Universidad de Cantabria

Facultad de Ciencias

DETECCIÓN DE LA ENFERMEDAD DE PARKINSON MEDIANTE ANÁLISIS DE LA VOZ CON ESPECTROGRAMAS MEL Y REDES NEURONALES CONVOLUCIONALES

(Parkinson's Disease Detection through Voice Analysis Using Mel Spectrograms and Convolutional Neural Networks)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Sergio Lanza Teja

Directora: Cristina Tîrnăucă

Codirector: Marcos Aguilella

Abril – 2025

Resumen

La enfermedad de Parkinson es un trastorno neurodegenerativo crónico que afecta al sistema nervioso central, caracterizado principalmente por la pérdida progresiva de funciones motoras. Entre sus síntomas más comunes se encuentran el temblor en reposo, la rigidez muscular, lentitud de los movimientos voluntarios y alteraciones en el habla. Estas manifestaciones vocales, que incluyen cambios en la prosodia, la intensidad y la articulación, pueden ser indicativas de la presencia temprana de la enfermedad, lo que convierte a la voz en una fuente potencialmente útil para su detección no invasiva.

El objetivo de este Trabajo de Fin de Grado es desarrollar un sistema automático capaz de detectar la enfermedad de Parkinson a partir del análisis de grabaciones de voz. Para ello, se han utilizado bases de datos con participantes sanos y pacientes diagnosticados, tanto en español como en italiano, con el fin de incorporar variabilidad fonética. Las señales de audio se han sometido a un proceso de preprocesamiento y transformación en espectrogramas de Mel, que permiten representar visualmente las características acústicas relevantes para el diagnóstico.

Para el desarrollo del sistema se ha empleado una red neuronal convolucional (CNN) diseñada específicamente para procesar imágenes de espectrogramas. El modelo ha sido entrenado y evaluado utilizando técnicas de aprendizaje profundo, alcanzando una precisión del 80 % y un AUC de 0,918. Además, se ha implementado una aplicación con interfaz gráfica que permite al usuario cargar grabaciones y obtener predicciones visuales e interactivas. El sistema demuestra un rendimiento robusto y se plantea como una herramienta prometedora para el apoyo al diagnóstico clínico.

Palabras clave: Diagnóstico médico, Aprendizaje profundo, Análisis de voz, Espectrogramas de Mel, Redes neuronales convolucionales, Detección del Parkinson.

Abstract

Parkinson's disease is a chronic neurodegenerative disorder that affects the central nervous system, primarily impairing motor functions. Common symptoms include resting tremor, muscle rigidity, slowness of voluntary movements, and speech disturbances. These vocal alterations—such as changes in prosody, intensity, and articulation—can serve as early indicators of the disease, making voice analysis a promising non-invasive diagnostic tool.

The aim of this Final Degree Project is to develop an automatic system capable of detecting Parkinson's disease through the analysis of voice recordings. To achieve this, voice datasets in Spanish and Italian were used, including recordings from both healthy individuals and patients diagnosed with Parkinson's disease, in order to ensure phonetic diversity. The audio signals underwent preprocessing and were transformed into Mel spectrograms, which visually represent the acoustic features relevant to diagnosis.

The system was built using a convolutional neural network (CNN) specifically designed to process spectrogram images. The model was trained and evaluated using deep learning techniques, achieving an accuracy of 80 % and an AUC of 0,918. Additionally, a graphical user interface was developed, allowing users to upload recordings and receive interactive visual predictions. The system demonstrates robust performance and presents itself as a promising tool to support clinical diagnosis.

Key words: Medical Diagnosis, Deep Learning, Voice Analysis, Mel Spectrograms, Convolutional Neural Networks, Parkinson's Detection.

Índice general

Índice de figuras

Capítulo 1

Introducción

La enfermedad de Parkinson es un trastorno neurodegenerativo progresivo que afecta principalmente al sistema motor, aunque sus consecuencias abarcan también funciones cognitivas, autonómicas y, especialmente, comunicativas. Este trastorno, cuya prevalencia se ha incrementado en las últimas décadas debido al envejecimiento poblacional, se manifiesta comúnmente a través de temblores, rigidez muscular y bradicinesia, y puede producir alteraciones en la voz desde fases tempranas de su evolución [2].

La disfonía, hipofonía y disprosodia son frecuentes en pacientes con Parkinson incluso en fases iniciales, y afectan considerablemente a su calidad de vida [7]. En este contexto, surge la idea de que la voz —una señal rica en características fisiológicas— podría ser utilizada como herramienta diagnóstica no invasiva y accesible. No obstante, esta tarea no es sencilla: la interpretación de patrones vocales asociados a enfermedades requiere de técnicas avanzadas de procesamiento de señales y aprendizaje automático.

En los últimos años, diversos estudios han explorado la posibilidad de utilizar grabaciones de voz como método auxiliar para el diagnóstico del Parkinson. Por ejemplo, Little et al. [5] desarrollaron un sistema basado en parámetros acústicos para discriminar entre pacientes y controles con alta precisión. Posteriormente, se han empleado técnicas más sofisticadas como los espectrogramas de Mel combinados con redes neuronales profundas, logrando mejoras notables en la clasificación [1]. Estos enfoques destacan la viabilidad de utilizar representaciones visuales del sonido como entrada para modelos de aprendizaje profundo, sentando las bases del presente trabajo.

Diversos estudios comparativos han evaluado el rendimiento de distintos enfoques de clasificación para la detección de la enfermedad de Parkinson a partir de voz, incluyendo modelos tradicionales de aprendizaje automático (como SVM o KNN), redes recurrentes como Bi-LSTM y arquitecturas convolucionales. Los resultados muestran que las CNN, especialmente cuando se combinan con LSTM, ofrecen una mayor precisión y una mejor capacidad para capturar patrones relevantes en representaciones espectrales del habla, superando sistemáticamente a otros modelos [3].

Este Trabajo de Fin de Grado (TFG) nace de la hipótesis de que es posible detectar patrones indicativos de la enfermedad de Parkinson en grabaciones de voz mediante el uso de *Deep Learning* (DL). Para ello, se parte de señales de audio que se transforman en representaciones visuales del contenido frecuencial (espectrogramas de Mel), las cuales son procesadas por redes neuronales convolucionales (CNN). Estas arquitecturas, ampliamente utilizadas en el reconocimiento de imágenes, permiten capturar patrones espaciales complejos de manera jerárquica y eficiente.

Como ejemplo ilustrativo, en el trabajo de Toro et al. [10] se presenta una comparación visual entre el espectrograma de una persona sana y otro correspondiente a una persona con Parkinson. En la Figura 1.1 se reproducen estas imágenes, en las que se aprecian diferencias notables en la distribución de la energía frecuencial, las pausas y la articulación de los sonidos. Estas diferencias justifican el uso de este tipo de representación como entrada al modelo.

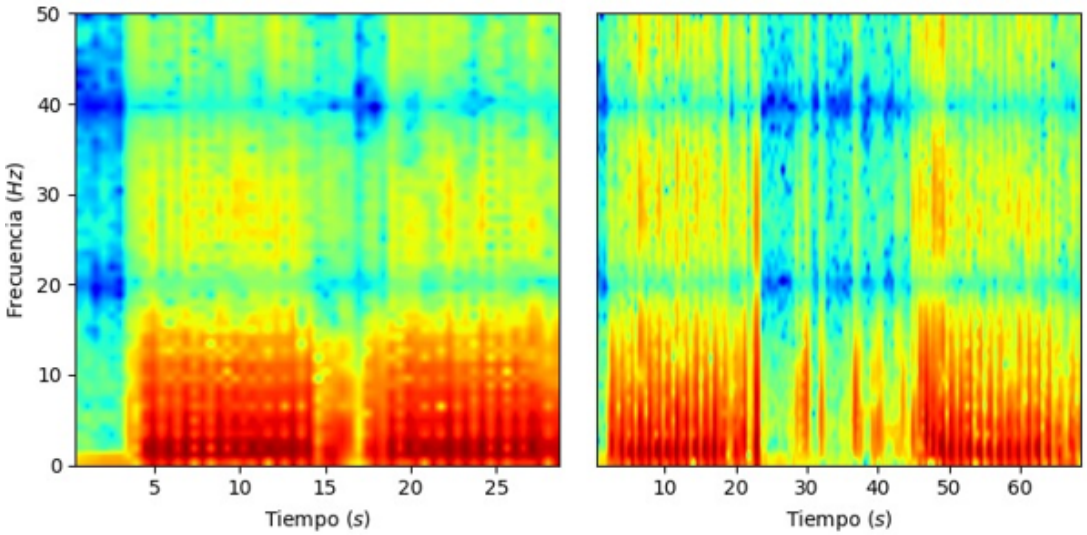

Figura 1.1: Comparación entre el espectrograma de una persona sana (izquierda) y una persona con Parkinson (derecha). Imagen adaptada de [10].

La elección de este enfoque se basa en que los espectrogramas, además de conservar la riqueza acústica del audio, permiten aplicar técnicas avanzadas de visión por computador para detectar estructuras relevantes que podrían pasar desapercibidas en un análisis convencional. De este modo, se aprovecha la potencia de DL para traducir señales acústicas en predicciones clínicas.

Más allá del desarrollo del modelo, este trabajo también pone énfasis en la creación de una herramienta accesible. Para ello, se ha desarrollado una aplicación con interfaz gráfica que permite a usuarios no especializados cargar grabaciones de voz, recibir un diagnóstico estimado y visualizar tanto los resultados como los espectrogramas procesados. Esta herramienta, además, ofrece funcionalidades como el análisis por lotes (carpetas completas), representaciones gráficas y segmentación de audios, convirtiéndose en un sistema práctico y potencialmente útil para tareas de apoyo al diagnóstico.

El objetivo final de este proyecto no es únicamente lograr una predicción precisa, sino también demostrar que es posible construir una herramienta funcional, replicable y extensible que combine aprendizaje profundo, procesamiento de voz y diseño de interfaces orientadas a la experiencia del usuario.

A lo largo de esta memoria se abordan de forma estructurada todos los aspectos que conforman el desarrollo del sistema propuesto. En el **Capítulo 2** se describen en detalle los datos utilizados, incluyendo la procedencia, las características de las grabaciones y el preprocesamiento aplicado para su estandarización. El **Capítulo 3** presenta la metodología de trabajo seguida, detallando las fases del proyecto y la planificación temporal. En el **Capítulo 4** se explican los fundamentos teóricos de las redes neuronales convolucionales, así como la arquitectura final diseñada para el modelo. A continuación, el **Capítulo 5** expone el desarrollo e integración del sistema, incluyendo la implementación técnica, el entrenamiento del modelo, la evaluación de resultados y los criterios de validación. Posteriormente, en el **Capítulo 6** se analiza la construcción de la aplicación con interfaz gráfica y sus funcionalidades orientadas a facilitar el uso por parte de usuarios no especializados. Finalmente, en el **Capítulo 7** se recogen las conclusiones obtenidas y se plantean posibles líneas de mejora y extensión del trabajo.

Para facilitar la comprensión del trabajo realizado, se ha publicado el código fuente completo de este proyecto en un repositorio de GitHub accesible en: https://github.com/slt866/TFG. El repositorio está organizado en tres carpetas principales: `Modelo`, que contiene los scripts de preprocesamiento, definición de la arquitectura y entrenamiento del modelo; `Entrenado`, donde se encuentra el archivo del modelo final entrenado en formato `.pth`; e `Interfaz`, que incluye la aplicación gráfica desarrollada en Python con Tkinter, así como scripts de predicción y pruebas.

Capítulo 2

Origen, estructura y caracteristicas de los datos utilizados

El desarrollo de este TFG se basa en el uso de señales de audio procedentes de participantes clasificados en dos categorías: HC *(Healthy Control)*, que corresponde a personas sin enfermedad, y PD *(Parkinson's Disease)*, que hace referencia a personas diagnosticadas con la enfermedad de Parkinson. El objetivo es utilizar estos datos para entrenar un modelo de aprendizaje profundo que permita detectar patrones asociados a la patología a partir de la voz.

Las grabaciones de voz utilizadas en este estudio se obtuvieron a partir de dos fuentes de datos independientes, diferenciadas por la nacionalidad de los participantes: españoles e italianos. Esta diversidad contribuye a la robustez del modelo al introducir variabilidad lingüística y fonética. Todas las grabaciones fueron realizadas en condiciones controladas para asegurar una calidad acústica adecuada y minimizar la interferencia de ruidos externos.

Para la obtención de los datos con participantes españoles, se utilizó un micrófono alimentado por USB (Logitech, modelo 980186-0403), colocado sobre una superficie estable a una distancia aproximada de 10 centímetros de la boca del participante. Las sesiones se llevaron a cabo en una sala silenciosa y se utilizó el software Audacity (versión 2.0.3) para la captura de las señales.

- **Conjunto de datos con participantes españoles:** Este conjunto está compuesto por grabaciones etiquetadas con nombres descriptivos que indican la tarea vocal realizada. Por ejemplo, el archivo `ka.wav` contiene la fonación repetitiva de la sílaba "ka". Estas actividades incluyen tanto emisiones vocálicas simples como tareas más complejas de habla espontánea o lectura estructurada. En el Cuadro 2.1 se muestra un resumen estadístico de duración de los ejercicios vocales para este conjunto. A continuación, se detallan las pruebas:

- **`/aueoi/`**: fonación de vocales concatenadas.
- **`/ka/`**, **`/pa/`**, **`/uy/`**, **`/ta/`**: fonación repetitiva de sílabas simples.
- **`/pataka/`**, **`/patachaka/`**: repetición de secuencias silábicas complejas.
- **`vocal`**: fonación sostenida de una vocal.
- **`fluencia categorial`**: el participante nombra en un minuto el mayor número de palabras pertenecientes a la categoría "animal".
- **`robo galletas`**: el participante debe describir una imagen entre veinte y sesenta segundos.
- **`habla libre`**: el participante habla de forma libre durante un intervalo de tiempo.
- **`lectura texto`**: el participante debe leer en voz alta un fragmento del libro "El Principito".

Ejercicio vocal	Recuento	Media (s)	Std (s)	Min (s)	Max (s)
aueoi	56	12,97	3,40	5,70	23,95
fluencia categorial	58	59,83	1,41	49,40	60,82
habla libre	57	37,49	18,56	11,28	89,28
ka	60	10,34	0,47	8,88	11,54
lectura texto	55	184,52	67,23	120,47	487,06
pa	58	10,28	0,35	9,13	11,28
patachaka	59	9,22	2,37	4,98	18,13
pataka	59	7,52	2,11	4,45	15,96
robo galletas	58	30,76	14,17	9,75	60,13
ta	58	10,27	0,36	8,58	11,34
uy	59	7,77	1,47	1,37	10,51
vocal	58	10,29	0,93	6,12	11,62

Cuadro 2.1: Resumen estadístico de duración de los ejercicios vocales (en segundos) para el conjuntos de datos con participantes españoles

- **Conjunto de datos con participantes italianos:** Las grabaciones del conjunto de datos italiano presentan una estructura de nombres codificada. Cada archivo comienza con un prefijo que representa la prueba realizada, seguido por una secuencia alfanumérica correspondiente a la identificación del sujeto, la fecha y otras posibles variables del estudio. A continuación, se describen los distintos prefijos utilizados, junto con la actividad vocal asociada y la duración de los audios correspondientes (Cuadro 2.2).

- **B1** y **B2**: primera y segunda lectura del texto fonémicamente equilibrado *Il ramarro della zia.*
- **D1** y **D2**: ejecución repetitiva de las sílabas *pa* y *ta*, respectivamente.
- **FB1**: lectura de frases fonémicamente equilibradas.
- **PR1**: lectura de palabras fonémicamente equilibradas.
- **VA1**, **VE1**, **VI1**, **VO1**, **VU1**: fonación sostenida de las vocales *a*, *e*, *i*, *o* y *u*, respectivamente.
- **VA2**, **VE2**, **VI2**, **VO2**, **VU2**: segunda fonación de cada vocal anterior.

Ejercicio vocal	Recuento	Media (s)	Std (s)	Min (s)	Max (s)
B1	61	76,64	49,43	38,3	250,31
B2	57	60,98	28,92	23,18	191,06
D1	46	6,14	1,07	0,66	7,35
D2	46	6,02	1,03	0,61	7,4
FB1	44	34,23	13,58	12,02	75,44
PR1	61	62,1	40,68	34,66	236,09
VA1	46	13,8	5,96	4,38	27,6
VA2	45	7,58	4,27	3,79	22,0
VE1	46	14,24	6,98	4,38	32,25
VE2	45	7,1	3,57	3,05	19,75
VI1	46	14,71	6,28	4,0	29,13
VI2	45	7,91	6,44	5,14	35,22
VO1	46	14,48	6,22	5,64	31,6
VO2	45	7,45	4,26	3,6	23,5
VU1	46	14,47	6,13	3,38	29,03
VU2	45	7,48	4,16	3,56	22,7

Cuadro 2.2: Resumen estadístico de duración de los ejercicios vocales (en segundos) para el conjuntos de datos con participantes italianos.

2.1. Representación de las señales de audio

Para poder aplicar técnicas de aprendizaje automático, y en particular redes neuronales convolucionales, es necesario transformar las señales de audio en una representación visual que pueda ser interpretada por dichos modelos. Existen diversas maneras de representar gráficamente una señal de audio. Una de las más sencillas es la forma de onda, que representa la amplitud de la señal a lo largo del tiempo. Esta representación bidimensional utiliza el eje horizontal para el tiempo y el eje vertical para la amplitud.

La Figura 2.1 ilustra una forma de onda generada a partir de una grabación del conjunto de datos utilizado. Esta visualización permite observar la envolvente temporal de la señal, aunque no proporciona información acerca del contenido frecuencial, lo cual limita su utilidad en tareas de análisis acústico más complejas.

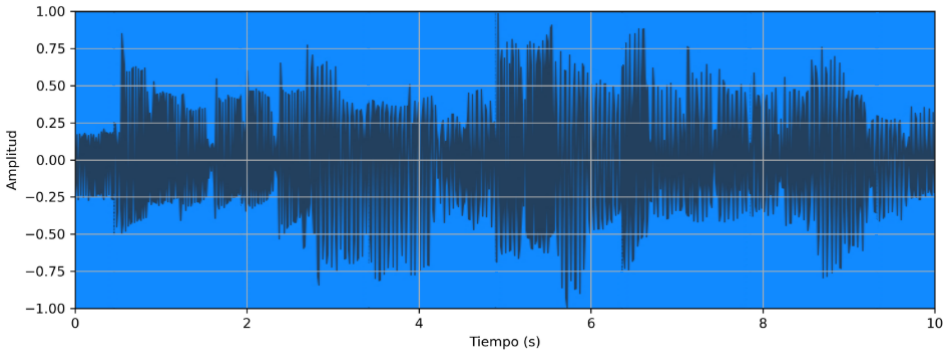

Figura 2.1: Representación temporal de una señal de voz humana. En el eje X se muestra el tiempo (en segundos) y en el eje Y la amplitud normalizada. Imagen de creación propia a partir de una grabación del dataset, generada con Python y Matplotlib.

Una alternativa mucho más informativa que la forma de onda es el uso de espectrogramas, concretamente espectrogramas de Mel, los cuales permiten una visualización tridimensional del sonido. En esta representación, el eje horizontal (X) indica el tiempo, el eje vertical (Y) representa las frecuencias, mientras que la intensidad de la señal se codifica mediante un mapa de colores. Como se puede observar en la Figura 2.2, los sonidos de mayor energía se muestran con colores más intensos, mientras que las áreas más silenciosas aparecen con tonalidades más suaves.

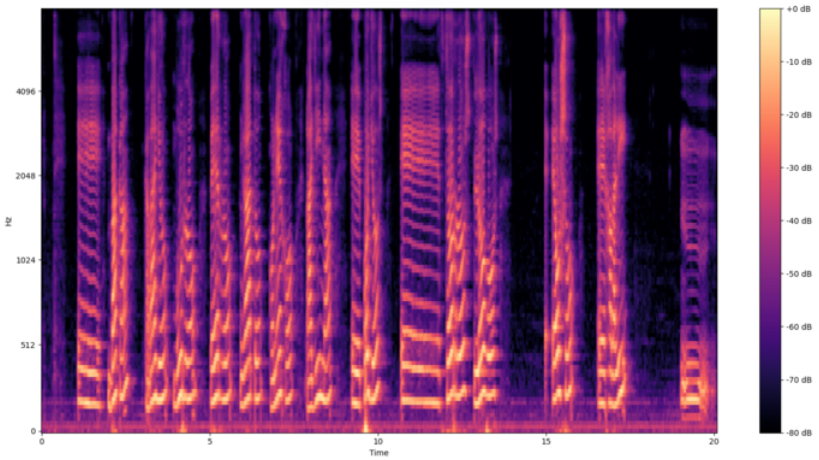

Figura 2.2: Espectrograma de Mel correspondiente a una señal de voz. En el eje X se representa el tiempo, en el eje Y la frecuencia (Hz); la intensidad se codifica mediante un mapa de colores. Imagen de creación propia a partir de una grabación del dataset, generada con la librería `librosa` (versión 0.10.2).

A diferencia de la forma de onda, esta estructura permite analizar cómo varía el contenido frecuencial a lo largo del tiempo, lo cual es un aspecto fundamental en el análisis de señales de voz. Gracias a esta capacidad para capturar simultáneamente tanto patrones temporales como espectrales, los espectrogramas se convierten en una herramienta esencial para tareas de clasificación acústica, como es el caso de este trabajo, que se centra en la detección de Parkinson a partir de señales de voz.

Los espectrogramas de Mel constituyen una variante especializada de los espectrogramas tradicionales. La diferencia principal radica en el uso de la escala de Mel para la representación de frecuencias. Esta escala se basa en la percepción auditiva humana, enfatizando las frecuencias bajas, a las que el oído es más sensible, y comprimiendo las altas.

Gracias a esta propiedad, los espectrogramas de Mel resultan especialmente eficaces para tareas de reconocimiento de voz y análisis de patrones acústicos, ya que ofrecen una representación más alineada con la forma en la que las personas perciben el sonido. Por este motivo, se han convertido en una herramienta ampliamente utilizada en el entrenamiento de modelos de aprendizaje profundo orientados al procesamiento del habla.

En este trabajo, los espectrogramas de Mel se generan mediante la biblioteca `librosa`, que permite obtener esta representación a partir de señales de audio preprocesadas. Sin embargo, antes de poder generar estos espectrogramas, es necesario aplicar un conjunto de etapas de preprocesamiento a las señales de audio originales.

2.2. Etapas de preprocesamiento

La conversión de los audios a espectrogramas de Mel requiere una serie de pasos previos orientados a estandarizar y optimizar las señales antes de su representación visual. Estas etapas aseguran la coherencia de los datos y mejoran la calidad de las entradas utilizadas durante el entrenamiento del modelo.

2.2.1. Resampleado

El resampleado es el proceso de convertir una señal de audio de una frecuencia de muestreo a otra. La **frecuencia de muestreo**, medida en hercios (Hz), indica cuántas muestras del audio se toman por segundo durante la digitalización de la señal. Por ejemplo, una frecuencia de 16 kHz (16 000 Hz) significa que se toman 16 000 muestras por segundo. Cuanto mayor es la frecuencia de muestreo, mayor es la precisión con la que se captura el sonido, aunque también se incrementa el tamaño del archivo y el coste computacional.

Este paso es fundamental porque las grabaciones de voz pueden haberse capturado con diferentes dispositivos y configuraciones técnicas, generando señales con distintas frecuencias de muestreo. Si estas diferencias no se normalizan, el modelo de aprendizaje profundo podría interpretar variaciones técnicas como diferencias relevantes, afectando negativa-

mente su rendimiento. Al convertir todas las señales a una misma frecuencia de muestreo, se garantiza que el modelo procese los datos de forma coherente y homogénea.

En este trabajo, todas las señales de audio se resamplearon a una frecuencia común de **16 kHz**, un valor ampliamente adoptado en tareas de procesamiento del habla por ofrecer una buena relación entre calidad perceptual y eficiencia computacional [4].

2.2.2. Eliminación de silencios

Posteriormente, se aplica un algoritmo de eliminación de silencios con el objetivo de reducir la presencia de información irrelevante en los audios. Esta etapa es fundamental para obtener espectrogramas más representativos y compactos, eliminando segmentos sin actividad vocal que podrían introducir ruido en el proceso de aprendizaje. Para ello, se emplea la función `librosa.effects.trim()`, que descarta regiones donde el volumen se sitúa por debajo de un umbral determinado, definido por el parámetro `top_db`.

En este trabajo, se ha utilizado un valor de `top_db = 20`, seleccionado de forma empírica tras realizar pruebas con distintos valores. Se evaluaron configuraciones con umbrales más altos (por ejemplo, 40 y 60 dB), que eliminaban silencios de forma más agresiva pero también suprimían partes útiles del habla, especialmente al final de las frases. Por el contrario, valores más bajos conservaban demasiado ruido de fondo. Finalmente, se optó por 20 dB por ofrecer un equilibrio adecuado entre la eliminación de silencios prolongados y la conservación de información vocal relevante.

Un umbral mayor podría suprimir partes importantes al final de las frases, que a menudo contienen información útil para el diagnóstico. Por tanto, esta elección contribuye a mejorar la calidad de las señales de entrada y, en consecuencia, el rendimiento del modelo.

2.2.3. Segmentacion de archivos

Una vez procesadas las señales, los archivos que superan los 10 segundos de duración se dividen en segmentos más cortos de longitud uniforme. Esta segmentación persigue dos objetivos principales: por un lado, garantizar que todas las muestras tengan una duración estándar al ser introducidas en la red neuronal, y por otro, aumentar el número total de ejemplos disponibles para el entrenamiento.

Por ejemplo, una grabación de 60 segundos puede dividirse en seis fragmentos de 10 segundos, lo que incrementa la cantidad de datos sin necesidad de recopilar nuevas grabaciones. Este aumento del conjunto de entrenamiento contribuye directamente a mejorar la capacidad de generalización del modelo.

2.2.4. Conversión a espectrograma de Mel

Completadas las etapas anteriores, cada segmento de audio es transformado en un espectrograma de Mel mediante la función `librosa.feature.melspectrogram()`. Este proceso convierte la señal acústica en una matriz de intensidades distribuidas por tiempo y frecuencia, que posteriormente se visualiza como una imagen en escala de colores.

Finalmente, los espectrogramas generados se almacenan como imágenes en formato `.png`, lo que permite su utilización directa como entradas en el modelo de clasificación basado en redes convolucionales.

Capítulo 3

Estrategia de trabajo y planificación

El desarrollo de este TFG se ha llevado a cabo siguiendo un enfoque iterativo y progresivo, estructurado en distintas fases a lo largo del curso académico. A continuación, se describe de forma general el proceso metodológico seguido para la implementación del sistema propuesto.

Durante los meses de octubre y noviembre se realizó una primera fase de estudio, centrada en la exploración y análisis de los conjuntos de datos de voz disponibles, así como en la revisión de técnicas de aprendizaje automático aplicadas a señales acústicas. En esta etapa se evaluaron distintas aproximaciones de clasificación de voz y se definió el enfoque final del proyecto: una red neuronal convolucional aplicada sobre espectrogramas de Mel.

En diciembre se iniciaron las primeras pruebas experimentales con las grabaciones de voz, lo que permitió detectar limitaciones en los datos, como la presencia de silencios prolongados o grabaciones de duración variable. Durante esta fase también se observaron interferencias no deseadas, como indicaciones verbales por parte del personal médico, que se filtraban en el audio y afectaban la calidad de la señal. Asimismo, se detectaron casos puntuales, como el de un paciente con demencia severa, cuyas grabaciones eran escasas, de corta duración y contenían un nivel elevado de silencio en comparación con otros participantes. Estos hallazgos motivaron la incorporación de un preprocesamiento específico para limpiar y segmentar los audios de manera consistente, incluyendo la eliminación de silencios con el objetivo de mejorar la calidad de la señal de entrada al modelo.

A partir de finales de diciembre y durante los meses de enero y febrero, se procedió al diseño e implementación del modelo. Inicialmente se desarrolló una versión preliminar en `Keras`, que ofreció resultados insatisfactorios. Posteriormente, se implementó una nueva versión utilizando `PyTorch`, que ofreció un mayor control y flexibilidad. Este modelo ha sido mejorado iterativamente desde su primera versión, incorporando mecanismos de regularización, normalización y mejoras en el preprocesamiento de datos.

En el mes de marzo se abordó la creación de la interfaz gráfica de usuario (GUI), integrando en ella el modelo previamente entrenado. Esta aplicación permite a los usuarios cargar audios individuales o carpetas completas y visualizar las predicciones junto a los espectrogramas generados.

Durante el mes de abril se llevó a cabo una fase de verificación y pruebas funcionales de la aplicación. En esta etapa se implementaron casos de prueba automatizados para asegurar el correcto funcionamiento de la interfaz gráfica y la fiabilidad del sistema en distintos escenarios de uso.

Finalmente, durante el mes de mayo se llevaron a cabo las últimas mejoras tanto en el modelo como en la interfaz gráfica, enfocándose en aspectos de optimización, usabilidad y presentación de resultados. Esta fase permitió consolidar una versión funcional y estable de la aplicación, completando así el ciclo de desarrollo previo a la elaboración del documento de memoria y su validación final.

Para representar de manera visual la planificación temporal del proyecto, se ha elaborado un **diagrama de Gantt** (Figura 3.1) que resume las distintas fases desarrolladas a lo largo del trabajo:

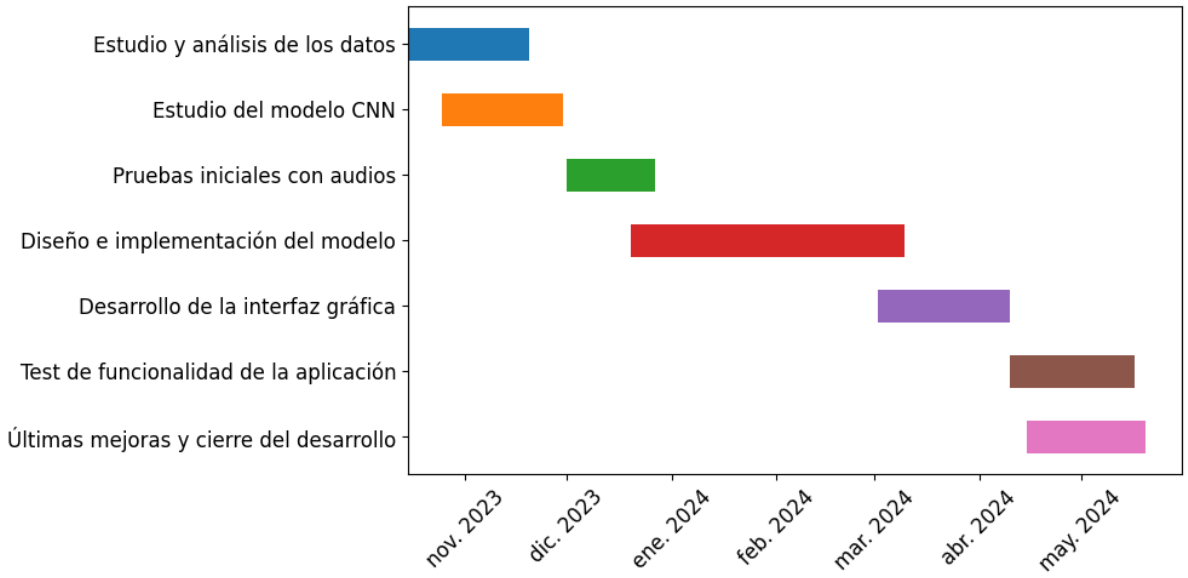

Figura 3.1: Planificación temporal del proyecto

Capítulo 4

Arquitectura y fundamentos del modelo convolucional

Este apartado describe en profundidad el modelo de red neuronal convolucional empleado para la clasificación de pacientes con enfermedad de Parkinson a partir de espectrogramas de voz. Se detallan tanto la arquitectura como los fundamentos teóricos sobre los que se sustenta, así como las decisiones de diseño adoptadas para su desarrollo.

4.1. Introducción al modelo

Antes de abordar la arquitectura específica utilizada, conviene introducir brevemente el concepto de red neuronal artificial. Estas redes son sistemas de procesamiento computacional inspirados en el funcionamiento de los sistemas nerviosos biológicos. Están formadas por múltiples nodos, también llamados neuronas artificiales, que se organizan en capas y se conectan entre sí mediante pesos sinápticos. Cada nodo recibe una o varias señales de entrada, las procesa mediante una función de activación, y transmite un resultado a los nodos de la capa siguiente. A través de un proceso iterativo de entrenamiento, basado habitualmente en retropropagación del error y optimización mediante descenso del gradiente, la red ajusta sus pesos para minimizar la diferencia entre la salida generada y la salida deseada. Este mecanismo le permite aprender de los datos y generalizar el conocimiento adquirido a nuevas muestras de entrada.

La Figura 4.1 muestra el esquema de una red neuronal artificial simple con una sola capa oculta, que sirve como base conceptual para entender estructuras más complejas como las redes convolucionales [1]. En este tipo de arquitecturas, cada neurona de una capa está conectada con todas las neuronas de la capa siguiente, permitiendo una propagación completa de la información.

[1] Imagen tomada de European Valley: https://europeanvalley.es/noticias/crear-red-neuronal-desde-las-matematicas/

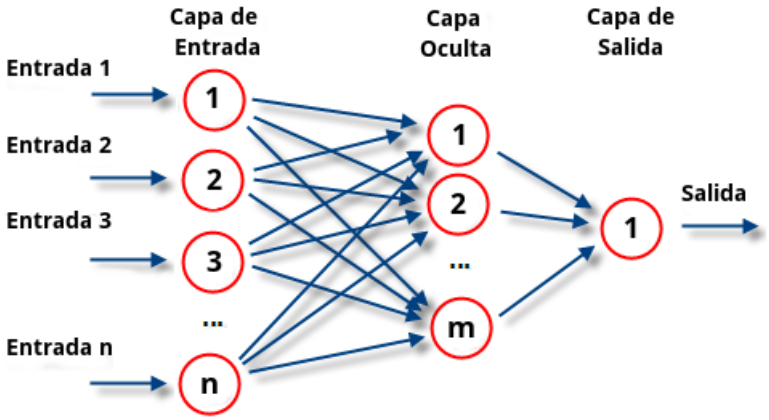

Figura 4.1: Red neuronal artificial con una capa oculta.

Dentro de las redes neuronales artificiales, las CNN destacan por su eficacia en el procesamiento de datos visuales, como imágenes. En este trabajo se ha optado por una CNN debido a que los datos de entrada —espectrogramas de voz— se pueden interpretar como imágenes. Las CNN permiten detectar patrones visuales específicos y aprender representaciones jerárquicas de las características de entrada. Estas arquitecturas han sido ampliamente estudiadas para comprender mejor su funcionamiento interno [11].

Una de las operaciones fundamentales en las CNN es la **convolución**, la cual constituye el núcleo del proceso de extracción de características. Esta operación consiste en aplicar un filtro (también denominado *kernel*) sobre pequeñas regiones locales de la imagen de entrada, desplazándose a lo largo de la misma. En cada posición, se realiza una multiplicación elemento a elemento entre el filtro y la región correspondiente de la entrada, seguida de una suma de los productos obtenidos. El resultado es un único valor que se asigna a la posición correspondiente en una nueva matriz denominada **mapa de características** o *feature map*.

Desde un punto de vista matemático, la convolución bidimensional discreta entre una imagen de entrada I y un filtro K de tamaño $l \times l$ se puede expresar como se muestra en la fórmula 4.1.

$$S(i,j) = (I * K)(i,j) = \sum_{m=0}^{l-1} \sum_{n=0}^{l-1} I(i+m, j+n) \cdot K(m,n) \tag{4.1}$$

donde $S(i,j)$ es el valor del mapa de salida en la posición (i,j), $I(i+m, j+n)$ representa el valor de la imagen en la posición desplazada, y $K(m,n)$ corresponde al valor del filtro en la posición (m,n). En cada posición, el filtro se "superpone" a una pequeña región del mapa de entrada, y se calcula la suma ponderada de los valores correspondientes. Este proceso se repite mientras el filtro se desliza sobre toda la imagen con un cierto paso (o *stride*).

Cada filtro en una CNN tiene como objetivo detectar una característica particular en los datos de entrada. Por ejemplo, algunos filtros pueden aprender a identificar bordes horizontales, otros detectan texturas, esquinas o transiciones bruscas en la intensidad. En capas más profundas, los filtros se vuelven progresivamente más complejos, capturando combinaciones de patrones más abstractos. De esta forma, la red va construyendo una representación jerárquica de la información, desde características de bajo nivel hasta conceptos de alto nivel relevantes para la tarea de clasificación.

Una característica clave de la convolución es que los filtros son compartidos a lo largo de toda la imagen, lo que implica que un mismo filtro se aplica en múltiples posiciones. Esta propiedad, conocida como **invarianza traslacional**, permite a la red reconocer un mismo patrón independientemente de su ubicación dentro de la imagen. Además, al usar filtros pequeños y compartidos, se reduce significativamente el número de parámetros del modelo en comparación con una red totalmente conectada, lo que hace a las CNN más eficientes y menos propensas al sobreajuste.

Por ejemplo, en el contexto de este trabajo, los espectrogramas representan la evolución temporal y frecuencial de una señal de voz. Aplicar convoluciones sobre estas representaciones permite detectar estructuras como zonas de energía concentrada, armónicos, o caídas bruscas de frecuencia, las cuales pueden estar relacionadas con alteraciones en la producción del habla propias de pacientes con Parkinson. De este modo, la operación de convolución no solo reduce la dimensionalidad de forma eficiente, sino que también captura información significativa para el diagnóstico basado en voz.

La Figura 4.2 ilustra gráficamente el mecanismo de convolución aplicado a una imagen, donde cada posición del filtro produce un valor de salida que forma parte del nuevo mapa de características. [2]

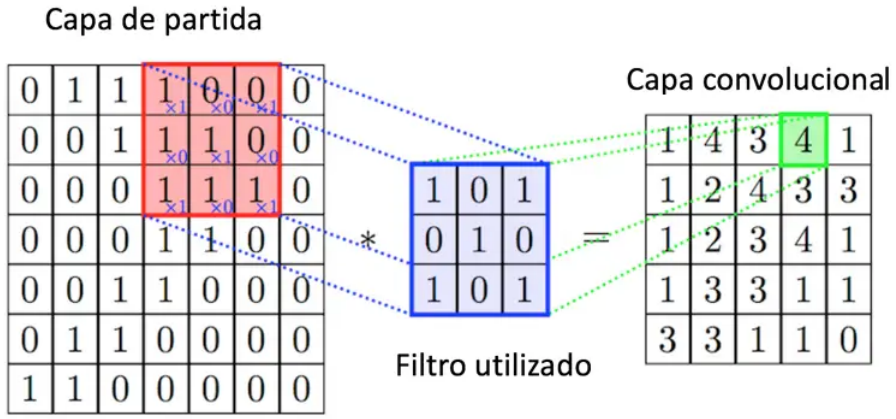

Figura 4.2: Ejemplo de funcionamiento de una convolución

Otra operación clave en las redes convolucionales es el **submuestreo**, también conocido como *pooling*. Su principal objetivo es reducir progresivamente las dimensiones espaciales de los mapas de características generados por las capas convolucionales. Esta reducción

[2]Imagen tomada de Diego Calvo: https://www.diegocalvo.es/red-neuronal-convolucional/

no solo disminuye el número de parámetros y el coste computacional del modelo, sino que también contribuye a controlar el sobreajuste y a reforzar la invariancia frente a pequeñas transformaciones o desplazamientos en la entrada.

Existen principalmente dos tipos de *pooling*: la agrupación máxima (*max-pooling*) y la agrupación media (*average pooling*). Ambas operan dividiendo el mapa de características en regiones no superpuestas —generalmente de tamaño 2×2 o 3×3— y aplicando una función de agregación sobre cada región para obtener un único valor representativo que se almacena en una nueva matriz de menor tamaño.

La agrupación máxima selecciona el valor más alto dentro de cada región, lo cual tiene el efecto de conservar las activaciones más fuertes detectadas por los filtros convolucionales. Este método resulta particularmente útil cuando se desea preservar la presencia de una característica sin importar su ubicación exacta. Su uso está ampliamente extendido debido a que ayuda a resaltar los rasgos más significativos del mapa de activación, aportando mayor robustez ante desplazamientos o distorsiones menores en los datos de entrada.

Por otro lado, la agrupación media calcula el valor promedio de cada región, proporcionando una representación más suave y general del mapa de activación. Aunque es menos sensible a picos locales, puede ser útil en tareas donde se desea una agregación más global de la información. No obstante, suele perder eficacia en contextos donde las características discriminatorihola Sergio, la sección 6.6.2as se manifiestan como valores máximos locales.

En este trabajo se ha optado por emplear la agrupación máxima, ya que se adapta mejor a la naturaleza del problema tratado. Dado que los espectrogramas de voz presentan patrones distintivos en determinadas regiones del tiempo-frecuencia, es importante conservar aquellas zonas con mayor intensidad, que suelen reflejar eventos acústicos relevantes. La agrupación máxima, ilustrada en la Figura 4.3, permite así mantener esta información clave mientras se reduce la complejidad de los datos, facilitando que las capas posteriores puedan extraer mejores representaciones[3].

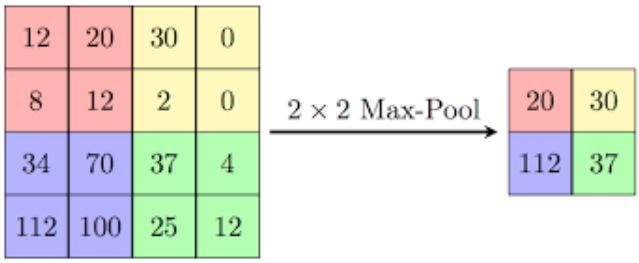

Figura 4.3: Ejemplo de funcionamiento de *max-pooling*.

Una vez introducidos los conceptos fundamentales de las redes neuronales, la operación de convolución y las técnicas de submuestreo, es posible describir cómo estos mecanismos se aplican en el procesamiento de los datos. En primer lugar, la imagen de entrada se somete a una serie de operaciones que incluyen la convolución y el *max-pooling*. Mediante la

[3]Imagen tomada de Papers with Code: https://paperswithcode.com/method/max-pooling

primera, se identifican patrones locales relevantes presentes en la imagen; posteriormente, el *max-pooling* permite reducir la dimensión del mapa de características manteniendo las señales más representativas. Este proceso de extracción y reducción de información facilita que la red aprenda de manera eficiente las particularidades de los espectrogramas utilizados, justificando así el uso de redes convolucionales en la clasificación de señales de voz asociadas a la enfermedad de Parkinson.

Además de estos componentes, se emplea una técnica de regularización denominada *dropout* ampliamente utilizada en redes neuronales profundas, introducida por Srivastava et al. [9], cuyo objetivo principal es prevenir el sobreajuste. Durante el entrenamiento, esta técnica consiste en desactivar aleatoriamente un porcentaje de neuronas (en este caso, el 45 %) en cada iteración. Al evitar que ciertas neuronas dependan excesivamente de otras, el modelo aprende representaciones más robustas y distribuidas. Esto obliga a la red a no apoyarse únicamente en ciertas vías internas, fomentando una mayor redundancia y diversidad en la activación de las neuronas. Como resultado, el modelo mejora su capacidad de generalización y rendimiento cuando se enfrenta a datos no vistos.

4.2. Estrategias de optimización y regularización en redes convolucionales

Además del diseño estructural de la arquitectura convolucional, el rendimiento de una red neuronal se ve influido de forma significativa por las técnicas utilizadas durante su entrenamiento. Estas estrategias complementarias tienen como objetivo mejorar la capacidad de generalización del modelo, estabilizar la convergencia y reducir el sobreajuste, aspectos especialmente relevantes cuando se trabaja con conjuntos de datos de tamaño limitado, como es el caso de este trabajo.

Uno de los mecanismos clave aplicados en este proyecto es la normalización por lotes (*batch normalization*), que se implementa tras cada capa convolucional. Esta técnica normaliza las activaciones de cada mini-lote durante el entrenamiento para que presenten media cero y varianza unitaria, lo que acelera la convergencia, reduce la sensibilidad a la inicialización de los pesos y permite utilizar tasas de aprendizaje más elevadas sin comprometer la estabilidad del modelo.

Otro componente fundamental es la regularización por decaimiento de pesos (*weight decay*), implementada mediante una penalización L2 sobre los pesos del modelo. Este mecanismo limita el crecimiento excesivo de los parámetros durante el aprendizaje, favoreciendo modelos más simples y con menor tendencia al sobreajuste.

Para llevar a cabo el entrenamiento se ha empleado el algoritmo `AdamW`, una variante mejorada del optimizador `Adam`. A diferencia de su versión original, `AdamW` desacopla explícitamente la actualización de los gradientes del término de regularización, lo que le permite mantener una evolución más estable de los pesos, especialmente en redes profundas.

Asimismo, para combatir la limitación en el tamaño del conjunto de datos y mejorar la capacidad del modelo para generalizar ante nuevas muestras, se ha incorporado un esquema de aumento de datos. En concreto, se aplican rotaciones aleatorias de hasta 10 grados y volteos horizontales sobre los espectrogramas durante el entrenamiento. Estas transformaciones permiten introducir variabilidad sin alterar las características esenciales de las señales originales.

Estas estrategias se complementan con otros aspectos importantes del proceso de entrenamiento, como la selección adecuada del tamaño de los *batches*, que influye directamente en la estabilidad de los gradientes y la eficiencia computacional. En este trabajo se ha utilizado un tamaño de *batch* de 16 muestras, como compromiso entre capacidad de generalización y uso de recursos, teniendo en cuenta las limitaciones de memoria impuestas por el uso de espectrogramas de alta resolución.

4.3. Diseño de la arquitectura del modelo

Antes de llegar al diseño definitivo de la arquitectura descrita en esta sección, se llevaron a cabo numerosas pruebas experimentales con diferentes configuraciones de red. A lo largo del proceso, se evaluaron múltiples alternativas de optimización, incluyendo el uso de distintos algoritmos como `Adam`, `SGD`, `RMSprop` y, finalmente, `AdamW`, que fue el que ofreció un mejor compromiso entre velocidad de convergencia y regularización del modelo. Asimismo, se realizaron ajustes en la cantidad de capas convolucionales, el número de filtros por capa, el tamaño de los *kernels* y la profundidad total de la red, con el objetivo de encontrar una estructura que maximizara el rendimiento sin incurrir en sobreajuste.

También se probaron diversas tasas de *dropout* para regularizar la red, con valores comprendidos entre 0,2 y 0,7, decantándose finalmente por una tasa de 0,45 tras analizar su impacto en la capacidad de generalización. Otros aspectos explorados incluyeron variaciones en las funciones de activación, esquemas de inicialización de pesos, normalización por lotes y técnicas de aumento de datos.

La arquitectura desarrollada en este trabajo está formada por cuatro bloques convolucionales principales. Cada uno de estos bloques incluye una capa convolucional encargada de extraer patrones espaciales de la imagen; después, se aplica una función de activación ReLU. Tras esto, se aplica una capa de normalización por lotes y se cierra cada bloque con una capa de *max-pooling*. Esta estructura permite una extracción progresiva y jerárquica de características desde niveles bajos hasta representaciones más abstractas.

La Figura 4.4 muestra un esquema general de esta arquitectura, destacando las principales etapas de procesamiento desde la entrada (espectrograma) hasta la salida (predicción binaria).

Figura 4.4: Arquitectura del modelo

Estos cuatro bloques funcionan como una etapa de extracción de características jerárquica, en la que las capas más cercanas a la entrada capturan patrones básicos (como bordes o transiciones), mientras que las capas más profundas identifican patrones más abstractos y específicos del problema (como la forma global del espectrograma o zonas de alta intensidad).

Tras la fase convolucional, se utiliza una capa de aplanamiento para convertir la salida tridimensional en un vector unidimensional que pueda ser procesado por las capas densas. Este vector se introduce en una capa totalmente conectada de 128 neuronas, que actúa como una etapa de integración de la información aprendida por los bloques anteriores. Posteriormente, se aplica una función de activación ReLU, y se introduce una capa de *dropout* con una tasa de desactivación del 45 %.

Finalmente, se añade una última capa totalmente conectada con dos neuronas, que genera la probabilidad de pertenencia a una de las dos clases: paciente con Parkinson o persona sana.

Esta arquitectura ha sido diseñada buscando un equilibrio entre complejidad y capacidad de generalización. La elección de cuatro bloques convolucionales, junto con mecanismos como la normalización por lotes y el *dropout*, permite obtener un modelo eficiente y preciso incluso con conjuntos de datos de tamaño limitado.

Capítulo 5

Desarrollo e integración del sistema basado en CNN

Este capítulo describe la implementación práctica del sistema desarrollado, detallando cómo se han materializado las decisiones de diseño expuestas en capítulos anteriores. Se presentan las herramientas y librerías utilizadas, la estructura del código, el preprocesamiento de los datos, el proceso de entrenamiento del modelo convolucional, así como los criterios de evaluación empleados. Además, se analizan los resultados obtenidos y se justifica la elección de la configuración final utilizada para la detección de la enfermedad de Parkinson a partir de espectrogramas de voz.

5.1. Estructura general del sistema

La implementación del modelo se ha llevado a cabo mediante un diseño modular, basado en clases y principios de programación orientada a objetos. La arquitectura está dividida en cinco componentes principales:

- **CNN**: define la arquitectura de la red neuronal convolucional.
- **AudioPreprocessor**: se encarga del procesamiento de las grabaciones, incluyendo eliminación de silencios, segmentación y conversión a espectrogramas.
- **ParkinsonDataset**: organiza las imágenes en un conjunto de datos compatible con PyTorch.
- **ParkinsonTrainer**: coordina el entrenamiento, testeo, generación de espectrogramas y almacenamiento del modelo.
- **Main.py**: actúa como punto de entrada de la aplicación, conectando todos los módulos anteriores.

Esta estructura favorece una alta cohesión de cada clase y un acoplamiento bajo, facilitando tanto la mantenibilidad como la extensibilidad del código.

5.2. Preprocesamiento de grabaciones

El módulo `AudioPreprocessor.py` se encarga de eliminar silencios de las grabaciones de voz y dividirlas en segmentos de duración fija de 10 segundos. Para ello, se utiliza la función `librosa.effects.split`, que detecta las regiones activas del audio (aquellas cuya energía supera un umbral de 20 dB).

Una vez eliminados los silencios, cada señal de audio resultante se divide en fragmentos consecutivos de exactamente 10 segundos. Esta duración se seleccionó por su equilibrio entre contener información suficiente para la predicción y mantener la uniformidad en el tamaño de entrada al modelo convolucional.

Cada uno de los segmentos generados se guarda con un nombre que permite identificar su procedencia. La nomenclatura utilizada sigue el siguiente formato:

`[Grupo]_[Paciente]_[AudioOriginal]_segX.png`

donde `Grupo` indica si pertenece a un paciente con Parkinson (`PD`) o a un control sano (`HC`), `Paciente` es el identificador del individuo, `AudioOriginal` es el nombre del archivo de audio original sin la extensión, y `segX` indica el índice del segmento correspondiente.

Por ejemplo, si un archivo llamado `ta.wav` del paciente `P01`, que pertenece al grupo `PD`, tiene una duración útil de 22 segundos (tras eliminar los silencios), este se divide en dos segmentos completos de 10 segundos cada uno:

- `PD_P01_ta_seg0.png` (10 segundos)
- `PD_P01_ta_seg1.png` (10 segundos)

El fragmento restante, de aproximadamente 2 segundos, se descarta, ya que en este trabajo se ha optado por conservar únicamente aquellos segmentos que alcanzan la duración completa establecida para la entrada del modelo.

Esta decisión, aunque conlleva una ligera reducción del número total de muestras disponibles, presenta varias ventajas desde el punto de vista del aprendizaje automático. En primer lugar, evita introducir ruido artificial en los datos de entrada, ya que el rellenado con ceros de fragmentos más cortos puede generar patrones irreales que el modelo podría interpretar como características relevantes. Además, trabajar únicamente con fragmentos de duración uniforme garantiza una mayor homogeneidad estructural en el conjunto de

datos, lo cual facilita el aprendizaje de patrones consistentes y reduce el riesgo de sobreajuste a muestras atípicas. En segundo lugar, los fragmentos breves suelen contener menos información acústica útil, por lo que su inclusión podría aportar más variabilidad que contenido discriminativo. Por último, desde el punto de vista computacional, esta estrategia permite simplificar el pipeline de preprocesamiento y entrenamiento, al no requerir lógica adicional para gestionar excepciones, escalados o normalización de entradas de tamaño variable.

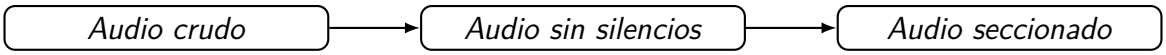

Figura 5.1: Proceso de preprocesamiento del audio

5.3. Conversión a espectrogramas

Los espectrogramas se generan a partir de los segmentos de audio utilizando la transformada de Fourier junto con un banco de filtros Mel, mediante la función `librosa.feature.melspectrogram`. A continuación, se aplica una **transformación logarítmica** mediante `librosa.power_to_db`, la cual convierte los valores de energía a una escala logarítmica expresada en decibelios (dB).

Este paso es fundamental porque la percepción humana del sonido no es lineal, sino logarítmica. Nuestro oído es más sensible a variaciones de volumen en niveles bajos que en niveles altos, por lo que esta transformación refleja mejor cómo los humanos perciben la intensidad sonora. Además, permite comprimir el rango dinámico de los valores del espectrograma, facilitando su representación visual y optimizando su tratamiento por parte del modelo de red neuronal.

Una vez convertidos, los espectrogramas se guardan como imágenes en formato `.png`. Para su visualización, se utiliza el mapa de colores `inferno`, que resalta con mayor claridad las zonas de alta energía.

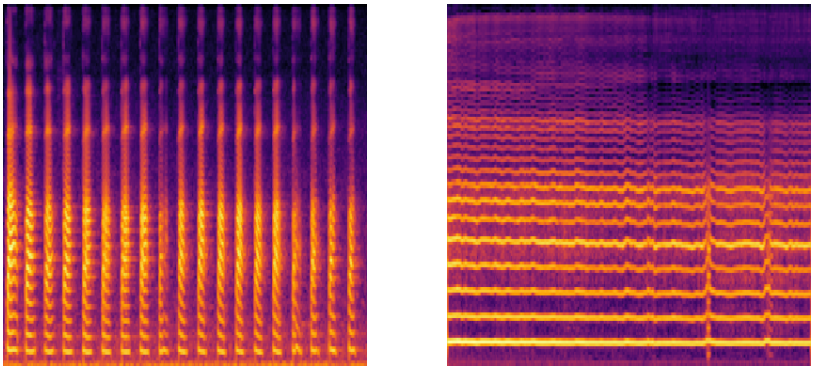

Figura 5.2: Ejemplos de espectrogramas procesados.

Como se puede observar en la Figura 5.2, lado izquierdo, el espectrograma presenta una estructura caracterizada por líneas verticales. Esto se debe a que el audio original consiste en la repetición continua de la sílaba “ka”. Tras el proceso de eliminación de silencios, el audio se reduce a segmentos compactos donde cada “línea vertical” representa una emisión individual de la sílaba. Aunque en cada línea vertical pueden observarse también componentes horizontales (frecuencias específicas de la sílaba), su breve duración temporal hace que estas se perciban menos evidentes.

En contraste, la imagen del lado derecho de la Figura 5.2 muestra un espectrograma con predominancia de líneas horizontales, correspondiente a un audio donde se mantiene una vocal durante un período prolongado. Estas líneas horizontales reflejan la presencia de frecuencias constantes en el tiempo, características típicas de sonidos mantenidos o estacionarios. En este tipo de espectrograma, es posible identificar claramente las frecuencias fundamentales y sus armónicos, que se manifiestan como bandas horizontales de mayor intensidad.

Esta diferenciación visual entre los espectrogramas es ilustrativa de cómo las distintas características acústicas del habla se representan en el dominio tiempo-frecuencia, y cómo el preprocesamiento influye en la forma en que se visualizan estos patrones. Esta información es esencial para que el modelo pueda aprender a distinguir patrones asociados a patologías como el Parkinson a partir de las imágenes de espectrogramas generadas.

5.4. División y transformación de datos

Para preparar correctamente los datos antes de entrenar la red neuronal, se realiza una división y transformación cuidadosa de los espectrogramas generados. En primer lugar, se separan los datos en conjuntos de entrenamiento y prueba, garantizando que no haya solapamiento de pacientes entre ambos. Esta división se realiza a nivel de paciente, extrayendo su identificador del nombre del archivo.

Es importante destacar que esta división por paciente no solo evita la fuga de información entre los conjuntos de entrenamiento y prueba, sino que también previene que el modelo aprenda a identificar a los individuos en lugar de generalizar patrones característicos de la enfermedad. Si no se realiza esta separación, el modelo puede ”memorizarçaracterísticas específicas de los pacientes presentes en ambos conjuntos, lo que conduce a métricas infladas y poco representativas del rendimiento real. Este aspecto metodológico suele pasarse por alto en algunos estudios, como el de Little et al. [5], donde no se aplica una separación estricta por paciente, lo que puede comprometer la validez de los resultados obtenidos.

Cada imagen de espectrograma se somete posteriormente a un conjunto de transformaciones con los siguientes objetivos:

- **Redimensionamiento a 256x256 píxeles:** los espectrogramas generados mediante `librosa` tienen un tamaño original de aproximadamente 387×385 píxeles, derivado de la configuración del gráfico utilizado para su visualización. Este tamaño no varía significativamente entre muestras, ya que todos los fragmentos de audio tienen la misma duración (10 segundos). Sin embargo, para asegurar una entrada uniforme al modelo y facilitar el uso de arquitecturas convolucionales estándar, todas las imágenes se redimensionan a una resolución fija de 256×256 píxeles mediante la transformación `Resize` de `torchvision`. Esta resolución se ha elegido por ser lo suficientemente detallada para preservar patrones relevantes en los espectrogramas, al mismo tiempo que mantiene una carga computacional razonable durante el entrenamiento.
- **Aumento de datos (*data augmentation*):** se aplican transformaciones aleatorias como rotaciones de hasta 10 grados y volteo horizontal. Estas técnicas permiten al modelo generalizar mejor, evitando que se memoricen características específicas del conjunto de entrenamiento.
- **Normalización de los valores de píxeles:** los valores RGB de las imágenes se normalizan canal por canal con una media de 0,5 y una desviación estándar de 0,5. Esta práctica es habitual en redes neuronales convolucionales porque acelera la convergencia del modelo y mejora la estabilidad numérica durante el entrenamiento.

Estas transformaciones son implementadas mediante la librería `torchvision.transforms` de PyTorch y se aplican dinámicamente al cargar cada imagen del dataset.

5.5. Diseño e implementación de la red neuronal

La implementación de la arquitectura propuesta se lleva a cabo en el módulo `CNN.py`, donde se define una clase `CNN` que hereda de `nn.Module`. Este archivo encapsula la lógica del modelo y lo estructura en dos fases diferenciadas: una etapa convolucional para la extracción de características y otra densa para la clasificación.

La red se inicializa con cuatro bloques convolucionales sucesivos, cada uno compuesto por una convolución bidimensional (`Conv2d`), una activación ReLU, una normalización por lotes (`BatchNorm2d`) y una capa de `MaxPooling`. Estos bloques están codificados explícitamente en el constructor `__init__` y agrupados funcionalmente mediante el método `forward_features()`, lo que permite modularizar el procesamiento de la entrada en dos fases claramente diferenciadas.

Tras la etapa convolucional, la salida tridimensional es aplanada (`nn.Flatten`) y procesada por una capa totalmente conectada de 128 neuronas. A esta se le aplica una función ReLU y una capa *dropout* con una tasa de desactivación del 45 %, cuyo objetivo es reducir el sobreajuste al forzar al modelo a no depender excesivamente de determinadas neuronas durante el entrenamiento.

La última capa densa (`Linear(128, 2)`) actúa como clasificador binario, produciendo las probabilidades asociadas a cada clase (persona sana o paciente con Parkinson) a partir de las características extraídas.

El método `forward()` se encarga de definir el recorrido completo de los datos por la red, primero invocando `forward_features()` para aplicar las convoluciones, y posteriormente ejecutando las capas densas hasta obtener la predicción final.

Esta estructura modular y jerárquica no solo favorece la claridad del código, sino que también permite una mayor facilidad para realizar modificaciones, pruebas de componentes individuales o adaptaciones futuras a nuevas tareas de clasificación.

5.6. Entrenamiento del modelo

El modelo convolucional se ha entrenado utilizando un conjunto total de 1792 espectrogramas generados a partir de las grabaciones de voz preprocesadas. De ellos, el 80 % (1416 muestras) se ha empleado para el entrenamiento, mientras que el 20 % restante (376 muestras) se ha reservado para la fase de evaluación. Esta partición se ha realizado asegurando que las muestras de un mismo paciente no estén presentes en ambos subconjuntos, evitando así la fuga de información y permitiendo una evaluación más realista de la capacidad de generalización del modelo.

El entrenamiento del modelo es gestionado por la clase `ParkinsonTrainer`, localizada en el archivo `Trainer.py`. Esta clase encapsula todo el flujo de trabajo, desde el preprocesamiento hasta la validación. El proceso completo sigue los siguientes pasos:

1. Se invoca el módulo de preprocesamiento para generar los espectrogramas a partir de las grabaciones de voz.
2. Se asignan los espectrogramas a conjuntos de entrenamiento y validación siguiendo una partición por paciente, lo que evita que muestras del mismo individuo aparezcan en ambos conjuntos.
3. A los espectrogramas se les aplica una transformación compuesta que incluye redimensionamiento, rotación aleatoria y normalización.
4. Se inicializa el modelo, la función de pérdida y el optimizador.
5. Se ejecuta el entrenamiento durante 30 épocas.

En cuanto al tiempo de ejecución, cada época tiene una duración aproximada de un minuto, por lo que el entrenamiento completo dura en torno a 30 minutos. Tras finalizar el entrenamiento, la evaluación del modelo sobre el conjunto de test tiene una duración de aproximadamente 10 segundos, y el cálculo del área bajo la curva ROC (AUC) requiere un tiempo adicional de unos 10 segundos.

Para minimizar la función objetivo se emplea `CrossEntropyLoss`. Como optimizador se utiliza `AdamW`, una variante de Adam que incluye regularización por decaimiento de pesos (`weight_decay`) para evitar el sobreajuste. En este trabajo se ha utilizado un `learning rate` de $0,0005$ y un `weight_decay` de $1e$-3.

Durante el proceso de entrenamiento, la biblioteca `tqdm` permite monitorizar el progreso de cada época en tiempo real. Para cada *batch* de entrenamiento, se muestra tanto la función de pérdida como la precisión acumulada, lo cual facilita la detección de estancamientos o divergencias.

La Figura 5.3 muestra la evolución del *accuracy* y la función de pérdida (*loss*) en cada *mini-batch* del conjunto de entrenamiento a lo largo de las 30 épocas. Se observa una tendencia ascendente en la precisión y una disminución progresiva del error, aunque en las últimas épocas el rendimiento se estabiliza. Esta estabilización sugiere que el modelo ha alcanzado un punto óptimo de aprendizaje.

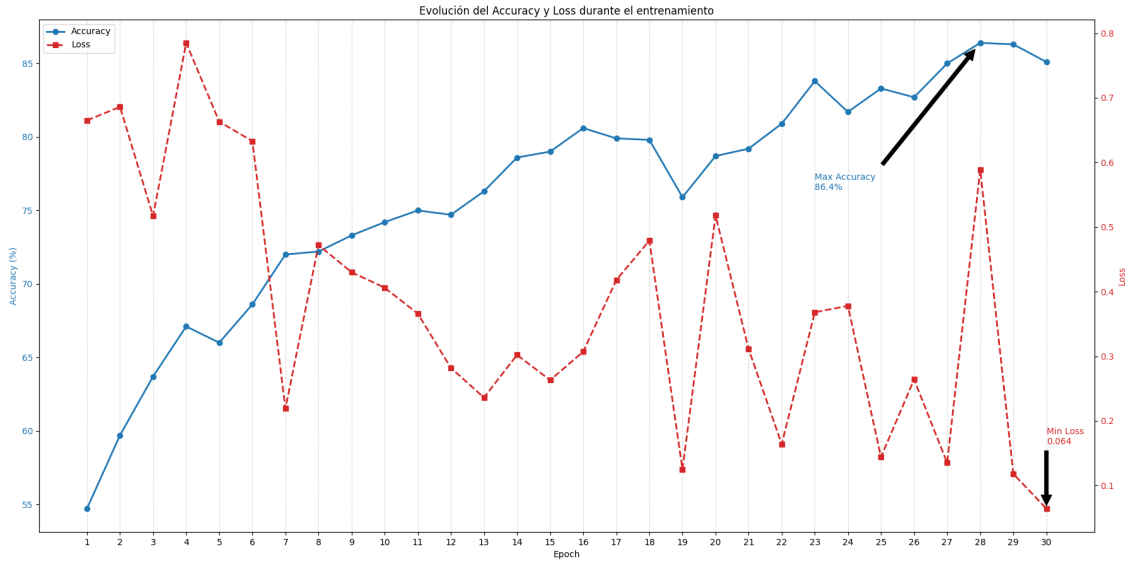

Figura 5.3: Evolución del *accuracy* y *loss* durante el entrenamiento del modelo a lo largo de 30 épocas.

5.7. Evaluación del modelo

Una vez completado el proceso de entrenamiento, el modelo se somete a una evaluación utilizando el conjunto de validación. Esta fase tiene como objetivo estimar su capacidad de generalización ante datos no vistos y detectar posibles signos de sobreajuste.

El procedimiento de evaluación es realizado por el método `test_model()` de la clase `ParkinsonTrainer`, que permite obtener las métricas clave del rendimiento del modelo y facilita la comparación entre diferentes configuraciones.

Las acciones que se llevan a cabo son:

- Cálculo de la pérdida media (*loss*) sobre el conjunto de test, usando la misma función de pérdida empleada durante el entrenamiento (`CrossEntropyLoss`).
- Cómputo de la precisión global (*accuracy*), que representa el porcentaje de predicciones correctas entre todas las muestras del conjunto de validación.
- Visualización de los resultados en consola a través de `tqdm`.

Además de estas métricas básicas, se han incorporado herramientas adicionales de análisis que permiten evaluar el rendimiento del modelo de forma más detallada.

5.7.1. Curva ROC y AUC

Una vez entrenado el modelo, se ha evaluado su capacidad de discriminación mediante la generación de la curva ROC (*Receiver Operating Characteristic*), que representa la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) para distintos umbrales de decisión. Esta curva permite analizar el comportamiento del modelo más allá de un único punto de corte, proporcionando una visión más completa de su rendimiento.

Es importante destacar que tanto la curva ROC como el valor del AUC (Área Bajo la Curva) se han calculado utilizando exclusivamente el conjunto de prueba, es decir, datos no vistos durante el entrenamiento. Esto garantiza que las métricas obtenidas reflejan la capacidad real de generalización del modelo.

La Figura 5.4 muestra la curva ROC obtenida tras la evaluación del modelo, donde se puede observar su capacidad de discriminación entre clases.

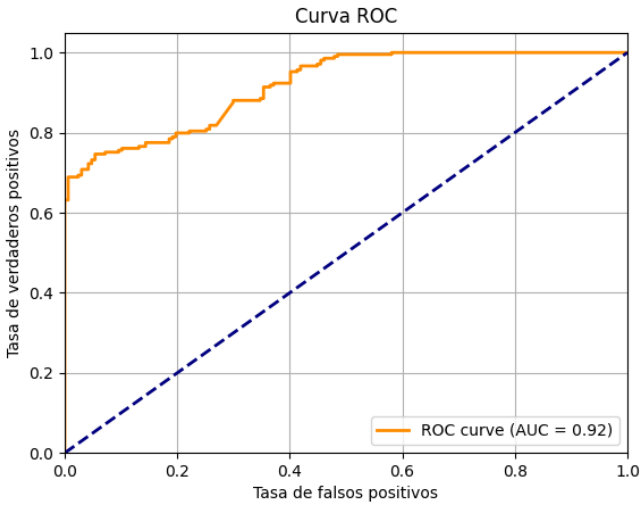

Figura 5.4: Curva ROC con un AUC de 0,9180.

El área bajo la curva (AUC-ROC) alcanzó un valor de **0,9180**, lo que indica una excelente capacidad del modelo para discriminar entre pacientes con Parkinson y personas sanas.

Además del *accuracy*, se calcularon las métricas de **precisión**, **recobrado** *(recall)* y **F1-*score*** para cada clase, tal y como se muestra en el Cuadro 5.1:

Clase	**Precisión**	***Recall***	***F1-score***	***Support***
Sano	0,76	0,81	0,79	167
Parkinson	0,84	0,80	0,82	209
Accuracy		0,81		
Macro promedio	0,80	0,81	0,80	376
Promedio ponderado	0,81	0,81	0,81	376

Cuadro 5.1: Métricas de clasificación en el conjunto de test.

5.7.2. Matriz de confusión por espectrograma

La matriz de confusión permite visualizar de forma clara cuántas muestras fueron clasificadas correctamente o erróneamente por el modelo para cada clase.

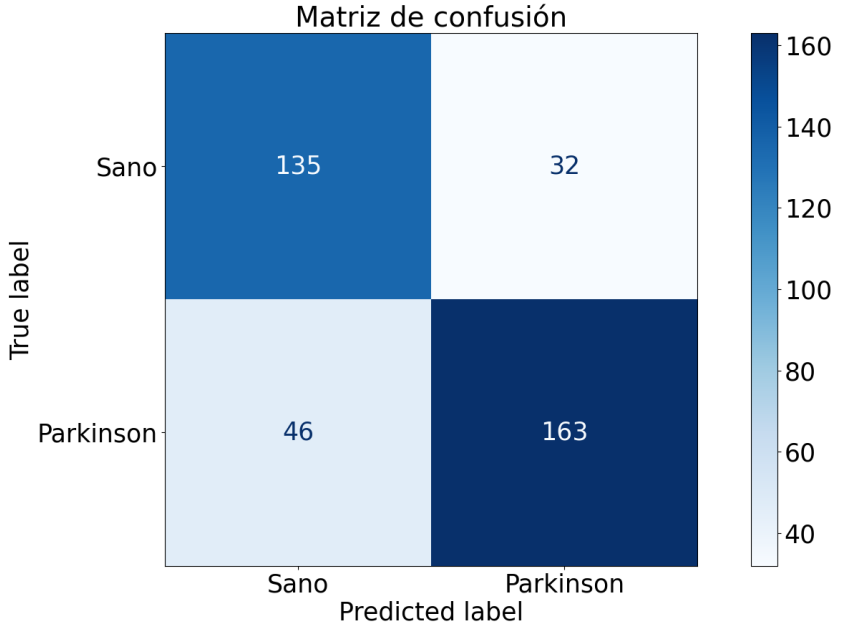

Figura 5.5: Matriz de confusión obtenida tras la evaluación del modelo.

Como se observa en la Figura 5.5, el modelo presenta un comportamiento equilibrado entre las dos clases, con una ligera superioridad en la detección de casos de Parkinson, lo cual es deseable en aplicaciones clínicas donde los falsos negativos pueden tener consecuencias más graves.

5.7.3. Evaluacion por paciente

Además de la evaluación por espectrograma individual, se ha realizado una evaluación adicional a nivel de paciente, agrupando todos los espectrogramas pertenecientes a una misma persona. Para ello, se ha calculado la probabilidad media de Parkinson entre todos los segmentos de audio de cada paciente, y se ha aplicado un umbral de 0,5 para determinar la clase final.

Esta estrategia permite obtener una visión más realista del rendimiento del modelo en un contexto clínico, donde el diagnóstico se realiza por paciente y no por fragmento de voz.

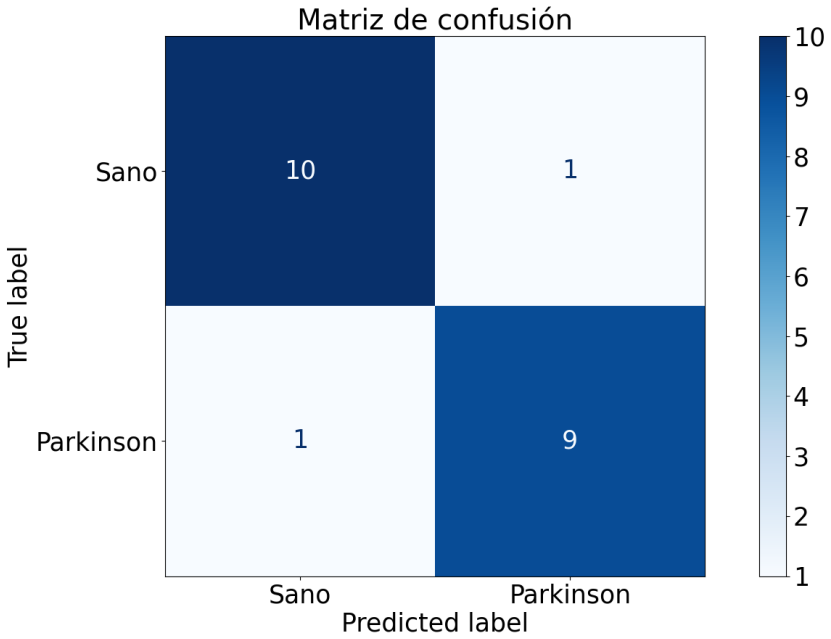

Figura 5.6: Matriz de confusión obtenida tras la evaluación del modelo.

La matriz de confusión obtenida por paciente se muestra en la Figura 5.6. En ella se observa que el modelo clasificó correctamente a 10 de los 11 pacientes sanos y a 9 de los 10 pacientes con Parkinson. Esto se traduce en una precisión del 91 % para la clase "Sano" y una precisión del 90 % para la clase "Parkinson". Las tasas de precisión por clase pueden consultarse en detalle en el Cuadro 5.2.

Clase	**Precisión**	***Recall***	***F1-score***	***Support***
Sano	0,91	0,91	0,91	11
Parkinson	0,90	0,90	0,90	10

Cuadro 5.2: Métricas de clasificación en el conjunto de test.

El modelo alcanza una precisión global del 90 % por paciente, con un *F1-score* promedio de 0,90. Estos resultados reflejan un buen equilibrio entre sensibilidad y especificidad.

Capítulo 6

Implementación de la aplicación

Siguiendo los principios establecidos por Sommerville en su libro *Software Engineering* [8], la implementación de la aplicación se ha llevado a cabo mediante una arquitectura modular y orientada a objetos, empleando un enfoque incremental y centrado en componentes reutilizables.

Dado el carácter exploratorio del presente trabajo, centrado en la validación de un enfoque técnico más que en la entrega de un producto final, no se llevó a cabo una fase formal de captura de requisitos. En lugar de ello, los objetivos del proyecto y las decisiones de diseño se guiaron por criterios de viabilidad técnica, disponibilidad de herramientas y recursos, y la necesidad de evaluar el rendimiento del modelo propuesto en un entorno controlado. Esta aproximación es habitual en proyectos de investigación o trabajos académicos, donde el foco principal reside en la experimentación y la demostración de conceptos, más que en la implementación de soluciones orientadas a usuarios finales o clientes específicos.

6.1. Arquitectura basada en componentes

Esta arquitectura facilita la separación de responsabilidades, el mantenimiento del software y la reutilización de módulos funcionales.

Los componentes principales son los siguientes:

- **Modelo**: contiene todos los elementos relacionados con la definición de la red neuronal. El único módulo relevante para la aplicación es:
  - `CNN.py`: define la arquitectura de la red convolucional utilizada tanto en entrenamiento como en inferencia.
- **Interfaz**: contiene la aplicación final que interactúa con el usuario a través de una interfaz gráfica implementada en `Tkinter`. Esta carpeta reutiliza el modelo entrenado previamente. Sus módulos principales son:

- `Gui.py`: define y controla la interfaz gráfica, permitiendo la selección de archivos o carpetas de audio.
- `Parkinson_predictor.py`: encapsula la carga del modelo entrenado y la lógica de predicción a partir de espectrogramas.
- `Spectrogram_creator.py`: convierte el audio en imágenes de espectrogramas.
- `Main.py`: punto de entrada de la aplicación gráfica.

6.2. Modelo de desarrollo incremental

El desarrollo de la aplicación se ha llevado a cabo utilizando un enfoque incremental. Esta estrategia permite construir el sistema por etapas, añadiendo nuevas funcionalidades en ciclos iterativos, lo que facilita la detección de errores y mejora la capacidad de respuesta ante cambios en los requisitos.

En una primera iteración se construyó una interfaz gráfica básica con capacidad para cargar archivos de audio de forma individual. Posteriormente, se incorporó el modelo entrenado, reutilizando la arquitectura definida en el componente `CNN.py` del sistema. A medida que se consolidaba la funcionalidad principal, se fueron agregando características adicionales, como la posibilidad de procesar carpetas completas, generar diagnósticos agregados, visualizar espectrogramas y representar gráficamente los resultados mediante diagramas circulares.

Este enfoque ha favorecido una validación continua del desarrollo, facilitando la identificación de problemas de integración y permitiendo realizar ajustes progresivos orientados a mejorar la experiencia de usuario. Asimismo, ha facilitado la integración modular de los distintos elementos de la aplicación, como los módulos de predicción, visualización de resultados y tratamiento de archivos.

6.3. Diseño orientado a objetos

La aplicación ha sido desarrollada siguiendo el paradigma de diseño orientado a objetos, tal como se recomienda en la ingeniería del software moderna [8]. Este enfoque permite organizar el sistema en torno a unidades funcionales cohesivas y reutilizables, con responsabilidades claramente definidas.

Cada componente del sistema encapsula tanto datos como comportamientos relacionados, lo que facilita su mantenibilidad y su extensión futura. La separación en módulos independientes también favorece la escalabilidad y permite realizar pruebas unitarias de forma más efectiva.

Por ejemplo, el modelo de red neuronal, la lógica de predicción y la interfaz gráfica se encuentran distribuidos en módulos independientes. Este diseño modular ha permitido,

entre otras ventajas, reutilizar directamente la definición de la red convolucional tanto en el proceso de entrenamiento como en la aplicación de predicción interactiva.

Gracias a esta estructura, ha sido posible añadir nuevas funcionalidades —como la selección de carpetas o la visualización gráfica de resultados— sin necesidad de reestructurar el sistema completo, lo que refleja los principios de bajo acoplamiento y alta cohesión promovidos por este enfoque. La Figura 6.1 muestra cómo se organizan e interconectan los principales módulos funcionales de la aplicación.

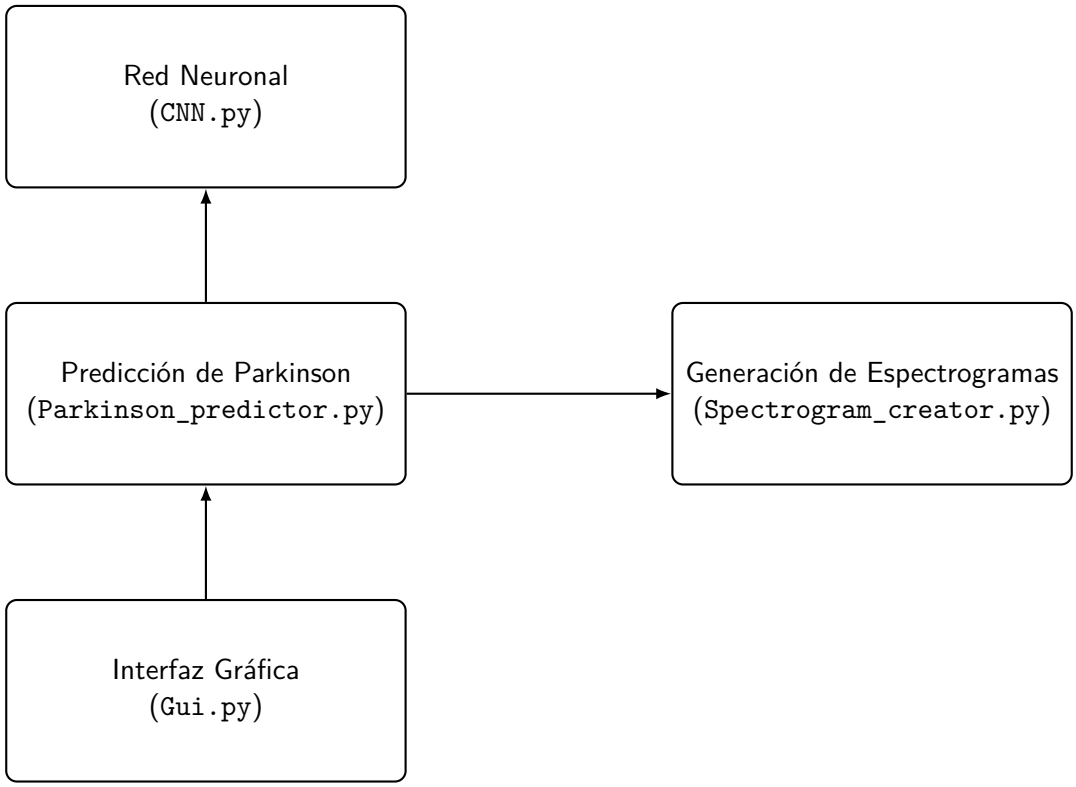

Figura 6.1: Relaciones entre módulos funcionales de la aplicación

6.4. Interacción entre componentes

La interacción entre los componentes sigue un flujo jerárquico bien definido, que comienza con la entrada del usuario en la interfaz gráfica y termina con la presentación de los resultados.

El flujo de funcionamiento cuando el usuario selecciona un archivo de audio o una carpeta completa es el siguiente:

1. La interfaz gráfica, definida en `Gui.py`, permite al usuario seleccionar un archivo o una carpeta con grabaciones. Este módulo actúa como punto de entrada, controlando la interacción usuario-sistema.

2. La solicitud se envía al componente de predicción, ubicado en `Parkinson_predictor.py`, que gestiona la lógica del flujo de predicción: carga del modelo, procesamiento del audio y obtención de resultados.

3. Este componente invoca a su vez a `Spectrogram_creator.py`, que genera imágenes espectrales de los segmentos de audio a través de transformadas de Fourier y escalas Mel, necesarias para el funcionamiento de la red convolucional.

4. Una vez creado el espectrograma, se transforma y normaliza con los parámetros establecidos y se utiliza como entrada para el modelo CNN previamente entrenado, el cual devuelve una prediccion.

5. Finalmente, los resultados son devueltos al módulo de interfaz gráfica, que se encarga de mostrarlos al usuario mediante texto e ilustraciones (por ejemplo, la gráfica y el espectrograma visualizado)

6. En el caso de analizar una carpeta, este flujo se repite por cada archivo, y luego se calcula una media ponderada de las predicciones individuales.

La Figura 6.2 muestra una representación esquemática del flujo de la aplicación:

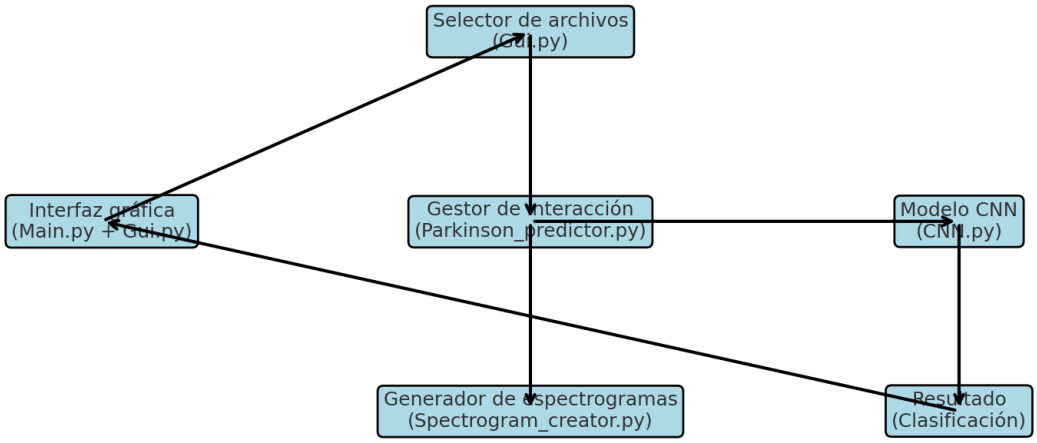

Figura 6.2: Flujo de interacción entre módulos de la aplicación

Este flujo modular asegura un desacoplamiento entre lógica de presentación, lógica de negocio y modelo, lo que sigue el principio de separación de responsabilidades y permite un fácil mantenimiento del sistema.

6.5. Interfaz gráfica de usuario

La aplicación desarrollada cuenta con una interfaz gráfica diseñada con la librería `Tkinter`, lo que permite una interacción directa e intuitiva por parte del usuario. A continuación,

se describen las principales vistas de la interfaz:

- Panel lateral izquierdo: contiene los botones para cargar un archivo de audio, una carpeta completa, reiniciar la interfaz o cerrar la aplicación.
- Panel central: muestra los resultados del análisis, incluyendo el diagnóstico final y una representación gráfica en forma de diagrama circular con los porcentajes de probabilidad.
- Panel lateral derecho: presenta visualmente el espectrograma asociado al archivo analizado, junto con su nombre y porcentaje de Parkinson.

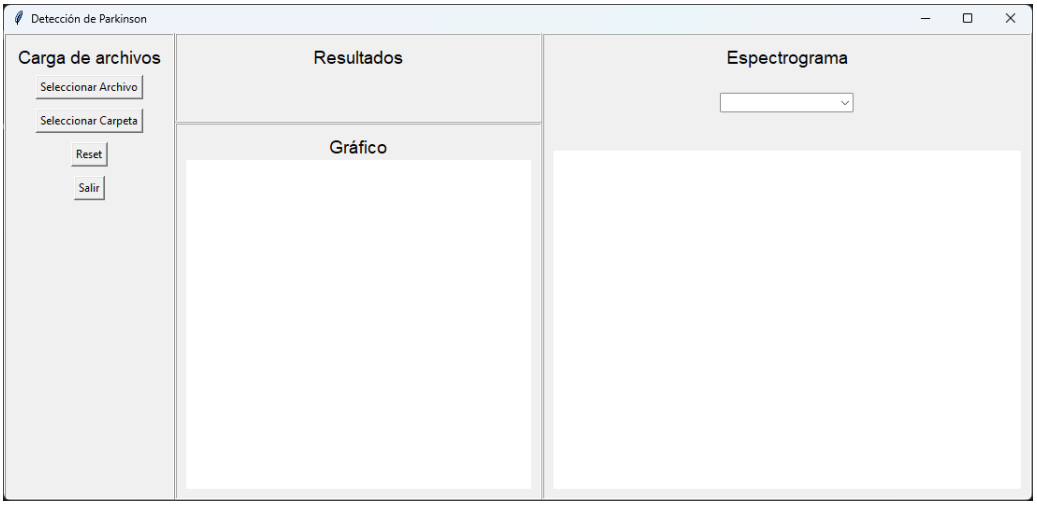

Figura 6.3: Vista inicial de la interfaz al ejecutar la aplicación

En la Figura 6.3 se muestra la ventana inicial de la aplicación, que permanece a la espera de que el usuario seleccione un archivo o carpeta de entrada.

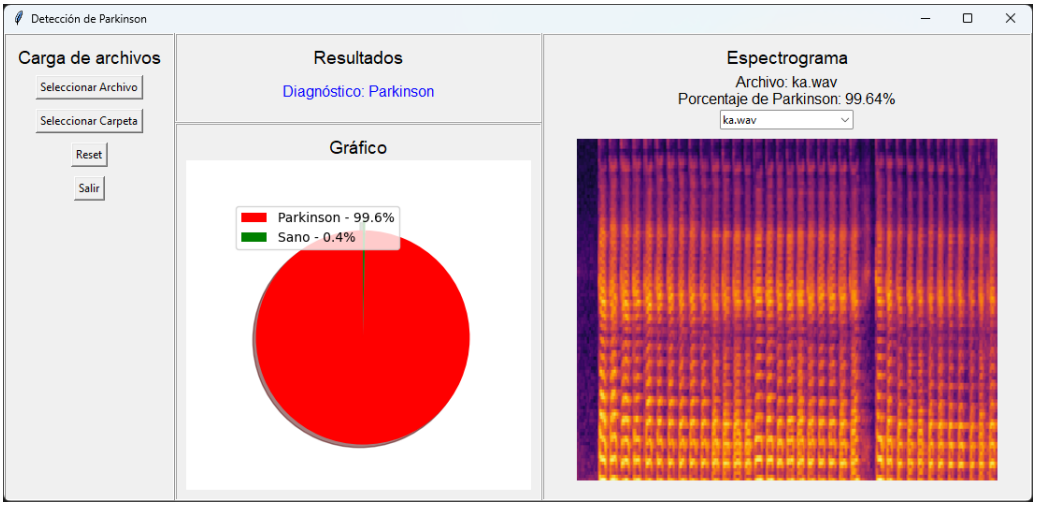

Figura 6.4: Interfaz tras seleccionar un archivo de audio individual

En la Figura 6.4 puede observarse cómo, al seleccionar un único archivo, se genera su espectrograma, se realiza la predicción de Parkinson y se muestran los porcentajes correspondientes junto con una visualización del espectrograma.

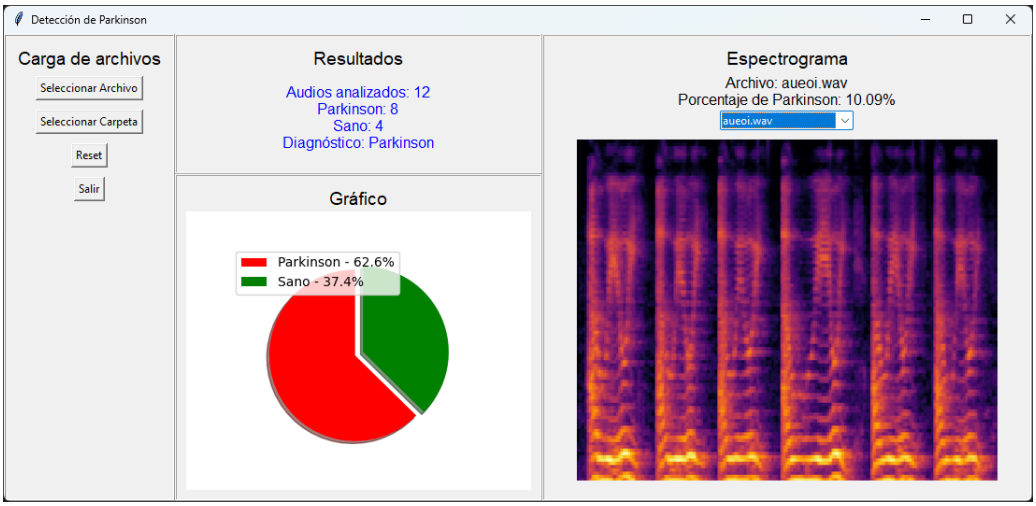

Figura 6.5: Interfaz tras seleccionar una carpeta con varios archivos

Finalmente, en la Figura 6.5 se observa el análisis de una carpeta completa. En este caso, el sistema procesa cada archivo de forma individual y calcula un diagnóstico global del paciente a partir de la media de las predicciones obtenidas para cada audio. Esta media se compara con el umbral de 0,5 para determinar si el paciente es clasificado como sano o con Parkinson. El resultado se muestra en el centro de la pantalla mediante un diagrama circular que representa el porcentaje global de Parkinson estimado para el conjunto de grabaciones.

Además, en el panel derecho de la interfaz se visualiza el porcentaje de Parkinson correspondiente a cada archivo individual. El usuario puede seleccionar cualquier audio desde el menú desplegable, lo que actualiza automáticamente la visualización del espectrograma asociado, así como el nombre del archivo y su predicción individual.

6.6. Consideraciones de usabilidad y diseño

El diseño de la interfaz gráfica se ha realizado teniendo en cuenta principios de usabilidad y experiencia de usuario, con el objetivo de proporcionar una aplicación accesible, comprensible y eficiente, incluso para usuarios sin experiencia técnica.

Según los principios expuestos por Sommerville en su capítulo sobre interfaces de usuario [8], se han seguido las siguientes directrices:

- **Simplicidad visual:** la disposición de los elementos en la pantalla se ha mantenido

clara y ordenada, distribuyendo los botones de acción a la izquierda y los resultados en la parte central y derecha, evitando la sobrecarga cognitiva.

- **Feedback inmediato:** una vez el usuario selecciona un archivo o carpeta, los resultados del análisis se actualizan de forma instantánea, mostrando los porcentajes y espectrogramas generados, lo que mejora la percepción de respuesta y control.
- **Consistencia:** las acciones están etiquetadas de forma coherente (por ejemplo, "Seleccionar Archivo" y "Seleccionar Carpeta"), lo que permite al usuario anticipar el comportamiento del sistema.
- **Minimización del error:** se han incluido comprobaciones básicas como validar que los archivos seleccionados tengan la extensión correcta (`.wav`) y mostrar mensajes apropiados si no se encuentra contenido válido en una carpeta.
- **Organización modular:** gracias a la estructura basada en clases como `Gui`, `ParkinsonPredictor` y `SpectrogramCreator`, es posible modificar o ampliar funcionalidades de forma localizada, sin afectar al resto de la interfaz.

El uso de gráficos (como el diagrama circular para visualizar el porcentaje de Parkinson) proporciona una representación visual accesible incluso para usuarios sin conocimientos técnicos, facilitando la interpretación de los resultados.

En conjunto, el diseño de la interfaz se ha centrado en facilitar la interacción del usuario, garantizando que el flujo completo —desde la carga del archivo hasta la obtención del diagnóstico— sea claro, visual y eficiente.

6.7. Pruebas unitarias de la interfaz gráfica

Para garantizar la fiabilidad y estabilidad de la aplicación desarrollada, se han implementado pruebas unitarias sobre la interfaz gráfica (GUI) utilizando el framework `unittest` de Python. Este conjunto de pruebas permite validar el correcto funcionamiento de los principales elementos de la interfaz, asegurando que las interacciones con el usuario producen los resultados esperados y que no se introducen errores inesperados durante la ejecución.

Las pruebas se definen en un módulo independiente, siguiendo una estrategia de aislamiento funcional y reutilización de componentes. Para evitar dependencias externas y efectos secundarios, se hace uso extensivo de la biblioteca `unittest.mock`, que permite simular el comportamiento del modelo de predicción y las operaciones de selección de archivos del sistema.

Entre las funcionalidades testadas destacan:

- Verificación del título de la ventana principal tras la inicialización de la aplicación.

- Comprobación de que el menú desplegable y las etiquetas de resultado se actualizan correctamente tras analizar un archivo de audio.
- Validación de la funcionalidad de reinicio de la interfaz (`reset`), asegurando que se limpian correctamente todos los elementos visuales.
- Simulación de la carga de múltiples archivos desde una carpeta y comprobación de que los resultados se agregan correctamente.
- Evaluación de la lógica asociada al cambio de selección en el menú desplegable, asegurando que se actualiza la información del espectrograma.

Cada prueba se ejecuta de forma aislada en un entorno gráfico controlado mediante `tkinter.Tk()`, y al finalizar se realiza una limpieza para liberar recursos y eliminar archivos temporales generados.

Esta batería de tests contribuye a mejorar la calidad del software, permitiendo detectar errores de forma temprana y garantizar la robustez de la interfaz ante distintos escenarios de uso.

6.8. Posibles mejoras y líneas futuras

La implementación actual proporciona una base funcional sólida para la detección automática de Parkinson a partir de voz. Sin embargo, existen diversas líneas de trabajo que podrían desarrollarse en el futuro para ampliar las funcionalidades, mejorar la accesibilidad y ofrecer un sistema más completo, robusto y personalizado.

6.8.1. Adquisición de audio en tiempo real desde la interfaz

Una de las mejoras más relevantes consiste en permitir al usuario realizar las pruebas de voz directamente desde la propia aplicación, sin necesidad de cargar archivos previamente grabados. Esta funcionalidad convertiría el sistema en una herramienta completamente autónoma y accesible, especialmente útil para entornos clínicos o usuarios sin experiencia técnica.

El sistema podría incluir un botón “Iniciar prueba”, que active un asistente guiado. Este asistente iría indicando al usuario las distintas pruebas fonéticas requeridas (por ejemplo: repetir la sílaba “ka”, mantener una vocal, leer un texto), cada una acompañada de un botón de grabación, uno de reinicio y uno de confirmación. Al finalizar, el usuario podría revisar las grabaciones, repetir alguna si fuera necesario y enviar todos los datos para su análisis automático.

Esta propuesta no solo mejora la experiencia de uso, sino que estandariza el proceso de recogida de datos, reduciendo errores humanos y facilitando la adopción en contextos reales.

6.8.2. Despliegue como aplicación web o móvil

Otra línea de desarrollo natural sería el despliegue del sistema como una aplicación web accesible desde navegadores, o como una app móvil compatible con sistemas Android o iOS. Esta adaptación ampliaría considerablemente el alcance de la herramienta, facilitando su uso en consultas médicas, instituciones sanitarias o incluso a nivel personal.

Además, el uso de tecnologías multiplataforma permitiría integrar funciones adicionales como almacenamiento en la nube, historial de pruebas o sincronización con dispositivos médicos.

6.8.3. Sistema de seguimiento longitudinal

Incorporar un sistema de seguimiento longitudinal permitiría a los usuarios realizar pruebas periódicas y visualizar la evolución de sus resultados en el tiempo. Esta funcionalidad tendría un gran valor tanto clínico como investigativo, ya que permitiría detectar patrones de evolución en etapas tempranas de la enfermedad.

La aplicación podría almacenar registros con fecha, resultados de cada prueba y gráficas comparativas. Con el tiempo, se podrían generar perfiles personalizados que reflejen la progresión del deterioro vocal y ayuden a ajustar tratamientos o estrategias de intervención.

6.8.4. Explicabilidad del modelo

Otra mejora relevante sería la incorporación de técnicas de explicabilidad de modelos (*model explainability*) que permitan al usuario o al profesional entender por qué el modelo ha tomado una determinada decisión.

Por ejemplo, podrían utilizarse técnicas como Grad-CAM [6] o visualizaciones de saliencia para resaltar las regiones del espectrograma que han tenido más peso en la predicción final. Esto aportaría transparencia al sistema y aumentaría la confianza en sus resultados, lo cual es fundamental en aplicaciones relacionadas con la salud.

Capítulo 7

Conclusión

Este TFG ha demostrado la viabilidad de utilizar técnicas de aprendizaje profundo, concretamente redes neuronales convolucionales, para la detección automática de la enfermedad de Parkinson a partir de grabaciones de voz. A lo largo del desarrollo del proyecto se ha abordado de forma integral el proceso, desde la adquisición y preprocesamiento de los datos hasta la implementación de un modelo funcional y su integración en una aplicación con interfaz gráfica.

El sistema propuesto ha alcanzado una precisión del 80 % y un AUC de 0,918 en la clasificación por espectrograma individual, lo que evidencia su capacidad para discriminar entre pacientes con Parkinson y personas sanas a partir de espectrogramas de Mel. Además, se ha realizado una evaluación complementaria a nivel de paciente, agrupando las predicciones de todos los audios correspondientes a una misma persona. En este caso, el modelo ha alcanzado una precisión global del 90 % con un *F1-score* promedio de 0,90, mostrando un rendimiento equilibrado entre ambas clases. Esta evaluación por paciente resulta muy relevante en contextos clínicos, donde el diagnóstico se realiza a nivel de paciente y no por fragmento de voz.

Estos resultados, junto con la robustez del modelo frente a la variabilidad fonética y la diversidad de tareas vocales, refuerzan el potencial de la voz como biomarcador no invasivo en contextos clínicos. Además del desarrollo del modelo, se ha puesto especial énfasis en la creación de una herramienta accesible y usable, orientada tanto a profesionales como a usuarios sin conocimientos técnicos. La arquitectura modular, el diseño orientado a objetos y la implementación de pruebas unitarias garantizan la mantenibilidad y escalabilidad del sistema.

Como líneas futuras, se plantean mejoras como la adquisición de audio en tiempo real, el despliegue multiplataforma, el seguimiento longitudinal de pacientes y la incorporación de técnicas de explicabilidad del modelo. Estas extensiones permitirían convertir la herramienta en un sistema aún más completo, transparente y útil para el apoyo al diagnóstico precoz del Parkinson.

En definitiva, este trabajo no solo ha permitido aplicar conocimientos adquiridos durante el grado en un caso práctico de alto impacto social, sino que también abre la puerta a futuras investigaciones y desarrollos en el ámbito del análisis de voz y la inteligencia artificial aplicada a la salud.

Bibliografía

[1] Lerina Aversano, Mario Luca Bernardi, Marta Cimitile, Martina Iammarino, Debora Montano, and Chiara Verdone. A machine learning approach for early detection of parkinson's disease using acoustic traces. In *2022 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–8, 2022.

[2] Lonneke ML de Lau and Monique MB Breteler. Epidemiology of parkinson's disease. *The Lancet Neurology*, 5(6):525–535, 2006.

[3] Rania Salah El-Sayed. A hybrid cnn-lstm deep learning model for classification of the parkinson disease. *International Journal of Applied Mathematics*, 53(4), 2023.

[4] Yunpeng Li, Marco Tagliasacchi, Oleg Rybakov, Victor Ungureanu, and Dominik Roblek. Real-time speech frequency bandwidth extension. In *ICASSP 2021 - IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

[5] Max Little, Patrick Mcsharry, Eric Hunter, Jennifer Spielman, and Lorraine Ramig. Suitability of dysphonia measurements for telemonitoring of parkinson's disease. *IEEE transactions on biomedical engineering*, 56(4):1015–1022, 2009.

[6] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128:336–359, 2020.

[7] Sabine Skodda, Uwe Schlegel, et al. Progression of dysprosody in parkinson's disease over time—a longitudinal study. *Movement Disorders*, 26(14):2423–2426, 2011.

[8] Ian Sommerville. *Software Engineering*. Pearson, 10 edition, 2020.

[9] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[10] M. Toro, J. Vargas, and D. Ruiz. Detección de parkinson mediante análisis de señales de voz usando técnicas de machine learning. *Ingeniería Solidaria*, 15(2):1–13, 2019.

[11] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833. Springer, 2014.