



Universidad de Valladolid

**Escuela Técnica Superior de Ingenieros de
Telecomunicación**

Trabajo de Fin de Grado

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

**ANÁLISIS DE TOS EN ENTORNOS RUIDOSOS
MEDIANTE TRANSFORMERS**

Autor: Juan
Quintanilla Bernabé

Tutores:
D. Juan Pablo Casaseca de la Higuera
D. Luis Miguel San José Revuelta

Valladolid, Mayo 2025

Agradecimientos

aaaaaa

Índice

Agradecimientos	I
Índice	II
Índice de Figuras	VI
Índice de Tablas	VII
Índice de Códigos	VIII
1 Introducción	1
1.1 Descripción del problema	1
1.2 Objetivos	1
1.3 Fases y métodos	2
1.4 Medios necesarios	3
1.5 Estructura del Documento	4
2 Estado del Arte	4
3 Marco Teórico	6
3.1 Inteligencia Artificial	6
3.2 Machine Learning	8
3.3 Deep Learning	9
3.3.1 Fundamentos del aprendizaje profundo	9
3.3.2 Bases matemáticas del Deep Learning	10
3.3.3 Optimización y estructura de redes neuronales	12
3.3.4 Arquitecturas redes neuronales profundas	14
3.3.5 Redes Convolucionales	16
4 Transformers	18
4.1 Introducción a los Transformers	18
4.2 Arquitectura general del Transformer	19
4.3 Mecanismos principales	21
4.3.1 Justificación Matemática	24
4.3.2 Requisitos dataset y eficiencia	25
4.4 Variantes de arquitectura	26
5 Metodología	27
5.1 Recursos hardware, software y librerías	27
5.2 Descripción del conjunto de datos	28
5.2.1 Grabaciones originales y desbalance de clases	29
5.2.2 Estadísticas generales del corpus	29
5.3 Preprocesamiento de audio	30
5.3.1 Normalización y recorte	30
5.4 Justificación ViT	31

5.5	Obtención de espectrogramas	31
5.6	Data Augmentation	32
5.7	Pitch Scaling	33
5.8	Noise Injection	34
5.9	Volume Scaling	36
5.10	Convolución con Respuesta al Impulso (IR)	37
5.11	Muestreo Circular	38
5.12	Codificación Posicional Aprendida	40
5.13	Resumen de parámetros y justificación técnica	40
5.14	División en conjuntos y validación	40
5.15	Fine-Tunning	44
5.16	Descripción de las métricas	45
5.17	Parámetros y pruebas realizadas	47
6	Implementación	47
6.1	Entrenamiento	47
6.2	Optimización de hiperparámetros con Hyperband	48
6.3	Arquitectura propuesta	50
6.4	Modelo CNN de referencia para comparación	53
7	Resultados	54
7.1	Detección	55
7.2	Impacto de las técnicas de <i>data augmentation</i>	55
7.3	Detección sobre el conjunto de datos completo	58
7.3.1	Curvas de pérdida y precisión	58
7.3.2	Rendimiento medio en validación cruzada	59
7.3.3	Análisis del sobreajuste	59
7.4	Comparativa tras el ajuste de parámetros	59
7.4.1	Curvas de pérdida y precisión	60
7.4.2	Rendimiento medio en validación cruzada	60
7.4.3	Análisis del sobreajuste	61
7.4.4	Comparativa Transformer vs CNN base	61
7.5	Clasificación	62
7.6	Descripción médica de clases clínicas	63
7.6.1	Aproximación al problema con enfrentamiento uno frente a uno	63
7.6.2	Resultados completos por pares, métricas prioritarias y riesgo de sobreajuste	64
7.6.3	Adaptación del modelo al problema multiclas	68
7.6.4	Comparativa Transformer vs CNN base	70
8	Conclusiones	70
8.1	Relevancia del Data Augmentation	70
8.2	Limitaciones del estudio	70
8.3	Aplicabilidad médica	70

9	Futuras líneas de trabajo	70
A	Anexos	71

Índice de Figuras

1	Sistema de Expertos MYCIN	5
2	Línea Temporal de los hitos de la Inteligencia Artificial	7
3	Paradigmas de la programación	8
4	Ejemplo de representaciones aprendidas por un clasificador de dígitos [14]	9
5	Rosenblatt perceptrón	10
6	Representación gráfica de las funciones de activación	11
7	Representación gráfica de las capas de una red neuronal	11
8	Superficie error-rendimiento y trayectoria del gradiente con el paso de las iteraciones	13
9	Esquema de una red neuronal con retropropagación y actualización de pesos.	13
10	Simulación del descenso de gradeinte dónde no alcanza el mínimo local	14
11	Arquitectura de red neuronal recurrente	15
12	Funcionamiento de un autoencoder: el dato de entrada se comprime en una representación latente y luego se reconstruye.	15
13	Ejemplo de la operación de convolución 2D [4]	16
14	Aplicación de diferentes filtros de convolución sobre un espectrograma de tos	17
15	Ejemplo de Average Pooling [25]	18
16	Arquitectura del Transformer propuesta por Vaswani	20
17	Ejemplo de codificación sinusoidal	22
18	Heatmap de la codificación posicional sinusoidal empleado en los Transformers	23
19	Arquitectura del ViT propuesta por Dosotovitsky [20]	27
20	Flujo de Trabajo	27
21	Fragmento con tos	32
22	Fragmento sin tos	32
23	Comparativa entre el espectrograma original y tras aplicar el desplazamiento en frecuencia	34
24	Comparativa entre el waveform original y tras aplicar desplazamiento en frecuencia	34
25	Comparativa de espectrograma antes de la adición de ruido y después	35
26	Comparativa del waveform antes de la adición de ruido y después	36
27	Escalado de el audio con constante aleatoria	36
28	Comparativa espectrograma original vs tras aplicarse la convolución	38
29	Comparativa del waveform original vs tras aplicarse la convolución	38
30	Espectrograma antes y tras aplicar el muestreo circular.	39
31	Waveform antes y tras aplicar el muestreo circular	39
32	Representación de los subconjuntos de entrenamiento, test y validación respecto a el dataset completo	41

33	Ejemplo de implementación de la validación cruzada por k-folds	42
34	Red neuronal estándar antes y después de aplicar descartes aleatorios (dropout)	43
35	Ejemplo de simulación de convergencia del gradiente según la tasa de aprendizaje <i>gamma</i>	44
36	Matriz de confusión para el caso de clasificación binaria	46
37	Estrategias de búsqueda de hiperparámetros. Izquierda: <i>grid search</i> . Centro: <i>random search</i> . Derecha: selección adaptativa tipo Successive Halving/Hyperband, que concentra recursos en regiones prometedoras del espacio.	48
38	División en parches 5x5 de espectrograma de dimensiones 45x100	51
39	Bloque codificador Transformer con atención multicabeza ($h = 2$), normalización, red feed forward intermedia de 384 y conexión residual. Basado en [80].	51
40	Curvas de entrenamiento con AdamW [48]. Se muestra la evolución de la pérdida y la precisión en entrenamiento y validación	52
41	Arquitectura de la red convolucional empleada	53
42	Entrenamiento sin datos sintéticos	56
43	Augmentación con desplazamiento de tono	56
44	Augmentación con volume scaling	57
45	Augmentación con convolución con respuesta al impulso	57
46	Curvas de error Test-Val antes de el ajuste de hiperparámetros	58
47	Curvas de error Test-Val tras de el ajuste de hiperparámetros	60
48	Evolución del error en los conjuntos de test y validación. Epoc vs Asma	65
49	Evolución del error en los conjuntos de test y validación. Ag vs Asma	66
50	Evolución del error en los conjuntos de test y validación. Ag vs Cáncer	66
51	Evolución del error en los conjuntos de test y validación. Epoc vs Asma	67
52	Evolución del error en los conjuntos de test y validación. Epoc vs Cancer	68
53	Evolución del error en los conjuntos de test y validación. Cancer vs Asma	68

Índice de Tablas

1	Comparativa entre CNN y Vision Transformers en eficiencia de datos, sesgos y necesidad de preentrenamiento.	25
2	Librerías utilizadas y su funcionalidad principal	28
3	Resumen del conjunto de datos de los utilizado. Se indica el número de muestras por clase y la duración total.	29
4	Hiperparámetros finales empleados en el modelo Transformer para clasificación de los	50
5	Resumen de la arquitectura Transformer propuesta.	52
6	Resumen de la arquitectura CNN de referencia.	54
7	Resultados del dataset original (sin augmentación).	55
8	Resultados con augmentación: <i>Noise Injection</i>	55
9	Resultados con augmentación: <i>Pitch Scaling</i>	56
10	Resultados con augmentación: <i>Primer intento</i>	59
11	Resultados con augmentación: <i>Pitch Scaling</i>	60
12	Resultados enfrentamiento E vs Ag.	65
13	Resultados enfrentamiento Ag vs As.	65
14	Resultados enfrentamiento Ag vs C.	66
15	Resultados enfrentamiento E vs As.	67
16	Resultados enfrentamiento E vs C.	67
17	Resultados enfrentamiento C vs As.	68

Índice de Códigos

1 Introducción

1.1 Descripción del problema

Detectar y clasificar automáticamente episodios de tos dentro de una señal de audio no es una tarea trivial. Las grabaciones de este tipo presentan una gran variabilidad: cada individuo tose de forma distinta, y además, en condiciones reales aparecen ruidos de fondo que pueden enmascarar el sonido. Este trabajo de fin de grado se centra precisamente en ese reto. La idea es desarrollar y validar un sistema basado en arquitecturas de tipo transformer capaz de reconocer de manera fiable los episodios de tos, incluso cuando la señal proviene de entornos clínicos donde el ruido es inevitable.

El punto de partida es un conjunto de datos recopilado en hospitales de Valladolid y Palencia. Reúne unas 14.700 grabaciones de tos y cerca de 150.000 fragmentos de otros sonidos no relacionados. Este fuerte desbalanceo entre clases supone un problema añadido: si el modelo se entrena sin precauciones, tenderá a priorizar los ejemplos mayoritarios y perderá precisión en la detección de la tos, justo la clase más importante.

Para superar estas limitaciones se ha optado por los transformers, un tipo de red neuronal eficaz en tareas secuenciales. Frente a modelos más tradicionales como las redes recurrentes o convolucionales, los transformers son capaces de capturar dependencias a largo plazo en la señal y han demostrado resultados superiores en reconocimiento del habla, clasificación de eventos acústicos y también en visión por ordenador. Estas evidencias apoyan su elección como base para el sistema propuesto.

Ahora bien, el buen rendimiento de este tipo de arquitecturas no depende solo del modelo, sino, en mayor medida de los datos de entrenamiento. En este contexto, resulta clave aplicar técnicas de "data augmentation". Mediante perturbaciones controladas como la adición de ruido, cambios en el tono o simulación de entornos acústicos distintos se consigue ampliar artificialmente el conjunto original. Estudios recientes muestran que estas técnicas mejoran de forma notable la robustez de los modelos y reducen el riesgo de sobreajuste, lo que en la práctica se traduce en un sistema que generaliza mejor.

El objetivo final de este proyecto es, por tanto, diseñar y poner a prueba un modelo transformer capaz de identificar y clasificar episodios de tos en condiciones ruidosas. La aspiración es que este sistema pueda convertirse en una herramienta útil tanto en la práctica clínica como en la monitorización remota de pacientes, contribuyendo a un diagnóstico temprano y a la diferenciación de enfermedades respiratorias a partir del análisis automático de patrones acústicos.

1.2 Objetivos

Este Trabajo de Fin de Grado busca desarrollar y poner a prueba un sistema capaz de identificar y clasificar de manera automática episodios de tos en graba-

ciones de audio con ruido de fondo. La idea no es únicamente mejorar la precisión en la detección bajo condiciones realistas, sino también acercar este tipo de soluciones a escenarios clínicos y a contextos de monitorización a distancia, donde su aplicación práctica puede resultar especialmente valiosa.

De forma más concreta, el proyecto persigue varios objetivos. El primero es diseñar un modelo de clasificación de tos que emplee arquitecturas de tipo transformer, conocidas por su capacidad para captar relaciones a largo plazo en datos secuenciales. Este enfoque se pondrá en paralelo con los resultados ya obtenidos en el laboratorio con redes convolucionales, de manera que la comparación permita valorar de forma objetiva qué ventajas y limitaciones ofrece cada aproximación.

El segundo objetivo está relacionado con las técnicas de aumento de datos aplicadas a señales de audio. Estas transformaciones permiten generar ejemplos adicionales a partir de las grabaciones originales, lo que ayuda a compensar la escasez de datos etiquetados y el fuerte desequilibrio entre clases. Se evaluará hasta qué punto estas técnicas contribuyen a que el modelo generalice mejor y mantenga un buen rendimiento en situaciones nuevas.

El tercer aspecto a estudiar es la influencia del tamaño del conjunto de entrenamiento en el rendimiento de los modelos. Se pretende observar cómo varía el comportamiento de redes convolucionales y transformers cuando se dispone de más o menos datos, lo que permitirá establecer recomendaciones sobre su uso en contextos donde el volumen de ejemplos es limitado.

Por último, el sistema se evaluará con métricas habituales en el ámbito del reconocimiento de eventos acústicos, como la sensibilidad, la especificidad, la precisión o el área bajo la curva ROC. Estas medidas servirán para cuantificar el grado de acierto en condiciones ruidosas y, en conjunto, mostrar el potencial de la propuesta como herramienta de apoyo al diagnóstico temprano y a la detección automática de patologías respiratorias.

1.3 Fases y métodos

El trabajo se fue desarrollando en varias etapas consecutivas, desde la preparación inicial de los datos hasta la validación de los resultados finales. En primer lugar, se reunió y procesó el conjunto de grabaciones, que incluía tanto episodios de tos como fragmentos de audio sin relevancia clínica. Para aumentar la variabilidad de las señales y aproximarse mejor a entornos reales, se aplicaron distintas técnicas de data augmentation que permitieron enriquecer el material disponible sin necesidad de nuevas campañas de grabación.

La siguiente etapa consistió en el diseño de la arquitectura. Partiendo de los modelos convolucionales que ya se habían utilizado en el laboratorio, se construyó una versión adaptada basada en transformers, pensada para trabajar sobre espectrogramas de audio. En un primer momento se ensayó la detección binaria (tos frente a no tos) y, una vez validada esta aproximación, se extendió el mismo

esquema metodológico a la clasificación multiclase. [22, 59, 41]

En la tercera fase se entrenaron los modelos y se evaluó su desempeño. Para ello se empleó validación cruzada con particiones definidas a nivel de paciente, de manera que las grabaciones de un mismo individuo no aparecieran simultáneamente en entrenamiento y prueba. Este criterio redujo el riesgo de sobreajuste y aportó una estimación más realista de la capacidad de generalización del sistema.

Por último, los resultados de la arquitectura basada en transformers se pusieron en relación con los obtenidos previamente mediante redes convolucionales. Esta comparación permitió valorar de manera directa las ventajas y limitaciones de cada enfoque, así como medir con más detalle la influencia de las técnicas de aumento de datos en el rendimiento final.

1.4 Medios necesarios

El proyecto necesitó una mezcla de recursos de hardware, software y librerías particulares que permitieron el procesamiento de grandes cantidades de datos acústicos y la capacitación de redes neuronales profundas.

Los experimentos se realizaron en los servidores Tanis, Isis y Tebas. Todos ellos contaban con procesadores Intel Xeon, GPU NVIDIA con soporte CUDA y Linux como sistema operativo. El empleo de GPU con 16 GB de memoria fue particularmente relevante porque posibilitó que los cálculos matriciales se realizaran mucho más rápido y que se entrenaran modelos con decenas de miles de espectrogramas en plazos apropiados. Asimismo, la memoria RAM y la capacidad de almacenamiento de estas máquinas fueron suficientes para gestionar el conjunto original de grabaciones y las versiones ampliadas.

La implementación se realizó íntegramente en Python, utilizando entornos virtuales para mantener aisladas las dependencias. Se recurrió a librerías de referencia en el campo del aprendizaje profundo y del procesado de audio: PyTorch y TensorFlow para la construcción y entrenamiento de redes, torchaudio y librosa para la manipulación de señales y la creación de espectrogramas. A estas se añadieron NumPy y SciPy para el cálculo numérico, scikit-learn para la validación cruzada y las métricas, y SoundFile para la lectura y escritura de archivos en formato WAV.

El conjunto de datos estaba compuesto por grabaciones clínicas de tos y fragmentos de control sin episodios relevantes, organizados cuidadosamente por paciente. A partir de este material se generaron nuevas versiones mediante técnicas de aumento de datos como inyección de ruido, escalado de volumen, modificaciones de tono o convolución con respuestas al impulso de distintos espacios, lo que permitió dotar al corpus de mayor diversidad y robustez.

Para el análisis de los resultados se emplearon herramientas de visualización que facilitaron tanto la interpretación como el seguimiento del entrenamiento. Matplotlib se utilizó en la representación de espectrogramas y gráficas, mientras

que TensorBoard permitió monitorizar en detalle la evolución de las métricas a lo largo de los experimentos.

1.5 Estructura del Documento

El trabajo se ha organizado en distintas etapas que marcan un recorrido progresivo. La primera parte cumple una función de introducción : en ella se presentan los objetivos, se expone el problema y se revisa el estado del arte, lo que permite situar el proyecto dentro del contexto actual de la investigación en procesamiento de audio y aprendizaje profundo.

A continuación, se incluyen los capítulos dedicados al marco teórico. En esta sección se repasan los conceptos esenciales de inteligencia artificial y aprendizaje automático, prestando especial atención a las redes neuronales y, de forma más concreta, a los transformers, que constituyen la base del enfoque adoptado.

La tercera parte reúne la metodología y la implementación. Aquí se explican los recursos de hardware y software utilizados, las características del conjunto de datos, los pasos de preprocesamiento y, finalmente, la construcción del modelo junto con las pruebas de validación necesarias.

El documento se cierra con los capítulos de resultados y conclusiones. En ellos se analizan los experimentos realizados, se discuten sus implicaciones y se plantean posibles líneas de trabajo para continuar la investigación. Los anexos, por último, ofrecen material adicional que sirve de apoyo a la parte práctica y complementa la memoria principal.

2 Estado del Arte

El recorrido de la inteligencia artificial en la medicina comenzó en los años setenta con los primeros sistemas expertos. Un ejemplo emblemático fue MYCIN [70], desarrollado para asistir en el diagnóstico automático. Aunque nunca llegó a utilizarse de forma rutinaria en hospitales, sí sirvió para demostrar el potencial de estas tecnologías y abrió el camino a investigaciones posteriores. Durante los ochenta y noventa, el aumento de la capacidad de cómputo permitió experimentar con redes neuronales y enfoques bayesianos, que ampliaron las posibilidades de aplicación en entornos clínicos. Sin embargo, ha sido en la última década cuando se ha producido un salto cualitativo, el aprendizaje profundo ha dado lugar a sistemas capaces de analizar imágenes médicas, textos clínicos o grandes volúmenes de datos de pacientes con un nivel de precisión difícil de imaginar en etapas anteriores.

Hoy en día encontramos ejemplos muy concretos de esta evolución. Se han desarrollado modelos que identifican neumonía en radiografías con una precisión comparable a la de un radiólogo [65]; otros trabajan sobre señales fisiológicas como los ECG o los EEG, y no faltan aplicaciones destinadas a la monitorización remota de pacientes. Aun así, no todo son ventajas: la escasez de bases de

MYCIN Expert System

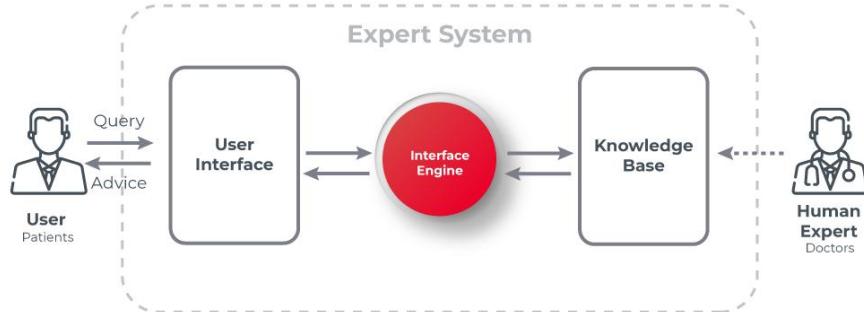


Figura 1: Sistema de Expertos MYCIN

datos suficientemente grandes y bien anotadas sigue siendo un problema serio. Además, muchos algoritmos funcionan como auténticas “cajas negras” [18], lo que dificulta la interpretación de sus resultados en un contexto clínico real. A esto se añaden la necesidad de validaciones rigurosas y el debate ético y regulatorio que inevitablemente acompaña al uso de información sanitaria sensible [76].

En el ámbito de la visión por computador, las redes convolucionales marcaron un antes y un después. A partir de 2012, con el trabajo de Krizhevsky y colaboradores [44], estas arquitecturas se consolidaron gracias a su capacidad para extraer rasgos visuales de manera jerárquica. Desde entonces se han utilizado con éxito en radiología [23], dermatología o patología digital, alcanzando en algunos casos resultados similares a los de especialistas humanos y con una eficiencia notable frente a las redes densas tradicionales. Un paso más allá llegó con los *Vision Transformers* (ViT), introducidos por Dosovitskiy et al. [20]. En lugar de trabajar con filtros locales, dividen las imágenes en parches tratados como secuencias, lo que permite a los mecanismos de auto-atención captar relaciones globales desde el inicio. Al principio necesitaban enormes bases de datos para rendir al máximo, pero propuestas como DeiT [78] demostraron que, con técnicas adecuadas de regularización y aumento de datos, podían entrenarse de forma mucho más eficiente. En medicina ya se han ensayado variantes aplicadas a resonancias magnéticas o retinografías, con resultados alentadores [12].

También el audio ha ganado fuerza como fuente de información clínica. La tos, voz o respiración son biomarcadores accesibles y no invasivos. Inicialmente se emplearon descriptores manuales junto con clasificadores como las SVM (Sup-

ported Vector Machines). Tras el uso generalizado del aprendizaje profundo en otros campos, fueron las redes convolucionales basadas en espectrogramas aquellas que lograron la detección de tos de forma precisa [53]. Recientemente, los transformers han supuesto grandes avances en el análisis de imagen médica [26], con el AST (Audio Spectrogram Transformer) destacando en la mayoría de las métricas relevantes.

El interés por la clasificación automática de la tos se disparó durante la pandemia de COVID-19 y, desde entonces, ha ido calando también en el estudio de patologías respiratorias crónicas. Hoy ya contamos con repositorios abiertos como COUGHVID [56] y, además, con colecciones clínicas más acotadas, como el corpus recogido en los hospitales de Valladolid y Palencia [38, 75]. Estos conjuntos permiten entrenar y validar modelos de detección; ahora bien, se ha visto que las CNN fallan al generalizar cuando cambia la población o el contexto de registro. En los últimos años, equipos de la Universidad de Valladolid han empezado a explorar aprendizaje adaptativo y técnicas de interpretabilidad para paliar esas limitaciones [5]. En paralelo, los transformers aplicados a espectrogramas han abierto un frente nuevo: más robustez y una mejor capacidad para capturar dependencias temporales amplias.

En ese marco se sitúa este Trabajo Fin de Grado. Comparamos, de forma sistemática, el rendimiento de transformers y redes convolucionales en la clasificación de episodios de tos bajo condiciones ruidosas. Para compensar la escasez de grandes bases clínicas, se recurre a un uso intensivo, y controlado, del aumento de datos [78]. La contribución central es un flujo experimental sólido y reproducible, que permite perfilar puntos fuertes y flaquezas de cada arquitectura y, en consecuencia, aportar evidencias sobre su viabilidad en escenarios reales de monitorización médica.

3 Marco Teórico

3.1 Inteligencia Artificial

La inteligencia artificial puede entenderse, en su definición más clásica, como el área de la informática que busca crear sistemas capaces de llevar a cabo tareas que, realizadas por personas, exigirían inteligencia. Esta visión quedó reflejada en la propuesta de la Conferencia de Dartmouth de 1956 [50], considerada de manera general como el punto de partida formal del campo. Con el paso del tiempo, la idea ha evolucionado: hoy se habla de IA como la disciplina que trata de dotar a las máquinas de la capacidad de aprender a partir de datos, adaptarse a nuevas situaciones y tomar decisiones sin que sea necesario programarlas explícitamente para cada caso.

Los objetivos de la IA han sido amplios desde sus orígenes. A lo largo de su desarrollo se han producido hitos que marcan etapas muy distintas: el Test de Turing en 1950 [79], la propia Conferencia de Dartmouth en 1956, la expansión de los sistemas expertos en los años ochenta, la victoria de Deep Blue frente

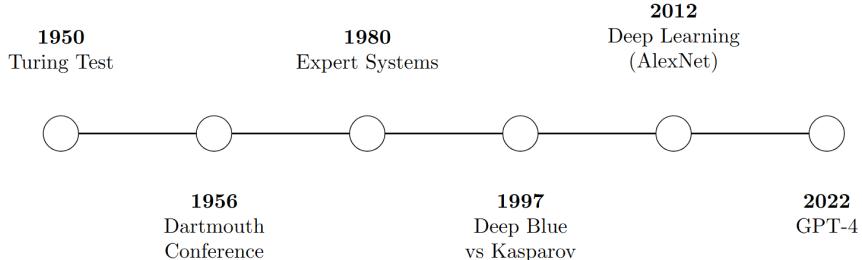


Figura 2: Linea Temporal de los hitos de la Inteligencia Artificial

a Kasparov en 1997 [10] o el resurgir de las redes profundas con AlexNet en 2012 [44]. Más recientemente, la llegada de los grandes modelos generativos ha vuelto a situar a la IA en el centro del debate tecnológico y social. Estos hitos se recogen en la línea del tiempo que acompaña este apartado y que resume visualmente la trayectoria del campo. 2

La evolución de la disciplina se suele dividir en dos grandes corrientes. Por un lado, el enfoque simbólico o basado en reglas, predominante entre los años cincuenta y ochenta, que trataba de representar el conocimiento mediante lógicas formales, árboles de decisión o sistemas expertos. En paralelo apareció la corriente conexionista, inspirada en el funcionamiento del cerebro humano, que dio lugar a las primeras redes neuronales. Estas tuvieron un desarrollo limitado durante décadas, pero a partir de 2010 se consolidaron gracias al aumento de la capacidad de cómputo y la disponibilidad de grandes volúmenes de datos, dando forma a lo que hoy denominamos aprendizaje profundo [14].

La diferencia con la programación tradicional ayuda a entender el alcance de este cambio. Mientras que en el enfoque clásico el programador define un conjunto de reglas que, aplicadas a los datos, producen las respuestas, en el aprendizaje automático se parte de datos y ejemplos de salida y es el propio sistema el que aprende las reglas que relacionan ambos. En algunos casos basta incluso con datos y una función de evaluación. Este giro metodológico explica el impacto que ha tenido la IA en problemas donde escribir las reglas de forma manual resulta ineficiente o directamente inviable. ??

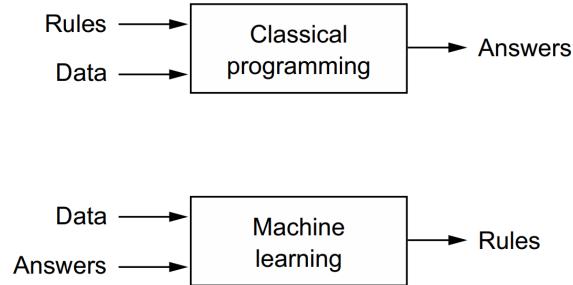


Figura 3: Paradigmas de la programación

3.2 Machine Learning

El aprendizaje automático se ha convertido en uno de los pilares centrales de la inteligencia artificial. Puede entenderse como el conjunto de técnicas que permiten a un sistema mejorar su desempeño en una tarea concreta a medida que se expone a más datos. Mitchell lo formuló de manera precisa al señalar que un programa “aprende” cuando, a partir de la experiencia, consigue mejorar su rendimiento en una clase de tareas con respecto a una medida de evaluación determinada [52].

Todo algoritmo de aprendizaje automático se construye alrededor de cuatro elementos básicos. El primero es el conjunto de datos, que actúa como fuente de conocimiento inicial. El segundo es el modelo, encargado de aproximar la relación entre entradas y salidas posibles. A ello se añade la función de pérdida, que mide la diferencia entre las predicciones y los valores reales, y, por último, un procedimiento de optimización que ajusta los parámetros del modelo para reducir ese error [28].

Los métodos de aprendizaje suelen agruparse en tres categorías principales. El aprendizaje supervisado, que es el más utilizado, requiere datos etiquetados y se aplica en tareas de regresión y clasificación. Dentro de este grupo se encuentran técnicas como la regresión lineal o logística, los árboles de decisión y las máquinas de vectores soporte. En contraste, el aprendizaje no supervisado trabaja con datos sin etiquetar y busca patrones ocultos o agrupaciones naturales; ejemplos clásicos son k-means, el clustering jerárquico o el análisis de componentes principales. Finalmente, el aprendizaje por refuerzo se centra en la interacción de un agente con un entorno, de manera que aprende estrategias a partir de recompensas y penalizaciones. Métodos como Q-learning, SARSA o, más recientemente, los algoritmos de deep reinforcement learning son representativos de esta línea [7, 32, 73].

El aprendizaje profundo puede considerarse una extensión de este marco general. Su rasgo distintivo es el uso de redes neuronales profundas capaces de extraer

representaciones jerárquicas de los datos. Este enfoque se detalla con mayor profundidad en el apartado siguiente.

3.3 Deep Learning

3.3.1 Fundamentos del aprendizaje profundo

El aprendizaje profundo puede considerarse una rama dentro del aprendizaje automático en el que se emplean redes neuronales con varias capas de procesamiento. A diferencia de muchos algoritmos clásicos, que se necesita seleccionar y definir manualmente las características relevantes, las redes profundas son capaces de descubrir estas representaciones complejas a partir de los datos. En la práctica, las primeras capas suelen captar patrones sencillos (por ejemplo, bordes en una imagen o picos de energía en una señal de audio), y conforme se avanza en la red se van combinando hasta formar estructuras más complejas y abstractas.⁴

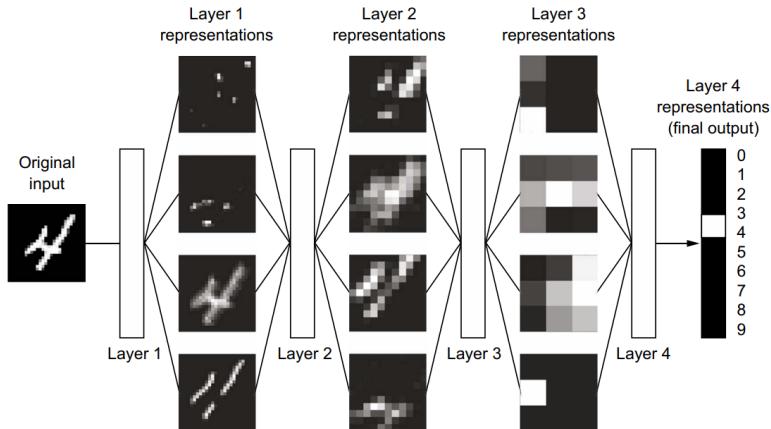


Figura 4: Ejemplo de representaciones aprendidas por un clasificador de dígitos [14]

Los primeros intentos de modelar este tipo de redes aparecieron en los años cuarenta y cincuenta, con trabajos como el de McCulloch y Pitts (1943) [51], que propuso un modelo lógico de neurona artificial, o el perceptrón de Rosenblatt (1958) [67], considerado el primer clasificador entrenable. El gran paso adelante llegó con la retropropagación del error, introducida por Werbos en 1974 [83], que permitió ajustar de manera eficiente los parámetros de las redes multicapa y abrió el camino a las arquitecturas modernas. A partir de 2010, el campo experimentó un auge sin precedentes gracias al acceso a grandes volúmenes de datos y al aumento de la potencia de cálculo.

El empleo de redes profundas presenta claras ventajas frente a otros enfoques.

Se adaptan bien a datos de gran dimensionalidad, como imágenes, texto o grabaciones de audio, y han alcanzado resultados sobresalientes en áreas como la visión por computador, el procesamiento del lenguaje natural o el análisis de señales. Su capacidad de generalización, derivada de la forma en que aprenden representaciones jerárquicas, explica su éxito en ámbitos muy distintos. Sin embargo, no están exentas de problemas. Esta arquitectura requiere conjuntos de datos amplios y etiquetados, su entrenamiento demanda un gran esfuerzo computacional y, además, actúan como cajas negras, dificultando su interpretabilidad y limitando su uso en entornos donde la transparencia es un requisito esencial.

3.3.2 Bases matemáticas del Deep Learning

El elemento más sencillo sobre el que se construyen las redes neuronales es el *perceptrón*. Un perceptrón recibe varias entradas, cada una asociada a un peso, suma esas entradas y añade un término de sesgo. El resultado pasa después por una función de activación que decide la salida final. De forma matemática puede escribirse como:

$$z = \sum_{i=1}^n w_i x_i + b, \quad \hat{y} = \sigma(z),$$

donde x_i son las entradas, w_i los pesos, b el sesgo y $\sigma(\cdot)$ la función de activación [67]. Visualmente se representa como en la figura adjunta 5

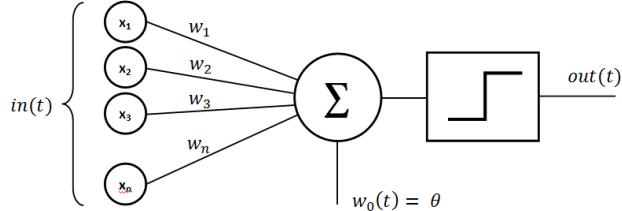


Figura 5: Rosenblatt perceptron

Las funciones de activación son fundamentales porque introducen no linealidad en el sistema. Sin ellas, una red de muchas capas se reduciría a una sola transformación lineal. Algunos ejemplos habituales son la función sigmoide, ReLU (Rectified Linear Unit) y tangente hiperbólica.⁶

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \text{ReLU}(x) = \max(0, x) \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

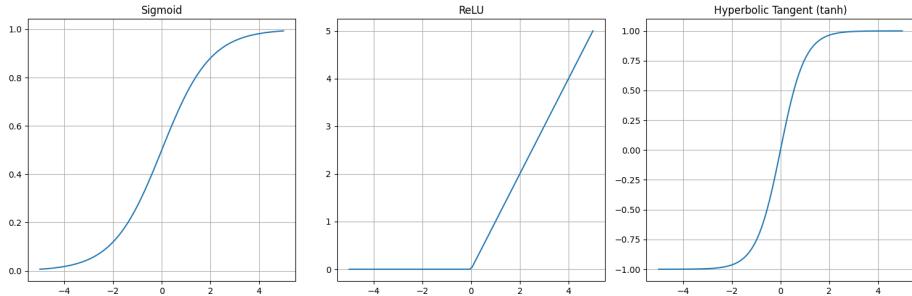


Figura 6: Representación gráfica de las funciones de activación

Cuando se conectan varios perceptrones en paralelo se forma una capa. La primera capa recibe los datos originales y la última capa genera la salida del modelo. Entre ambas pueden existir varias capas ocultas, que van transformando progresivamente la información. Si denotamos por $\mathbf{h}^{(l)}$ la salida de la capa l , se cumple

$$\mathbf{h}^{(l)} = \sigma\left(W^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right),$$

con $\mathbf{h}^{(0)} = \mathbf{x}$ como vector de entrada y $W^{(l)}, \mathbf{b}^{(l)}$ los parámetros de la capa.

Una red neuronal profunda no es más que la composición de varias de estas transformaciones.⁷ De hecho, existe un resultado teórico que afirma que un perceptrón multicapa con al menos una capa oculta y funciones de activación adecuadas puede aproximar cualquier función continua con la precisión deseada. Esto no significa que siempre encontremos la solución óptima, pero sí garantiza que la red tiene capacidad suficiente para representar problemas muy complejos. [16]

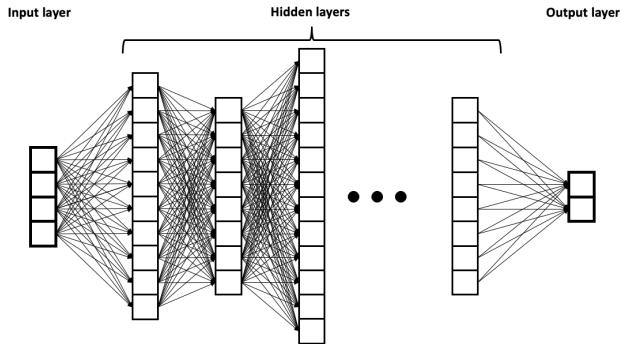


Figura 7: Representación gráfica de las capas de una red neuronal

Finalmente se comentará la diferencia entre anchura y profundidad. Una red muy ancha puede resolver problemas complejos, pero hacerlo con varias capas suele ser más eficiente: la profundidad permite crear representaciones jerárquicas y reutilizar características aprendidas en niveles previos.

3.3.3 Optimización y estructura de redes neuronales

El entrenamiento de una red neuronal se basa en ajustar sus parámetros para que las predicciones que genera se acerquen lo máximo posible a los valores reales. Para medir esa diferencia se define una *función de pérdida*, denotada como $L(\hat{y}, y)$, donde \hat{y} es la salida del modelo y y la etiqueta verdadera. Según la naturaleza del problema se emplean funciones distintas: en regresión suele usarse el error cuadrático medio (MSE), mientras que en clasificación se recurre a la entropía cruzada o *cross-entropy*, que penaliza con más fuerza los errores de predicción en clases minoritarias [28].

En la práctica, no se optimiza el error sobre toda la distribución de datos, sino una estimación conocida como riesgo empírico. Esta se calcula a partir de un conjunto finito de ejemplos de entrenamiento.

Para llevar a cabo esta minimización se emplean algoritmos basados en gradiente. El descenso de gradiente (GD) calcula la dirección de mayor decrecimiento de la pérdida y ajusta los parámetros en consecuencia. En problemas prácticos se suele utilizar su versión estocástica (SGD), donde las actualizaciones se hacen sobre pequeños conjuntos de datos (*mini-batches*). Esto introduce cierto ruido en el proceso, pero acelera el cálculo y evita atascarse en mínimos locales [7].

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} J(\theta^{(t)}) \quad \theta^{(t+1)} = \theta^{(t)} - \eta \nabla_{\theta} J(\theta^{(t)}; x_i, y_i)$$

- θ : vector de parámetros del modelo.
- η : tasa de aprendizaje (*learning rate*).
- $J(\theta)$: función de coste global.
- $J(\theta; x_i, y_i)$: función de coste evaluada sobre un único ejemplo (x_i, y_i) .

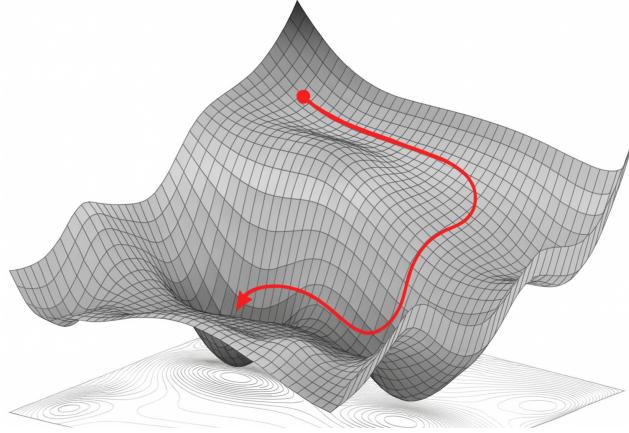


Figura 8: Superficie error-rendimiento y trayectoria del gradiente con el paso de las iteraciones

Para calcular los gradientes [83] Werbos propuso el algoritmo de la retropropagación del error [83]. Consiste en aplicar la regla de la cadena a la composición de funciones que compone la red (desde la capa de salidas hacia la entrada). Obteniéndose el gradiente parcial que indica como modificarse en la siguiente iteración.

$$w \leftarrow w - \eta \frac{\partial L}{\partial w},$$

donde η es la tasa de aprendizaje que controla el tamaño del paso (figura 9).

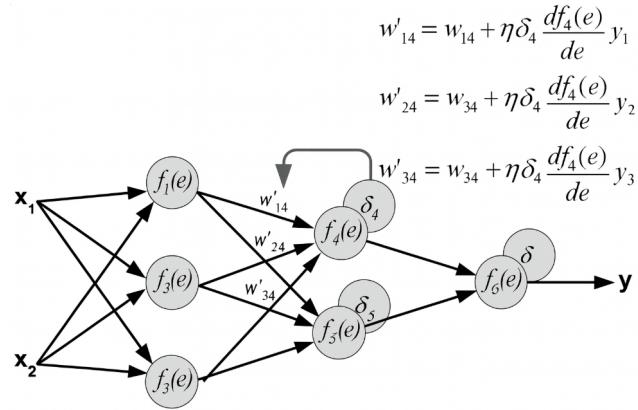


Figura 9: Esquema de una red neuronal con retropropagación y actualización de pesos.

El entrenamiento de redes profundas presenta dificultades (figura 10). Un problema conocido aparece cuando los gradientes, en redes con muchas capas, se van reduciendo hasta casi desaparecer o, al contrario, crecen sin control, lo que impide que el aprendizaje avance con normalidad. A esto se añaden otras complicaciones: la existencia de múltiples mínimos locales en la función de pérdida, o la fuerte dependencia de los resultados respecto a la elección de hiperparámetros como la tasa de aprendizaje. Estas limitaciones han impulsado la creación de métodos de optimización más avanzados que el SGD clásico. Entre ellos destacan algoritmos como Adam o RMSProp.

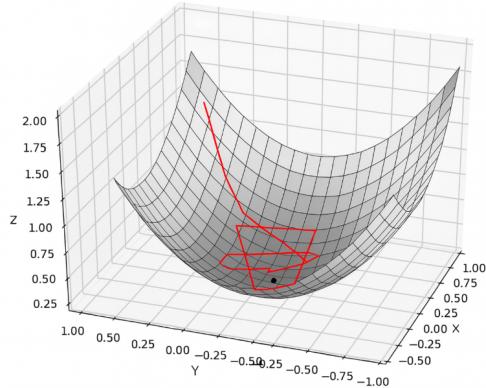


Figura 10: Simulación del descenso de gradeinte dónde no alcanza el mínimo local

3.3.4 Arquitecturas redes neuronales profundas

A lo largo de las últimas décadas se han desarrollado distintos tipos de redes neuronales profundas, cada una adaptada a un tipo de problema concreto. Aunque comparten la misma idea de fondo —una combinación de capas que transforman progresivamente la información de entrada hasta producir una salida—, difieren en la manera en que procesan los datos y en las aplicaciones para las que resultan más eficaces.

- **Perceptrón multicapa (MLP):** punto de partida de las redes profundas. Arquitectura basada en capas totalmente conectadas, donde cada neurona de una capa se conecta con todas las de la siguiente. Funcionan bien con problemas de baja dimensión o datos tabulares, pero no son eficientes con datos que tienen estructura espacial o temporal compleja. [67]
- **Redes recurrentes (RNN):** diseñadas para información secuencial. Útiles para modelar dependencias temporales en series temporales, texto o audio. Problema principal: dificultad para capturar dependencias largas. Destacan las LSTM, aunque han perdido protagonismo frente a los Trans-

formers, más eficientes en el manejo de secuencias largas mediante mecanismos de atención.(figura 11)

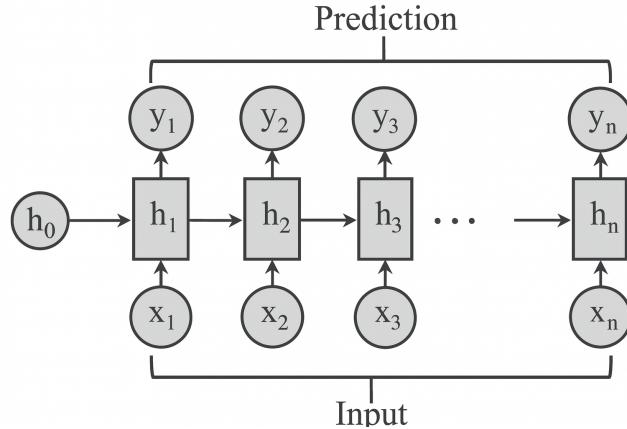


Figura 11: Arquitectura de red neuronal recurrente

- **Autoencoders:** Su objetivo es reducir la dimensionalidad de los datos mediante un codificador, aprender representaciones más simples y reconstruirlas con el decodificador. Son útiles para la detección de anomalías (figura 12)

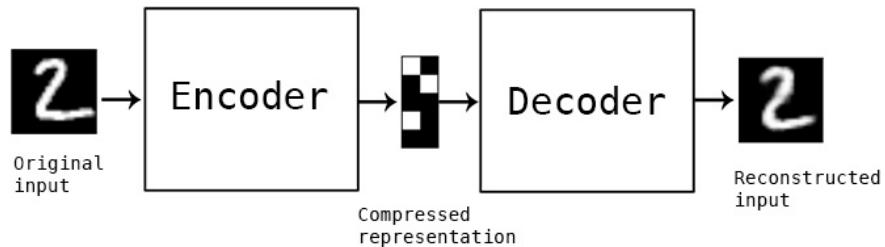


Figura 12: Funcionamiento de un autoencoder: el dato de entrada se comprime en una representación latente y luego se reconstruye.

- **Redes convolucionales (CNN):** estándar en la clasificación de imágenes.[44] Se han tomado como referencia en este proyecto para comparar resultados y métricas con estudios anteriores del LPI.
- **Transformers:** Es la arquitectura elegida. Su capacidad para modelar dependencias de largo alcance los ha convertido en la arquitectura dominante en texto, audio e incluso visión por computador. Se alimentan de grandes volúmenes de datos.[80]

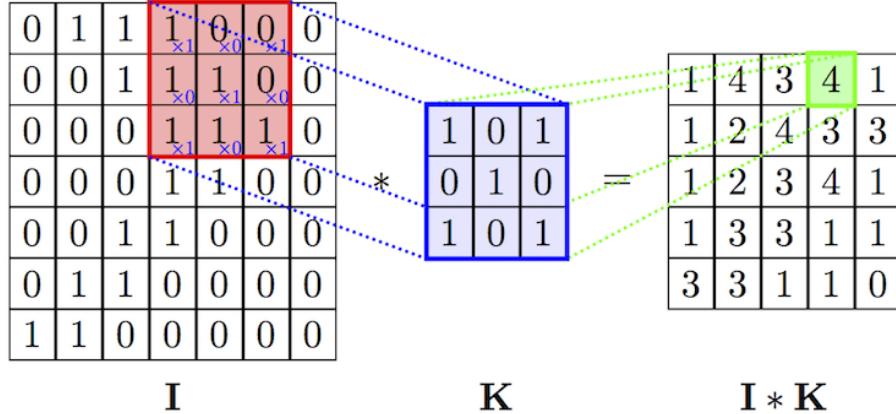


Figura 13: Ejemplo de la operación de convolución 2D [4]

3.3.5 Redes Convolucionales

Las redes convolucionales (CNN) aparecieron con la idea de aprovechar la estructura espacial que tienen los datos en forma de matriz, como las imágenes o los espectrogramas de audio. A diferencia de un perceptrón multicapa, que conecta todas las entradas con todas las neuronas de la siguiente capa, en una CNN cada neurona sólo se conecta con una pequeña región local. De esta manera se reduce drásticamente el número de parámetros y, al mismo tiempo, se gana capacidad para detectar patrones que se repiten en distintas zonas de la señal. [14][44]

El fundamento de las CNN es la operación de convolución, que da nombre a esta arquitectura. Se puede representar como un filtro que se aplica sobre la matriz de entrada, calculando en cada posición una combinación ponderada por los valores de la ventana. Según los valores del filtro servirán para detectar bordes, texturas o cambios en intensidad. Estos parámetros constituyen los pesos de la red aprendidos durante la etapa de entrenamiento. Caracterizada para dos dimensiones se expresa como:

$$S(i, j) = (X * K)(i, j) = \sum_m \sum_n X(i - m, j - n) K(m, n),$$

donde X es la entrada, K el filtro y S la salida o mapa de características.

La operación de convolución es especialmente útil porque presenta una serie de propiedades clave para detectar patrones en imágenes. El hecho de aplicar filtros sobre regiones pequeñas resulta muy eficiente para detección de bordes, cambios de frecuencia, texturas. También, dado que el filtro se aplica en todas las posiciones de la señal, permite reconocer un mismo patrón independientemente de

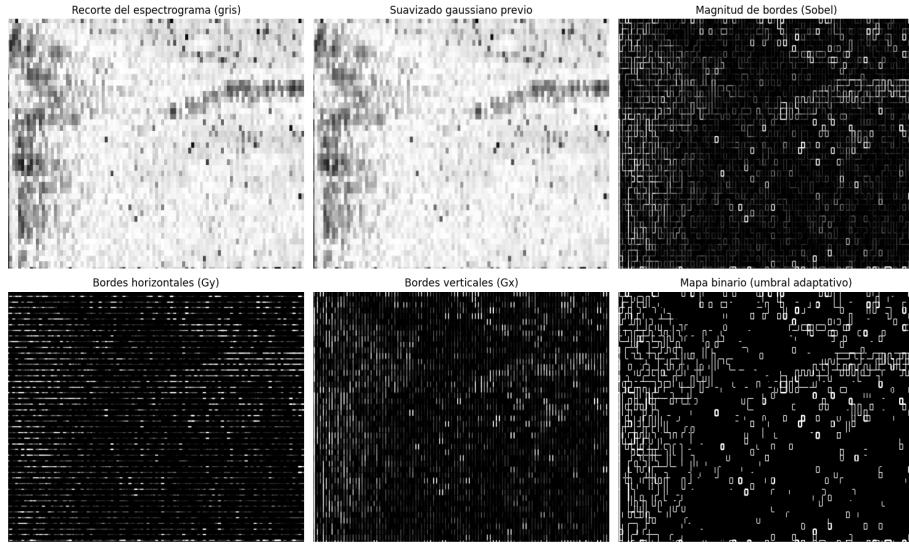


Figura 14: Aplicación de diferentes filtros de convolución sobre un espectrograma de tos

que aparezca en lugares distintos. Finalmente, al no ser redes densamente conectadas, comparten pesos, reduciendo el número de parámetros notablemente, facilitando el entrenamiento en grandes volúmenes de datos y su capacidad de generalización. [14]

Tras la convolución, el resultado es la entrada de una función de activación, que introduce no linealidad. Para simplificar y hacer más manejable la representación, se usan capas de *pooling* [25] 15. Esto aporta cierta invariancia: la red seguirá reconociendo un borde aunque aparezca desplazado algunos píxeles.

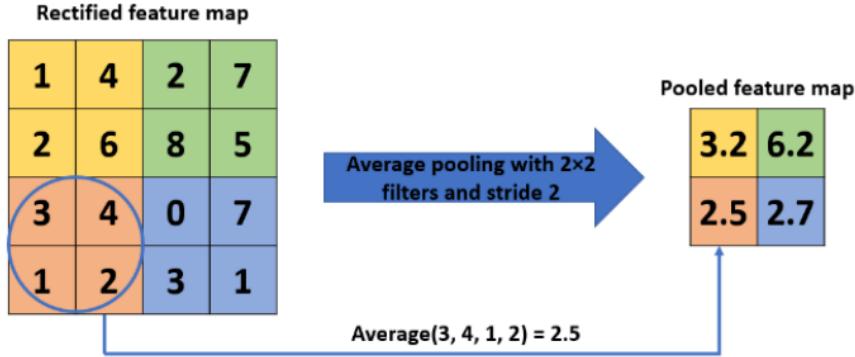


Figura 15: Ejemplo de Average Pooling [25]

Tal y como se señaló anteriormente, lo interesante de las CNN es su carácter jerárquico. Las primeras capas identifican rasgos sencillos (líneas, esquinas, regiones brillantes). En capas más profundas esos elementos se combinan y aparecen estructuras cada vez más ricas, como formas concretas o partes de un objeto. Al final, las últimas capas son capaces de reconocer conceptos de alto nivel de abstracción.

Aunque en este trabajo los Transformers han sido la base del modelo, las CNN siguen siendo una referencia esencial. En problemas como la clasificación de los presentan ventajas claras: necesitan menos datos para entrenarse, su coste computacional es menor y suelen ser más fáciles de interpretar, sobre todo en las capas iniciales donde los filtros tienen un significado visual directo. Por estas razones se han tomado como punto de comparación en los experimentos y se han usado como referencia para las métricas estándar en clasificación de imágenes.

4 Transformers

4.1 Introducción a los Transformers

Hasta 2017, para el procesamiento de secuencias las redes neuronales recurrentes con sus respectivas variantes mencionadas anteriormente (LSTM) [37], constituyan el estado del arte. Sin embargo presentaban dos problemas principales que hacían inviable su escalabilidad a problemas complejos. Las RNN procesaban secuencialmente la entrada, impidiendo paralelizar el procesamiento y acelerar el entrenamiento para tratar con secuencias largas. Por otra parte, sufrían problemas para aprender dependencias de largo alcance debido al desvanecimiento de gradiente.

La propuesta de Vaswani supuso un cambio de paradigma [80]. Reemplazan-

do por completo las conexiones recurrentes y convoluciones por mecanismos de atención. Los obstáculos [80]. La idea clave fue reemplazar la recurrencia por mecanismos de atención, eliminando por completo las conexiones recurrentes y convoluciones. En el Transformer, cada elemento de la secuencia puede interactuar directamente con cualquier otro mediante atención, independientemente de la distancia que los separe. Esto permite capturar de forma más eficaz las relaciones de largo alcance en una frase o secuencia. Al mismo tiempo, al no tener que procesar secuencialmente, el modelo puede paralelizar el procesamiento de todos los tokens de entrada, acelerando significativamente el entrenamiento [80]. De hecho, Vaswani et al. demostraron en traducción automática que un Transformer entrenado en paralelo igualaba o superaba la calidad de modelos recurrentes, con mucho menos tiempo de entrenamiento.

Debido a su éxito, esta arquitectura no se ha limitado a el procesamiento de lenguaje natural. Se adaptó para visión por ordenador (Vision Transformer) [20] , y de forma similar a imágenes análisis de audio (AST: Audio Spectrogram Transformer) [26]. Se ha demostrado la efectividad de los mecanismos de atención independientemente de la naturaleza del problema, obteniendo resultados punteros en clasificación de imágenes [20] y eventos acústicos [26].

4.2 Arquitectura general del Transformer

En la publicación titulada ".^atention is all you need"[80] presenta la siguiente arquitectura 16. Basado en dos partes: un encoder situado a la izquierda, y un decoder en la derecha. Cabe recalcar que en ciertas aplicaciones se emplea únicamente el encoder.

El **encoder** está formado por N bloques idénticos apilados, cada uno con dos componentes: una capa de auto-atención multi-cabeza y una red *feed-forward* aplicada a cada posición. Ambas subcapas incluyen conexiones residuales y normalización, lo que estabiliza el entrenamiento. La salida del encoder es una representación contextualizada de toda la secuencia de entrada.

El **decoder** también consta de N bloques, pero con tres subcapas: auto-atención enmascarada (para que un token no pueda usar información futura), atención encoder-decoder (que conecta con las representaciones producidas por el encoder) y, finalmente, la red *feed-forward*. El resultado pasa por una capa lineal y un softmax que produce una distribución de probabilidades sobre el vocabulario de salida. De este modo, el decoder puede generar secuencias autoregresivamente, como en la traducción automática.

En ambos lados se añaden *embeddings* de entrada (tokens de texto, parches de imagen o bloques de espectrograma) junto con codificaciones posicionales, que proporcionan al modelo la noción de orden. El esquema muestra cómo estos componentes se combinan en un flujo de información que va desde los datos iniciales hasta las predicciones de salida, ilustrando la modularidad y simetría de la arquitectura.

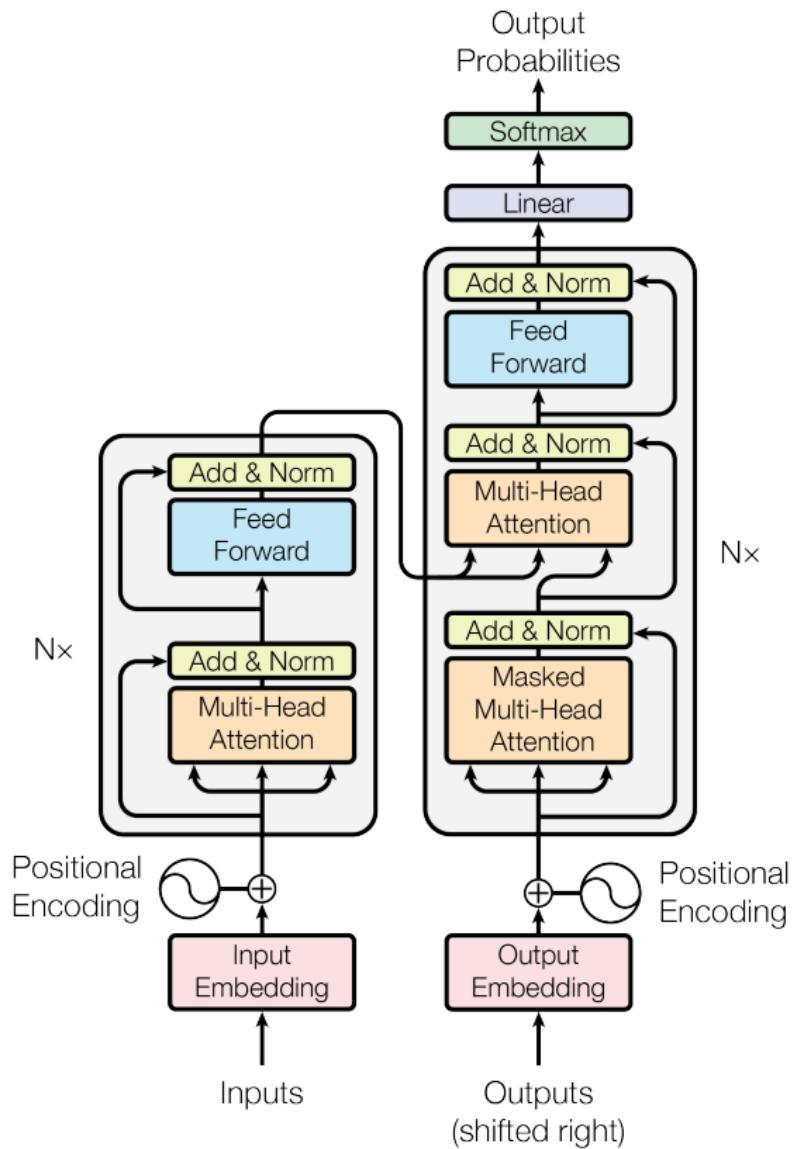


Figura 16: Arquitectura del Transformer propuesta por Vaswani

4.3 Mecanismos principales

Atención y atención multi-cabeza. Un mecanismo de atención calcula una combinación ponderada de todos los elementos de la secuencia de entrada según su relevancia para un elemento dado (consulta). Formalmente, la atención recibe un vector *query* (consulta), un conjunto de vectores *key* (claves) y sus correspondientes *value* (valores), y produce como salida una suma ponderada de los valores [80]. Las ponderaciones se obtienen mediante una función de compatibilidad entre la consulta y cada clave. En el Transformer se utiliza una atención por producto escalar escalado, cuya fórmula puede expresarse como:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- Q : matriz de consultas (*queries*).
- K : matriz de claves (*keys*).
- V : matriz de valores (*values*).
- d_k : dimensión de las claves, usada para escalar el producto QK^T .

La fórmula de la atención escalada se realiza a partir de tres matrices. Q , la matriz de consultas (query) que contiene información que busca cada token de entrada en relación con los demás. La matriz de claves, K , que determina que información ofrece cada token con el resto de la secuencia. Por último la matriz de valores V , contiene el contenido o información que se transmite si los tokens están relacionados. El producto QK^T calcula la relación de un token con el resto de la secuencia. El denominador es un factor de escala dependiente de la dimensión que estabiliza los gradientes. La función softmax se obtiene una función de probabilidad que resaltan las posiciones más relevantes. Estos pesos se ponderarán con los valores de V , resultando como salida una mezcla de los elementos de la secuencia enfocados en lo más importante de cada consulta.

El Transformer amplía este concepto con la atención de múltiples cabezas (*multi-head attention*). En lugar de una única atención, el modelo entrena varias “cabezas” de atención en paralelo [80]. Cada cabeza proyecta las Q, K, V originales a subespacios de menor dimensión (con diferentes pesos entrenables) y calcula su propio conjunto de pesos de atención.

Los resultados de todas las cabezas se concatenan y combinan mediante una capa lineal final. Este mecanismo permite que cada cabeza enfoque distintos aspectos o relaciones de la secuencia en paralelo, superando la posible pérdida de información que ocurriría si solo una cabeza promediara todas las atenciones. En conjunto, la atención multi-cabeza confiere al modelo un gran poder de representación al combinar esas perspectivas.

Codificación posicional. Una diferencia crucial del Transformer respecto a las RNN es que no existe ninguna recurrencia ni convolución que imponga un

orden en la secuencia. Al ingresar todos los tokens simultáneamente, el modelo no “sabe” en qué posición viene cada uno. Para solventar esto, se añaden codificaciones posicionales a los *embeddings* de entrada [80]. En el Transformer original se usan codificaciones sinusoidales fijas: a cada posición *pos* en la secuencia se le asigna un vector calculado con funciones seno y coseno de distintas frecuencias. Estas codificaciones tienen la misma dimensión que los *embeddings* de palabra, y se suman a ellos antes de entrar al primer bloque del modelo. La motivación de usar senos y cosenos es que facilitan que el modelo aprenda relaciones de posición relativa (por ejemplo, que dos elementos separados por *k* posiciones tengan codificaciones con una relación aproximadamente lineal).

En la práctica, las codificaciones posicionales permiten al Transformer distinguir qué token es primero, segundo, etc., preservando así la estructura de la secuencia. Los transformers introducen todos los tokens de forma simultánea, paralelizando los cálculos y permitiendo procesar secuencias largas. Sin embargo, nace la necesidad de añadir codificaciones posicionales para indicar el orden de la secuencia, ya que en muchos contextos, como el procesado de texto el orden contiene información indispensable. Aunque existen más estrategias, en el transformer original presentado por Vaswani [80], se emplean codificaciones sinusoidales. A cada posición en la secuencia se le asigna un vector calculado con funciones seno y coseno de distintas frecuencias, vectores con misma dimensión que los embeddings de las propias palabras y se les suman antes de entrar al modelo. De esta forma aprenden relaciones, permitiendo distinguir el orden de los tokens y preservando la estructura de la secuencia.

	<i>k</i>	<i>i</i> =0	<i>i</i> =0	<i>i</i> =1	<i>i</i> =1
The	0	$P_{00}=\sin(0) = 0$	$P_{01}=\cos(0) = 1$	$P_{02}=\sin(0) = 0$	$P_{03}=\cos(0) = 1$
dog	1	$P_{10}=\sin(1/1) = 0$	$P_{11}=\cos(1/1) = 0.54$	$P_{12}=\sin(1/10) = 0.10$	$P_{13}=\cos(1/10) = 1.0$
ran	2	$P_{20}=\sin(2/1) = 0.91$	$P_{21}=\cos(2/1) = -0.42$	$P_{22}=\sin(2/10) = 0.20$	$P_{23}=\cos(2/10) = 0.98$
fast	3	$P_{30}=\sin(3/1) = 0.14$	$P_{31}=\sin(3/1) = -0.99$	$P_{32}=\sin(3/10) = 0.30$	$P_{33}=\cos(3/10) = 0.96$

Figura 17: Ejemplo de codificación sinusoidal

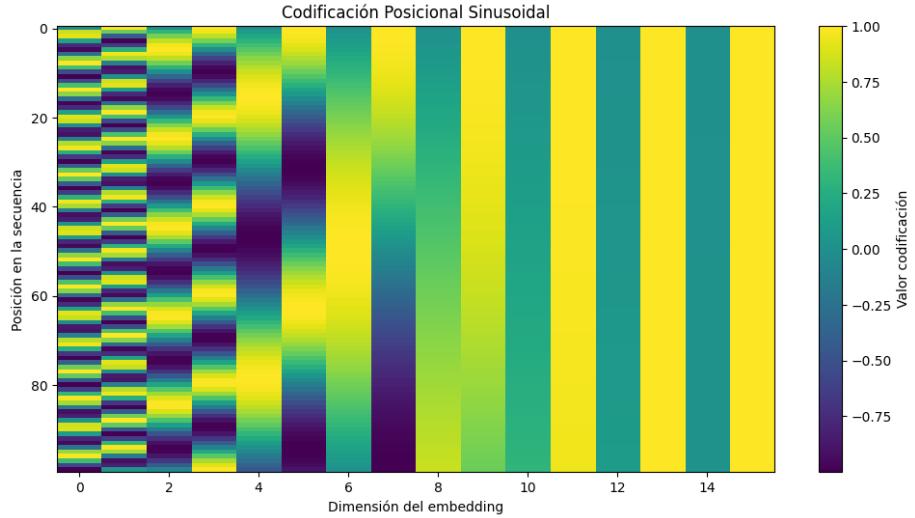


Figura 18: Heatmap de la codificación posicional sinusoidal empleado en los Transformers

En este gráfico generado se muestra cómo la codificación posicional sinusoidal asigna a cada posición de la secuencia un patrón distinto en las posiciones del embedding usando combinaciones de ondas de distinta frecuencia. Permitiendo así incorporar al transformer el orden de los tokens aunque no tenga recurrencia ni convolución.

- Eje horizontal: Dimensión del embedding
- Eje vertical: posición en la secuencia
- Colores: Codifican el valor numérico en esa posición y dimensión

Estructura del modelo El Transformer se organiza en bloques que se repiten. En su forma original, consta de un encoder y un decoder separados [80].

Cada bloque de encoder incluye una capa de auto-atención multi cabeza y tras ella, una red feed-forward aplicada secuencialmente. Esta red esta formada por un perceptrones con dos capas lineales y activación intermedia. Tras cada subcapa se aplica un *skip connection* residual seguido de normalización por capas (*layer normalization*). Estos mecanismos multi-head attention, feed-forward, residuales y normalización se repiten en cada capa. En el Transformer base se utilizan $N = 6$ capas encoder apiladas. El decoder, usado en tareas como traducción, incluye auto-atención, feed-forward y además una atención encoder–decoder, donde las consultas provienen del decoder y las claves/valores del encoder. Se aplica también un enmascaramiento (*masking*) en la auto-atención para asegurar que al predecir un token no se usen futuros tokens (autoregresivo). En nuestro contexto de clasificación de audio, normalmente se emplea únicamente

el encoder.

4.3.1 Justificación Matemática

Una ventaja de la arquitectura Transformer es que, pese a su novedad conceptual, sigue siendo un modelo de red neuronal completamente diferenciable de extremo a extremo. Cada componente descrito (atención, *softmax*, combinaciones lineales, etc.) es una función diferenciable, y en conjunto el modelo puede verse como una composición amistada de funciones diferenciables sobre la entrada. Esto significa que podemos entrenar al Transformer mediante retropropagación de errores (*backpropagation*), igual que a cualquier red neuronal clásica. Gracias a la regla de la cadena, el gradiente del error se puede propagar a través de cada subcapa (atención, capas lineales, etc.), ajustando los pesos internos para minimizar una función de pérdida definida (por ejemplo, entropía cruzada en clasificación).

En términos formales, si denotamos por L la pérdida a minimizar y θ representa el conjunto de todos los pesos del modelo (matrices de proyección W_q, W_k, W_v, W^O de las cabezas de atención, pesos de las capas *feed-forward*, *embeddings*, etc.), el entrenamiento consiste en iterativamente actualizar θ en la dirección negativa del gradiente:

$$\theta \leftarrow \theta - \eta \nabla_{\theta} L,$$

donde η es la tasa de aprendizaje. Esta actualización por descenso de gradiente estocástico se aplica repetidamente usando los gradientes calculados por *backpropagation*. Por ejemplo, los parámetros de las proyecciones en cada cabeza de atención reciben gradientes que dependen de cuánta atención otorgó esa cabeza a las posiciones correctas o incorrectas, y así el modelo aprende a resaltar las relaciones útiles. Lo importante es que no hay pasos no diferenciables: incluso la operación de *softmax* en la atención es diferenciable (tiene una derivada conocida), al igual que la normalización por capas. Por ello, el espacio de parámetros del Transformer se puede optimizar con las mismas técnicas estándar de aprendizaje profundo. En resumen, desde un punto de vista matemático, el Transformer es coherente con el paradigma de aprendizaje supervisado: define una función $f_{\theta}(x)$ diferenciable que mapea una entrada x (secuencia de tokens) a una salida (por ejemplo, distribución de probabilidad de clases), y entrena los parámetros θ para minimizar la pérdida mediante gradiente descendente.

Vale la pena notar que, a pesar de tener muchos componentes, el número de operaciones secuenciales mínimas por capa es menor que en una RNN. En la práctica, la complejidad de cada capa de auto-atención es $O(n^2)$ respecto al número de tokens (n) debido al producto QK^T , pero esta operación se puede ejecutar de forma matricial completamente paralela en hardware GPU. Esto contrasta con la complejidad temporal $O(n)$ secuencial de una RNN por token [3]. Así, matemáticamente el Transformer ofrece mejor escalabilidad a costa de un mayor uso de memoria.

4.3.2 Requisitos dataset y eficiencia

La necesidad de un elevado número de parámetros conlleva una serie de implicaciones sobre los datos necesarios para entrenarlos. En general, los Transformers necesitan volúmenes de datos muy grandes para alcanzar su máximo rendimiento, ya que no asumen tantas estructuras predefinidas en los datos como otros modelos [21, 42]. Por ejemplo, en visión, una CNN aprovecha la estructura local de las imágenes mediante convoluciones con pesos compartidos, lo que actúa como un sesgo de localidad. Este sesgo hace que las CNN aprendan patrones con menos datos, pues esencialmente conocen que un patrón visual puede aparecer en cualquier parte de la imagen. En cambio, un Transformer visual trata cada *patch* de imagen más libremente, sin imponer explícitamente esa invariancia espacial. Como resultado, los Transformers requieren un mayor volumen de datos y tienden a sobreajustarse si el conjunto de entrenamiento es pequeño [42]. Estudios comparativos han señalado que, con conjuntos de datos reducidos, los modelos convolutionales suelen superar a los Transformers, precisamente gracias a sus sesgos incorporados [42].

Aspecto	CNN	Vision Transformers (ViT)
Eficiencia de datos	Miles de imágenes	Decenas de millones de imágenes
Sesgos	Invariancia local/traslacional impuesta por convoluciones	Tratan cada patch con menos restricciones.
Preentrenamiento	No siempre necesario: rinden bien en datasets medianos.	Requieren de grandes volúmenes de datos para ajustar el modelo[21].
Pocos datos	Suelen superar a Transformers por sus sesgos estructurales.	Mayor riesgo de sobreajuste; rendimiento inferior a ResNet equivalente con datos limitados.

Tabla 1: Comparativa entre CNN y Vision Transformers en eficiencia de datos, sesgos y necesidad de preentrenamiento.

De hecho, el ViT original obtuvo resultados sobresalientes en ImageNet solo después de preentrenarse en un conjunto privado de 300 millones de imágenes y luego ajustarse a ImageNet [21].

Una consecuencia de lo anterior es que, cuando no se dispone de datos masivos, es crucial aplicar técnicas de regularización y aumento de datos para entrenar Transformers. Se ha observado que los Transformers dependen más fuertemente de la regularización y *data augmentation* (AugReg) en comparación con las CNN, especialmente en conjuntos pequeños [77]. Por ejemplo, en visión es habitual entrenar ViT en ImageNet usando esquemas agresivos de aumento de imagen (recortes aleatorios, *mixup*, *cutout*, *color jitter*, etc.) y regularizadores como decaimiento de peso o *dropout* elevados, para prevenir el sobreajuste [77]. El trabajo de Trouvon [78] mostró que combinando una intensa augmentación con más cómputo, era posible que un ViT igualara el desempeño que tendría si se le diera 10 veces más datos sin augmentación [77]. En otras palabras, el aumento de datos e incentivos de regularización puede suplir, hasta cierto pun-

to, la falta de datos en Transformers, imitando los beneficios de un dataset más grande. Por contraste, en CNN muchas de esas técnicas son útiles pero menos críticas, ya que la propia arquitectura ya evita cierto sobreajuste mediante sus restricciones estructurales.

En nuestro caso concreto (clasificación de audio de tos con Transformers), disponemos de un conjunto de datos relativamente reducido. Siguiendo las observaciones anteriores, es indispensable aplicar técnicas de *data augmentation* sobre las muestras de audio para ampliar artificialmente el volumen de datos efectivo y mejorar la generalización del modelo. Esto puede incluir transformar los espectrogramas con cambio de tono, adición de ruido, mezclado de muestras (*mixup*), entre otras estrategias. Dichas técnicas, junto con regularizadores (p. ej. *dropout* en las capas de atención, penalización L2, etc.), actúan como contrapeso al riesgo de sobreajuste.

De este modo, aunque nuestro dataset de tos no sea suficientemente grande, el Transformer puede entrenarse de forma robusta extrayendo patrones en los espectrogramas y evitando aprender solo las particularidades. En resumen, la necesidad de grandes datos que tienen los Transformers se puede mitigar hasta cierto punto con augmentación y preentrenamiento; en este Trabajo de Fin de Grado justificamos explícitamente el uso intensivo de augmentación de datos para compensar la carencia de volumen de datos y aprovechar el potencial de esta arquitectura.

4.4 Variantes de arquitectura

El Vision Transformer (ViT) es un modelo propuesto por Dosovitskiy que aplica la arquitectura Transformer al ámbito de la visión por computador [20]. La idea central del ViT es tratar una imagen como una secuencia de parches, de manera análoga a cómo en procesamiento de lenguaje natural se trata un texto como una secuencia de palabras.

Concretamente, una imagen de entrada se divide en una cuadrícula de parches de tamaño fijo , y cada parche se aplana en un vector de características [42]. A cada vector de parche se le aplica una proyección lineal para llevarlo a la dimensión de trabajo del Transformer [42].

De forma análoga a los modelos de NLP, se suma un *embedding* posicional a cada parche para indicar su ubicación en la imagen [42]. Además, se introduce un token especial de clasificación ([CLS]) al inicio de la secuencia, similar al usado en BERT, cuyo embedding final se emplea como representación de la imagen completa.

La secuencia resultante de *embeddings* de parches se alimenta a un encoder Transformer estándar (sin decoder), compuesto por múltiples capas de autoatención y redes *feed-forward*, que permiten que el modelo atienda globalmente a toda la imagen.

En esencia, el ViT prescinde por completo de las convoluciones: aprende desde

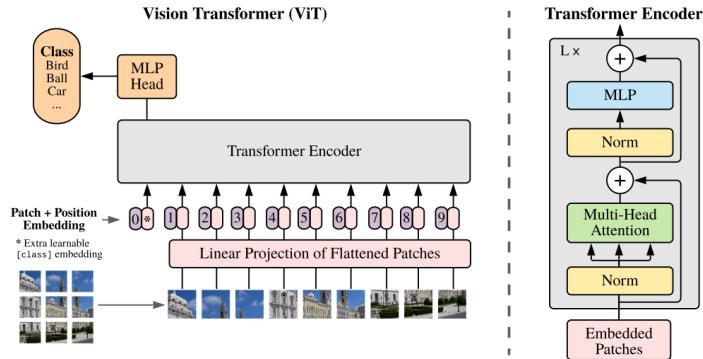


Figura 19: Arquitectura del ViT propuesta por Dosotovitsky [20]

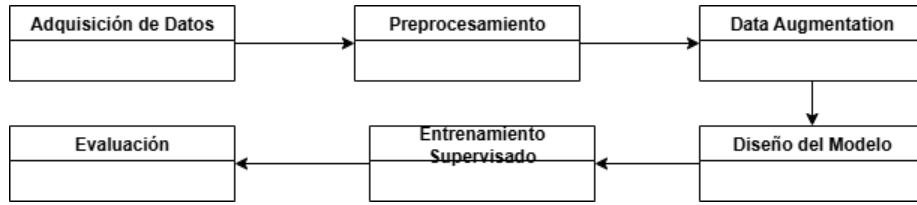


Figura 20: Flujo de Trabajo

ceros qué regiones de la imagen deben relacionarse entre sí mediante la autoatención. (figura 19)

5 Metodología

En esta sección se detalla el procedimiento seguido para abordar el problema primero de la detección automática de tos y posteriormente de su clasificación. Se parte de grabaciones clínicas en formato de audio, se someten a un procesamiento para aumentar la robustez del sistema y tras convertirse a espectrogramas, sirven como entrada a los modelos. Finalmente es entrenado y validado. La figura 20 ilustra las fases clave del flujo de trabajo.

5.1 Recursos hardware, software y librerías

Todos los experimentos se realizaron en un entorno Linux x86_64 con una GPU NVIDIA para acelerar los cálculos. El código fue desarrollado en Python 3.9 utilizando bibliotecas especializadas en deep learning y procesamiento de audio. En particular, se empleó TensorFlow junto con Keras como framework de aprendizaje profundo para definir y entrenar los modelos neuronales.

Para la gestión y aumento del conjunto de datos, se desarrollaron una serie de scripts en Python, incluidos en el anexo, estructurando el código en módulos independientes: por ejemplo, un módulo para aplicar cada técnica de data augmentation, otro para generar espectrogramas en formato imagen a partir de los audios WAV, y otro para entrenar/evaluar el modelo con validación cruzada.

El entorno de hardware constaba de un servidor con procesador Intel Xeon y una GPU nvidia Tesla con 16 GB de memoria, lo que permitió manejar los tensores de entrada en paralelo. Asimismo, se aprovechó la API CUDA a través de TensorFlow para realizar la computación en GPU.

En cuanto a software, el sistema operativo fue Ubuntu 20.04 LTS y se utilizó un entorno virtual de Python para aislar las dependencias. Todos los experimentos fueron ejecutados sin requerir conexión.

Framework / Librería	Uso principal
Librosa	Extracción de espectrogramas, efectos de audio
SoundFile	Lectura y escritura de archivos WAV
scikit-learn	Validación cruzada (GroupKFold), métricas
NumPy	Cálculo matricial y operaciones vectorizadas
SciPy	Transformada de Fourier, procesamiento de señal
Matplotlib	Generación de gráficas y visualizaciones

Tabla 2: Librerías utilizadas y su funcionalidad principal

5.2 Descripción del conjunto de datos

El conjunto de datos de audio utilizado para el desarrollo y evaluación del modelo proviene de un estudio previo de monitorización de tos. Esta formado por grabaciones etiquetadas. Para la detección contamos con alrededor de 17.000 muestras de tos y 114.000 fragmentos sin tos. Cada muestra es un archivo .wav de un segundo de duración. Estas grabaciones pertenecen a 20 sujetos diferentes, introduciendo una variabilidad inter-individual en las características de la tos, ya que estas tienen una alta componente vocal. El procesado de los audios jugará un papel importante dado que queremos un modelo que detecte toses para posteriormente clasificar, no que detecte pacientes.

Las muestras están organizadas originalmente en dos directorios principales:

- **DATA:** contiene 20 subcarpetas, una por paciente (identificados como *Pal01*, *Pal02*, ..., *Pal20*). En cada carpeta se encuentran múltiples archivos WAV de tos de ese paciente, cada uno de 1 segundo.
- **NO_TOS:** directorio que agrupa fragmentos de audio de 1 s donde no hay tos (pueden contener ruido ambiental, habla u otros sonidos, pero fueron verificados como segmentos sin eventos de tos). Estos no están separados por paciente, sino mezclados, sumando un total de 114k archivos.

AQUI TENGO QUE METER HISTOGRAMAS DE LA VARIANZA ENTRE NÚMERO DE DATOS POR PACIENTE

5.2.1 Grabaciones originales y desbalance de clases

Las grabaciones originales presentan un claro desbalance de clases: la proporción de muestas de “no tos” es aproximadamente 6.7 veces mayor que la de “tos”. Esto refleja la realidad de muchos entornos de detección, en los que los eventos de interés (tos) son una minoría en comparación con periodos de no tos o silencio. Este desbalance podría llevar a que un modelo trivial que siempre predijera “no hay tos” obtuviese una precisión alta (87 % de aciertos), pero obviamente sería inútil en la práctica. Por tanto, fue prioritario abordar este desbalance mediante técnicas tanto de muestreo como de métricas adecuadas AUC, sensibilidad o especificidad.

Otro factor a tener en cuenta es que algunos audios de tos fueron grabados en entornos controlados relativamente silenciosos, mientras que otros contienen ruido de fondo significativo (conversaciones, ruidos de tráfico, etc.). Esto añade dificultad a la tarea pero a la vez hace al conjunto de datos más representativo de un despliegue real.

Cada paciente aportó un número diferente de toses (por ejemplo, algunos pacientes tienen cientos de ejemplos de tos y otros apenas decenas). Esta disparidad inter-paciente se tuvo en cuenta en la validación cruzada, para evitar que un paciente con muchas muestras dominase el entrenamiento.

5.2.2 Estadísticas generales del corpus

En la Tabla ?? se resumen algunas estadísticas generales del corpus de audio utilizado:

Clase	Número de muestras	Duración total
Tos (positivos)	~17 000	4.7 horas
No Tos (negativos)	~114 000	31.7 horas
Total	131 000	36.4 horas

Tabla 3: Resumen del conjunto de datos de tos utilizado. Se indica el número de muestras por clase y la duración total.

Como se observa, el conjunto de datos suma en torno a 36 horas de audio. Todas las muestras tienen exactamente 1 segundo, gracias a un preprocesamiento de segmentación realizado en el estudio original del cual provienen los datos. Este preprocesamiento original garantizó que los archivos de tos contuvieran al menos una tos clara en ese segundo (en la mayoría de casos la tos ocurre cerca del inicio del segmento, seguido de silencio), y que los archivos de no-tos no incluyeran ninguna tos (solo silencio o ruido no relacionado). Esta característica

que la tos aparezca típicamente al comienzo del segmento positivo motivó una de las técnicas de data augmentation que describiremos más adelante (muestreo circular) para evitar sesgos.

En cuanto a la distribución por paciente, en promedio cada paciente aportó 850 muestras de tos (aunque con alta variabilidad). Los pacientes con menos toses rondan las 100-200 muestras, mientras que los más prolíficos tienen del orden de 2000. Esto implica que, sin una estrategia adecuada, el modelo podría sobreajustarse a aquellos pacientes con muchas toses. Por ello, en la estructuración del conjunto de datos aumentado se optó por un enfoque que equilibra el número de muestras de cada paciente aplicando técnicas de aumento proporcionalmente.?? Revisar códigos

5.3 Preprocesamiento de audio

Antes de extraer características y entrenar los modelos, se aplicó un preprocesamiento uniforme a todos los archivos de audio para normalizar condiciones y eliminar posibles sesgos triviales:

5.3.1 Normalización y recorte

Todas las señales de audio se normalizaron en amplitud. En concreto, cada waveform fue escalada dividiendo por el valor absoluto máximo de la onda, de modo que cada archivo tuviera amplitud en el rango $[-1, 1]$. Esta normalización por pico previene saturación (*clipping*) al generar los espectrogramas y asegura que las diferencias de volumen absoluto entre muestras no dominen el aprendizaje (el modelo debería aprender características más robustas que simplemente “esta tos suena más fuerte que el ruido”). Cabe mencionar que esta normalización se aplicó después de ciertas técnicas de aumento que modifican la amplitud, para compensar la atenuación/amplificación introducida).

Dado que cada archivo ya venía segmentado a 1 segundo exacto, no fue necesario realizar recortes temporales significativos ni padding. No obstante, se aseguró que todos los archivos tuvieran exactamente la misma longitud de 1.00 s (algunos pocos archivos podían tener una diferencia de milisegundos por cuestiones de muestreo). Se recortaron o llenaron con silencio los casos en que la duración difería ligerísimamente para mantener la uniformidad dimensional.

Por último, se convirtió todas las señales a un mismo formato de muestreo: frecuencia de muestreo de 16 kHz y 16 bits por muestra PCM mono. Esta frecuencia es suficiente para cubrir el rango de frecuencia relevante de la tos humana (las componentes principales suelen estar por debajo de 8 kHz). Algunos audios originales estaban a 44.1 kHz; estos se resamplearon a 16 kHz con filtrado antialiasing usando `librosa.resample` para reducir el tamaño de datos y la redundancia de frecuencias ultrasónicas no necesarias. ??

5.4 Justificación ViT

Conviene justificar el uso de ViT en un problema donde los datos de entrada son audios [20]. Aunque estos pueden procesarse directamente en forma de señal temporal, en la práctica es más eficaz transformarlo a la representación tiempo/frecuencia. Destacan, entre estas representaciones, los espectrogramas log-Mel, habiéndose consolidado como el estándar en tareas de clasificación de audio. En este proceso convertimos una onda unidimensional en una imagen 2D, permitiendo analizar simultáneamente la evolución temporal y la distribución de energía en frecuencia. Una estructura matricial especialmente conveniente para redes convolucionales o basadas en atención. El escalado Mel simula la sensibilidad humana, aumentando la resolución en frecuencias bajas, y menor en las altas.

5.5 Obtención de espectrogramas

Proceso para convertir el dataset de audios en un array de espectrogramas:

1. Se aplicó sobre cada audio una transformada de Fourier FFT para obtener su espectro de magnitud. Se utilizó una ventana de Hann de 10 ms de duración (≈ 160 muestras a 16 kHz) sin solapamiento entre ventanas consecutivas. Esto produjo 100 ventanas a lo largo de 1 s. La elección de 0 % de solapamiento fue intencional para simplificar la segmentación en parches para el Transformer.
2. El espectro de potencia obtenido se convirtió a la escala mel utilizando 128 bandas mel inicialmente, cubriendo el rango de 0 Hz hasta 8 kHz (frecuencia de Nyquist de 16 kHz).
3. Se tomó el logaritmo de la amplitud (log-mel) añadiendo un pequeño $\epsilon = 10^{-6}$ para evitar logaritmos de cero.
4. Se normalizó cada espectro dividiéndolo por su valor máximo, de modo que todos los coeficientes quedaran acotados entre 0 y 1. Esta normalización por espectro elimina factores globales de escala entre diferentes muestras.
5. Para reducir la dimensionalidad se tomaron únicamente las primeras 45 bandas mel, desde la frecuencia mínima hasta aproximadamente 8 kHz. Estudios previos indican que la energía principal de la voz se concentra típicamente bajo 5–6 kHz [?]; aquí se incluyeron hasta 8 kHz para margen de seguridad, resultando en 45 bandas de mel.
6. Finalmente, se ajustó la dimensión temporal a 100 marcos: si algún espectro tenía ligeramente menos de 100 columnas (por pequeños efectos de bordes en la STFT), se realizó *padding* simétrico con valores mínimos; si tuviera más, se recortaría. En la práctica, todos quedaron de tamaño 45×100 .

El resultado es que cada audio se transformó en una matriz de características de 45×100 . Estas matrices se almacenaron y se usaron como entradas al modelo. Se optó por calcular y guardar todos los espectrogramas antes del entrenamiento para acelerar las iteraciones de entrenamiento y facilitar la experimentación. Para ello, se creó un fichero tipo pickle que contenía un vector de dimensiones $(N, 45, 100)$ con todos los espectrogramas (donde $N \approx 131000$ es el total de muestras) y sus etiquetas correspondientes.

En la Figura ?? se muestra un ejemplo de espectrograma log-mel obtenido a partir de una tos, comparado con el espectrograma de un fragmento sin tos. Se puede apreciar visualmente la diferencia: la tos presenta una explosión de energía de banda ancha 21 seguida de armónicos y decaimiento, mientras que el segmento sin tos muestra solo ruido de fondo difuso de menor intensidad 22.

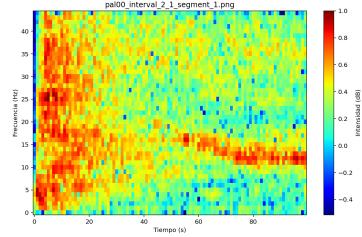


Figura 21: Fragmento con tos

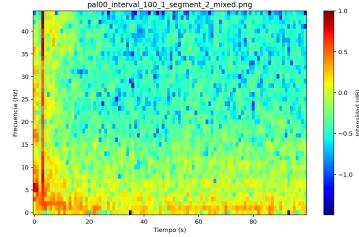


Figura 22: Fragmento sin tos

5.6 Data Augmentation

Dada la limitada cantidad de datos de tos y la gran variabilidad de condiciones posibles en entornos reales, se emplearon intensivamente técnicas de data augmentation específicas de audio [? ?]. El objetivo del data augmentation fue generar versiones ligeramente modificadas de las grabaciones originales de manera que el modelo aprendiera a ser invariante a ciertas transformaciones y ganara robustez sin necesidad de obtener nuevas muestras reales. Tras una revisión bibliográfica de métodos exitosos en clasificación de audio [?], se seleccionaron cinco técnicas principales de aumento:

1. **Pitch Scaling**
2. **Noise Injection** (inyección de ruido)
3. **Volume Scaling** (escalado/ajuste de volumen)
4. **Convolución con respuesta al impulso** (simulación de reverberación de sala)
5. **Muestreo Circular** (desplazamiento temporal cíclico de la señal)

También se consideraron otras técnicas clásicas como time stretching y inversión de fase, pero se descartaron tras una evaluación inicial: el primero alteraba la

duración de la señal y, por tanto, rompía la uniformidad temporal requerida, dado que nuestros espectrogramas deben tener longitud fija, mientras que la inversión de la señal no produce ningún cambio apreciable en el espectrograma de magnitud, únicamente invierte la fase, y por tanto no aportaría variabilidad relevante, dado que en los espectrogramas solo se considera la magnitud. A continuación, se detallan las técnicas implementadas y sus parámetros.

5.7 Pitch Scaling

Esta técnica consiste en modificar la frecuencia fundamental de la señal de audio. En la práctica, equivale a hacer que la tos suene más aguda o grave que la original, simulando variaciones fisiológicas entre individuos, o incluso variaciones por edad o sexo.

Esta transformación se implementó mediante la función ??, aplicando un desplazamiento en semitonos al espectro de la señal. Los semitonos, que son el parámetro de control de la función, son el salto de frecuencia entre dos teclas adyacentes en cualquier instrumento de teclado, dividiendo cualquier escala en 12 pasos logarítmicos.

La transposición de n semitonos en la escala temperada equivale a un factor de razón de frecuencias

$$r = 2^{n/12}. \quad (1)$$

Dado un tono de frecuencia f_0 , la frecuencia resultante tras el desplazamiento es

$$f' = r f_0 = f_0 2^{n/12}. \quad (2)$$

En el dominio de la frecuencia continua, el escalado de tono ideal actúa como

$$Y(\omega) = X\left(\frac{\omega}{r}\right), \quad (3)$$

lo que desplaza todas las componentes a una razón común r .

El ajuste de este parámetro fue fundamental, dado que si se desplaza el tono demasiado, se introducen distorsiones no realistas, contaminando el conjunto de datos. Finalmente el número de semitonos que se desplazaba cada señal era un parámetro aleatorio con distribución uniforme entre -5 y 5 semitonos. Este desplazamiento reduce la componente vocal y espectral de la tos sin modificar la estructura temporal de la tos.

Para cada señal se elige aleatoriamente el número de semitonos en un intervalo simétrico:

$$n_{\text{steps}} \sim \mathcal{U}(a, b), \quad a = -5, b = 5, \quad (4)$$

En la figura ?? se muestra la forma de la onda y el espectrograma de un audio de ejemplo antes y después de la transformación. Se puede observar el efecto que tiene tanto en el dominio del tiempo como de la frecuencia. Concretamente, en

esta simulación el parámetro que determina el desplazamiento frecuencial tomó un valor positivo, por lo que se puede observar como se desplaza ligeramente hacia arriba toda la imagen, hacia frecuencias mayores.

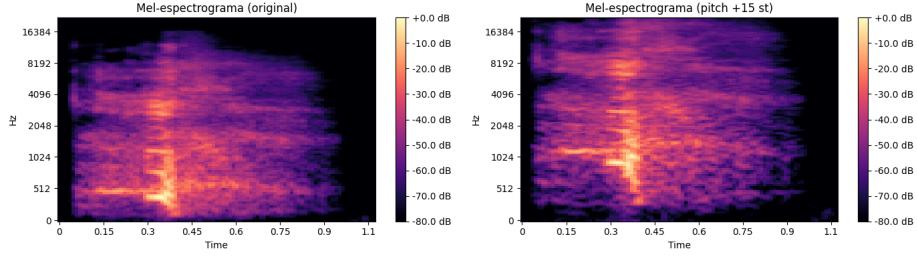


Figura 23: Comparativa entre el espectrograma original y tras aplicar el desplazamiento en frecuencia

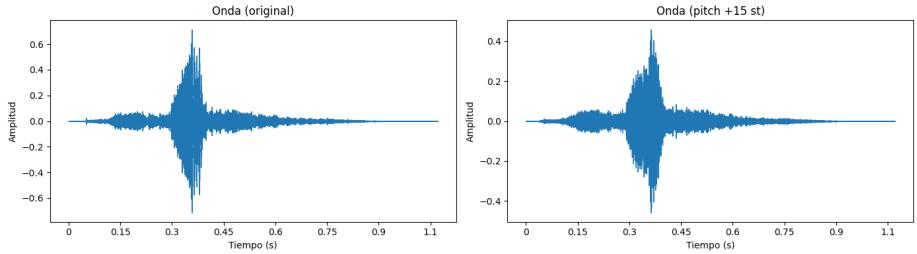


Figura 24: Comparativa entre el waveform original y tras aplicar desplazamiento en frecuencia

5.8 Noise Injection

Esta técnica consiste en añadir ruido de fondo sobre la señal original. Entre las distintas variantes, se empleo ruido blanco Gaussiano. Esto simula entornos acústicos contaminados, no ideales, dónde hay interferencias de fondo. Al exponer el modelo a toses con distintos niveles de ruido le forzamos a que aprenda las características intrínsecas de la tos. ??

Se añade ruido gaussiano blanco de media cero a la señal original:

$$y(t) = x(t) + \eta(t), \quad \eta(t) \sim \mathcal{N}(0, \sigma^2), \quad (5)$$

Debemos escoger el valor que tomará la intensidad del ruido, fundamental para que el ruido no sea insignificante, pero no lo suficientemente potente como para enmascarar la tos. Este parámetro se calibró en términos de la relación señal a ruido (SNR).

Sea la potencia media de la señal

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} x[n]^2 = r_x^2, \quad (6)$$

donde r_x es el valor cuadrático medio (RMS) de x . Para ruido gaussiano blanco de varianza σ^2 se tiene

$$P_\eta = \mathbb{E}[\eta[n]^2] = \sigma^2. \quad (7)$$

La relación señal–ruido en forma lineal y en decibelios es

$$\text{SNR} = \frac{P_x}{P_\eta} = \frac{r_x^2}{\sigma^2}, \quad \text{SNR}_{\text{dB}} = 10 \log_{10} \left(\frac{r_x^2}{\sigma^2} \right) = 20 \log_{10} \left(\frac{r_x}{\sigma} \right). \quad (8)$$

En función del SNR se concluyó adecuado que, al igual que el parámetro para el *pitch scaling*, funcionara como un parámetro aleatorio en el rango (0.01,0.03). Esto se corresponde con un nivel entre 10 y 20 dB.

También se consideró usar ruido coloreado (como ruido rosa o rojo, que tienen mayor densidad espectral en bajas frecuencias). Sin embargo, estos tienden a solapar la energía principal de la tos (500 Hz–2 kHz) añadiendo componentes en las mismas bandas, lo cual podía sesgar el aprendizaje del modelo hacia esos patrones de ruido de baja frecuencia. Se decidió restringirse a ruido blanco más neutral espectralmente.

Finalmente reescaló la salida para limitar el pico a un umbral $\tau \in (0, 1)$:

$$y_{\text{clip}}[n] = \frac{\tau}{\max(1, \max_m |y[m]|)} y[n]. \quad (9)$$

En la imagen posterior podemos ver cómo esta técnica añade energía de forma uniforme en el espectrograma, obligando al modelo a centrarse en los picos de energía de la tos. ??

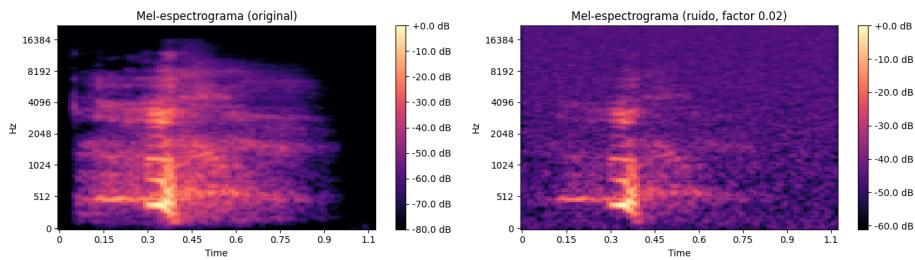


Figura 25: Comparativa de espectrograma antes de la adición de ruido y después

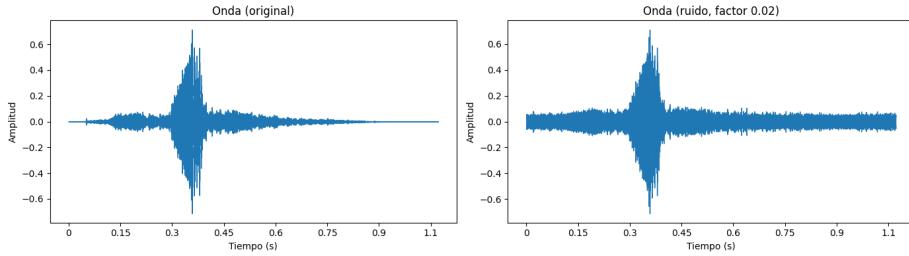


Figura 26: Comparativa del waveform antes de la adición de ruido y después

5.9 Volume Scaling

Multiplicando la señal por un factor obtenemos distintas simulaciones dónde la fuente esta a distinta distancia del micrófono o existen cambios en la ganancia de la grabación. Es una técnica sencilla, pero eficaz a la hora de controlar la sensibilidad del modelo a la intensidad de la tos. Aprendiendo así la invarianza a la escala en amplitud. Sin embargo la posterior normalización del audio antes de generar

El **escalado de volumen** consiste en multiplicar la amplitud de la señal por un factor constante, simulando situaciones en las que la fuente está a distinta distancia del micrófono o hay cambios en la ganancia de grabación. Es una de las técnicas más sencillas, pero efectiva para controlar la sensibilidad del modelo a la intensidad de la tos [?]. Si el modelo se entrenara solo con toses muy fuertes, podría aprender a ignorar toses débiles. Al exponerlo a versiones atenuadas (o amplificadas) de las mismas toses, debería aprender invariancia a la escala de amplitud.

Sin embargo, estas variaciones se eliminaban cuando se normalizaban los espectrogramas. Por lo tanto, para no generar duplicados, esta técnica finalmente no se aplicó sobre el conjunto de datos final.

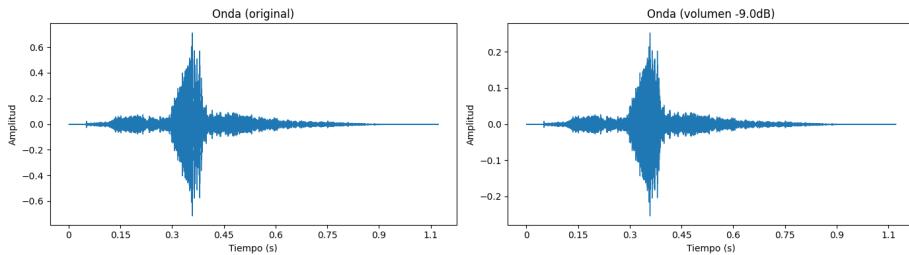


Figura 27: Escalado de el audio con constante aleatoria

5.10 Convolución con Respuesta al Impulso (IR)

Por último, se simularon diferentes entornos acústicos. De esta forma generamos artificialmente muestras como si hubieran sido grabados en distintos sitios y bajo distintas condiciones. Para ello convolvieron las señales de los con las respuestas al impulso de salas reales. Convolver con la respuesta al impulso equivale a que este hubiera sido grabado en esa sala bajo la condición de que el sistema sea lineal e invariante en el tiempo. Entonces su comportamiento queda completamente descrito por la respuesta al impulso.

Cualquier señal de entrada $x(t)$ (p. ej., una tos “seca”) puede descomponerse como superposición de impulsos:

$$x(t) = \int_{-\infty}^{\infty} x(\tau) \delta(t - \tau) d\tau,$$

y por **linealidad e invariancia temporal** la salida es la suma de las respuestas a cada impulso desplazado:

$$y(t) = \int_{-\infty}^{\infty} x(\tau) h(t - \tau) d\tau = (x * h)(t).$$

Así, $y = x * h$ es exactamente la señal que obtendríamos *grabando x en ese recinto con esa colocación fuente-micrófono (ruidos aditivos aparte)*.

El conjunto de respuestas al impulso se descargó de la base de datos pública OpenAIR, un repositorio académico. En total se recogieron 244 respuestas, abarcando desde recintos que introducían notables reverberaciones hasta entornos casi anecoicos o abiertos. A cada audio se le seleccionaba una IR aleatoria y se realizó una convolución lineal. Sobre cada audio se seleccionaba una respuesta aleatoria sobre el conjunto. El principal inconveniente de esta técnica era que alteraba el número de muestras de la señal, siendo la señal final más larga que la original. En el dominio discreto, si $x[n]$ tiene longitud N y $h[n]$ longitud M , la convolución lineal produce:

$$y[n] \in \mathbb{R}^{N+M-1}.$$

Es decir, la señal reverberada es más larga que la tos original debido a la cola de reverberación introducida por la IR.

Control de la duración:

Para mantener una duración fija de 1 s, se aplica:

$$\tilde{y}[n] = \begin{cases} y[n], & 0 \leq n < N_0 \\ 0, & \text{en caso de padding} \end{cases}$$

donde N_0 corresponde al número de muestras en 1 segundo (p. ej., $N_0 = 16,000$ a 16 kHz).

- Si $y[n]$ es más largo que N_0 , se **trunca**.

- Si es más corto (por bordes de STFT, etc.), se **rellena (padding)** con valores bajos o simétricos.

Implementación práctica (FFT):

$$y \approx \text{IFFT}(\text{FFT}(x) \text{FFT}(h)).$$

Posteriormente se normaliza para evitar saturación:

$$\hat{y}(n) = \frac{y(n)}{\max |y(n)|}.$$

La motivación de esta técnica es que en aplicaciones reales, la tos puede ocurrir en salas con eco o con distinta acústica, y un detector entrenado solo con toses “secas” podría fallar al encontrarse reverberación. Al entrenarlo con ejemplos reverberados, se mejora la robustez espacial del modelo. Como ejemplo, la Figura inferior 29e muestra una tos a la cual se le ha aplicado la IR de una gran sala: se observan “colas” de energía posteriores al pulso principal, difuminando los bordes del evento de tos en el espectrograma. Aun así, la ventana de 1 s aseguraba que la energía no se disipa completamente y quedara información suficiente.

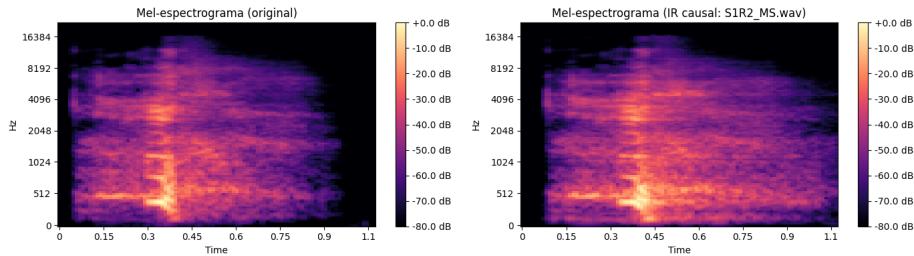


Figura 28: Comparativa espectrograma original vs tras aplicarse la convolución

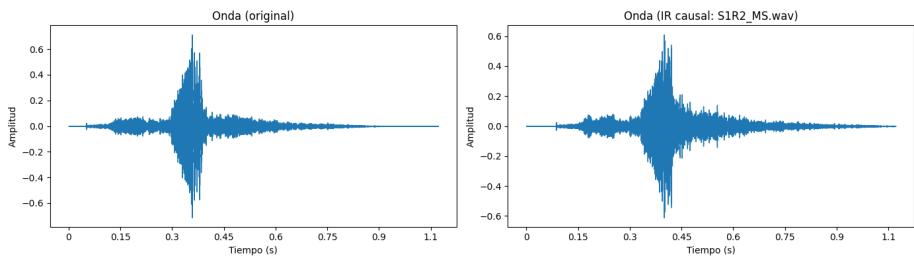


Figura 29: Comparativa del waveform original vs tras aplicarse la convolución

5.11 Muestreo Circular

Por último, se aplicó muestreo circular sobre todo el conjunto, tanto los originales como los . Como se mencionó, en las grabaciones originales la tos suele

ocurrir al inicio de cada segmento de 1 s, seguidos de silencio. Esto podría crear un sesgo en el modelo, interpretando cualquier sonido fuerte como en los primeros instantes como tos, en lugar de aprender las características espectrales de la tos.

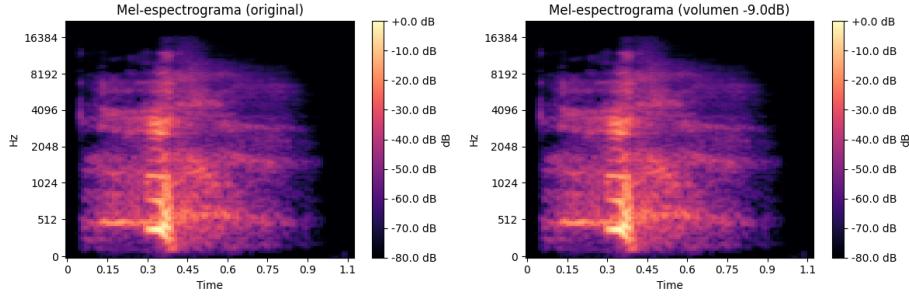


Figura 30: Espectrograma antes y tras aplicar el muestreo circular.

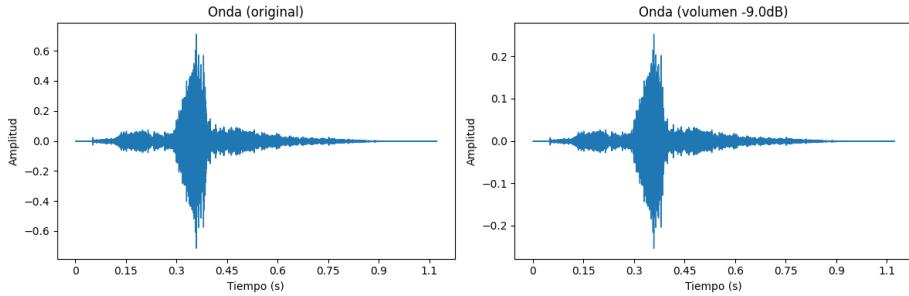


Figura 31: Waveform antes y tras aplicar el muestreo circular

Esta técnica no altera la distribución de frecuencias ni la forma de la tos, solo su posición temporal dentro del segmento.

Sea una señal discreta de tos $x[n]$ de longitud N (por ejemplo, $N = 16,000$ muestras para 1 s a 16 kHz). El desplazamiento temporal de $x[n]$ en k muestras se define como:

$$x_k[n] = x[(n - k) \bmod N],$$

donde el operador $\bmod N$ asegura que los valores que “salen” por un extremo de la señal reaparezcan por el otro.

Ejemplo en tiempo continuo (idealizado):

Si $x(t)$ es la señal de 1 s, el muestreo circular equivale a:

$$x_k(t) = \begin{cases} x(t - k), & 0 \leq t - k < 1 \\ x(t - k + 1), & \text{si } t - k \geq 1 \end{cases}$$

con tiempos normalizados en segundos y desplazamiento $k \in \{0,25, 0,5, 0,75\}$ s.

Interpretación:

- La forma espectral de $x[n]$ no cambia, solo su localización temporal dentro de la ventana.
- Formalmente, la magnitud del espectro $|X(\omega)|$ se conserva:

$$X_k(\omega) = e^{-j\omega k} X(\omega) \Rightarrow |X_k(\omega)| = |X(\omega)|.$$

- Por tanto, la técnica obliga al modelo a aprender patrones invariantes al tiempo, ya que la energía de las tos puede aparecer en cualquier parte del segmento de 1s.

Como se ve en las imágenes anteriores, ?? simplemente consiste en un desplazamiento temporal de la señal sin modificar sus características espectrales.

5.12 Codificación Posicional Aprendida

5.13 Resumen de parámetros y justificación técnica

El nuevo conjunto de datos se generó aplicando estas transformaciones. Sobre cada muestra original se produjeron una con cada técnica (cambio de tono, convolución con la IR, adición de ruido) y luego sobre el conjunto final se aplicó el muestreo circular sobre ellos con un factor de 5. Esto quiere decir que, por el muestreo circular, por cada audio que se procesaba se generaban cinco nuevos con distintos desplazamientos temporales. Este procesado de datos permitió multiplicar el volumen total de los datos en un factor x15, acercándonos a las necesidades de la arquitectura escogida. En nuestro caso, los resultados confirmaron que ningún método de aumento por sí solo bastaba para eliminar el sobreajuste, pero la aplicación conjunta de todos ellos sí logró una mejora sustancial .

5.14 División en conjuntos y validación

La organización inicial del corpus partía de dos grupos diferenciados: por un lado, la carpeta DATA, que contenía a su vez un directorio por paciente (`pal00, pal01, ..., pal20`), y dentro de cada uno de ellos los fragmentos de audio correspondientes a episodios de tos; por otro, la carpeta `no_toses`, que incluía segmentos de control sin eventos de tos. A medida que se aplicaban distintas técnicas de *data augmentation*, se generaban nuevas carpetas que replicaban la misma estructura de DATA, de modo que cada técnica producía un conjunto ampliado de audios distribuidos por paciente. El siguiente paso fue integrar estos subconjuntos en un solo directorio sobre el que se aplicó el muestreo circular. El siguiente paso será la división en los conjuntos de entrenamiento , test y validación. [28]

El conjunto de entrenamiento permitirá ajustar los parámetros del modelo, validación supervisará su aprendizaje y con el subconjunto de test se evaluará el rendimiento final sobre datos nunca vistos. La importancia de este caso residía en asegurar que no se solaparan entre pacientes, todas las muestras de un mismo paciente deberán pertenecer únicamente a uno de los conjuntos. En el caso contrario, conseguiríamos un modelo que memorizara patrones específicos sobre los pacientes, siendo incapaz de generalizar su aprendizaje a futuros nuevos pacientes.

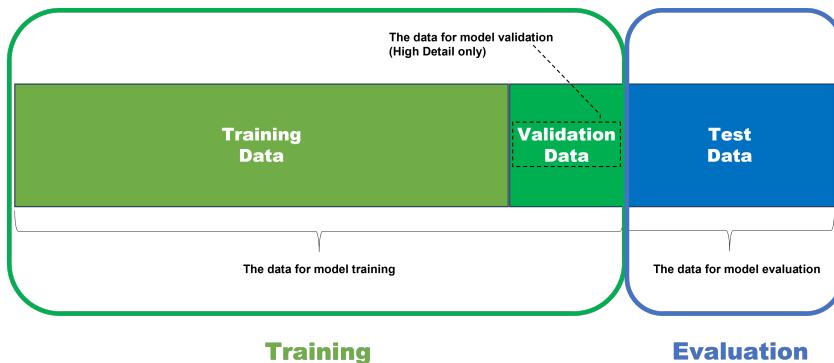


Figura 32: Representación de los subconjuntos de entrenamiento, test y validación respecto a el dataset completo

Esta división se realizó siguiendo el esquema de validación cruzada por k-folds[43]. Los pacientes fueron divididos en k grupos aproximadamente equivalentes. En cada iteración un grupo se reserva para la validación mientras que los restantes se utilizan para el entrenamiento. Este proceso se repite hasta que todos han ocupado el papel de validación. Gracias a esta técnica reducimos la varianza asociada a una única partición de datos, además de proporcionar una mejor estimación sobre su rendimiento.

Desde un punto de vista biomédico, esta elección metodológica resulta crítica. Validar a nivel de paciente y mediante k-folds aproxima de manera más realista las condiciones de despliegue futuro de un modelo de clasificación de los, en el que se esperan sujetos distintos a los observados durante el entrenamiento. Para ilustrar este esquema, en la Figura 33 se muestra un diagrama del procedimiento de validación cruzada empleado, así como un ejemplo de distribución de clases tras el balanceo.

Entrenamiento de Transformers: función de coste, épocas, dropout y tasa de aprendizaje

Entrenar un Transformer, en esencia, consiste en minimizar una función de coste sobre un conjunto de datos con ayuda de un optimizador y algunas técnicas de

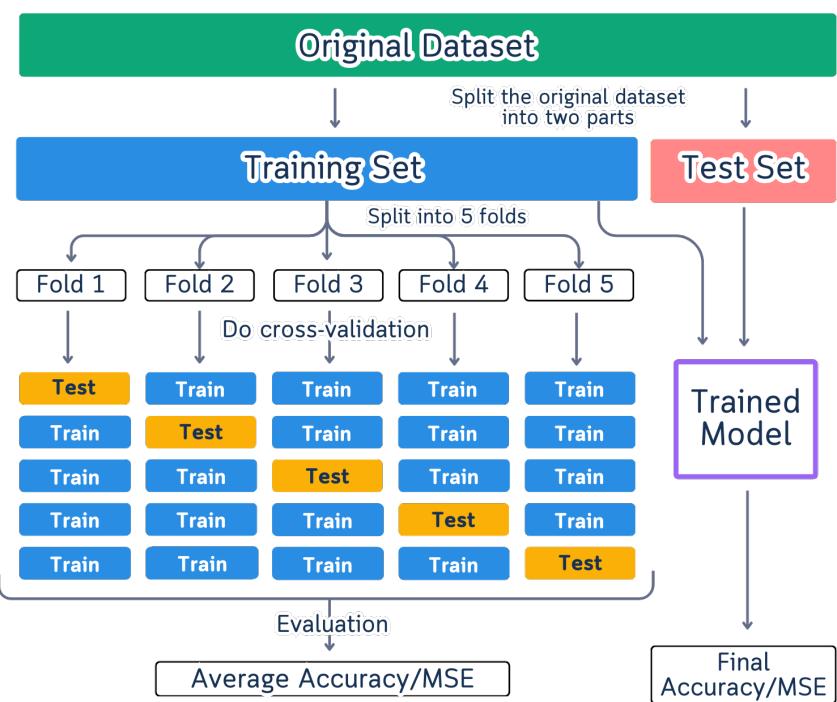


Figura 33: Ejemplo de implementación de la validación cruzada por k-folds

regularización para evitar el sobreajuste [81, 49]. Aunque la implementación concreta pueda variar, conviene fijar bien los conceptos.

Épocas. Una época es una pasada completa por el conjunto de entrenamiento. Pocas épocas suelen dejar un modelo con alto sesgo (underfitting), demasiadas, con alta varianza (memoriza ruido). La pauta más fiable es observar las curvas de entrenamiento y validación. Si la pérdida de entrenamiento sigue bajando pero la de validación sube, conviene activar (early stopping). Con conjuntos de datos muy grandes, es común reportar por pasos (steps) más que por épocas. En fine-tuning de modelos preentrenados, a menudo bastan unas pocas épocas (1–5). Desde cero, pueden requerirse muchas más.

Dropout. La tasa de descarte (dropout) desactiva aleatoriamente una fracción de activaciones durante el entrenamiento y se apaga en inferencia 34. En Transformers puede insertarse en la atención, en la red feed-forward e incluso en embeddings. Su efecto neto es reducir coadaptaciones y mejorar la generalización. Tasas excesivamente bajas tienden al sobreajuste, y excesivamente altas merman capacidad (underfitting). [72, 49].

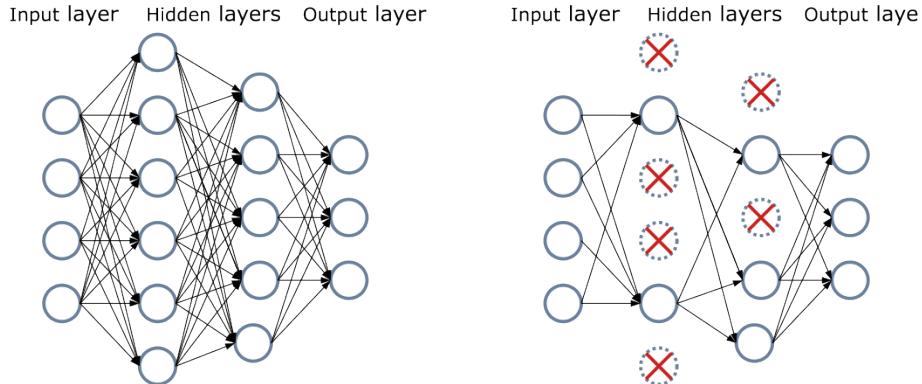


Figura 34: Red neuronal estándar antes y después de aplicar descartes aleatorios (dropout)

Tasa de aprendizaje. La learning rate escala el tamaño de cada actualización. Tasas altas aceleran pero pueden provocar oscilaciones o divergencia; tasas bajas estabilizan pero ralentizan e incluso encallan en óptimos pobres. Con lotes grandes, puede escalarse la tasa de aprendizaje, aunque a veces empeora la generalización si no se compensa con regularización o ruido [49, 58].

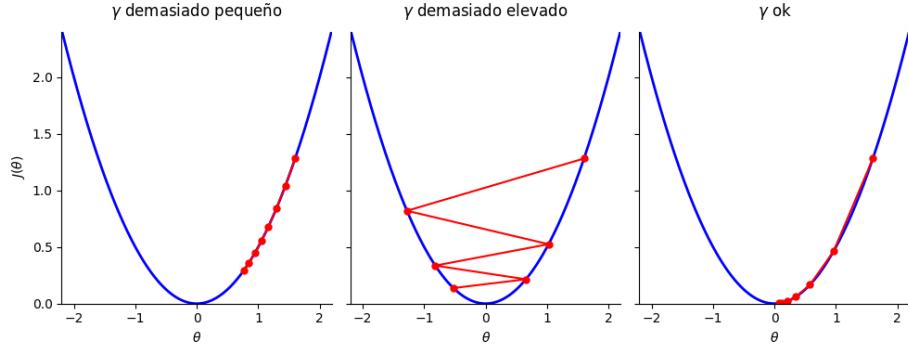


Figura 35: Ejemplo de simulación de convergencia del gradiente según la tasa de aprendizaje γ

En conjunto, entrenar bien un Transformer es menos una serie de pasos que un equilibrio: pérdida alineada con el objetivo, número de épocas vigilado con validación, regularización (dropout/weight decay) y una política de aprendizaje que acompañe al modelo desde la exploración inicial hasta el fine-tunning final.

5.15 Fine-Tuning

El término fine-tunning se refiere al proceso de ajustar un modelo ya entrenado o preconfigurado con el objetivo de adaptarlo a una tarea específica. Es decir, al contrario del entrenamiento desde cero, donde los parámetros se inicializan con valores aleatorios, en el fine-tunning se parte con una configuración previa, posteriormente esos pesos se ajustan para afrontar el nuevo problema, necesitando un menor volumen de datos. Esta herramienta es especialmente útil cuando los conjuntos de datos son limitados, como en contextos biomédicos, resultando ventajoso emplear arquitecturas ya probadas [28].

Una opción era reutilizar una arquitectura de clasificación de imágenes preentrenada (ViT, Resnet, Hugging Face [1]) y posteriormente ajustarla a la detección de tos con los espectrogramas. Esta estrategia funciona especialmente bien si además se acompaña con un preentrenamiento específico en audio. [26, 2]. Sin embargo, en los espectrogramas contiene invariancias que no son compatibles con las imágenes naturales. Mientras que mayoritariamente en los ViT [20] un desplazamiento vertical u horizontal no cambia su clase, en nuestros datos suponen desplazamiento en tiempo y frecuencia, aunque para nuestra tarea si que admite invarianza temporal, no vertical, siendo clave para la clasificación y detección de sonidos.

Por este motivo, se optó por un entrenamiento progresivo apoyado en dos fases. En primer lugar, se utilizó un tunner de hiperparámetros basado en Hyperband [?], encargado de explorar automáticamente distintas configuraciones del Transformer: número de cabezas de atención, dimensión de los embeddings,

profundidad de las capas densas y porcentaje de descarte (dropout). Con este procedimiento se identificaron combinaciones prometedoras en términos de precisión de validación. A partir de esos resultados, se llevó a cabo la fase de fine-tuning propiamente dicha, donde se fijaron los hiperparámetros óptimos y se entrenaron modelos definitivos en los cinco folds de validación cruzada.

Para adaptar el modelo a la clasificación binaria modificamos únicamente el tramo final de la arquitectura. Tras los bloques Transformer añadimos un par de capas densas de tamaño intermedio y rematamos con una salida de dos neuronas con activación softmax una por clase. Además, aplicamos regularización L2 en las densas y sumamos dropout, como se ve en el script de entrenamiento, para contener el sobreajuste. Nada grandilocuente, pero efectivo: el aprendizaje se volvió más estable y la capacidad de generalización mejoró. Durante el finetuning se experimentó con distintos grados de “congelación” de parámetros. En algunos ensayos, únicamente las capas finales fueron reentrenadas, manteniendo fijas las proyecciones de los parches y la codificación posicional. En otros, todo el modelo se entrenó de manera conjunta. Se observó que permitir un ajuste completo ofrecía mejores resultados en este caso, probablemente porque el dominio de entrada (espectrogramas de tos) difiere de los conjuntos genéricos donde se suelen preentrenar estas arquitecturas.

5.16 Descripción de las métricas

Para valorar el rendimiento de los modelos se empleó un conjunto de métricas habituales en clasificación binaria, todas derivadas de la matriz de confusión 3x3 y sus cuatro casillas verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos— [62]. No hay misterio ahí: de esa tabla salen casi todos los indicadores que importan.

La primera es la tasa de acierto (accuracy), que indica qué proporción de predicciones fueron correctas sobre el total de muestras evaluadas. Fácil de interpretar pero puede llevar a errores cuando las clases están desbalanceadas algo bastante común en detección de tos, donde abundan los segmentos sin tos, porque un modelo que acierta casi siempre el no-tos puede inflar la cifra sin realmente detectar lo relevante.

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

En un contexto clínico pesan más sensibilidad y especificidad. La sensibilidad mide la capacidad del sistema para identificar correctamente los positivos. Si la sensibilidad cae, se escapan episodios clínicamente importantes; mal asunto para monitorización. La especificidad, por su parte, refleja cuántos negativos (no-tos) reconoce de forma correcta, reduciendo falsas alarmas que, de otro modo, terminarían saturando al personal. Estas métricas son especialmente relevantes en casos en los que el riesgo no es igual en cada error.

	Positive	TP	FN
Actual value			
Negative	Positive	FP	TN
	Predicted value		

Figura 36: Matriz de confusión para el caso de clasificación binaria

$$\text{Sensibilidad (Recall)} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Especificidad} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

La precisión muestra la relación de los positivos reales entre todos los positivos detectados. La métrica F1 consensa la precisión y sensibilidad mediante su media armónica. Esta es útil cuando ambas dimensiones importan y las clases no están balanceadas.

$$\text{Precisión (Precision)} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{F1} = 2 \cdot \frac{\text{Precisión} \cdot \text{Recall}}{\text{Precisión} + \text{Recall}}$$

Por último, se consideró el área bajo la curva ROC (AUC-ROC), que evalúa la capacidad discriminativa del modelo al recorrer distintos umbrales de decisión. El AUC ofrece una visión global del rendimiento, mostrando cómo varía la tasa de verdaderos positivos frente a la de falsos positivos al mover el umbral; en biomédica es casi estándar porque permite comparaciones razonablemente justas entre modelos con distinta calibración [9].

$$\text{AUC-ROC} = \int_0^1 \text{TPR}(\text{FPR}) \, d(\text{FPR})$$

5.17 Parámetros y pruebas realizadas

6 Implementación

6.1 Entrenamiento

Para entrenar los modelos se siguió una estrategia cuidadosa de monitorización de métricas y uso de técnicas de regularización con el fin de prevenir sobreajuste. En cada entrenamiento, se utilizó la pérdida por entropía cruzada binaria como función de costo para la tarea de detección de tos (clasificación binaria).

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^N \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

El optimizador empleado fue AdamW, una variante del algoritmo Adam que incorpora decaimiento de pesos desacoplado [?]. A diferencia de Adam clásico, AdamW aplica la regularización L^2 de forma independiente al cálculo del gradiente, lo que se ha reportado que mejora la generalización del modelo [?].

$$\theta_{t+1} = \theta_t - \eta \cdot \text{Adam}(g_t) - \eta \lambda \theta_t$$

- θ_t son los parámetros del modelo en el paso t ,
- $g_t = \nabla_{\theta} \mathcal{L}$ es el gradiente de la función de pérdida,
- $\text{Adam}(g_t)$ representa la actualización estándar de Adam (con sus promedios adaptativos),
- λ es el factor de decaimiento de pesos (regularización L^2),
- η es la tasa de aprendizaje.

Se inicializó la tasa de aprendizaje en 3×10^{-4} y se utilizó un "scheduler" para reducirla si la métrica de validación no mejoraba en varias épocas consecutivas. Además, se aplicó 'early-stopping'. Si después de 10 épocas el valor de pérdida de validación no mostraba mejora, el entrenamiento se detenía anticipadamente para evitar seguir ajustando sobre datos ya memorizados.

$$\eta \leftarrow \gamma \cdot \eta, \quad 0 < \gamma < 1$$

Si $\mathcal{L}^{(val)}$ no mejora durante P épocas consecutivas, \implies detener entrenamiento.

Cada entrenamiento se llevó a cabo por un máximo de 50 épocas como tope, con lotes de tamaño 32. Durante el entrenamiento, en cada época se calculaban las

métricas de precisión, sensibilidad, especificidad y AUC tanto sobre los datos de entrenamiento (mediante validación interna) como sobre el pliegue de validación correspondiente, para monitorizar el progreso. Todas estas precauciones permitieron detectar rápidamente el sobreajuste (ver apartado anterior) y comparar diferentes configuraciones de modelo.

Debido a la disponibilidad de una GPU para cómputo intensivo, se aprovechó la aceleración por hardware para reducir los tiempos de entrenamiento. Previamenente al inicio del entrenamiento, el entorno seleccionaba automáticamente la GPU con mayor memoria libre para realizar las operaciones (evitando así problemas si la máquina tenía varias GPUs ocupadas). En general, cada entrenamiento de 50 épocas tardó del orden de 8 a 10 segundos por época en la GPU seleccionada, procesando los $\sim 130k$ espectrogramas de 1 segundo en cada iteración.

$$g_t = \frac{1}{|B|} \sum_{i \in B} \nabla_{\theta} \ell(x_i, y_i; \theta), \quad |B| = 32$$

6.2 Optimización de hiperparámetros con Hyperband

Para la optimización de hiperparámetros de empleo el algoritmo Hyperband [45]. Esta estrategia destaca por su capacidad de explorar un gran número de combinaciones sin sacrificar el coste computacional. Comparando con la búsqueda por cuadrícula (grid search [37] o búsqueda aleatoria [6], Hyperband asigna recursos de forma adaptativa, empieza entrenando muchas configuraciones con pocas épocas y, tras medir su rendimiento, elimina agresivamente las peores para concentrar los esfuerzos en las mejores. Internamente, cada iteración ejecuta el procedimiento de "Sucessive Halving" [40]: en cada etapa se conserva aproximadamente una fracción $1/\eta$ de configuraciones y se multiplica por η el número de recursos por configuración.

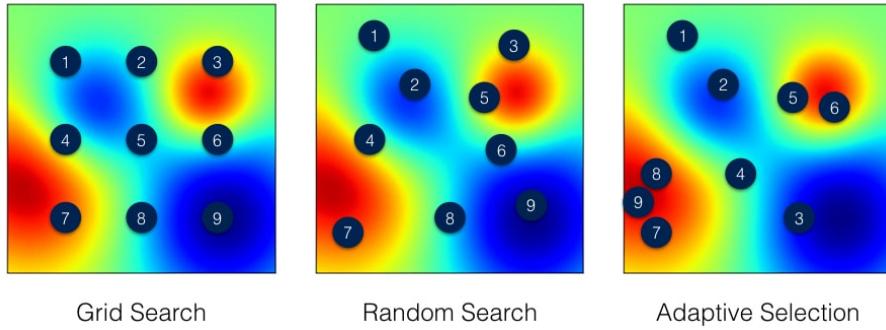


Figura 37: Estrategias de búsqueda de hiperparámetros. Izquierda: *grid search*. Centro: *random search*. Derecha: selección adaptativa tipo Successive Halving/Hyperband, que concentra recursos en regiones prometedoras del espacio.

Hyperband se controla con dos valores: R (máximo de épocas por configuración) y $\eta > 1$ (factor de reducción):

$$R := \text{máximo de épocas por configuración}, \quad \eta > 1. \quad (10)$$

El número de rondas es:

$$s_{\max} = \lfloor \log_{\eta} R \rfloor. \quad (11)$$

En la ronda $s \in \{s_{\max}, \dots, 0\}$ se comienza con:

$$n = \left\lceil \frac{s_{\max} + 1}{s + 1} \eta^s \right\rceil, \quad r = R \eta^{-s}. \quad (12)$$

En cada etapa $i = 0, \dots, s$:

$$n_i = \lfloor n \eta^{-i} \rfloor, \quad r_i = r \eta^i. \quad (13)$$

Aproximadamente, el coste por ronda es constante:

$$\sum_{i=0}^s n_i r_i \approx (s_{\max} + 1) R, \quad (14)$$

lo que reparte el presupuesto entre rondas con muchas configuraciones pero pocas épocas y con pocas configuraciones pero muchas épocas).

El tunner explora hiperparámetros clave de la arquitectura Transformer [81]: número de cabezas de atención, dimensión del *embedding*, tamaño de la capa *feed-forward*, tasa de *dropout* y tamaño de parche (*patch size*). La métrica objetivo es la **val_accuracy**. Con las Ecs. (12)–(13), la búsqueda procede así:

1. Generar n configuraciones con distintos valores de los hiperparámetros.
2. Entrenar cada configuración durante r épocas.
3. Ordenarlas por **val_accuracy**; descartar una fracción $1 - 1/\eta$ y reasignar $r\eta$ épocas a las supervivientes.
4. Repetir el proceso hasta completar la ronda.

Permite evaluar cientos de combinaciones sin entrenarlas por completo, con un ahorro sustancial de tiempo y coste computacional frente a alternativas clásicas [71].

El resultado de esta optimización fueron los siguientes hiperparámetros: ??

Hiperparámetro	Valor final
Tamaño de entrada	45×100 (espectrograma log-Mel)
Número de clases	2 (tos / no tos)
Tamaño de parche (PATCH_SIZE)	5
Dimensión del embedding (EMBED_DIM)	64
Número de cabezas de atención (NUM_HEADS)	2
Dimensión del feed-forward (FF_DIM)	384
Número de bloques Transformer	1
Dropout (DROPOUT_RATE)	0.1
Capa densa intermedia	128 neuronas, activación ReLU
Regularización	L2 ($\lambda = 0,01$) en capas densas
Capa de salida	Softmax (2 nodos)
Optimizador	AdamW
Tasa de aprendizaje	3×10^{-4}
Función de pérdida	Categorical Cross-Entropy
Batch size	128
Épocas máximas	50

Tabla 4: Hiperparámetros finales empleados en el modelo Transformer para clasificación de tos

6.3 Arquitectura propuesta

La arquitectura propuesta es un Transformer sobre representaciones tiempo-frecuencia de tamaño 45×100 (canal único). Se divide cada espectrograma en parches no solapados de 5×5 píxeles, generando $N = (45/5) \cdot (100/5) = 180$ tokens.

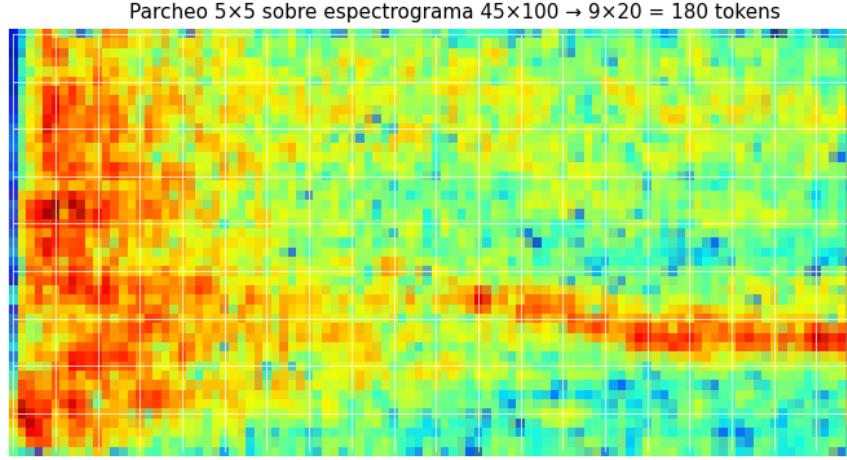


Figura 38: División en parches 5x5 de spectrograma de dimensiones 45×100

Cada parche se proyecta a un espacio de dimensión $d = 64$ mediante una capa densa (con regularización L2=0,01), y se suma una codificación posicional aprendida de la misma dimensión. Sobre esta secuencia se apila un bloque codificador Transformer con multi-atención de $h = 2$ cabezas (clave de dimensión d), seguida de normalización de capa, una red feed-forward de tamaño intermedio 384, y tasa de descarte 0,1 en la atención y en la proyección de salida. Finalmente, la secuencia codificada se aplana y pasa por una capa densa de 128 unidades (ReLU, L2=0,01) con tasa de descarte 0,1, y una capa softmax binaria para clasificar presencia/ausencia de tos. Todos estos valores se corresponden con la implementación empleada en este trabajo.

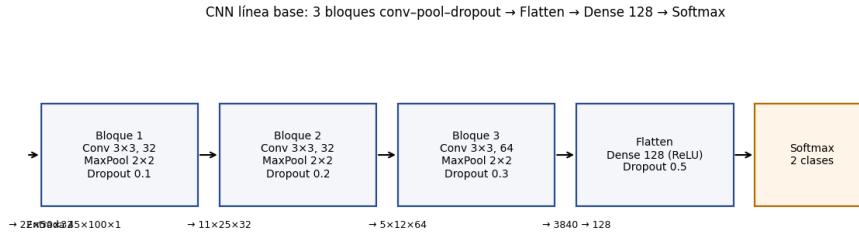


Figura 39: Bloque codificador Transformer con atención multicabeza ($h = 2$), normalización, red feed forward intermedia de 384 y conexión residual. Basado en [80].

El modelo se entrena con AdamW .tasa de aprendizaje $3 \cdot 10^{-4}$, función de pérdida categorical_crossentropy. Se aplica parada temprana con paciencia

de parada 5 épocas y un régimen máximo de 50 épocas, lote de 128 ejemplos, y validación cruzada de 5 particiones para estimar desempeño y variabilidad.

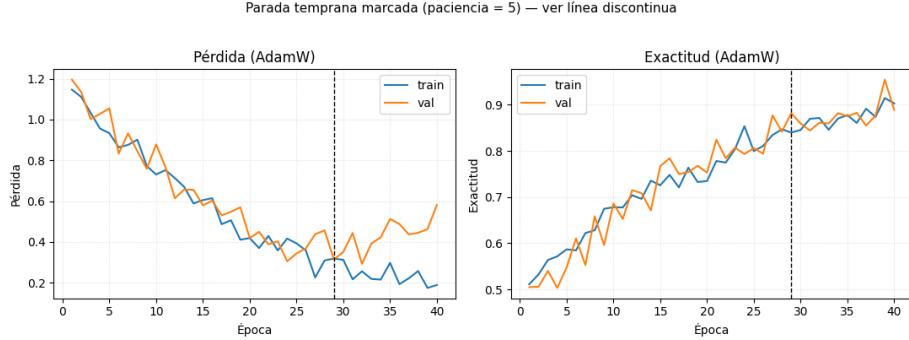


Figura 40: Curvas de entrenamiento con AdamW [48]. Se muestra la evolución de la pérdida y la precisión en entrenamiento y validación

Conceptualmente, el diseño sigue la línea de los Transformers para secuencias [80] y su adaptación a imágenes mediante parches (ViT) [20], trasladándola al dominio acústico como en los modelos AST [27]. La elección de AdamW responde a su mejor tratamiento del decaimiento de pesos frente a AdamW clásico [48]. Esta configuración equilibra capacidad de modelado y coste computacional, lo que es clave en un escenario con ruido real y variabilidad interpaciente, como el considerado en este TFG.

Tabla 5: Resumen de la arquitectura Transformer propuesta.

Característica	Configuración
Entrada	Espectrogramas $45 \times 100 \times 1$
Tokenización	Parches 5×5 , $N = 180$ tokens
Proyección de parches	Dense $d = 64$, L2=0.01
Codificación posicional	Aprendida
Bloques Transformer	1 bloque codificador
Atención	Multi-cabeza ($h = 2$), dropout 0.1
Feed-forward	Dimensión intermedia 384, ReLU
Normalización	LayerNorm + conexiones residuales
Capa densa	128 unidades, ReLU, L2=0.01, dropout 0.1
Salida	Softmax binaria
Optimizador	AdamW, lr $3 \cdot 10^{-4}$
Pérdida	categorical_crossentropy

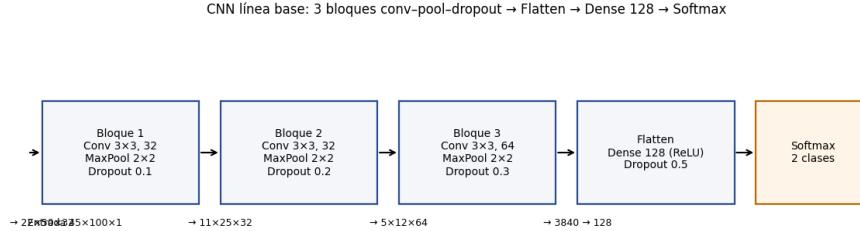


Figura 41: Arquitectura de la red convolucional empleada

6.4 Modelo CNN de referencia para comparación

Como línea base se emplea una CNN desarrollada en el Laboratorio de Procesado e Imagen sobre los mismos espectrogramas $45 \times 100 \times 1$. Es una red compacta con tres bloques convolucionales 3×3 y activación ReLU; los dos primeros tienen 32 filtros y el tercero 64.

Cada bloque incorpora max pooling 2×2 y dropout progresivo (0.1; 0.2; 0.3). Tras aplanar, se añade una densa de 128 unidades (ReLU) con dropout 0.5 y una salida softmax binaria. El modelo se compila con `categorical_crossentropy` y Adam, usando exactitud como métrica.

Para una comparación justa con el Transformer, se replica el protocolo de entrenamiento: lote 128, máximo 50 épocas y parada temprana sobre validación. Aun siendo deliberadamente sencilla, esta CNN resume buenas prácticas en clasificación de audio con espectrogramas, donde las convoluciones capturan patrones locales tiempo-frecuencia de forma eficaz [36, 60]. Sirve así como referencia interpretable y ampliamente reportada para cuantificar la ganancia atribuible a la atención y a la tokenización por parches del modelo propuesto.

Tabla 6: Resumen de la arquitectura CNN de referencia.

Característica	Configuración
Entrada	Especrogramas $45 \times 100 \times 1$
Nº Bloques	3
Dimensión convolución	2×2
Nº Filtros	32
Activación	ReLU
Pooling	MaxPool, 2×2
Tasa de descarte	0.1
Dimensiones Feed-Forward	128 unidades, ReLU, Dropout 0.5
Salida	Softmax binaria
Optimizador	Adam
Pérdida	categorical_crossentropy
Métrica	Exactitud
Entrenamiento	Batch 128, máx. 50 épocas, early stopping

7 Resultados

A continuación presentaremos los resultados más significativos encontrados durante el desarrollo del modelo, tanto para el problema de detección como de clasificación.

Para la detección, se analizaron los efectos de las estrategias de *data augmentation* de forma separada y conjunta. Se revisó cómo varió el sobreajuste del modelo bajo distintos conjuntos de datos y condiciones de entrenamiento.

Más adelante se contrastaron los resultados con arquitecturas previamente desarrolladas por el Laboratorio de Procesado e Imagen anteriormente mencionado, sin olvidar las particularidades del problema específico a tratar.

En cuanto a la clasificación, primero se plantea el problema de la definición de las clases clínicas, fundamental crear clases balanceadas para generar conjuntos de entrenamiento de calidad. Seguidamente, se describen dos enfoques distintos; comparaciones por pares entre clases, o un transformer multiclas que compare todas con todas.

Finalmente, como en el problema anterior, se presentan comparativas con la CNN. En conjunto, esta sección pretende ofrecer una mirada amplia al rendimiento del sistema, permitiendo tanto reconocer sus aciertos como señalar con claridad sus puntos débiles.

7.1 Detección

7.2 Impacto de las técnicas de *data augmentation*

Para medir el efecto de cada estrategia se partía del dataset original y se añadía, por separado, una única técnica en cada experimento. Este diseño permite aislar el impacto de cada transformación y entender qué tipo de variación aporta robustez al clasificador, antes de combinarlas. La motivación es estandarizar en aprendizaje profundo: la augmentation actúa como regularizador, amplía el soporte del conjunto de entrenamiento y reduce el sobreajuste al acercar el entrenamiento a las condiciones reales de despliegue [28, 14]. En audio, además, está bien documentado que los modelos mejoran cuando se simulan entornos acústicos y variaciones de la fuente [36, 53].

Tabla 7: Resultados del dataset original (sin augmentación).

Modelo	Exactitud	Memoria (MB)	Tiempo (s)	Sensibilidad	Especificidad	AUC
model-0	0.898565	41.988613	6.207804	0.884169	0.912962	0.949180
model-1	0.883380	41.987980	4.752613	0.875092	0.891668	0.942415
model-2	0.861703	41.987823	4.410996	0.826290	0.897115	0.927812
model-3	0.887202	41.987988	4.568282	0.866137	0.908266	0.944801
model-4	0.884197	41.988087	4.527950	0.873589	0.894805	0.939941
Media	0.883809	41.988098	4.893929	0.865056	0.901363	0.940830

La **inyección de ruido** introduce variaciones de fondo que ayudan a no confundir ruido ambiente con tros. En la práctica, observamos validaciones más estables y una reducción del *gap* entrenamiento–validación en la tabla de métricas respecto al *baseline*.

Tabla 8: Resultados con augmentación: *Noise Injection*.

Modelo	Exactitud	Memoria (MB)	Tiempo (s)	Sensibilidad	Especificidad	AUC
model-0	0.887780	41.988609	5.998950	0.875585	0.899975	0.949306
model-1	0.889183	41.987923	4.670399	0.888461	0.889906	0.955539
model-2	0.804654	41.987679	4.386551	0.666070	0.943237	0.851726
model-3	0.860901	41.988098	4.538124	0.816809	0.904994	0.933198
model-4	0.884246	41.987991	4.591663	0.881844	0.886647	0.949502
Media	0.865353	41.988060	4.837137	0.825754	0.904952	0.927854

Métricas obtenidas con **pitch scaling**;

Tabla 9: Resultados con augmentación: *Pitch Scaling*.

Modelo	Exactitud	Memoria (MB)	Tiempo (s)	Sensibilidad	Especificidad	AUC
model-0	0.956229	41.988594	25.888992	0.951425	0.961033	0.988180
model-1	0.958163	41.988022	25.071600	0.949570	0.966755	0.986128
model-2	0.966654	41.987709	24.566749	0.942939	0.990369	0.986905
model-3	0.956557	41.987816	25.212633	0.936825	0.976289	0.984442
model-4	0.954571	41.987778	24.493543	0.950448	0.958694	0.986260
Media	0.958435	41.987984	25.046703	0.946241	0.970628	0.986383

Aunque las métricas parecen indicar que la implementación de estas técnicas por separado mejoró el rendimiento de los modelos, es conveniente analizar la evolución del error en los conjuntos de test y validación para saber si este generaliza bien.

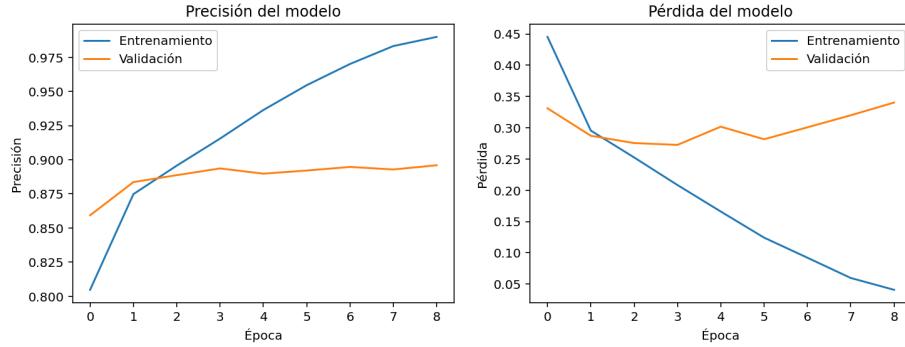


Figura 42: Entrenamiento sin datos sintéticos

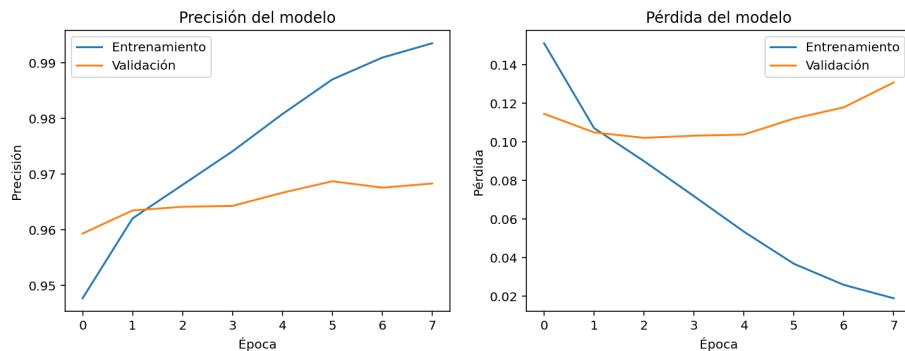


Figura 43: Augmentación con desplazamiento de tono

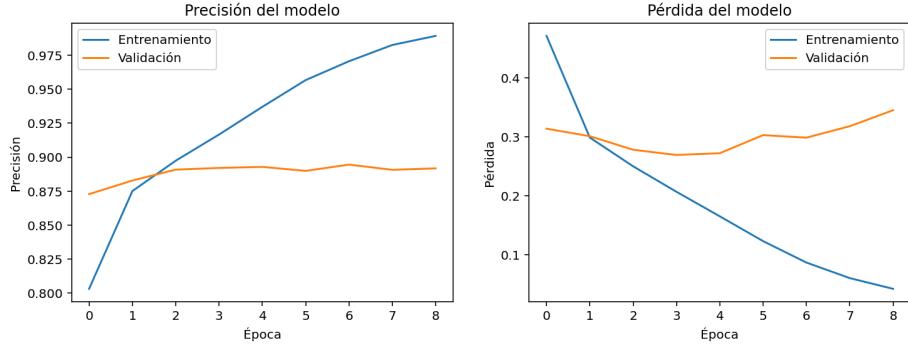


Figura 44: Aumentación con volume scaling

La **convolución con RIR** reproduce salas y reverberaciones diferentes; aunque el entrenamiento es algo más inestable, la validación resulta más realista (Fig. 45). En contraste, el modelo sin datos sintéticos muestra un incremento sostenido de la precisión de entrenamiento mientras la validación se estanca, señal clara de sobreajuste (Fig. 42).

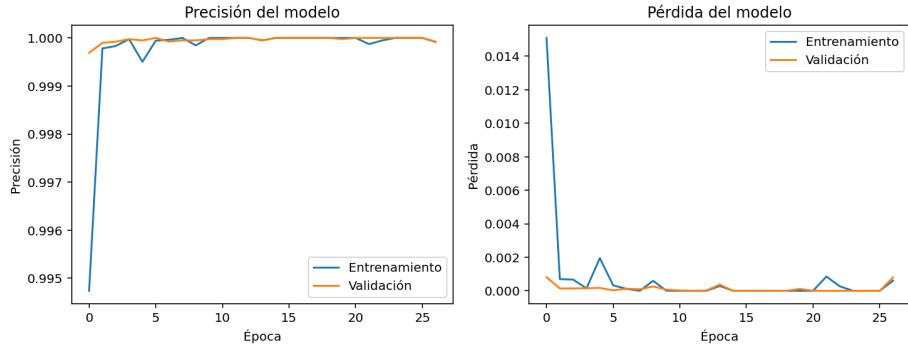


Figura 45: Aumentación con convolución con respuesta al impulso

Probar cada técnica por separado fue útil para identificar su efecto sobre sensibilidad especificidad y decidir combinaciones. En conjunto, los aumentos actúan de forma complementaria: ruido y RIR protegen frente al entorno; el tono y volumen cubren variabilidad intrapaciente.

Sin embargo, en todas ellas la precisión en entrenamiento sigue subiendo de forma sostenida (hasta casi el 100 %), mientras que la precisión en validación se estanca muy pronto y no mejora de manera significativa. De forma paralela, la pérdida en entrenamiento disminuye rápidamente, pero la pérdida en validación se estabiliza e incluso muestra ligeras oscilaciones o tendencia ascendente. Esto indica que el modelo memoriza los datos de entrenamiento en lugar de aprender

representaciones generalizables.

7.3 Detección sobre el conjunto de datos completo

Una vez comprobado el efecto de las técnicas de aumento de datos por separado, se emplearon el conjunto completo generado. Como se mencionó anteriormente el objetivo de esto es aumentar la variabilidad de las condiciones acústicas y eliminar el desbalance de clases, reduciendo así sesgos en el entrenamiento. La generación de datos sintéticos compensó la escasez de ejemplos reales y permitió acercar el modelo a escenarios prácticos, mientras que el muestreo circular evitó que las clases quedaran desbalanceadas.

Además, se presentan las curvas de error en validación y prueba, así como las métricas antes y después del ajuste de parámetros, lo que facilita evidenciar posibles signos de sobreajuste y valorar la mejora lograda tras la optimización.

7.3.1 Curvas de pérdida y precisión

En la Figura 46 se muestran las curvas de entrenamiento del primer intento con el dataset completo, tras aplicar data augmentation y muestreo circular. Puede apreciarse un indicio de sobreajuste: la precisión en entrenamiento asciende progresivamente hasta superar el 98 %, mientras que en validación se estabiliza en torno al 92–93 %.

De forma paralela, la pérdida de entrenamiento desciende con rapidez, mientras que la de validación se mantiene casi constante e incluso tiende a aumentar después de algunas épocas. Este comportamiento sugiere que el modelo aprende muy bien los patrones del conjunto de entrenamiento, pero empieza a perder capacidad de generalización a medida que se prolonga el entrenamiento.

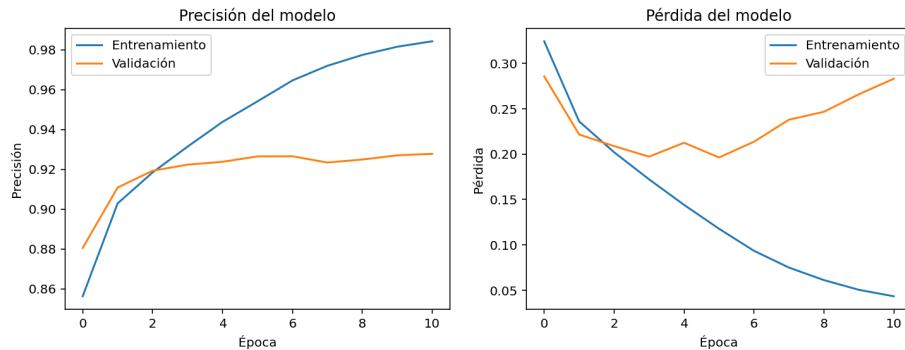


Figura 46: Curvas de error Test-Val antes de el ajuste de hyperparámetros

7.3.2 Rendimiento medio en validación cruzada

Las métricas agregadas en validación cruzada (Tabla 10) refuerzan este diagnóstico. Con una media de exactitud del 90.37 % y un AUC de 0.962 (véase Tabla 10).. Sin embargo, al analizar sensibilidad (88.0 %) y especificidad (92.7 %) se detectan oscilaciones relevantes entre pliegues, probablemente asociadas al sesgo por paciente.

La validación cruzada permitió detectar este fenómeno y confirmar que el modelo no era robusto en todos los escenarios, alineándose con las recomendaciones de Kohavi [43] para una estimación fiable en entornos biomédicos.

Tabla 10: Resultados con augmentación: *Primer intento*.

Modelo	Exactitud	Memoria	Tiempo	Sensibilidad	Especificidad	AUC
model-0	0.909756	38.155113	35.408004	0.858574	0.960938	0.968243
model-1	0.867391	38.155472	31.671340	0.830047	0.904735	0.940538
model-2	0.897819	38.155769	30.461218	0.855093	0.940545	0.954534
model-3	0.883181	38.155602	32.045799	0.859266	0.907095	0.942245
model-4	0.864142	38.155762	35.124707	0.819244	0.909039	0.938732
Media	0.884458	38.155544	32.942214	0.844445	0.924470	0.948858

7.3.3 Análisis del sobreajuste

El sobreajuste observado puede atribuirse a varias causas: complejidad excesiva de la arquitectura, regularización insuficiente y carencia de un ajuste de hiperparámetros. Este comportamiento es común en tareas biomédicas con datos heterogéneos, donde el modelo tiende a aprender correlaciones ligadas a condiciones de grabación o a características propias de algunos pacientes [62].

Para reducir el efecto del sobreajuste, se realizó un estudio sobre los motivos de este sobreajuste en los datos y parámetros del modelo.

7.4 Comparativa tras el ajuste de parámetros

Para mitigar este efecto se realizó una fase de ajuste de hiperparámetros (*fine-tuning*) utilizando el algoritmo Hyperband [45], que permitió seleccionar de manera eficiente combinaciones más eficientes. Los resultados tras este proceso (Tabla 11) muestran una mejora clara en la estabilidad.

La exactitud media se situó en 88.6 % con un AUC de 0.951, ligeramente inferior en términos absolutos al primer intento, pero mucho más equilibrada entre pliegues. Además, la sensibilidad (86.4 %) y la especificidad (90.9 %) quedaron en rangos similares, lo que indica que el modelo aprendió representaciones más generalizables.

En la Figura ?? se aprecia cómo las curvas de entrenamiento y validación se mantienen próximas durante las épocas, signo de un aprendizaje más estable y sin sobreajuste marcado.

7.4.1 Curvas de pérdida y precisión

Para analizar el rendimiento de los modelos, no basta con revisar las métricas obtenidas, sino la coherencia de su comportamiento durante el entrenamiento. Un buen modelo es aquel cuya precisión en validación es similar a la de entrenamiento, lo que indica capacidad de generalización y ausencia de sobreajuste. Además, sus curvas deben ser estables, sin oscilaciones bruscas, y la pérdida de validación debe mantenerse baja y converger de forma clara. En conjunto, estos indicadores reflejan que el modelo aprende patrones relevantes y no memoriza datos específicos.

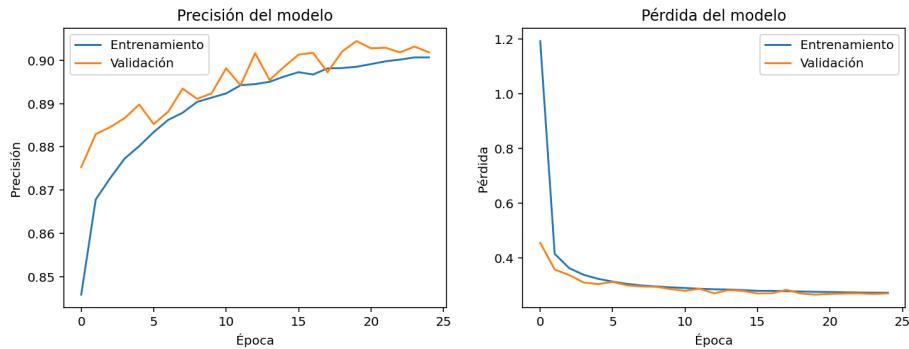


Figura 47: Curvas de error Test-Val tras de el ajuste de hyperparámetros

7.4.2 Rendimiento medio en validación cruzada

Es necesario comprobar que la reducción de complejidad del modelo y el ajuste de parámetros en el modelo no repercute negativamente sobre las métricas del modelo.

Tabla 11: Resultados con augmentación: *Pitch Scaling*.

Modelo	Exactitud	Memoria	Tiempo	Sensibilidad	Especificidad	AUC
model-0	0.917346	41.987862	56.776823	0.880288	0.954404	0.972745
model-1	0.893508	41.987823	49.729802	0.886036	0.900979	0.956818
model-2	0.913494	41.987679	42.494498	0.897491	0.929497	0.965565
model-3	0.907777	41.987976	41.748255	0.909502	0.906053	0.965480
model-4	0.886557	41.988083	41.649906	0.827923	0.945190	0.951117
Media	0.903736	41.987885	46.479857	0.880248	0.927225	0.962345

7.4.3 Análisis del sobreajuste

Las causas identificadas de sobreajuste y sus soluciones fueron:

- *Exceso de capacidad del modelo*: El Transformer inicial tenía más cabezas y parámetros de los necesarios para la cantidad de datos disponible. Esto le permitía memorizar variaciones específicas de ciertos pacientes. La solución fue simplificar la arquitectura (reducción de 5.5M a 1.2M parámetros), lo que obligó al modelo a usar su capacidad de manera más general. Tras esto, la variabilidad de métricas entre folds disminuyó significativamente, señal de que el modelo ya no se ajustaba solo a ciertos datos.
- *Datos insuficientemente variados*: Sin augmentation adecuado, el modelo podía aprender características que no se sostienen fuera del conjunto entrenado. La introducción de pequeñas variaciones rompió estos patrones propios.
- *Data leakage* : Se verificó que la implementación de GroupKFold estaba correcta, para asegurar que no hubiese inadvertidamente muestras de un mismo paciente en entrenamiento y validación. Esto habría inflado artificialmente las métricas de entrenamiento haciéndolas parecer buenas pero luego fallando en test. Aunque , no se encontró fuga de datos, la precaución quedó incorporada en el pipeline de forma permanente.

En resumen, el contraste entre ambos experimentos demuestra la importancia del ajuste de hiperparámetros y de la regularización en la clasificación de señales biomédicas. Aunque los valores brutos de exactitud pudieran parecer mejores en la configuración inicial, el modelo ajustado ofrece una generalización mucho más sólida, condición imprescindible para su aplicación en entornos clínicos reales.

7.4.4 Compatativa Transformer vs CNN base

Para entender las ventajas del enfoque Transformer, se comparó su rendimiento con el modelo CNN de referencia descrito en la metodología. Ambos modelos fueron entrenados y evaluados bajo las mismas condiciones (particiones y conjunto de datos).

Los resultados mostraron que el Transformer superó a la CNN en la mayoría de métricas:

- La **precisión** media de la CNN fue de 85,2 %, unos 3 puntos porcentuales inferior a la del Transformer (88,4 %).
- La **sensibilidad** de la CNN se quedó en ~ 78 %, bastante por debajo del 86 % del Transformer. Esto indica que la CNN perdió más toses (falsos negativos) que el Transformer. Analizando los errores, se vio que la CNN tenía dificultad con toses muy cortas o atípicas, a menudo confundiéndolas con ruido. El Transformer, gracias a su mecanismo de atención global, capturaba mejor esas señales sutiles en contexto.

- La **especificidad** de la CNN fue similar, alrededor de 90 %. Es decir, ambos modelos controlaron bien los falsos positivos, quizás porque distinguir “silencio/ruido” de “tos” es relativamente fácil una vez aprendidas las características básicas. La mejora principal del Transformer estuvo en la sensibilidad, detectando más verdaderas tos.
- La **AUC** de la CNN (0.93 promedio) también estuvo ligeramente por debajo de la del Transformer (0.95), lo que confirma que globalmente el Transformer tiene mejor capacidad discriminativa.

Además de las métricas, se comparó la eficiencia: el modelo Transformer es algo más pesado computacionalmente que la CNN (1.2M vs 0.5M parámetros), pero en inferencia ambos operan en tiempo prácticamente real en la GPU («1 s por 1000 muestras). En CPU, la CNN es más rápida, pero aún el Transformer procesaría decenas de muestras por segundo, suficiente para un sistema online moderado.

»Esta comparativa sugiere que el **Transformer aportó valor en capturar dependencias de larga duración** en los espectrogramas que la CNN pudo haber pasado por alto. Por ejemplo, en secuencias de tos múltiple, la CNN a veces marcaba solo una de ellas correctamente, mientras que el Transformer detectaba tanto la primera como la segunda, quizás porque “entendió” la relación temporal (toses que vienen en ráfaga). También en presencia de ruido transitorio (p.ej. un golpe), la CNN se confundía más, mientras que el Transformer aprendió a ignorarlo posiblemente atendiendo a otras partes del espectrograma menos afectadas.

»En cualquier caso, la CNN sigue siendo un modelo más sencillo y podría ajustarse con técnicas adicionales (p.ej. más capas, mejor normalización). No obstante, con la configuración escogida, el **Transformer se confirma como la mejor opción** para nuestro caso, justificada la complejidad ligeramente mayor por su mayor rendimiento. Esto está en línea con tendencias recientes que muestran Transformers superando a CNNs en tareas de audio cuando hay suficientes datos o augmentation para entrenarlos correctamente .

»7.5 Clasificación

»El conjunto de datos empleado en este trabajo está compuesto por registros clínicos etiquetados en cuatro categorías correspondientes a diferentes enfermedades respiratorias: *Cáncer, Asma, EPOC y Ag.*

»El primer paso sería la definición del problema y las clases. El principal problema encontrado durante el análisis de los datos fue el desbalance de datos entre las distintas clases. Este desbalance puede afectar negativamente tanto el proceso de entrenamiento como la capacidad del modelo para generalizar. En pasados estudios se comprobó que se tiende a desfavorecer a las clases menos representadas [33].

»HISTOGRAMA Y COMENTARIO DEL MISMO

»Para ilustrar esta distribución desigual, se incluye un histograma que refleja la frecuencia de instancias por clase, facilitando así la visualización del grado de desbalance presente en el conjunto de datos.

»7.6 Descripción médica de clases clínicas

»La tos clínica se describe por su timbre, sonoridad, duración de la fase espiratoria y por la presencia de componentes “musicales” o húmedos. La señal suele mostrar una descarga explosiva inicial, seguida de turbulencia y una cola de derramamiento, rasgos útiles para comparar patologías [54, 69]. En EPOC predomina una tos crónica y productiva, con ráfagas y energía relativamente baja asociada a secreciones y colapso espiratorio; los espectrogramas ayudan a distinguirla de otras entidades [69, 82]. En infecciones agudas la tos tiende a ser húmeda y en salvas por hipersensibilidad y aumento de moco. En cáncer de pulmón la tos es persistente pero heterogénea y rara vez aporta un patrón acústico por sí sola, por lo que se integra con clínica e imagen [35].

»La evidencia actual indica que, aunque existen patrones útiles para clasificación asistida por computadora, no hay rasgos acústicos inequívocamente por enfermedad. Por ello, el análisis de tos se integra con clínica, espirometría y técnicas de imagen [35, 69].

»7.6.1 Aproximación al problema con enfrentamiento uno frente a uno

»Como paso inicial hacia un clasificador multiclasa robusto, se adoptó un esquema de descomposición en enfrentamientos binarios *uno frente a uno* (OvO). Este enfoque transforma un problema con K clases clínicas en $K(K - 1)/2$ problemas de dos clases, entrenados de manera independiente con la misma canalización de preprocessado y extracción de características. La estrategia OvO es un estándar contrastado en la literatura por su capacidad para enfatizar fronteras de decisión finas entre pares de categorías cercanas y por su buen rendimiento con modelos lineales y no lineales, especialmente en escenarios con clases no disjuntas. [39, 61, 66].

»Este planteamiento resulta especialmente útil en un contexto clínico, donde interesa examinar de forma específica la separabilidad entre patologías que tienden a confundirse. El entrenamiento y la evaluación por pares permiten estimar, con métricas apropiadas, la discriminabilidad entre dos diagnósticos concretos y localizar patrones de error que podrían quedar enmascarados en la agregación multiclasa. Para cuantificar el rendimiento de manera coherente entre pares, se emplean métricas insensibles al umbral y compatibles con la extensión multiclasa, como el área bajo la curva ROC calculada mediante comparaciones por pares [31].

»El enfoque OvO contribuye además a manejar el desbalance global del conjunto de datos. Aunque no elimina el problema, trabajar con dos clases a la vez facilita controlar la proporción de ejemplos en cada enfrentamiento. Estas prácticas

están ampliamente respaldadas para aprendizaje con clases desbalanceadas y mejoran la sensibilidad sin degradar de forma marcada la especificidad [34, 11]. En nuestro caso, se fijó para cada par una política de balanceo consistente con las prevalencias clínicas y con las restricciones de validación cruzada por paciente, de forma que las comparaciones fuesen justas y replicables.

»En términos computacionales, el coste de entrenamiento crece cuadráticamente con el número de clases, pero los modelos binarios son paralelizables y las estrategias de inferencia eficiente, como los grafos acíclicos dirigidos de márgenes grandes (*DAGSVM*), reducen el número de comparaciones necesarias en tiempo de prueba [61]. En conjunto, el enfrentamiento uno frente a uno constituye un paso intermedio sólido para estudiar la estructura del problema, diagnosticar las confusiones más probables y fundamentar el diseño del modelo multiclasificación definitivo sobre evidencias empíricas controladas.

»**7.6.2 Resultados completos por pares, métricas prioritarias y riesgo de sobreajuste**

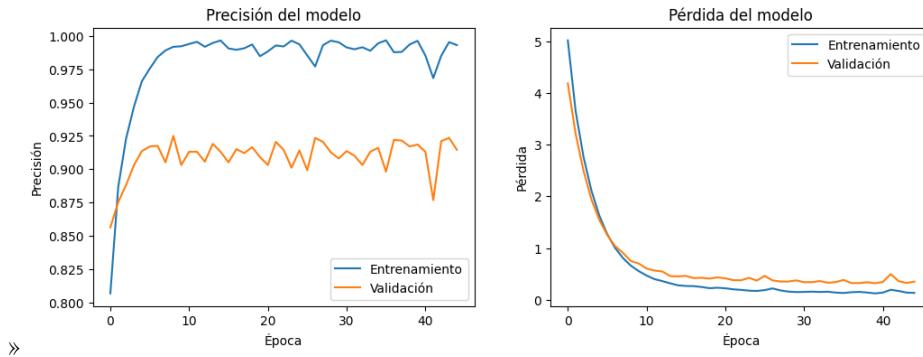
»Los seis enfrentamientos muestran un patrón estable con AUC altos en la mayoría de pares, cercanos a 0.97 en E frente a Ag, E frente a As y E frente a C, y un descenso claro en C frente a As, con AUC próximo a 0.92. La sensibilidad supera a la especificidad de forma sistemática en casi todos los pares, lo que sugiere un sesgo operativo que favorece la detección de la clase positiva. Dado que la exactitud puede resultar engañosa en presencia de desbalance y prevalencias clínicas desiguales, la interpretación se apoya en AUC, sensibilidad y especificidad equilibradas, *balanced accuracy*, F1 y en el coeficiente de correlación de Matthews, que resume la concordancia en los cuatro cuadrantes de la matriz de confusión [24, 68, 13, 63].

»Las curvas de aprendizaje muestran pérdidas que convergen con rapidez y una ganancia marginal tras la época veinte. En varios pares se observa una brecha sostenida entre la precisión de entrenamiento, muy próxima a la unidad, y la de validación en torno a 0.90. Este comportamiento es compatible con sobreajuste moderado en conjuntos del orden de miles de muestras por clase y con elevada variabilidad intraclasa. Para reducirlo, conviene fijar parada temprana sobre la pérdida de validación, aplicar regularización L_2 , *dropout* y *label smoothing*, y reforzar la variabilidad con *SpecAugment* y *mixup* en espectrogramas [64, 74, 55, 57, 84].

»A efectos de uso clínico, la recomendación es calibrar probabilidades, ajustar el umbral de decisión al coste de error del escenario y reportar rendimiento con validación cruzada por paciente, media y desviación estándar, e intervalos de confianza del 95 por ciento para la AUC mediante el método de DeLong [17]. Cuando la prevalencia de la clase positiva sea baja, las curvas de precisión frente a exhaustividad aportan una imagen más fiel que la curva ROC [68].

» Tabla 12: Resultados enfrentamiento E vs Ag.

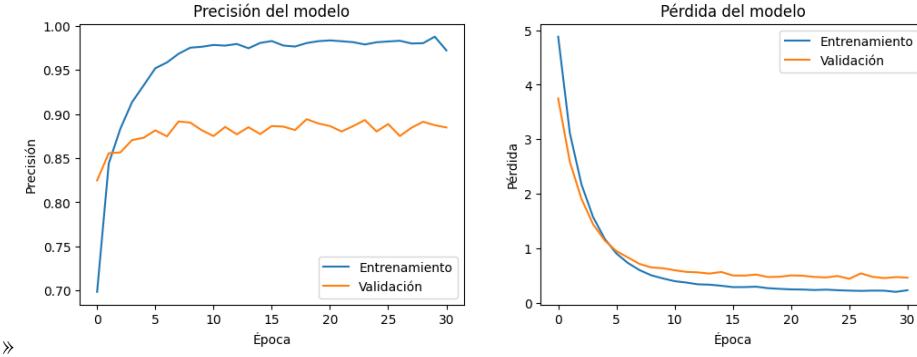
Modelo	»Exactitud	»Memoria (MB)	»Tiempo (s)	»Sensibilidad	»Especificidad	AUC
model-0	»0.905752	»18.187950	»1.369363	»0.913647	»0.895119	»0.966702
model-1	»0.909292	»18.188393	»1.814836	»0.936006	»0.873313	»0.969808
model-2	»0.913034	»18.188240	»1.272233	»0.921712	»0.901350	»0.969402
model-3	»0.890684	»18.188499	»1.301307	»0.860393	»0.931464	»0.966158
model-4	»0.890241	»18.188438	»1.231583	»0.967219	»0.786604	»0.966224
Media	»0.901801	»18.188304	»1.397865	»0.919796	»0.877570	»0.967659



» Figura 48: Evolución del error en los conjuntos de test y validación. Epoc vs Asma

» Tabla 13: Resultados enfrentamiento Ag vs As.

Modelo	»Exactitud	»Memoria (MB)	»Tiempo (s)	»Sensibilidad	»Especificidad	AUC
model-0	»0.886293	»18.188126	»2.791990	»0.930945	»0.841641	»0.957016
model-1	»0.886033	»18.188240	»1.179202	»0.894081	»0.877985	»0.953409
model-2	»0.868899	»18.188332	»1.112855	»0.865005	»0.872793	»0.946303
model-3	»0.888889	»18.188408	»1.151582	»0.934579	»0.843198	»0.958428
model-4	»0.870717	»18.188309	»1.144811	»0.936137	»0.805296	»0.951896
Media	»0.880166	»18.188283	»1.476088	»0.912150	»0.848183	»0.953410



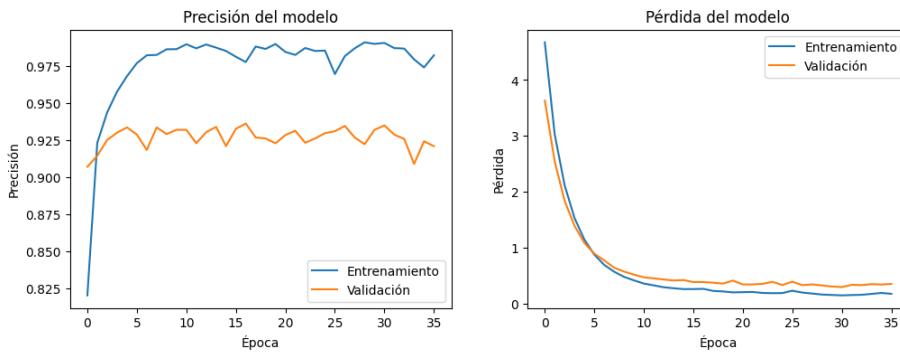
»Figura 49: Evolución del error en los conjuntos de test y validación. Ag vs
»Asma

»

»Tabla 14: Resultados enfrentamiento Ag vs C.

Modelo	»Exactitud	»Memoria (MB)	»Tiempo (s)	»Sensibilidad	»Especificidad	AUC
model-0	»0.923936	»18.188065	»1.876653	»0.928868	»0.919003	»0.980087
model-1	»0.928609	»18.188339	»1.154938	»0.949117	»0.908100	»0.977728
model-2	»0.928609	»18.188240	»1.209921	»0.921080	»0.936137	»0.979288
model-3	»0.923416	»18.188408	»1.173686	»0.904465	»0.942368	»0.982002
model-4	»0.909398	»18.188370	»1.153677	»0.968328	»0.850467	»0.976583
Media	»0.922793	»18.188284	»1.313775	»0.934372	»0.911215	»0.979137

»



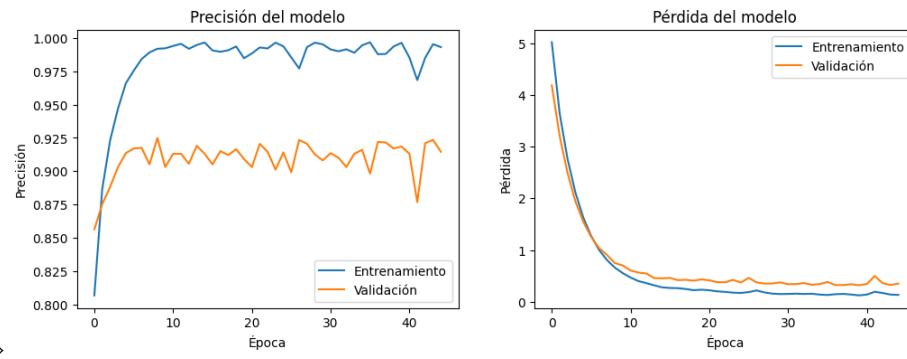
»Figura 50: Evolución del error en los conjuntos de test y validación. Ag vs
»Cáncer

»

»Tabla 15: Resultados enfrentamiento E vs As.

Modelo	»Exactitud	»Memoria (MB)	»Tiempo (s)	»Sensibilidad	»Especificidad	AUC
model-0	»0.911800	»18.187950	»1.746633	»0.969023	»0.854531	»0.978042
model-1	»0.894319	»18.188377	»0.938668	»0.852146	»0.936458	»0.972915
»	»0.912957	»18.188362	»0.949458	»0.958665	»0.867250	»0.976059
model-3	»0.903418	»18.188560	»1.032717	»0.964229	»0.842607	»0.970428
model-4	»0.905803	»18.188347	»0.882190	»0.881558	»0.930048	»0.972977
Media	»0.905659	»18.188319	»1.109933	»0.925124	»0.886179	»0.974084

»



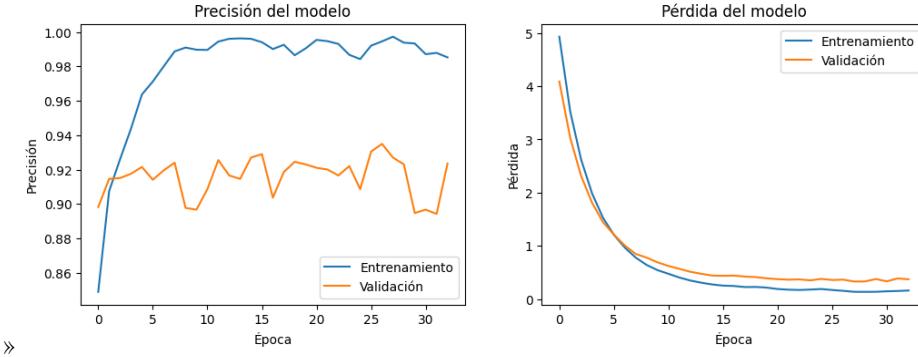
»

»Figura 51: Evolución del error en los conjuntos de test y validación. Epoc vs
»Asma

»

»Tabla 16: Resultados enfrentamiento E vs C.

Modelo	»Exactitud	»Memoria (MB)	»Tiempo (s)	»Sensibilidad	»Especificidad	AUC
model-0	»0.922924	»18.188011	»1.776406	»0.952343	»0.893482	»0.975003
model-1	»0.922924	»18.188179	»0.897751	»0.925278	»0.920572	»0.979116
»	»0.916137	»18.188370	»0.830451	»0.973768	»0.858506	»0.975043
model-3	»0.917329	»18.188400	»0.892776	»0.953100	»0.881558	»0.974665
model-4	»0.923688	»18.188354	»0.915651	»0.978537	»0.868839	»0.975644
Media	»0.920600	»18.188263	»1.062607	»0.956605	»0.884591	»0.975894



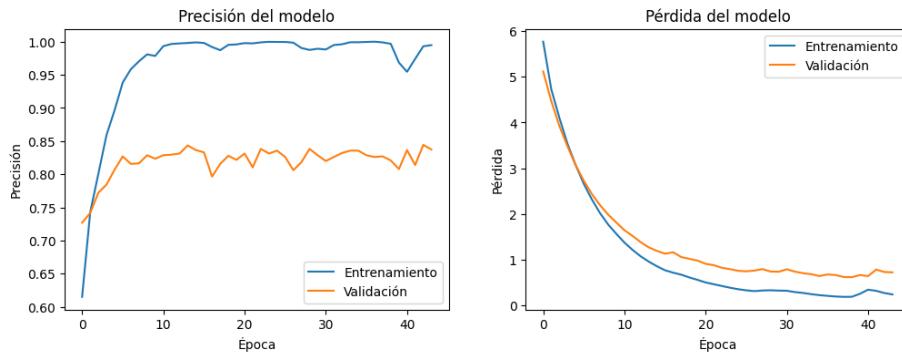
»Figura 52: Evolución del error en los conjuntos de test y validación. Epoc vs »Cancer

»

»Tabla 17: Resultados enfrentamiento C vs As.

Modelo	»Exactitud	»Memoria (MB)	»Tiempo (s)	»Sensibilidad	»Especificidad	AUC
model-0	»0.847599	»18.187935	»1.425333	»0.851182	»0.844011	»0.928420
model-1	»0.826026	»18.188232	»0.698666	»0.766017	»0.885953	»0.913132
model-2	»0.836351	»18.188408	»0.745568	»0.871866	»0.800836	»0.920993
model-3	»0.804318	»18.188301	»0.740501	»0.756267	»0.852368	»0.907481
model-4	»0.846797	»18.188370	»0.693528	»0.824513	»0.869081	»0.930610
Media	»0.832218	»18.188249	»0.860719	»0.813969	»0.850450	»0.920127

»



»Figura 53: Evolución del error en los conjuntos de test y validación. Cancer vs »Asma

»7.6.3 Adaptación del modelo al problema multiclas

»Tras caracterizar las fronteras de decisión por pares, el problema se formuló como una clasificación multiclas que abarca simultáneamente las cuatro pato-

logías. La arquitectura mantiene el bloque de autoatención multi-cabeza y la incrustación de parches con codificación posicional, pero se sustituyó la capa de salida binaria por una capa densa con cuatro neuronas y activación *softmax*, de modo que el modelo produce un vector de probabilidades normalizado en el simplex. La función de pérdida principal es la entropía cruzada categórica, criterio equivalente a la maximización de la verosimilitud bajo una suposición de clase verdadera codificada en caliente y ampliamente aceptado en clasificación multiclas [8, 29].

»Para favorecer la generalización se introdujeron regularizadores y tácticas de entrenamiento coherentes con la literatura reciente en modelos de atención y visión por transformadores. El bloque de proyección y las capas de alimentación se penalizan con norma cuadrática, se emplea *dropout* en atención y en las capas densas, y se entrena con AdamW, que desacopla el decaimiento de pesos de la estimación del momento y mejora la estabilidad frente a sobreajuste en tamaños de datos moderados [47, 81, 19]. La parada temprana monitoriza la pérdida de validación con restauración de los mejores parámetros. Para aumentar la diversidad muestral en el dominio de espectrogramas se recomienda enmascarado temporal y frecuencial del tipo *SpecAugment* así como mezclas convexas *mixup* entre ejemplos, dos técnicas que reducen la varianza y suavizan las fronteras de decisión [57, 84]. Adicionalmente, la suavización de etiquetas atenúa el exceso de confianza del *softmax* y suele mejorar la calibración de probabilidades [55, 30].

»El desbalance entre enfermedades se trató en el objetivo y en el muestreo. En la pérdida se introducen ponderaciones por clase inversamente proporcionales a la frecuencia, alternativa simple y efectiva. En escenarios con desbalance severo puede sustituirse por pérdida focal, que aumenta el peso de los ejemplos difíciles y minoritarios, o por su variante con *reweighting* según número efectivo de muestras [46, 15]. En el entrenamiento por lotes se aplicó muestreo estratificado por paciente, preservando la independencia entre pliegues y evitando fugas que inflarían el rendimiento. La evaluación se reporta con métricas macro promediadas, que asignan el mismo peso a cada clase, junto con AUC por pares y exactitud equilibrada, para ofrecer una visión robusta cuando las prevalencias difieren entre grupos. Finalmente, la experiencia del esquema OvO se reutiliza calibrando la salida multiclas y, cuando es pertinente, ajustando el umbral operativo por clase de acuerdo con el coste clínico de los errores.

»7.6.4 Comparativa Transformer vs CNN base

»8 Conclusiones

»8.1 Relevancia del Data Augmentation

»8.2 Limitaciones del estudio

»8.3 Aplicabilidad médica

»9 Futuras líneas de trabajo

» A Anexos

» Referencias

- [1] »Image classification — hugging face transformers. https://huggingface.co/docs/transformers/en/tasks/image_classification. Accedido: 9-Sep-2025.
- [2] »Transfer learning with yamnet for environmental sound classification. https://www.tensorflow.org/tutorials/audio/transfer_learning_audio. Actualizado: 16-Aug-2024; Accedido: 9-Sep-2025.
- [3] »ar5iv - transformer scalability and rnn comparison. Disponible en ar5iv.labs.arxiv.org, 2025. Consultado en 2025.
- [4] »An example of convolution operation in 2d. ResearchGate figure, 2025. Figura extraída del artículo relacionado disponible en ResearchGate.
- [5] »Jesús Amado, Pablo Casaseca-de-la Higuera, and Jorge Monge-Álvarez. Explainable ai for cough classification: a cnn-based approach with spectrogram analysis. *Biomedical Signal Processing and Control*, 93:106053, 2024.
- [6] »James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.
- [7] »Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [8] »Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [9] »Andrew P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.
- [10] »Murray Campbell, A. Joseph Hoane, and Feng-hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1–2):57–83, 2002.
- [11] »Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.
- [12] »Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L. Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.
- [13] »Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):6, 2020.

- [14] »François Chollet. *Deep Learning with Python*. Manning, 2017.
- [15] »Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.
- [16] »George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.
- [17] »Elizabeth R. DeLong, David M. DeLong, and Daniel L. Clarke-Pearson. Comparing the areas under two or more correlated roc curves: A nonparametric approach. *Biometrics*, 44(3):837–845, 1988.
- [18] »Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.
- [19] »Alexey Dosovitskiy, Lucas Beyer, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.
- [20] »Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.
- [21] »Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.
- [22] »Jaime Duque Domingo, Roberto Medina Aparicio, and Luis Miguel González Rodrigo. Cross validation voting for improving cnn classification in grocery products. *IEEE Access*, 10:20913–20925, February 2022. Open Access.
- [23] »Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542:115–118, 2017.
- [24] »Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.
- [25] »Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.
- [26] »Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. In *Proc. Interspeech*, pages 571–575, 2021.

- [27] »Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer. In *Proceedings of Interspeech*, 2021.
- [28] »Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [29] »Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.
- [30] »Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330, 2017.
- [31] »David J. Hand and Robert J. Till. A simple generalisation of the area under the roc curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, 2001.
- [32] »Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.
- [33] »Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
- [34] »Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.
- [35] »Sagar Hegde, Walter T. McNicholas, Albert Chua, et al. Cough sounds in screening and diagnostics: A scoping review. *NPJ Primary Care Respiratory Medicine*, 33(1):31, 2023.
- [36] »Shawn Hershey, Sourish Chaudhuri, Daniel P.W. Ellis, Jort F. Gemmeke, et al. Cnn architectures for large-scale audio classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.
- [37] »Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
- [38] »Carlos Hoyos-Barceló, Jorge Monge-Álvarez, and Pablo Casaseca-de-la Higuera. Real-time cough detection system based on hu moments and audio segmentation. *Computer Methods and Programs in Biomedicine*, 165:53–62, 2018.
- [39] »Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multiclass support vector machines. Technical Report Technical Report, Department of Computer Science, National Taiwan University, 2002.
- [40] »Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 240–248. PMLR, 2016.

- [41] »Jorge Jimenez García, María García Gadañón, Gonzalo César Gutierrez Tobal, Leila Kheirandish Gozal, Fernando Vaquerizo Villar, Daniel Álvarez González, Félix del Campo Matías, David Gozal, and Roberto Hornero Sánchez. A 2d convolutional neural network to detect sleep apnea in children using airflow and oximetry. *Elsevier*, 2022. Disponible en el Repositorio Documental UVa.
- [42] »Will Kenton. Why transformers need more data than cnns. Disponible en Medium, 2020. Consultado en 2025.
- [43] »Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 1137–1145, 1995.
- [44] »Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.
- [45] »Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2017.
- [46] »Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [47] »Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.
- [48] »Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR)*, 2019. Disponible en: <https://openreview.net/forum?id=Bkg6RiCqY7>.
- [49] »Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.
- [50] »John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon. A proposal for the dartmouth summer research project on artificial intelligence, 1955. Unpublished project proposal.
- [51] »Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.
- [52] »Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
- [53] »Jorge Monge-Álvarez, Carlos Hoyos-Barceló, Pablo Lesso, and Pablo Casaseca-de-la Higuera. Robust detection of audio-cough events using local hu moments. *IEEE Journal of Biomedical and Health Informatics*, 23(1):184–196, 2019.

- [54] »Sofiana Mootassim-Billah, Julien D’Haese, Andreas Schindler, and Marc Remacle. Acoustic analysis of voluntary coughs, throat clearings, and reflexive coughs. *Head & Neck*, 2023.
- [55] »Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? *Advances in Neural Information Processing Systems*, 32:4694–4703, 2019.
- [56] »Lara Orlandic, Tomás Teijeiro, and David Atienza. The coughvid crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms. *Scientific Data*, 8:156, 2021.
- [57] »Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. Specaugment: A simple data augmentation method for automatic speech recognition. In *Interspeech*, pages 2613–2617, 2019.
- [58] »Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 2013.
- [59] »Diego Asay Pérez Alonso. Técnicas avanzadas de aprendizaje profundo para la detección y análisis de tos en pacientes respiratorios. Trabajo fin de máster en ingeniería de telecomunicación, Universidad de Valladolid, Escuela Técnica Superior de Ingenieros de Telecomunicación, Valladolid, España, 2023. Director: Casaseca de la Higuera, Juan Pablo.
- [60] »Karol J. Piczak. Environmental sound classification with convolutional neural networks. In *Proceedings of the 2015 ACM Multimedia Conference*, pages 1015–1018, 2015.
- [61] »John Platt, Nello Cristianini, and John Shawe-Taylor. Large margin dags for multiclass classification. In *Advances in Neural Information Processing Systems*, volume 12, pages 547–553, 2000.
- [62] »David M. W. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.
- [63] »David M. W. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.
- [64] »Lutz Prechelt. Early stopping—but when? In *Neural Networks: Tricks of the Trade*, pages 55–69. Springer, 1998.
- [65] »Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017.

- [66] »Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.
- [67] »Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
- [68] »Takaya Saito and Marc Rehmsmeier. The precision–recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLOS ONE*, 10(3):e0118432, 2015.
- [69] »Antoine Serrurier, Christiane Neuschaefer-Rube, and Rainer Röhrig. Past and trends in cough sound acquisition, automatic detection and automatic classification: A comparative review. *Sensors*, 22(8):2896, 2022.
- [70] »Edward H. Shortliffe. *Computer-Based Medical Consultations: MYCIN*. Elsevier, 1976.
- [71] »Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, 2012.
- [72] »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(1):1929–1958, 2014.
- [73] »Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018.
- [74] »Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [75] »Diego Sánchez Prieto. Detección y clasificación automática de los mediante técnicas de deep learning. Trabajo fin de grado, Universidad de Valladolid, 2023.
- [76] »Eric Topol. *Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again*. Basic Books, 2019.
- [77] »Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021.
- [78] »Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning (ICML)*, 2021.

- [79] »Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.
- [80] »Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS 2017)*, volume 30, pages 5998–6008, 2017.
- [81] »Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [82] »Guangyao Wen, Peng Xu, Chao Zhang, et al. Assessing chronic obstructive pulmonary disease risk using cough audio signals. *NPJ Digital Medicine*, 2025.
- [83] »Paul J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Phd thesis, Harvard University, 1974.
- [84] »Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.