1.0.106
nvidia-nccl-cu12	2.19.3
nvidia-nvjitlink-cu12	12.2.140
nvidia-nvtx-cu12	12.1.105
oauthlib	3.2.2
opencv-python	4.6.0.66
openpyxl	3.1.2
opt-einsum	3.3.0
packaging	23.2
pandas	2.2.0
pandocfilters	1.5.1
parso	0.8.3
partd	1.2.0
patsy	1.0.1
pexpect	4.9.0
pillow	10.2.0
platformdirs	4.2.0

Librería	Versión
prompt-toolkit	3.0.43
protobuf	4.23.4
psutil	5.9.8
ptyprocess	0.7.0
pure-eval	0.2.2
pyaml	24.9.0
pyarrow	15.0.2
pyasn1	0.5.1
pyasn1-modules	0.3.0
pycparser	2.21
pygments	2.17.2
pymop	0.2.4
pyparsing	3.1.1
python-dateutil	2.8.2
pytorch-lightning	2.2.1
pyts	0.13.0
pytz	2024.1
pyyaml	6.0.1
pyzmq	25.1.2
referencing	0.33.0
requests	2.31.0
requests-oauthlib	1.3.1
rpds-py	0.18.0
rsa	4.9
scikit-base	0.11.0
scikit-learn	1.3.0
scikit-optimize	0.10.2
scipy	1.14.1
seaborn	0.13.2
shap	0.46.0
six	1.16.0
sktime	0.34.1
slicer	0.0.8
soundfile	0.12.1
soupsieve	2.5
stack-data	0.6.3
statsmodels	0.14.4
stumpy	1.13.0
sympy	1.12
tensorboard	2.15.1
tensorboard-data-server	0.7.2
tensorflow	2.15.0.post1
tensorflow-estimator	2.15.0
tensorflow-io-gcs-filesystem	0.36.0

---

Librería	Versión
termcolor	2.4.0
tf-explain	0.3.1
threadpoolctl	3.2.0
tinycss2	1.2.1
toolz	1.0.0
torch	2.2.2
torchcam	0.4.0
torchmetrics	1.3.2
tornado	6.4
tqdm	4.65.2
traitlets	5.14.1
triton	2.2.0
tsfresh	0.20.3
tsinterpret	0.4.7
tslearn	0.6.3
typing-extensions	4.9.0
tzdata	2023.4
urllib3	2.2.0
wcwidth	0.2.13
webencodings	0.5.1
werkzeug	3.0.1
wfdb	4.1.2
wrapt	1.14.1
xarray	2024.3.0
xmljson	0.2.1
yarl	1.9.4



