Universidad Nacional de Ingeniería

Facultad de Ingeniería Eléctrica y Electrónica

TESIS

Implementación de un sistema inteligente aplicando algoritmos de Deep Learning para la detección de enfermedad pulmonar obstructiva crónica mediante sonidos pulmonares

Para obtener el título profesional de Ingeniero Electrónico.

Elaborado por

Nilton Charlie Sare Vargas

0009-0003-5321-2453

Asesor

Dr. Ing. Carlos Celestino Medina Ramos

0000-0001-9747-3928

LIMA – PERÚ

2024

Citar/How to cite	Sare Vargas [1]
Referencia/Reference Estilo/Style: IEEE (2020)	[1] N. Sare Vargas, "*Implementación de un sistema inteligente aplicando algoritmos de Deep Learning para la detección de enfermedad pulmonar obstructiva crónica mediante sonidos pulmonares"* [Tesis]. Lima (Perú): Universidad Nacional de Ingeniería, 2024.

Citar/How to cite	(Sare, 2024)
Referencia/Reference Estilo/Style: APA (7ma ed.)	Sare, N. (2024). *Implementación de un sistema inteligente aplicando algoritmos de Deep Learning para la detección de enfermedad pulmonar obstructiva crónica mediante sonidos pulmonares*. [Tesis, Universidad Nacional de Ingeniería]. Repositorio institucional Cybertesis UNI.

Dedicatoria

A mis padres y abuelos, por la formación en mis valores, y que siempre me brindaron su apoyo desde la distancia.

Agradecimientos

Al Dr. Ing. Carlos Medina Ramos por su apoyo durante el desarrollo de este proyecto. También a todos mis amigos que me compartieron ideas y consejos para mejorar este trabajo.

Resumen

La Enfermedad Pulmonar Obstructiva Crónica (EPOC) es una de las principales causales de muerte a nivel global. Por lo tanto, es importante identificarla en sus etapas iniciales para proporcionar un tratamiento adecuado. Por ello, en esta tesis, se ha ideado un sistema inteligente basado en aprendizaje profundo para identificar patrones distintivos en los sonidos pulmonares, y predecir de manera precisa y eficiente la presencia de la EPOC.

Para el logro anterior, se utilizaron bases de datos de sonidos pulmonares provenientes del desafío ICBHI realizado en 2017 y de un grupo de investigadores del King Abdullah University Hospital. Adicionalmente, se aplicó una etapa de preprocesamiento a los datos, que involucró el uso de los espectrogramas de Mel y los coeficientes cepstrales en frecuencia de Mel (MFCC) para capturar características relevantes de los audios. Además, para resolver el problema de desbalance en el conjunto de datos, se aplicaron técnicas de aumento de datos.

Para el logro del objetivo general, se construyó un modelo de red neuronal convolucional (CNN) de doble rama; adicionalmente se adaptaron las arquitecturas de ResNet e InceptionV3, las cuales fueron entrenadas con diferentes algoritmos de optimización. Al ensayar estas redes, se obtuvieron métricas favorables, ante lo cual, se resalta el modelo CNN de doble rama, que empleó las características MFCC, inyección de ruido y desplazamiento de tono como técnica de aumento de datos, y el optimizador SGD. Con este modelo se logró la exactitud de 97.87%, precisión de 96.98%, sensibilidad de 96.62% y F1-score de 96.47% en la validación.

Palabras clave — Deep Learning, CNN, doble rama, sonidos pulmonares, EPOC, MFCC, Mel

Abstract

Chronic Obstructive Pulmonary Disease (COPD) is one of the leading causes of death globally. Therefore, it is important to identify it in its early stages to provide appropriate treatment. Hence, in this thesis, an intelligent system based on Deep Learning has been devised with the purpose of identifying distinctive patterns in lung sounds and accurately and efficiently predicting the presence of COPD.

To achieve this, databases of lung sound from the ICBHI challenge held in 2017 and from researchers at King Abdullah University Hospital were used. Additionally, a preprocessing stage was applied to the data, involving the use of Mel spectrograms and Mel-frequency cepstral coefficients (MFCC) to capture relevant features of the audio. Furthermore, data augmentation techniques were applied to address the problem of imbalance in the dataset. For the achievement of the general goal, a double-branch convolutional neural network (CNN) model was built, and the architectures of ResNet and InceptionV3 were adapted, which were trained with different optimization algorithms. Upon testing these networks, favorable metrics were obtained, highlighting the double-branch CNN model, which employed MFCC features, noise injection and pitch shifting as data augmentation techniques and the SGD optimizer. With this model, an accuracy of 97.87%, precision of 96.98%, sensitivity of 96.62%, and an F1-score of 96.47% were achieved during validation.

Keywords - Deep Learning, CNN, double-branch, lung sounds, COPD, MFCC, Mel

Tabla de Contenido

Pág.

Resumen v

Abstract vi

Introducción xvi

Capítulo I. Parte introductoria del trabajo 1

1.1 Generalidades 1

1.2 Descripción del Problema de Investigación 1

1.3 Planteamiento del problema 2

1.3.1 Problema General 2

1.3.2 Problemas Específicos 2

1.3.3 Variables 3

1.4 Objetivos del Estudio 3

1.4.1 Objetivo General 3

1.4.2 Objetivos Específicos 3

1.5 Hipótesis del Estudio 3

1.5.1 Hipótesis General 3

1.5.2 Hipótesis Específicos 3

1.6 Antecedentes Investigativos 4

1.6.1 Antecedentes Nacionales 4

1.6.2 Antecedentes Internacionales 4

Capítulo II. Marcos teórico, conceptual y filosófico 7

2.1 Marco Teórico 7

2.1.1 Base teórica de la enfermedad pulmonar obstructiva crónica 7

2.1.2 Base teórica de Deep Learning 15

2.2 Marco Conceptual 58

2.3 Marco filosófico 61

2.3.1 Humanismo y tecnología 61
2.3.2 Ética y responsabilidad 61
Capítulo III. Desarrollo del trabajo de investigación 62
3.1 Recopilación y análisis de dataset de sonidos pulmonares 64
3.2 Preprocesamiento de los datos 65
3.2.1 Unión de los datasets y reagrupación de categorías 65
3.2.2 Segmentación 66
3.2.3 Filtrado 67
3.3 Aumento de datos 68
3.3.1 Noise Injection + Pitch Shifting 68
3.3.2 Enmascaramiento en frecuencia y tiempo 69
3.4 Extracción de características 71
3.4.1 Espectrograma de Mel 71
3.4.2 MFCC 71
3.5 Entrenamiento y validación de los modelos CNN 72
3.5.1 Modelo propio 72
3.5.2 ResNet 80
3.5.3 Inception 89
Capítulo IV. Análisis y discusión de resultados 97
4.1 Discusión de resultados 101
4.2 Contrastación de las hipótesis 102
4.2.1 Hipótesis 1: Con la implementación de redes neuronales convolucionales, se detectará la EPOC mediante sonidos pulmonares. 102
4.2.2 Hipótesis 2: Con la implementación de arquitecturas de redes pre entrenadas, se identificará la óptima para la detección de la EPOC mediante sonidos pulmonares 102

4.2.3 Hipótesis 3: Con la aplicación de diversos algoritmos de optimización se logra determinar la arquitectura más significativa en términos de precisión..... 103
4.2.4 Hipótesis 4: Con la verificación del modelo entrenado aplicando sonidos pulmonares de pacientes con sospecha de EPOC, se confirmará la eficacia del sistema inteligente. 103
Conclusiones 104
Recomendaciones 105
Referencias Bibliográficas 106
Anexos 118

Lista de Figuras

Pág.

Figura 1: Zonas pulmonares afectados por la enfisema y bronquitis crónica 9
Figura 2: Puntos más comunes de auscultación y su secuencia 10
Figura 3: Forma de onda de las crepitantes finas. 12
Figura 4: Forma de onda de las crepitantes gruesas. 12
Figura 5: Forma de onda de las sibilancias. 13
Figura 6: Forma de onda de los Roncus 13
Figura 7: Estructura de una red neuronal convolucional 16
Figura 8: Señal de audio y su Transformada discreta de Fourier 18
Figura 9: Etapa de enventanado y aplicación de DFT por segmentos a la señal $\mathbf{x}(\mathbf{n})$ 19
Figura 10: Diagrama de bloques para la obtención del espectrograma de Mel 20
Figura 11: Escala de Mel 21
Figura 12: Banco de filtros de Mel de tamaño 20 21
Figura 13: Diagrama de bloques para la extracción de los coeficientes MFCC 22
Figura 14: Proceso de convolución entre la entrada y el filtro 23
Figura 15: Proceso de convolución entre la entrada y el filtro más el bias 24
Figura 16: Espectrograma de Mel en escala de grises. 25
Figura 17: Imagen original (a) y aplicación del Filtro Prewitt vertical (b) 26
Figura 18: Imagen original (a) y aplicación del Filtro Prewitt horizontal (b) 26
Figura 19: Imagen original (a) y aplicación del Filtro Sobel vertical (b) 27
Figura 20: Imagen original (a) y aplicación del Filtro Sobel horizontal (b) 27
Figura 21: Imagen original (a) y aplicación del Filtro Laplaciano (b) 28
Figura 22: Imagen original (a) y aplicación de la Máscara de Robinson en dirección NW (b) 29
Figura 23: Imagen original (a) y aplicación de la Máscara de Robinson en dirección SE (b) 29

Figura 24: Imagen original (a) y aplicación de la Máscara de Krisch en dirección NW (b) 30
Figura 25: Imagen original (a) y aplicación de la Máscara de Krisch en dirección SE (b) 30
Figura 26: Operación Average Pooling 31
Figura 27: Operación Max Pooling 32
Figura 28: Arquitectura CNN de doble rama 33
Figura 29: Representación gráfica de una red neuronal 34
Figura 30: Representación gráfica de la estructura matemática de una red neuronal 35
Figura 31: Batch Normalization 37
Figura 32: Función escalón 38
Figura 33: Función ReLu 39
Figura 34: Función sigmoide 40
Figura 35: Función tangente hiperbólica 41
Figura 36: Comportamiento de Descenso de gradiente 45
Figura 37: Comportamiento de GD con valores fijados de $\boldsymbol{\varphi}$ 45
Figura 38: Descenso de gradiente vs Gradiente conjugado 47
Figura 39: GD sin Momentum y con Momentum 49
Figura 40: GD, GD con Momentum y Adagrad 50
Figura 41: RMSprop vs Adagrad 51
Figura 42: Enmascaramiento en tiempo y frecuencia 53
Figura 43: Módulo Inception para GoogleNet 55
Figura 44: Módulo Inception para las versiones InceptionV2 e V3 55
Figura 45: Arquitectura de Inception V3 56
Figura 46: Diagrama del bloque residual 56
Figura 47: Etapas del Sistema inteligente basado en Deep Learning para la detección de EPOC 62
Figura 48: Esquema de principio para entrenamiento de los algoritmos a usar en el sistema inteligente basado en Deep Learning. 63

Figura 49: Información de las bases de datos ICBHI y KAUH 65
Figura 50: Número de pacientes por categoría 66
Figura 51: Número de audios respiratorios por categoría 66
Figura 52: Número de audios respiratorios segmentados por categoría 67
Figura 53: Extracto del código de aumento de datos con Noise Injection y Pitch Shifting 68
Figura 54: Número de audios respiratorios luego de la primera técnica de aumento de datos 69
Figura 55: Extracto del código para el aumento de datos con enmascaramiento en tiempo y frecuencia 70
Figura 56: Extracto del código para el preprocesamiento de los audios 72
Figura 57: Arquitectura diseñada del modelo propio 72
Figura 58: Arquitectura del modelo propio y cantidad de parámetros 73
Figura 59: Función de error del modelo propio entrenado con el primer aumento de datos 74
Figura 60: Métrica de exactitud del modelo propio entrenado con el primer aumento de datos 75
Figura 61: Métrica de precisión del modelo propio entrenado con el primer aumento de datos 75
Figura 62: Métrica de sensibilidad del modelo propio entrenado con el primer aumento de datos 76
Figura 63: Función de error del modelo propio entrenado con el segundo aumento de datos 77
Figura 64: Métrica de exactitud del modelo propio entrenado con el segundo aumento de datos 77
Figura 65: Métrica de precisión del modelo propio entrenado con el segundo aumento de datos 78

Figura 66: Métrica de sensibilidad del modelo propio entrenado con el segundo aumento de datos 78
Figura 67: Funciones de Keras para la construcción de la arquitectura ResNet..... 81
Figura 68: Últimas capas del modelo Resnet20 y cantidad de parámetros..... 82
Figura 69: Función de error del modelo ResNet entrenado con el primer aumento de datos 83
Figura 70: Métrica de exactitud del modelo ResNet entrenado con el primer aumento de datos 83
Figura 71: Métrica de precisión del modelo ResNet entrenado con el primer aumento de datos 84
Figura 72: Métrica de sensibilidad del modelo ResNet entrenado con el primer aumento de datos 84
Figura 73: Función de error del modelo ResNet entrenado con el segundo aumento de datos 86
Figura 74: Métrica de exactitud del modelo ResNet entrenado con el segundo aumento de datos 86
Figura 75: Métrica de precisión del modelo ResNet entrenado con el segundo aumento de datos 87
Figura 76: Métrica de sensibilidad del modelo ResNet entrenado con el segundo aumento de datos 87
Figura 77: Últimas capas del modelo InceptionV3 y cantidad de parámetros 89
Figura 78: Extracto del código para el ajuste del tamaño de las entradas para InceptionV3 90
Figura 79: Función de error del modelo InceptionV3 entrenado con el primer aumento de datos 91
Figura 80: Métrica de exactitud del modelo InceptionV3 entrenado con el primer aumento de datos 91

Figura 81: Métrica de precisión del modelo InceptionV3 entrenado con el primer aumento de dato 92
Figura 82: Métrica de sensibilidad del modelo InceptionV3 entrenado con el primer aumento de dato 92
Figura 83: Función de error del modelo InceptionV3 entrenado con el segundo aumento de datos 94
Figura 84: Métrica de exactitud del modelo InceptionV3 entrenado con el segundo aumento de datos 94
Figura 85: Métrica de precisión del modelo InceptionV3 entrenado con el segundo aumento de datos 95
Figura 86: Métrica de sensibilidad del modelo InceptionV3 entrenado con el segundo aumento de datos 95
Figura 87: Matriz de confusión al evaluar los datos de entrenamiento 98
Figura 88: Matriz de confusión al evaluar los datos de validación 99
Figura 89: Matriz de confusión al evaluar los datos de prueba 99
Figura 90: Tiempo de respuesta en la predicción de un paciente con EPOC 100
Figura 91: Tiempo de respuesta en la predicción de un paciente sano 101

Lista de Tablas

Pág.

Tabla 1: Cinco primeras causas de mayor muerte en el mundo 1

Tabla 2: Características de los sonidos pulmonares normales 11

Tabla 3: Resumen de resultados del entrenamiento del modelo propio con el primer aumento de datos 76

Tabla 4: Resumen de resultados del entrenamiento del modelo propio con el segundo aumento de datos 79

Tabla 5: Resumen de resultados del entrenamiento del modelo ResNet con el primer aumento de datos 85

Tabla 6: Resumen de resultados del entrenamiento del modelo ResNet con el segundo aumento de datos 88

Tabla 7: Resumen de resultados del entrenamiento del modelo InceptionV3 con el primer aumento de datos 93

Tabla 8: Resumen de resultados del entrenamiento del modelo InceptionV3 con el segundo aumento de datos 96

Tabla 9: Comparativa entre resultados de los modelos entrenados en este trabajo 97

Introducción

La enfermedad pulmonar obstructiva crónica (EPOC) es una afección que incide sobre el sistema respiratorio y es causal de un elevado índice de mortalidad a nivel global. En este sentido, la detección temprana de este mal es crucial para su tratamiento y control, de manera que se pueda evitar el deterioro de la salud del paciente.

Una de las técnicas comúnmente utilizada por los médicos para diagnosticar la EPOC es la auscultación pulmonar usando estetoscopio. Sin embargo, su subjetividad puede llevar a resultados inconsistentes.

En este contexto, los algoritmos de clasificación de Deep Learning son una herramienta prometedora para la detección precisa de la EPOC. Estos algoritmos tienen la capacidad de aprender y detectar patrones complejos en los sonidos pulmonares, lo que los vuelve una opción ideal para la detección temprana de esta enfermedad.

Esta tesis aborda el tema de detección temprana de la EPOC mediante el desarrollo de un algoritmo de clasificación de Deep Learning utilizando un gran conjunto de datos de sonidos pulmonares de pacientes con y sin EPOC, sirviendo de apoyo en la toma de decisión del especialista al momento del diagnóstico, y mejorar la objetividad en la interpretación de los sonidos pulmonares. Los resultados obtenidos pueden tener un impacto positivo en la detección de la EPOC en sus etapas tempranas, coadyuvando en la mejora de la calidad de vida de los individuos afectados.

Capítulo I. Parte introductoria del trabajo

1.1 Generalidades

La enfermedad pulmonar obstructiva crónica tiene gran repercusión en todo el mundo, causando considerable número de afectados. Solo en el año 2019, provocó el fallecimiento de más de 3 millones de personas a nivel global (Global Initiative for Chronic Obstructive Lung Disease [GOLD], 2023). Su detección oportuna representa un serio desafío para la salud pública. Se prevé que debido a la exposición crónica a sus factores de riesgo, las cifras de incidencia y mortalidad continúen aumentando significativamente en las próximas décadas (World Health Organization [WHO], 2023). Además, en algunas regiones del Perú, la precariedad de las condiciones de salud y la falta de equipos y especialistas obstaculizan la atención pertinente a la población (Ministro de Salud [Minsa], 2023). Por lo tanto, la EPOC es un dilema de gran importancia que precisa de atención con urgencia.

1.2 Descripción del Problema de Investigación

Las enfermedades que afectan al sistema respiratorio son un asunto de gran importancia cuando hablamos en términos de la salud, debido a que estas tienen una amplia contribución a la alta tasa de mortalidad a nivel global. Entre estas enfermedades, la EPOC y las infecciones del tracto respiratorio inferior (ITRI) se destacan entre las principales causas de muerte (WHO, 2021), mostrándose más detalle en la Tabla 1.

Tabla 1

Cinco primeras causas de mayor muerte en el mundo

Puesto	Primero	Segundo	Tercero	Cuarto	Quinto
Causa	Cardiopatía isquémica	Accidente cerebrovascular	Enfermedad pulmonar obstructiva crónica	Infecciones del tracto respiratorio inferior	Condiciones neonatales
Total de muertes en 2019	8 880 000	6 190 000	3 220 000	2 590 000	1 960 000

Nota: Adaptado de (WHO, 2021)

A lo largo de los años, las investigaciones con respecto a esta enfermedad pulmonar han aumentado, estos estudios indican que aproximadamente dos tercios de los pacientes en riesgo de desarrollar EPOC reciben diagnósticos incorrectos en la atención médica primaria (Laucho-Contreras & Cohen-Todd, 2020) y, como en toda enfermedad, la detección temprana de EPOC puede resultar crucial para reducir su incidencia y evitar que empeore clínicamente (Choi & Rhee, 2020).

Una manera de detectar las enfermedades pulmonares es a través de la técnica de auscultación pulmonar, que implica el uso de un estetoscopio con el cual el especialista evaluará los sonidos pulmonares (Jung et al., 2021). No obstante, es importante considerar que esta técnica es subjetiva y podría llevar a diagnósticos erróneos.

Dado este contexto, el diagnóstico temprano y correcto de la EPOC es fundamental para la salud de cualquier persona. En ese sentido, el desarrollo de un algoritmo de detección de esta enfermedad será una valiosa herramienta para los especialistas, otorgándoles una ayuda para tomar decisiones más precisas y acertadas (Naqvi & Choudhry, 2020).

1.3 Planteamiento del problema

1.3.1 Problema General

¿De qué manera se implementa un sistema inteligente aplicando algoritmos de Deep Learning que permita detectar la EPOC mediante sonidos pulmonares?

1.3.2 Problemas Específicos

- ¿Cómo se implementan las arquitecturas de las redes neuronales convolucionales que permitan detectar la EPOC mediante sonidos pulmonares?
- ¿De qué modo se implementan las arquitecturas de redes pre entrenadas para detectar la EPOC mediante sonidos pulmonares?
- ¿De qué forma se aplican los algoritmos de optimización que permitan elegir la arquitectura más significativa para detectar la EPOC mediante sonidos pulmonares?

- ¿De qué modo se verifica el modelo entrenado para detectar la EPOC mediante sonidos pulmonares y permita confirmar la eficacia del sistema inteligente?

1.3.3 Variables

- V. Dependiente: Enfermedad pulmonar obstructiva crónica.
- V. Independiente: Sistema inteligente aplicando algoritmos de Deep Learning.

1.4 Objetivos del Estudio

1.4.1 Objetivo General

Implementar un sistema inteligente aplicando algoritmos de Deep Learning que permita detectar la EPOC mediante sonidos pulmonares.

1.4.2 Objetivos Específicos

- Implementar las arquitecturas de redes neuronales convolucionales para detectar la EPOC mediante sonidos pulmonares.
- Implementar arquitecturas de redes pre entrenadas para identificar la óptima para detectar la EPOC mediante sonidos pulmonares.
- Aplicar algoritmos de optimización que permitan determinar la arquitectura más significativa en términos de precisión para detectar la EPOC mediante sonidos pulmonares.
- Verificar el modelo entrenado aplicando sonidos pulmonares de pacientes con sospecha de EPOC para confirmar la eficacia del sistema inteligente.

1.5 Hipótesis del Estudio

1.5.1 Hipótesis General

Con la implementación de un sistema inteligente aplicando algoritmos de Deep Learning, se detectará la EPOC mediante sonidos pulmonares.

1.5.2 Hipótesis Específicos

- Con la implementación de redes neuronales convolucionales, se detectará la EPOC mediante sonidos pulmonares.

- Con la implementación de arquitecturas de redes pre entrenadas, se identificará la óptima para la detección de la EPOC mediante sonidos pulmonares.
- Con la aplicación de diversos algoritmos de optimización se logra determinar la arquitectura más significativa en términos de precisión.
- Con la verificación del modelo entrenado aplicando sonidos pulmonares de pacientes con sospecha de EPOC, se confirmará la eficacia del sistema inteligente.

1.6 Antecedentes Investigativos

1.6.1 Antecedentes Nacionales

"Sistema informático móvil inteligente para la detección temprana y control de enfermedades respiratorias en pacientes del sector privado de salud en la ciudad de Trujillo" (Liberato Bernal & Quilcat Pesantes, 2021)

Resumen: Los autores idearon y desarrollaron un aplicativo móvil para el monitoreo constante de los sonidos respiratorios de los pacientes. Además, desarrollaron un modelo de aprendizaje profundo con la capacidad de analizar estos audios respiratorios e identificar sibilancias, permitiendo que los especialistas diagnostiquen enfermedades respiratorias de manera más eficiente.

Conclusiones: La aplicación móvil desarrollada por los autores permitirá que los pacientes puedan ser atendidos y diagnosticados por el especialista de manera remota. Además, su modelo de aprendizaje profundo basado en CNN obtuvo un 86% de exactitud y 0.29 de pérdida, por lo cual resulta en una gran ayuda en la toma de decisión del médico.

1.6.2 Antecedentes Internacionales

1. **"Automatic identification of respiratory diseases from stethoscopic lung sound signals using ensemble classifiers"** (L. Fraiwan et al., 2021)

Resumen: Los autores de este artículo aplicaron 4 modelos para la clasificación de enfermedades respiratorias los cuales fueron Decision Tree, Linear Discriminant, Support Vector Machine (SVM) y K-Nearest Neighbors (KNN). Además, utilizaron el método ensemble o de conjunto para mejorar el comportamiento de dichos modelos.

Conclusiones: los modelos propuestos permitieron obtener resultados satisfactorios en cuanto a su exactitud de predicción, de los cuales el mejor fue SVM con una exactitud de clasificación de 98.20% y en particular, un 94.95% de exactitud al predecir la EPOC.

2. **"Detecting Respiratory Diseases from Recorded Lung Sounds by 2D CNN"** (Hazra & Majhi, 2020)

Resumen: En este trabajo, los autores utilizaron los coeficientes cepstrales en la escala de Mel para extraer las características del espectro de los sonidos pulmonares y una CNN bidimensional y optimizador Adam con los cuales clasificaron enfermedades respiratorias.

Conclusiones: el método propuesto obtuvo una exactitud de clasificación de 92.39%. En particular lograron detectar la EPOC con un 95% de exactitud.

3. **"Automated Lung Sound Classification Using a Hybrid CNN-LSTM Network and Focal Loss Function"** (Petmezas et al., 2022)

Resumen: en este trabajo, los autores utilizaron la transformada de Fourier de tiempo corto para la obtención de los espectrogramas de los sonidos respiratorios de la base de datos de ICBHI 2017. Además, para la clasificación de estas características diseñaron una arquitectura híbrida de redes CNN y LSTM.

Conclusiones: para una división de datos de entrenamiento y prueba de 60/40, obtuvieron un 47.37%, 82.46%, 64.92% y 73.69% de sensibilidad, especifidad, score y precisión respectivamente. Este resultado puede mejorarse utilizando técnicas de aumento de datos en el Dataset de sonidos respiratorios.

4. **"Acquisition and Classification of Lung Sounds for Improving the Efficacy of Auscultation Diagnosis of Pulmonary Diseases"** (Tessema et al., 2022)

Resumen: los autores incorporaron sus propias muestras a la base de datos de ICBHI 2017, los cuales recolectaron utilizando un estetoscopio electrónico. Excluyeron de esta nueva base de datos los audios que no fueron adquiridos con un estetoscopio de un solo canal. Además, emplearon la transformada Wavelet discreta para transformar los audios en una representación en forma de imagen.

Conclusiones: en este estudio, utilizaron diversos modelos de machine learning para clasificar los sonidos pulmonares, obteniendo 99%, 99.04% y 99.2% de exactitud, sensibilidad y especifidad respectivamente.

Capítulo II. Marcos teórico, conceptual y filosófico

2.1 Marco Teórico

2.1.1 Base teórica de la enfermedad pulmonar obstructiva crónica

La EPOC es una enfermedad crónica, progresiva, prevenible y tratable, caracterizándose por la obstrucción o bloqueo del flujo aéreo en los pulmones o alteración estructural de los alvéolos, lo que dificulta respirar y con posibilidades de empeorar con el tiempo (National Heart Lung and Blood Insitute [NHLBI], 2022).

Los síntomas presentes en un paciente con EPOC no siempre son los mismos para todos (American Lung Association, 2023), aunque hay algunos muy comunes como la difícil respiración y tos crónica, y con el tiempo pueden limitar la capacidad del paciente para realizar actividades cotidianas (Lores Gutiérrez, 2006).

A pesar de ser una enfermedad que no tiene cura, existen tratamientos para controlar sus síntomas y evitar su progreso, mejorando así la calidad de vida del paciente (Mayo Clinic, 2021).

1. Factores de riegos de la EPOC

El desarrollo de la EPOC puede deberse a distintos factores de riesgo, como la exposición a contaminantes del ambiente, humo del tabaco, la edad e incluso por condiciones genéticas u otras enfermedades (NHLBI, 2022).

a) Tabaquismo

El tabaquismo resulta en el factor de riesgo más común e importante para el desarrollo de la EPOC, sobre todo en personas del sexo masculino, ya que estas tienen mayor prevalencia en esta adicción (Correa et al., 2019).

Además, mantenerse cerca del humo de segundo mano, ya sea de los cigarrillos o del humo exhalado por un fumador, puede conllevar al desarrollo de la EPOC (Centers for Disease Control and Prevention, 2022).

b) Contaminación ambiental

Otro de los factores más comunes para el desarrollo de la EPOC es la exposición al humo generado al quemar materia orgánica, como madera, estiércol u otras fuentes que se suelen utilizar, sobre todo en áreas rurales especialmente para cocinar (Salvi & Barnes, 2010).

Es importante destacar que las ciudades no están exentas de la contaminación ambiental, que incluye partículas de dióxido de nitrógeno (NO2), monóxido de carbono (CO), dióxido de azufre (SO2) y material particulado (PM2.5). Estas partículas pueden ser inhaladas por los residentes urbanos, lo que aumenta el riesgo de desarrollar EPOC o empeorar sus síntomas (Parra-Sánchez et al., 2020).

c) Condiciones genéticas

Un factor menos común para la manifestación de la EPOC es la deficiencia de Alfa-1 antitripsina (AAT), siendo el hígado el responsable de fabricarla con el objetivo de proteger los pulmones de los factores de riesgo mencionados en los dos incisos anteriores (NHLBI, 2022).

Además, la deficiencia de AAT conduce a la destrucción de los tejidos pulmonares de manera progresiva e irreversible lo que provoca la aparición de la EPOC (Martínez Luna et al., 2020).

2. Patología de la EPOC

La EPOC es heterogénea, compleja y dinámica. Es heterogénea por que puede manifestarse bajo distintas alteraciones pulmonares o extrapulmonares, siendo las principales el enfisema y la bronquitis crónica, afectando a los alvéolos y las vías aéreas respectivamente (López-Giraldo et al., 2014) tal y como se ilustra en la Figura 1.

a) Enfisema

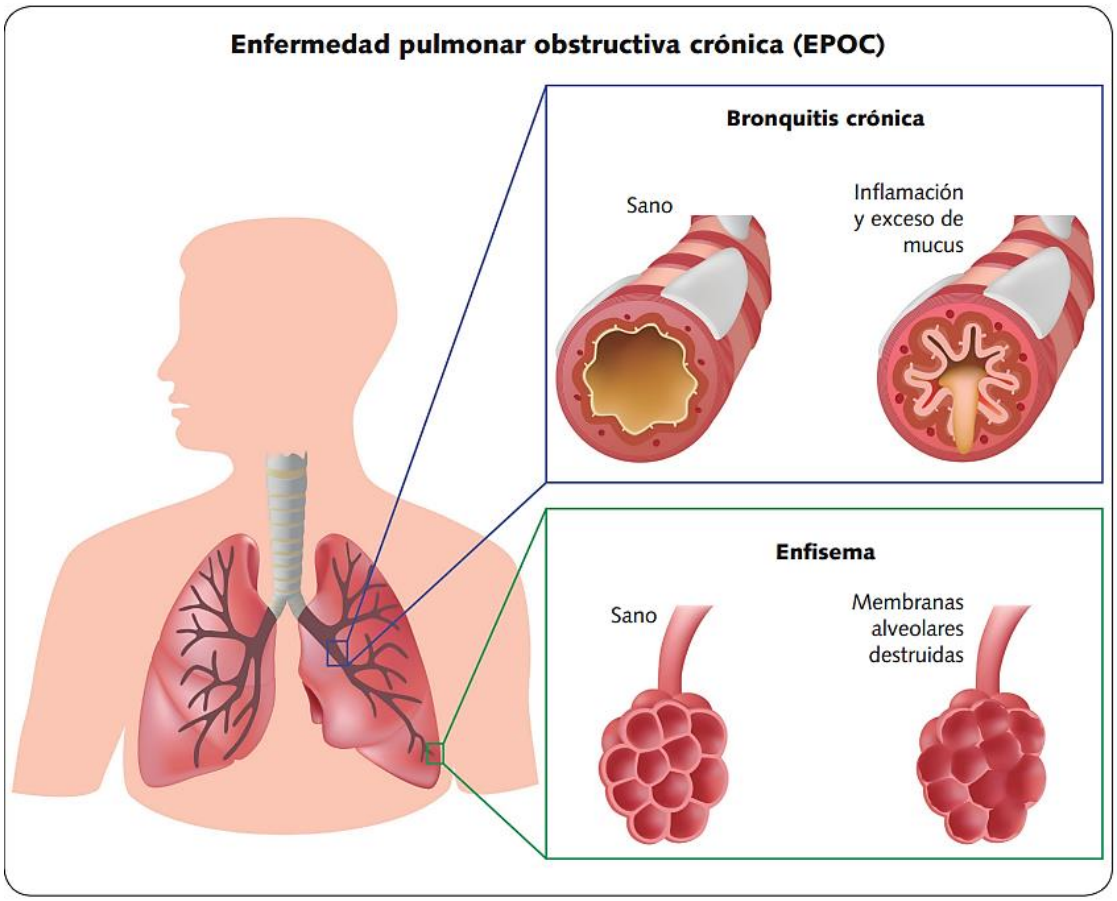
Los alvéolos aumentan irreversiblemente su tamaño, dañándose, provocando la destrucción de las paredes alveolares y a su vez manifestándose como disnea (WHO, 2023) y (Gómez Ayala, 2016).

b) Bronquitis crónica y bronquiolitis

Inflamación de los bronquios o vías respiratorias que llegan a obstruir el paso del aire. Esta inflamación provoca la producción excesiva de esputo y expectoración (WHO, 2023) y (Gómez Ayala, 2016).

Figura 1

Zonas pulmonares afectados por la enfisema y bronquitis crónica



Nota: fuente (Gómez Ayala, 2016)

La EPOC se puede diferenciar de otras enfermedades respiratorias, como el asma, ya que esta se puede presentar a cualquier edad, mientras que la EPOC suele desarrollarse en personas mayores. También puede diferenciarse debido a la naturaleza de la obstrucción del flujo aéreo ya que en la EPOC no se suele recuperar completamente la capacidad pulmonar con el tratamiento o puede llegar a ser irreversible si no es tratado a tiempo (Elsevier, 2020).

3. Diagnóstico

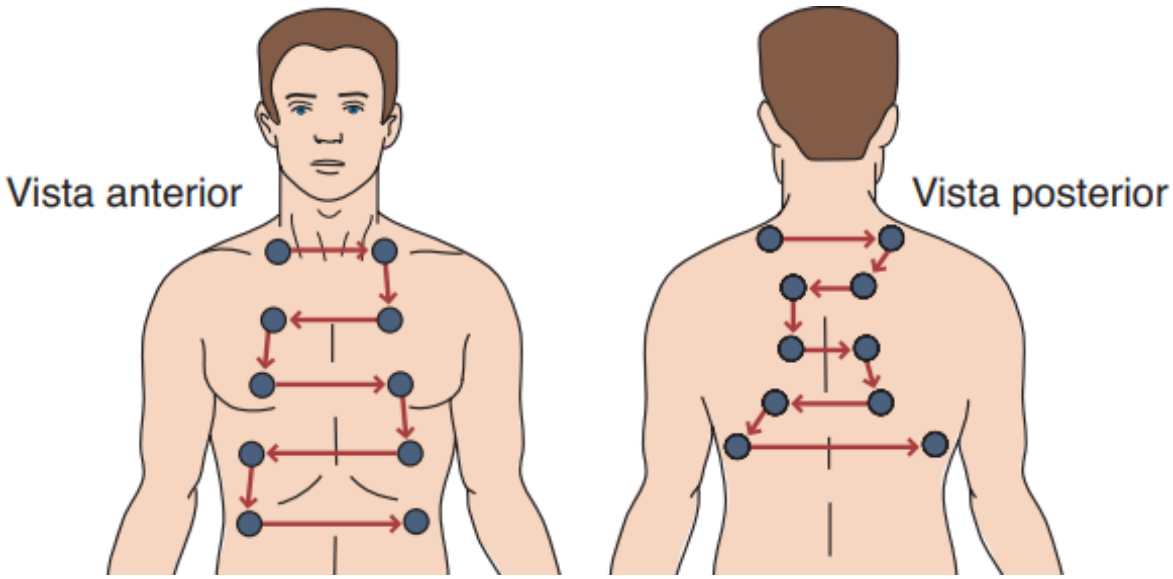
El especialista sabrá que un paciente tiene EPOC observando sus antecedentes médicos, como enfermedades previas, síntomas, evaluando los sonidos respiratorios mediante un examen físico, realizando pruebas de función pulmonar y, de ser necesario, a través de imágenes pulmonares (NHLBI, 2022) y (Chacón-Chaves et al., 2003).

En el examen físico, se emplea la técnica de auscultación como método de evaluación de los sonidos respiratorios. Esto implica el uso de un estetoscopio para escuchar estos sonidos y en base a eso brindar un diagnóstico. Utilizando esta técnica, el médico evaluará y detectará cambios sutiles en los ruidos respiratorios (Bertrand et al., 2020), auscultando directamente sobre la piel del paciente, generalmente sentado, en las posiciones correctas siendo los puntos más comunes los mostrados en la Figura 2.

Además, para realizar la auscultación, se debe de preparar un entorno silencioso evitando los ruidos ambientales y, de ser posible, que el paciente respire por la boca.

Figura 2

Puntos más comunes de auscultación y su secuencia



Nota: fuente (Myers, 2011)

Para la evaluación de los ruidos respiratorios, se deben de explorar características específicas (Vargas, 2015):

a) Fase: Inspiratoria o espiratoria.

b) Frecuencia: Número de muestras en la fase inspiratoria o espiratoria que sucede en una unidad de tiempo.

c) Simetría: Simétrico o asimétrico.

d) Tono: Alto o bajo.

e) Ritmo: Rítmico o arrítmico.

4. Sonidos o ruidos respiratorios

En la auscultación se analizan los ciclos respiratorios, que constan con 2 fases: la fase inspiratoria y la espiratoria. Estos ciclos respiratorios, a su vez, se pueden clasificar como ruidos normales y anormales.

Los ruidos normales se presentan como ruido traqueal, bronquial, broncovesicular y vesicular, y sus características se detallan en la Tabla 2.

Tabla 2

Características de los sonidos pulmonares normales

Tipo	Localización	Características	Frecuencia
Vesicular	Zona posterior del tórax.	Suave, susurrante De tono bajo: Inspiratorio>Espiratorio Duración: Inspiratorio>Espiratorio (~2:1)	100 – 1000 Hz Caída de energía: 100 - 200 Hz
Bronquial	Zona anterior del tórax.	Fuerte y de tono alto Duración: Espiratorio>Inspiratorio	100 – 5000 Hz Caída de energía: 800 Hz
Broncovesicular	Mayoría de los espacios pulmonares.	Duración: Espiratorio≈Inspiratorio	Intermedio entre Vesicular y bronquial.
Traqueal	Tráquea	Áspero y de tono alto Duración: Espiratorio≈Inspiratorio	100 – 5000 Hz Caída de energía: 800 Hz

Nota: adaptado de (Holt, 2021) *y* (Sarkar et al., 2015).

Por otro lado, los sonidos pulmonares anormales que se escuchan a través de la auscultación pulmonar se clasifican comúnmente en 6, asociadas a su vez a diferentes enfermedades pulmonares.

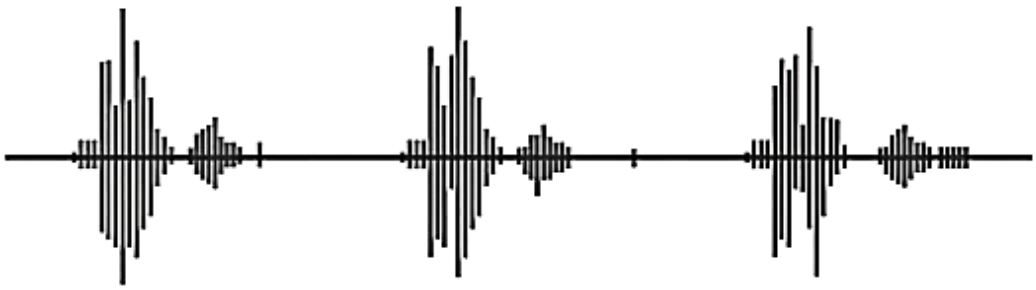
a) Crepitaciones finas

Ruidos de tono alto, discontinuos, de corta duración (~5ms), con una frecuencia de 650 Hz y que se presentan en la fase inspiratoria, y su forma de onda se muestra en la

Figura 3. Se puede escuchar estos ruidos anormales auscultando en las bases de los pulmones y pueden asociarse a pacientes con Fibrosis pulmonar, Neumonía o Bronquiolitis (Bertrand et al., 2020; Holt, 2021) y (Kim et al., 2021).

Figura 3

Forma de onda de las crepitantes finas.



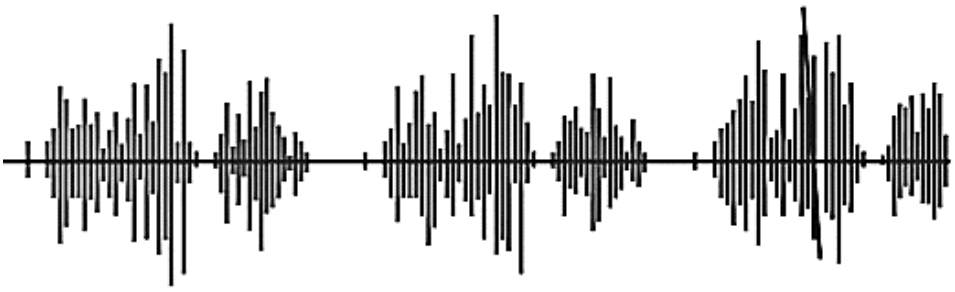
Nota: fuente (Holt, 2021).

b) Crepitaciones gruesas

Ruidos de tono bajo, discontinuos, de corta duración (~15ms), con una frecuencia de 350 Hz, presentes en ambas fases, pero predominantemente en la fase inspiratoria, y su forma de onda se muestra en la Figura 4. Se puede escuchar estos ruidos anormales auscultando en las bases de los pulmones y pueden asociarse a pacientes con Bronquitis aguda, Bronquiectasia o Traqueotomía (Bertrand et al., 2020; Holt, 2021) y (Kim et al., 2021).

Figura 4

Forma de onda de las crepitantes gruesas.



Nota: fuente (Holt, 2021).

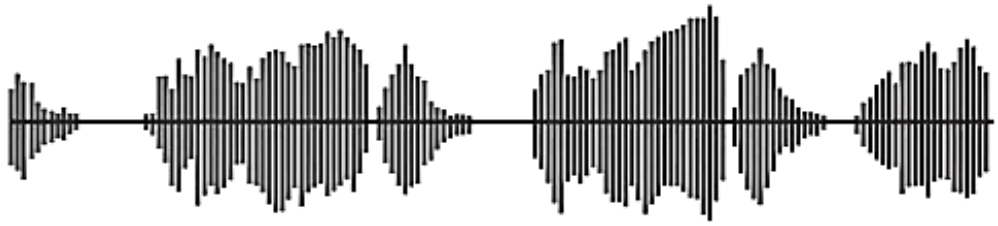
c) Sibilancias

Ruidos de tono alto, continuos, de mayor duración (~80ms), con una frecuencia entre 100 y 5000 Hz, con una frecuencia dominante de 400 Hz, y que se presentan en

ambas fases, pero predominantemente en la fase espiratoria, y su forma de onda se muestra en la Figura 5. Se puede escuchar estos ruidos anormales auscultando en la mayoría de los espacios pulmonares y pueden asociarse a pacientes con Asma o EPOC (Bertrand et al., 2020; Holt, 2021) y (Kim et al., 2021).

Figura 5

Forma de onda de las sibilancias.



Nota: fuente (Holt, 2021)*.*

d) Roncus

Ruidos de tono bajo, retumbante, continuos, de mayor duración (~80ms), con una frecuencia de 150 Hz, y que se presentan en ambas fases, pero predominantemente en la fase espiratoria, y su forma de onda se muestra en la Figura 6. Se puede escuchar estos ruidos anormales auscultando en la mayoría de los espacios pulmonares y pueden asociarse a pacientes con Bronquitis o Neumonía (Bertrand et al., 2020; Holt, 2021) y (Kim et al., 2021).

Figura 6

Forma de onda de los Roncus.

Nota: fuente (Holt, 2021)*.*

e) Estridor

Ruidos de tono alto, continuos, de mayor duración (~250ms), con una frecuencia 500 Hz, y que se presentan en la fase espiratoria. Se puede escuchar estos ruidos anormales auscultando en la tráquea y pueden asociarse a pacientes con Epiglotitis (Bertrand et al., 2020; Holt, 2021) y (Kim et al., 2021).

f) Frote o roce pleural

Ruidos de tono bajo, explosivo y rítmico, continuos, de corta duración (~15ms), con una frecuencia 350 Hz, y que se presentan en ambas fases. Se puede escuchar estos ruidos anormales auscultando en la mayoría de los espacios pulmonares y pueden asociarse a pacientes con Pleuritis, Neumonía o Tumor Pleural (Bertrand et al., 2020; Holt, 2021) y (Kim et al., 2021).

Entre estos ruidos anormales, las sibilancias son indicadores de una posible presencia de la EPOC, y para diagnosticarlo como tal, dependerá de la historia clínica y síntomas del paciente, así como a los factores de riesgo a los que estuvo expuesto (Martínez Luna et al., 2020).

2.1.2 Base teórica de Deep Learning

El aprendizaje profundo, una disciplina dentro del campo del Machine Learning, emplea redes neuronales profundas para aprender y extraer automáticamente características relevantes y patrones complejos de un conjunto de datos. Estas redes se inspiran en el comportamiento y funcionamiento del cerebro humano (International Business Machines Corporation [IBM], 2023), utilizando una arquitectura jerárquica compuesta por capas de unidades de procesamiento, y, a su vez, estas unidades representan a las neuronas del cerebro y son las encargadas de identificar patrones útiles en los datos (Espinoza Villafuerte, 2019).

Cuando se trabaja con imágenes, una variante ampliamente utilizada es la red neuronal convolucional (CNN). Estas redes son frecuentemente utilizadas para obtener información o características únicas en imágenes, como, por ejemplo, la representación en imagen de señales de audio, eliminando en su camino información redundante (Ortiz de Landaluce, 2021).

Se le da el nombre de red neuronal convolucional ya que aplican la operación de convolución, de símbolo *****, y que se define matemáticamente mediante la siguiente ecuación (Pazos Ruiz, 2020):

$$(f * h)(t) = \int_{-\infty}^{\infty} f(t-y)h(y)dy \quad (1)$$

Y se cambia la integral por una sumatoria si se encuentra en un caso discreto.

$$(f * h)(n) = \sum_{m=-\infty}^{\infty} f(n-m)h(m) \quad (2)$$

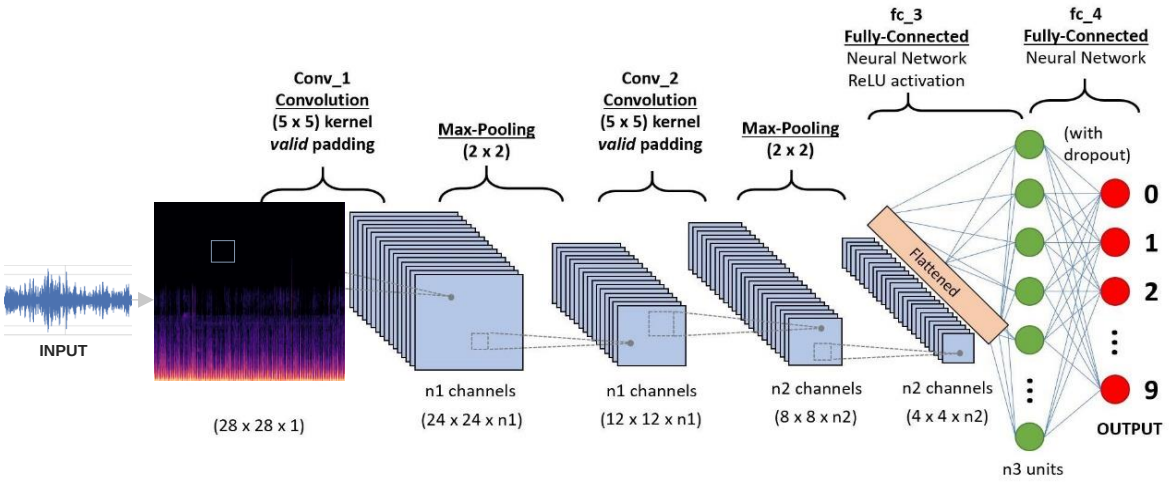
Una propiedad importante de la operación convolución es la conmutación (Baraniuk, n.d.), es decir:

$$(f * h)(n) = (h * f)(n) = \sum_{m=-\infty}^{\infty} f(m)f(n-m) \quad (3)$$

La red neuronal convolucional presenta dos partes importantes en su estructura, una parte de extracción de características y otra de clasificación de estas características, tal y como se visualiza en la Figura 7.

Figura 7

Estructura de una red neuronal convolucional



Nota: adaptado de (Fuchslocher Courard, 2021)

1. Extracción de características

Esta parte de la red neuronal convolucional contiene los procesos de filtro (convolución) y agrupación (ej. Max-Pooling), los cuales pueden posicionarse de manera secuencial según sea necesario para alcanzar un resultado óptimo. En el caso de trabajar con audios, previamente se debe realizar una etapa de preprocesamiento para obtener las imágenes que representen estos audios, como un espectrograma.

a) Preprocesamiento de datos

Diferentes métodos de procesamiento de audio se utilizan para representarlos en imágenes, como la transformada de Fourier de tiempo corto, el cochleagrama, el espectrograma de Mel, los coeficientes cepstrales en frecuencia de Mel o MFCC, entre otros.

i. Transformada de Fourier de Tiempo Corto (STFT)

El punto de partida del STFT es la Trasformada de Fourier, FT por sus siglas en inglés, una técnica matemática que se utiliza para convertir una señal en el dominio del

tiempo a su representación en el dominio de la frecuencia (Osgood, 2019). Sin embargo, esta herramienta funciona muy bien con señales estacionarias, lo que no es común en el mundo real (Flórez et al., 2009), como es el caso de los audios respiratorios.

La FT se expresa matemáticamente de la siguiente manera:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi f t} dt \quad (4)$$

Además, es importante conocer la variante de la FT que es la Transformada Discreta de Fourier (DFT), que se aplica cuando la señal es finita, es decir, limitada en tiempo y en ancho de banda (Osgood, 2019). La expresión matemática es similar a la ecuación (4).

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{j2\pi nk}{N}} \quad (5)$$

Sin embargo, al igual que la FT, la DFT también presenta limitaciones cuando se trabaja con señales no estacionarias, ya que proporcionan una representación en frecuencia de toda la señal y asume que esta es estacionaria durante todo su dominio temporal (Osgood, 2019). En la Figura 8, una señal de audio respiratorio de una persona sana y su DFT son presentadas. Sin embargo, en los 20 segundos de audio, es imposible distinguir las frecuencias fundamentales de cada ciclo respiratorio.

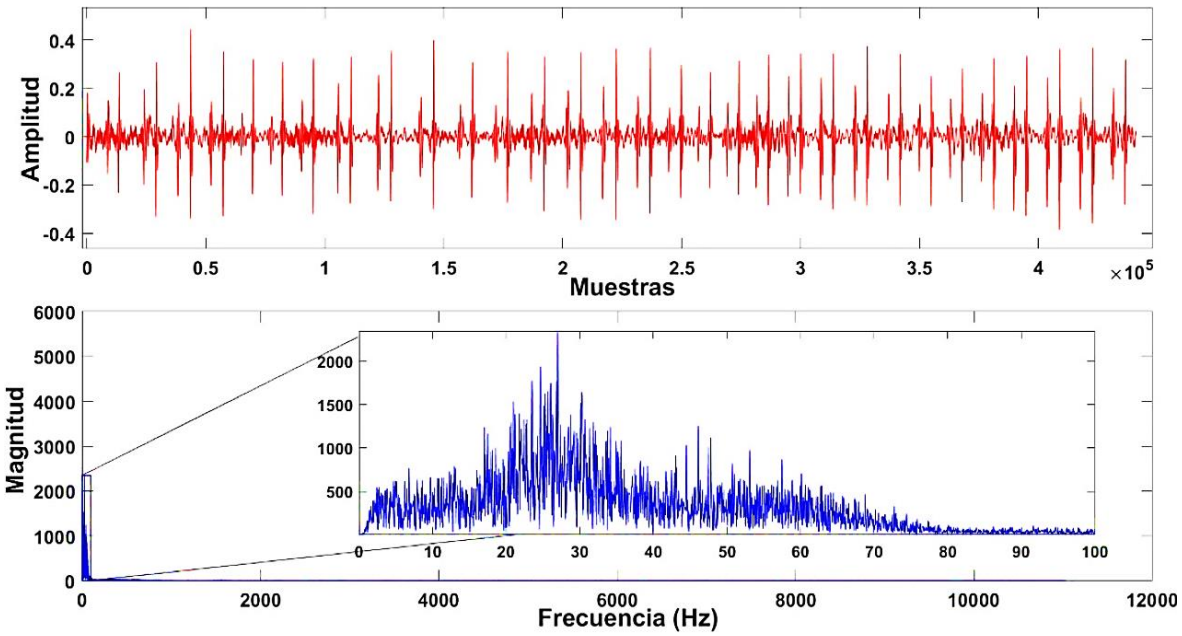
El problema mencionado se puede abordar utilizando una técnica conocida como Transformada de Fourier de Tiempo Corto (STFT). Esta técnica implica segmentar la señal de audio en N muestras, cada una con un tamaño de L puntos, y luego aplicar la DFT a cada una de estas muestras.

La señal se segmenta utilizando una función de ventana $\omega(n)$, obteniendo la ecuación (6), que expresa matemáticamente la STFT (Huang et al., 2019).

$$X(m) = \sum_{n=0}^{L-1} x(n - mL) \cdot \omega(n) \cdot e^{-j\omega n} \quad (6)$$

Figura 8

Señal de audio y su Transformada discreta de Fourier



Nota: La frecuencia de muestreo es 22050 Hz, por tanto, hay más de 400000 muestras.

Además de la longitud de la ventana (L), dos parámetros más se presentan, los cuales son el desplazamiento de la ventana (H), que indica cuantos puntos avanzará la ventana en cada paso, y la superposición de las ventanas (Ov), indicando la cantidad de puntos de un segmento que se superpondrán con el siguiente.

La Figura 9 muestra la aplicación de una función de ventana $\omega(n)$**,** que se multiplica con la señal $x(n)$, para luego aplicar la DFT a cada segmento resultante, lo que puede revelar información detallada sobre cómo cambian las frecuencias en el tiempo, lo que es especialmente útil para señales no estacionarias como los audios respiratorios.

La elección de la longitud de la ventana en la STFT tiene un impacto en la resolución en frecuencia y en la resolución en el tiempo. A mayor longitud de ventana coloquemos, mayor será la resolución en frecuencia y menor resolución en el tiempo. Lo contrario ocurre cuando el tamaño de la ventana es más corto (Jeon et al., 2020).

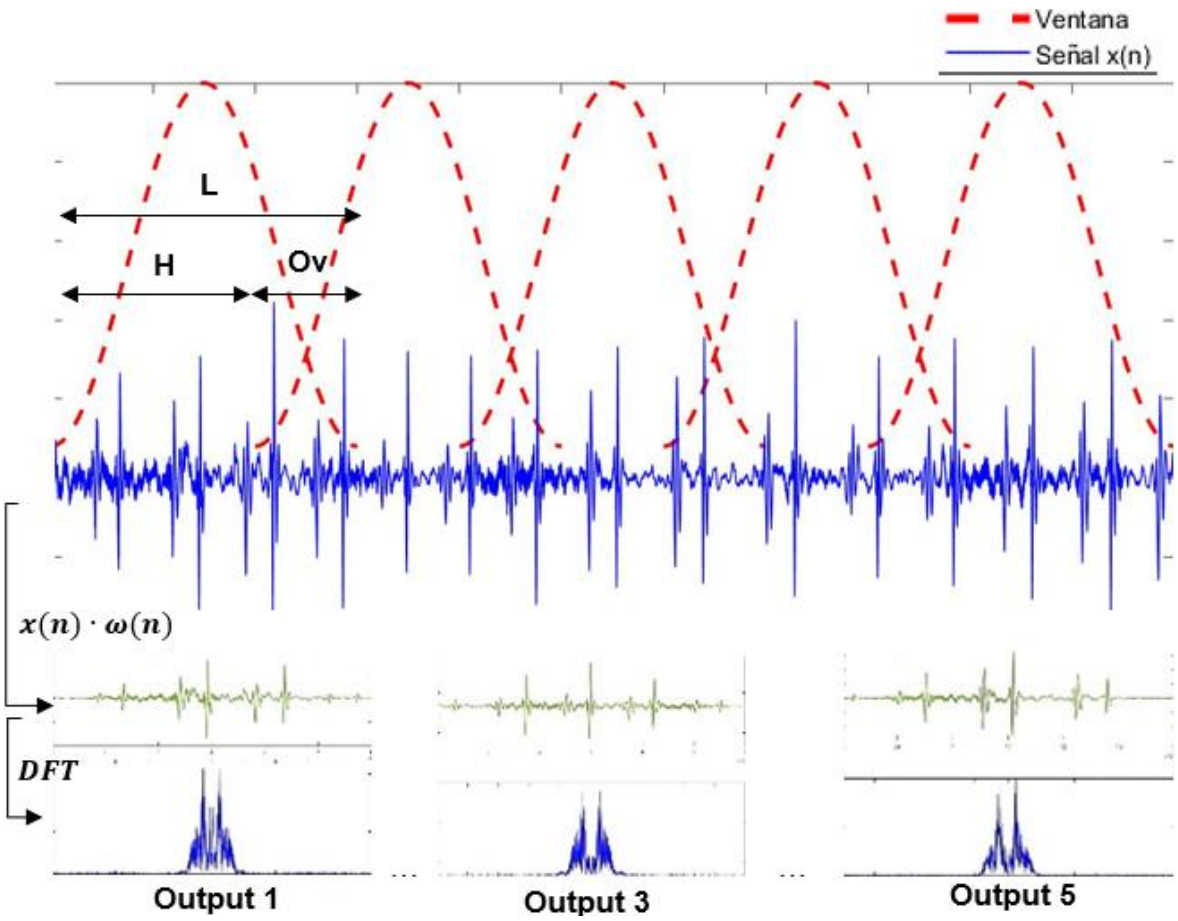
Comúnmente, la ventana de Hamming es la que se utiliza como función de ventana (Braun, 2001), y su valor en el punto n está definida mediante la siguiente ecuación:

$$\omega[n] = \begin{cases} 0.54 - 0.46 \cdot cos\left(\frac{2\pi n}{N-1}\right) & 0 \leq n \leq N-1 \\ 0 & otros \end{cases} \quad (7)$$

Siendo N el tamaño total de la ventana.

Figura 9

Etapa de enventanado y aplicación de DFT por segmentos a la señal $\boldsymbol{x(n)}$



Además de Hamming, la ventana de Hanning y la rectangular también son utilizadas, aunque estas son menos regulares, Matemáticamente se representan por las expresiones (**8**) y (**9**) respectivamente (Braun, 2001).

$$\omega[n] = \begin{cases} 0.5\left[1 - cos\left(\frac{2\pi n}{N}\right)\right] & 0 \leq n \leq N-1 \\ 0 & otros \end{cases} \quad (8)$$

$$\omega[n] = \begin{cases} 1 & 0 \leq n \leq N-1 \\ 0 & otros \end{cases} \quad (9)$$

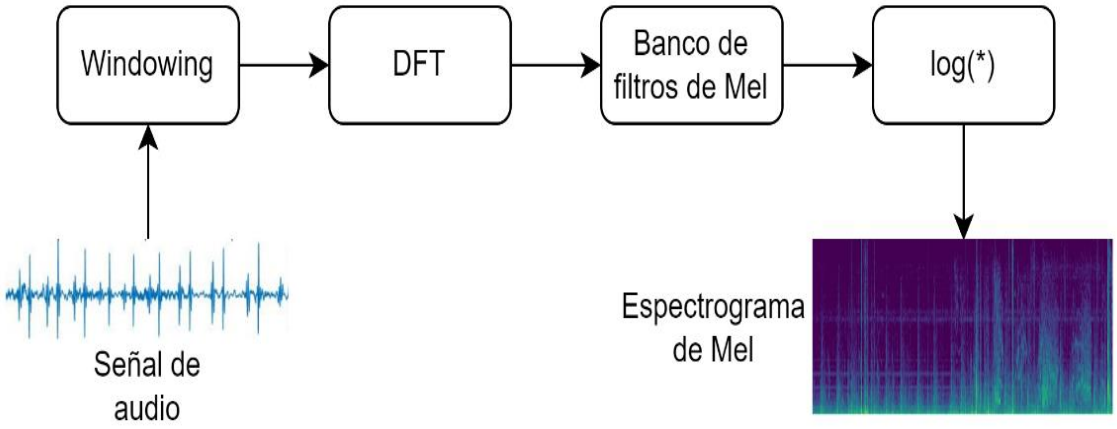
ii. Espectrograma de Mel

Esta representación de espectrograma utiliza métodos basados en el sistema sensorial humano, es decir, se ajusta a la percepción auditiva humana, lo que la hace más sensible a las frecuencias bajas y menos sensible a las altas (Guha et al., 2020).

La Figura 10 presenta un diagrama con las etapas para la obtención del espectrograma de Mel, que toma etapas de lo observado en la STFT.

Figura 10

Diagrama de bloques para la obtención del espectrograma de Mel



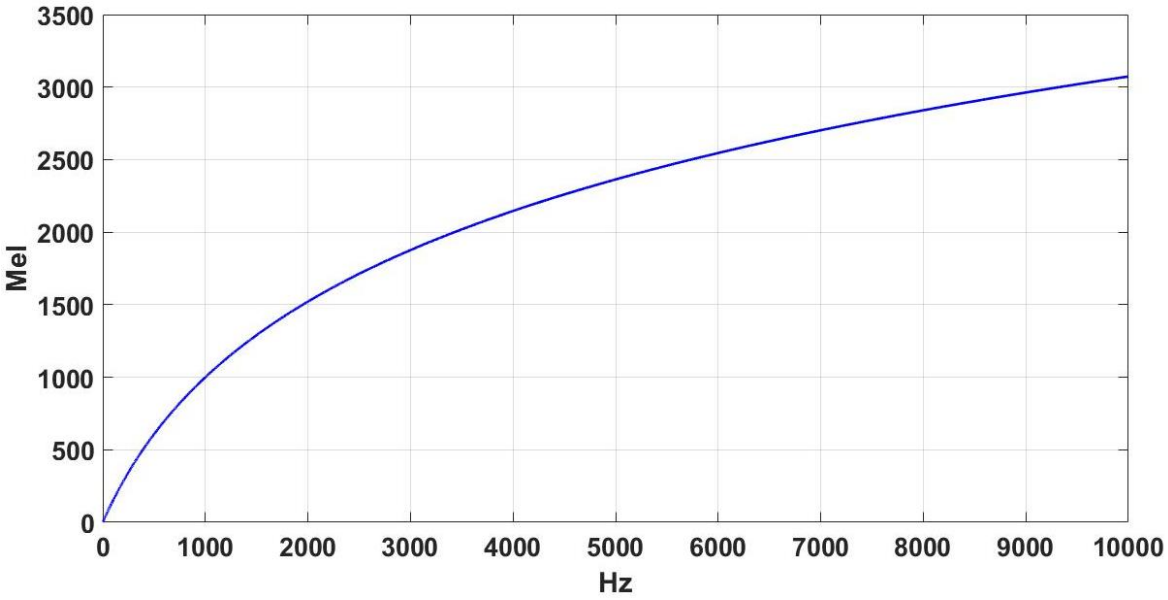
Nota: Computacionalmente, es mejor utilizar "la Transformada Rápida de Fourier" (FFT) en vez de la DFT.

Este tipo de espectrograma utiliza la escala Mel, que se visualiza en la Figura 11, y podemos aplicar la ecuación (10) para representar las frecuencias en esta escala (Guha et al., 2020):

$$m_f = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \qquad (10)$$

Figura 11

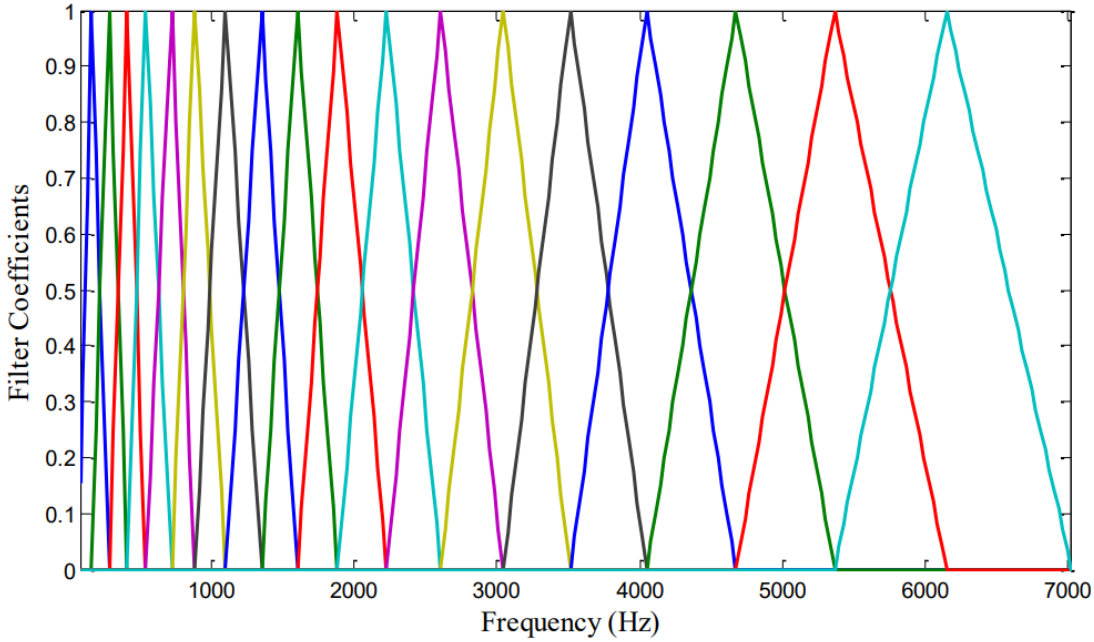
Escala de Mel



Para la obtención del espectrograma de Mel visto en el diagrama de la Figura 10, un paso crucial es la aplicación de un banco de filtros de Mel como el mostrado en la Figura 12, el cual, como se mencionó antes, se asemeja a la percepción auditiva humana (Tak et al., 2017).

Figura 12

Banco de filtros de Mel de tamaño 20



Nota: fuente (Yusnita et al., 2013).

iii. Coeficientes cepstrales en frecuencia de Mel (MFCC)

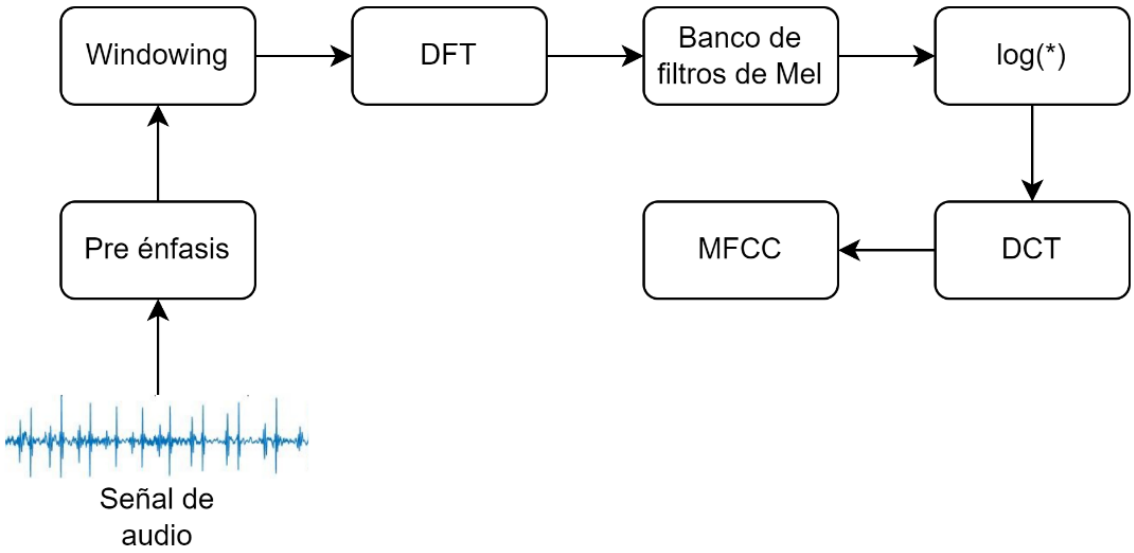
Durante mucho tiempo, los MFCC han sido un conjunto de características ampliamente utilizadas en el campo del procesamiento de señales de audio y reconocimiento del habla, ya que permiten comprimir de manera eficiente el espectro de frecuencias de dichas señales (Logan, 2000).

Los coeficientes MFCC se obtienen de las señales de audio siguiendo varios pasos, los cuales se observan en la Figura 13.

Dos etapas adicionales son agregadas a lo visto en los espectrogramas de Mel: Pre-énfasis y la Transformada de Coseno Discreta (DCT).

Figura 13

Diagrama de bloques para la extracción de los coeficientes MFCC



Pre-énfasis es una etapa de filtrado que enfatiza las frecuencias altas de la señal o, en otras palabras, es un filtro pasa alto (Barai et al., 2019). La ecuación que lo define es:

$$\hat{x}[n] = x[n] - \rho \cdot x[n-1] \quad (11)$$

Donde ρ denota el coeficiente de preénfasis y su valor se sitúa dentro del intervalo de 0.9 a 1.

La DCT se aplica para obtener los coeficientes MFCC y se expresa mediante la siguiente ecuación:

$$c_n = \sum_{b=1}^{B} \log\left(\hat{X}(b)\right) \cos\left(n\left(b - \frac{1}{2}\right)\frac{\pi}{B}\right) \quad (12)$$

Donde $\hat{X}(b)$ es la expresión obtenida al aplicar, sobre el audio, las etapas hasta el banco de filtros y B es el número de filtros (Barai et al., 2019).

b) Proceso de filtro

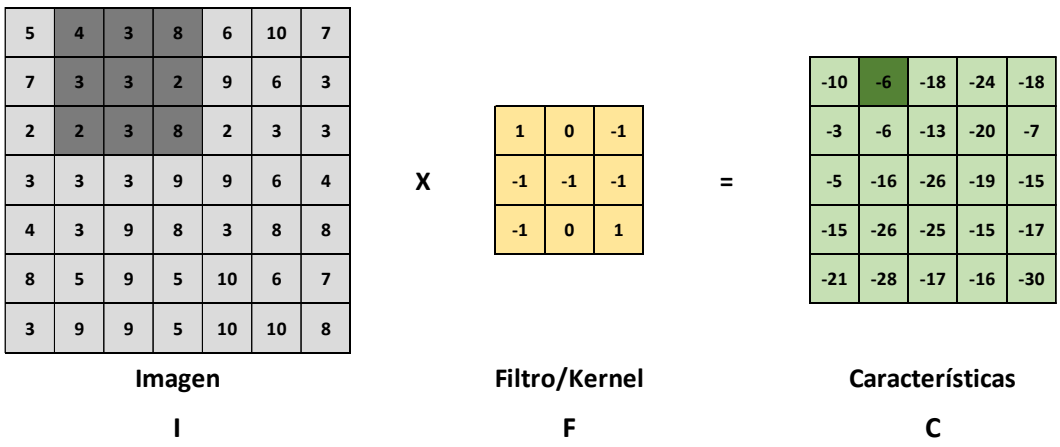
Para distinguir una imagen, nuestro cerebro a menudo se enfoca en una característica clave: los bordes. Estos bordes nos proporcionan información sobre los límites de los objetos en la imagen, lo que es esencial para reconocer y comprender lo que estamos viendo (McIlhagga, 2018).

En el contexto de las CNN, se aplican filtros convolucionales sobre una imagen, que funcionan de igual manera a nuestro cerebro, buscando activamente patrones o características específicas como bordes, texturas y formas (Olu Ipinlaye, 2023).

Los filtros convolucionales son un conjunto de pesos en una matriz de nxn, que se aplican, a través de la operación de convolución, sobre la matriz que define la imagen de entrada. Ya se ha definido esta operación de convolución en la ecuación (2), la cual implica realizar multiplicaciones y sumatorias ponderadas, recorriendo el filtro por toda la imagen con el fin de obtener una matriz resultante de características (Tejero Caballo, 2020).

Figura 14

Proceso de convolución entre la entrada y el filtro



Si se desglosa la operación de los cuadros sombreados en la Figura 14, se obtiene lo siguiente:

$$\boldsymbol{C} = \boldsymbol{I} * \boldsymbol{F} = 4 \cdot 1 + 3 \cdot 0 + 8 \cdot (-1) + 3 \cdot (-1) + 3 \cdot (-1) +$$

$$2 \cdot (-1) + 2 \cdot (-1) + 3 \cdot 0 + 8 \cdot 1 = -6$$

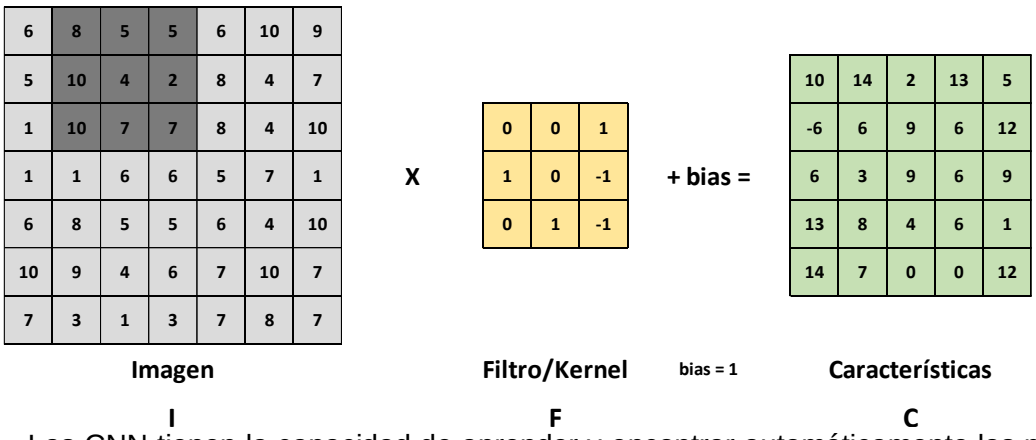
En términos generales, el tamaño de la matriz de características es $\left(\left(\frac{(n-f+2P)}{S} + 1\right), \left(\frac{(n-f+2P)}{S} + 1\right)\right)$. Donde S es el paso o stride, que indica cuantos pixeles recorre el filtro en cada paso, y P es el relleno o padding aplicado a la imagen, es decir, el número de filas y columnas que se agregan alrededor de la imagen (D. McCaffrey, 2018).

Los filtros utilizados suelen ser pequeños en comparación con la entrada, y en la Figura 14, por ejemplo, se utiliza un filtro de tamaño 3x3 en una imagen de 7x7. El padding es igual a 0 y el stride a 1. El resultado es una matriz de características de tamaño 5x5.

Además, es importante mencionar que hay un parámetro llamado bias o sesgo, que es un término adicional en la operación de convolución (Tejero Caballo, 2020). Simplemente, al resultado obtenido de la convolución, se le suma el bias tal y como se muestra en la Figura 15.

Figura 15

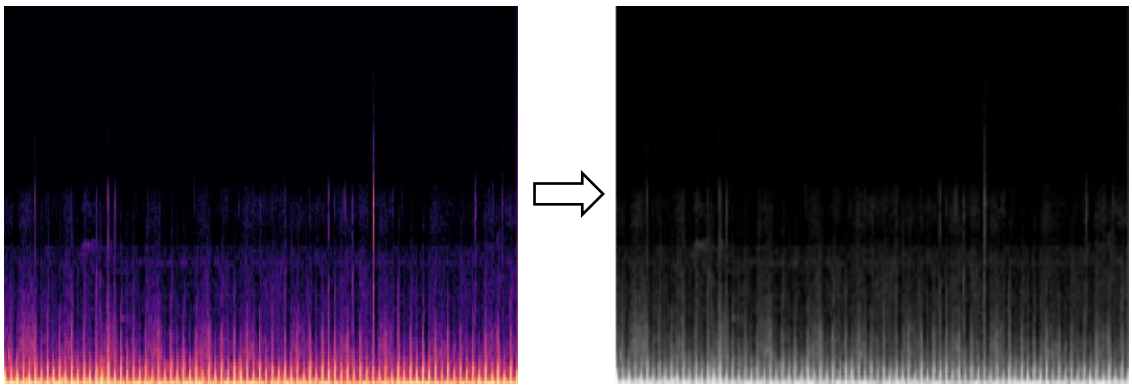
Proceso de convolución entre la entrada y el filtro más el bias



Las CNN tienen la capacidad de aprender y encontrar automáticamente los pesos respectivos de los filtros, por lo que facilita la implementación. No obstante, si se quiere comprender como operan los filtros convolucionales, es posible utilizar algunos tipos en esta etapa (Olu Ipinlaye, 2023), cada uno con tareas particulares en la detección de bordes de las imágenes, y a continuación se explicará un poco acerca de estos aplicados a la Figura 16.

Figura 16

Espectrograma de Mel en escala de grises.



i. Filtro Prewitt

El filtro Prewitt es un tipo de filtro diferencial de primer orden que calcula la derivada de la intensidad de pixeles en dos direcciones: Vertical y Horizontal (Hatice Çatal et al., 2016).

Estos filtros provienen de la definición de la derivada, que se expresa de la siguiente manera:

$$\frac{\partial I(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x}$$

$$\frac{\partial I(x,y)}{\partial y} = \lim_{\Delta y \to 0} \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y}$$

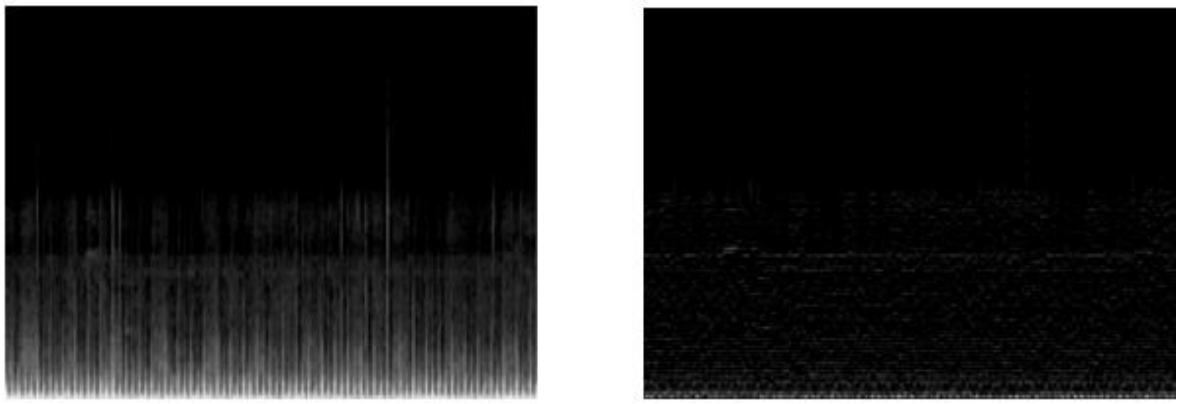
Estos límites indican que se compara la función $I(x,y)$ con un vecino $I(x+\Delta x)$ en una dirección, y con un vecino $I(y+\Delta y)$ en otra dirección. Esto, representado en una matriz se expresa de las siguientes maneras:

$$Vertical: \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$Horizontal: \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Figura 17

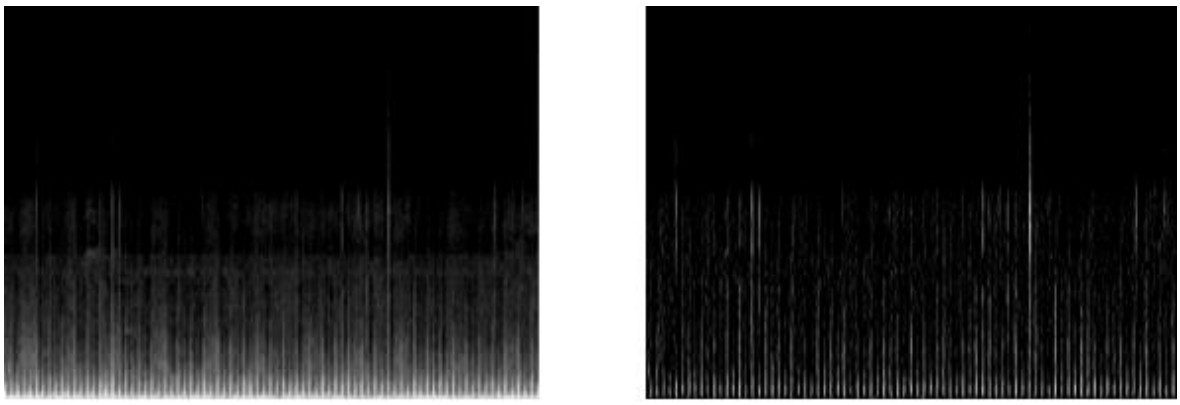
Imagen original (a) y aplicación del Filtro Prewitt vertical (b)



a) ***b)***

Figura 18

Imagen original (a) y aplicación del Filtro Prewitt horizontal (b)



a) ***b)***

El filtro vertical se utiliza para encontrar los bordes recorriendo la imagen de arriba hacia abajo. Como resultado, en la Figura 17 (b), se distinguen los límites verticales del espectrograma de Mel.

Por otro lado, la aplicación del filtro horizontal implica un recorrido de la imagen de izquierda a derecha, lo que permite resaltar de mejor manera los límites horizontales del espectrograma, como es muestra la Figura 18.

ii. Filtro Sobel

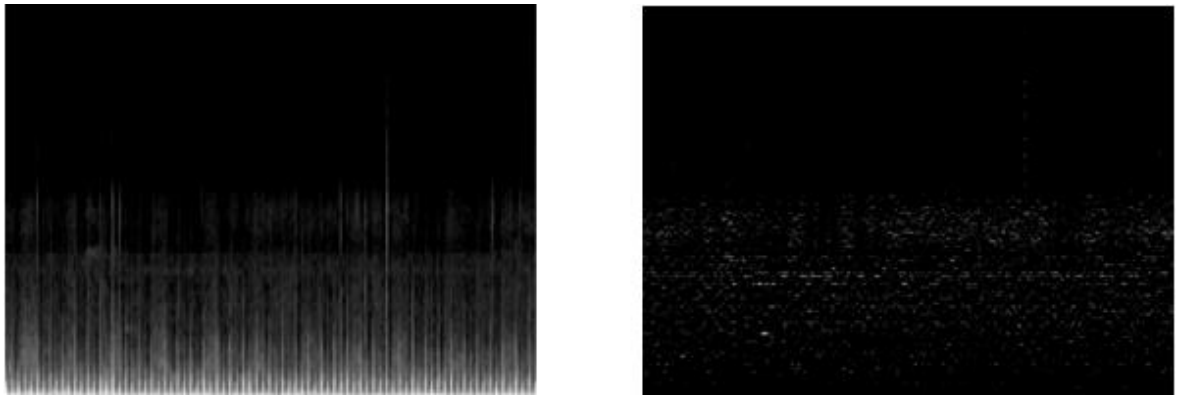
Al igual que el filtro Prewitt, el filtro Sobel es un filtro diferencial de primer orden. Sin embargo, a diferencia del filtro Prewitt, el filtro Sobel es más sensible a cambios en la intensidad en los bordes en una dirección (Hatice Çatal et al., 2016).

$$Vertical: \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$Horizontal: \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Figura 19

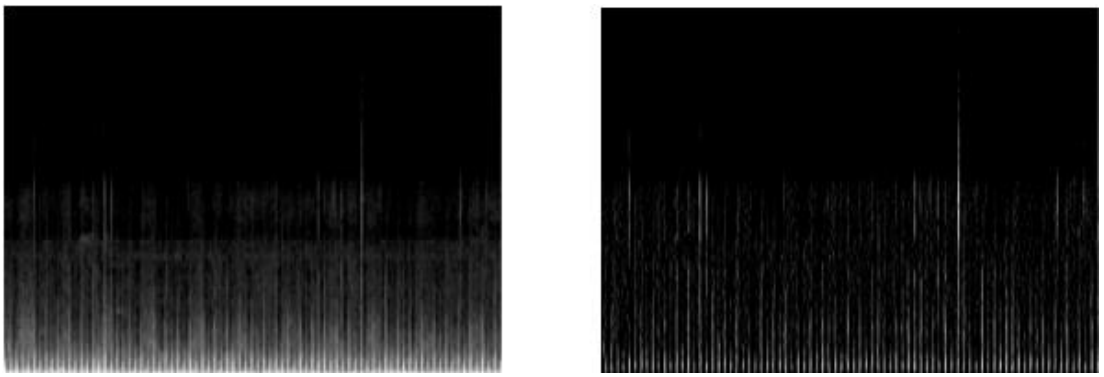
Imagen original (a) y aplicación del Filtro Sobel vertical (b)



a) ***b)***

Figura 20

Imagen original (a) y aplicación del Filtro Sobel horizontal (b)



a) ***b)***

En la Figura 19 y Figura 20 se pueden observar los límites verticales y horizontales respectivamente, que son un poco más claros comparados a los resultados obtenidos con el filtro Prewitt.

iii. Filtro Laplaciano

Es un filtro diferencial de segundo orden que se basa de la idea del Operador Laplaciano (Olu Ipinlaye, 2023). La máscara típica del filtro Laplaciano es la siguiente:

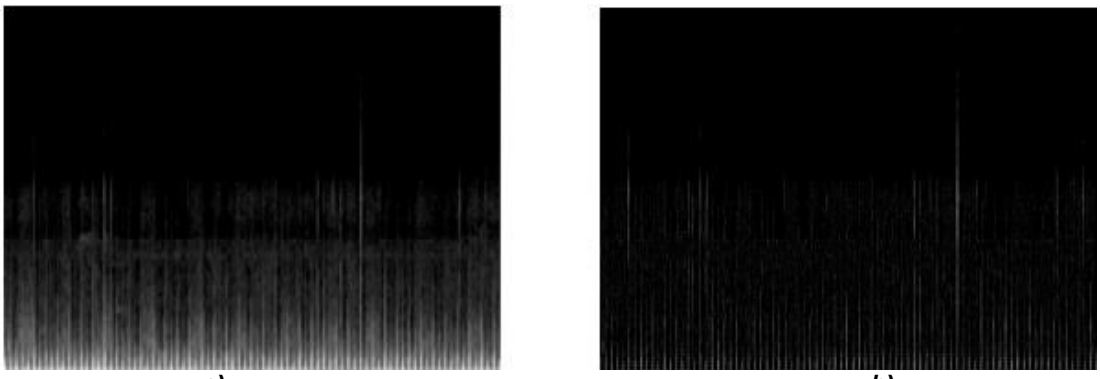
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Además, presenta variantes como las que se muestran a continuación:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad o \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figura 21

Imagen original (a) y aplicación del Filtro Laplaciano (b)



a) ***b)***

La Figura 21 muestra la aplicación de la máscara típica del Filtro Laplaciano sobre la imagen original.

iv. Máscara de Robinson

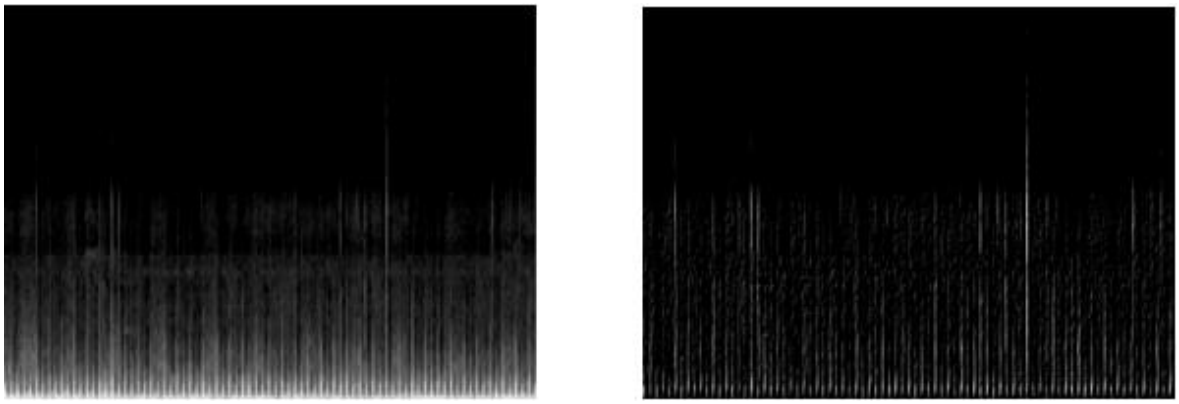
Este tipo de filtros es útil para reconocer los bordes en 8 distintas direcciones, mostrándose ejemplos en la Figura 22 y Figura 23, la cuales representan los puntos cardinales (Abdullah et al., 2019).

$$\boldsymbol{N}: \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \end{bmatrix} \quad \boldsymbol{NE}: \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{2} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ \mathbf{-2} & \mathbf{-1} & \mathbf{0} \end{bmatrix} \quad \boldsymbol{E}: \begin{bmatrix} \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ \mathbf{-2} & \mathbf{0} & \mathbf{2} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \end{bmatrix} \quad \boldsymbol{SE}: \begin{bmatrix} \mathbf{-2} & \mathbf{-1} & \mathbf{0} \\ \mathbf{-1} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{2} \end{bmatrix}$$

$$\boldsymbol{S}: \begin{bmatrix} \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{2} & \mathbf{1} \end{bmatrix} \quad \boldsymbol{SW}: \begin{bmatrix} \mathbf{0} & \mathbf{-1} & \mathbf{-2} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{2} & \mathbf{1} & \mathbf{0} \end{bmatrix} \quad \boldsymbol{W}: \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{2} & \mathbf{0} & \mathbf{-2} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \end{bmatrix} \quad \boldsymbol{NW}: \begin{bmatrix} \mathbf{2} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{-1} \\ \mathbf{0} & \mathbf{-1} & \mathbf{-2} \end{bmatrix}$$

Figura 22

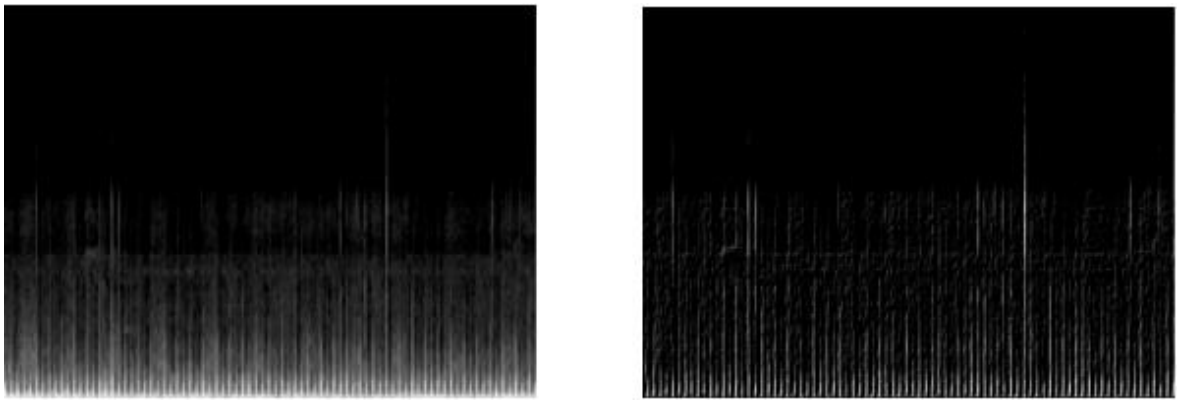
Imagen original (a) y aplicación de la Máscara de Robinson en dirección NW (b)



a) ***b)***

Figura 23

Imagen original (a) y aplicación de la Máscara de Robinson en dirección SE (b)



a) ***b)***

v. Máscara de Krisch

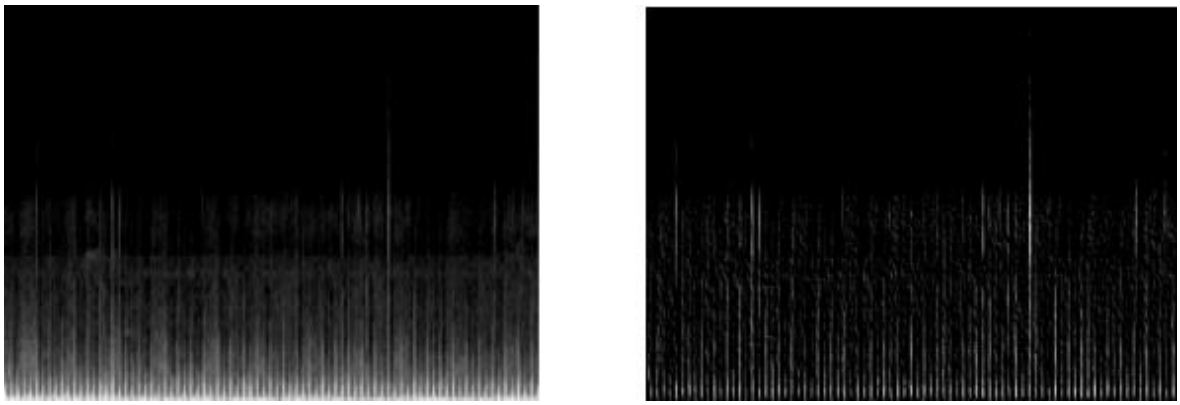
Es un filtro en 8 direcciones al igual que la Máscara de Robinson, pero con diferencias en su estructura y sus valores. Ejemplos de aplicación se muestran en la Figura 24 y Figura 25.

$$N:\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & -3 & 5\end{bmatrix}\quad NE:\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & 5\\ -3 & 5 & 5\end{bmatrix}\quad E:\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & -3\\ 5 & 5 & 5\end{bmatrix}\quad SE:\begin{bmatrix}-3 & -3 & -3\\ 5 & 0 & -3\\ 5 & 5 & -3\end{bmatrix}$$

$$S:\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix}\quad SW:\begin{bmatrix}5 & 5 & -3\\ 5 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}\quad W:\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}\quad NW:\begin{bmatrix}-3 & 5 & 5\\ -3 & 0 & 5\\ -3 & -3 & -3\end{bmatrix}$$

Figura 24

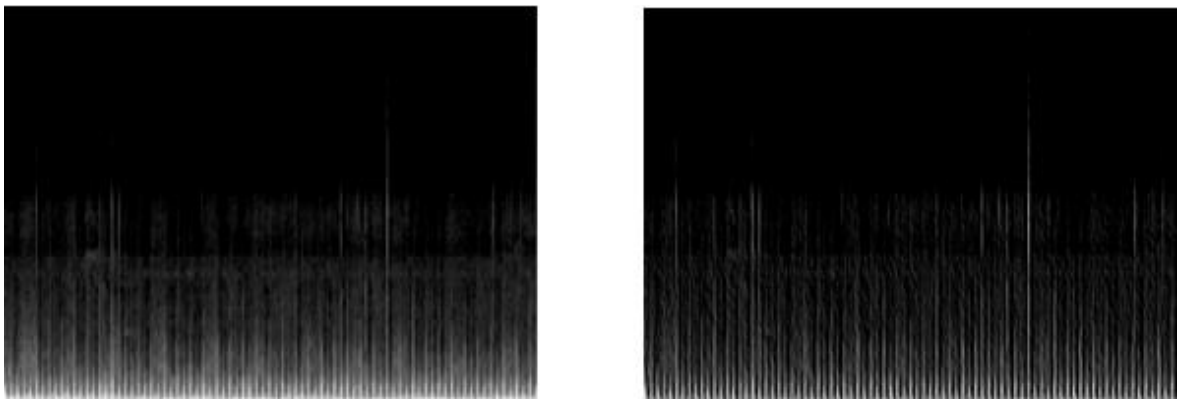
Imagen original (a) y aplicación de la Máscara de Krisch en dirección NW (b)



a) ***b)***

Figura 25

Imagen original (a) y aplicación de la Máscara de Krisch en dirección SE (b)



a) ***b)***

c) Proceso de agrupación

Las características de mayor importancia de las imágenes suelen obtenerse luego de aplicarse múltiples capas de convolución, esto implica un aumento en la complejidad del modelo, resultando un mayor coste computacional y riesgo de overfitting (Intel, 2023). Por esta razón, se suele colocar una capa de agrupación entre cada capa de convolución, para reducir su dimensión y coste de computadora.

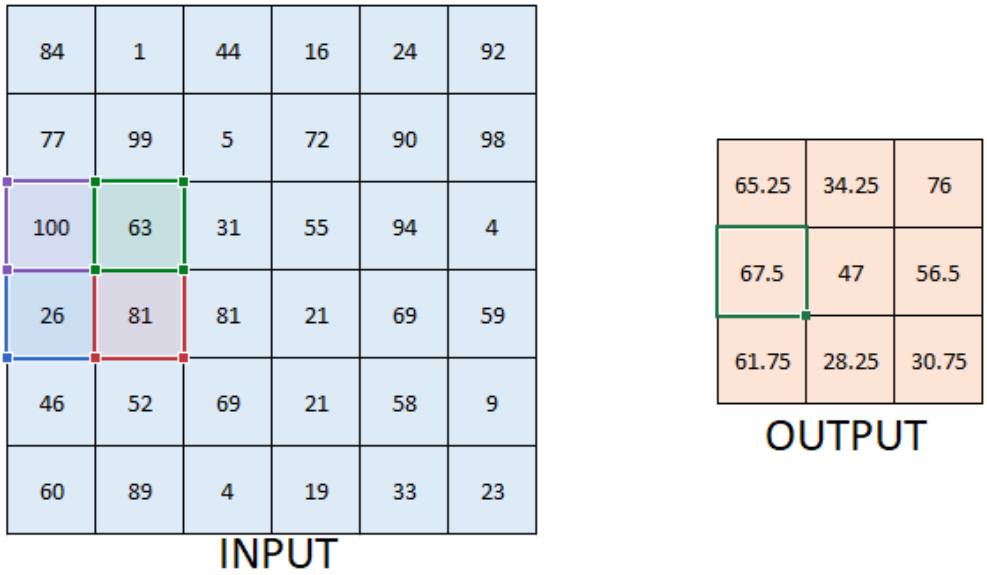
Existen varios métodos de agrupación como el Average Pooling y Max Pooling.

i. Average Pooling

Con Average Pooling, los valores de una región mxm se promedian, recorriendo toda la matriz entrante (Wang et al., 2017). Con esta media se suaviza los datos de entrada y se prefiere usarlo cuando se desea tener una visión más general de los datos.

Figura 26

Operación Average Pooling

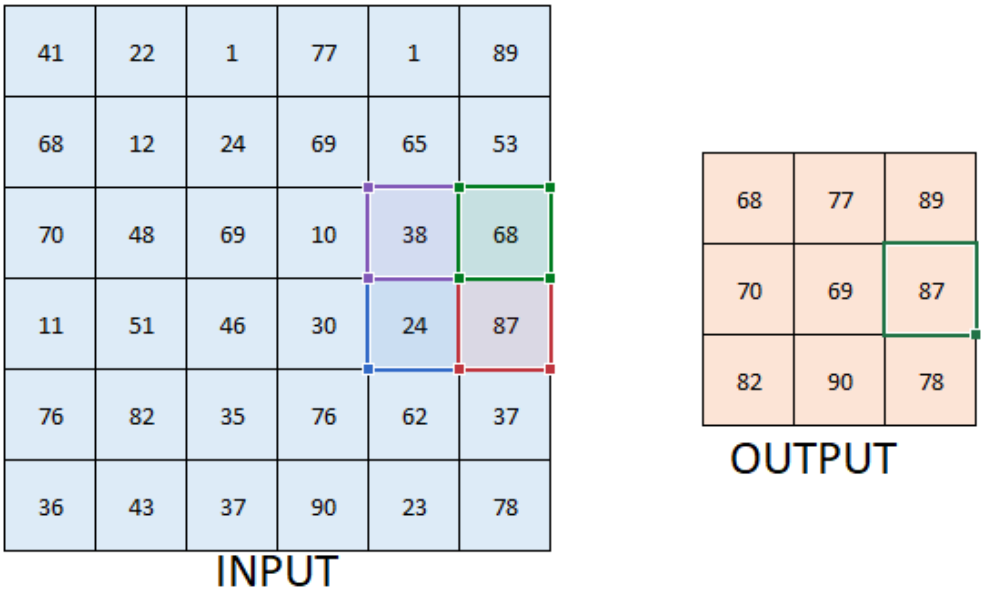


ii. Max Pooling

Esta operación es más sencilla aún ya que se tiene que escoger el mayor valor por cada región recorrida (Wang et al., 2017). Con este método, se mantienen las características más importantes de la matriz, reduciendo al mismo tiempo la dimensión de esta y a diferencia del average pooling, este método retiene menos información.

Figura 27

Operación Max Pooling



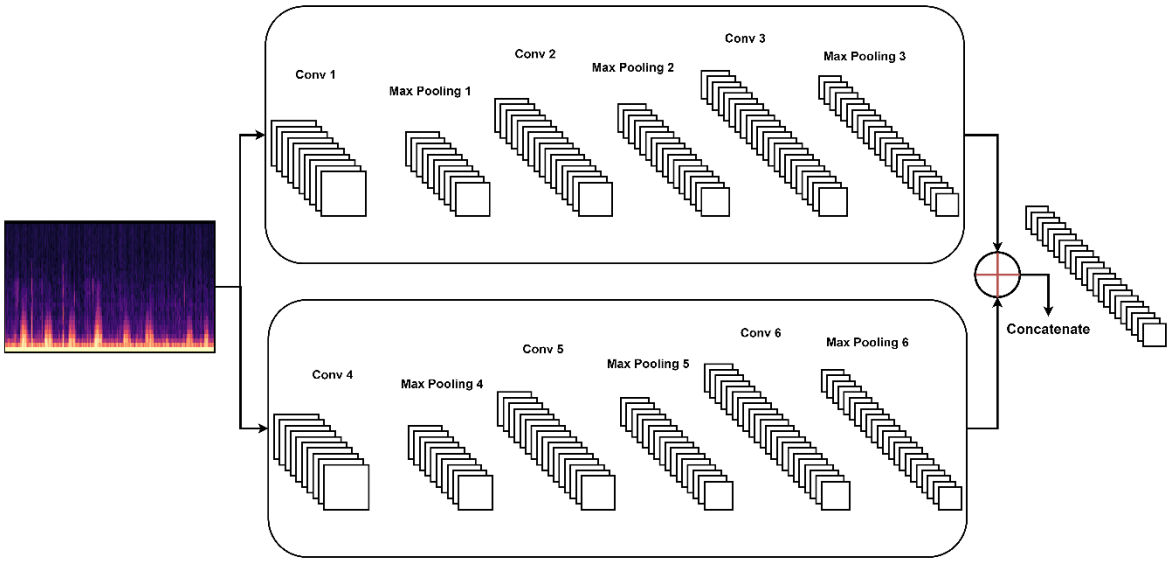
En los ejemplos mostrados en la Figura 26 y Figura 27 se tomaron regiones de 2x2 y un stride de 2 para recorrer la matriz de tamaño 6x6. Por cada región 2x2, se realiza la operación de agrupación correspondiente y se coloca en la matriz de salida.

Después de explorar la etapa de extracción de características en una CNN, surge una preocupación importante, que es el desafío de trabajar con conjuntos de datos limitados y la amenaza de overfitting o sobreajuste, especialmente cuando se diseñan capas profundas en la red (Del Coco et al., 2017).

Para abordar esta problemática, una solución prometedora es adoptar un enfoque de múltiples ramas en la arquitectura de la CNN. Esto implica dividir la red en varias ramas, cada una con una profundidad reducida en comparación con una red convencional. Cada rama se especializa en la extracción de características a una escala particular, lo que permite capturar patrones de los datos de entrada a diferentes niveles de detalle.

Figura 28

Arquitectura CNN de doble rama



La Figura 28 muestra una arquitectura CNN de doble rama, que utilizan las mismas imágenes como entrada. Sin embargo, existen variantes híbridas donde cada rama de la CNN recibe datos diferentes. Por ejemplo, una rama podría procesar los espectrogramas de Mel, mientras que la otra podría trabajar con los MFCC.

2. Clasificación de características

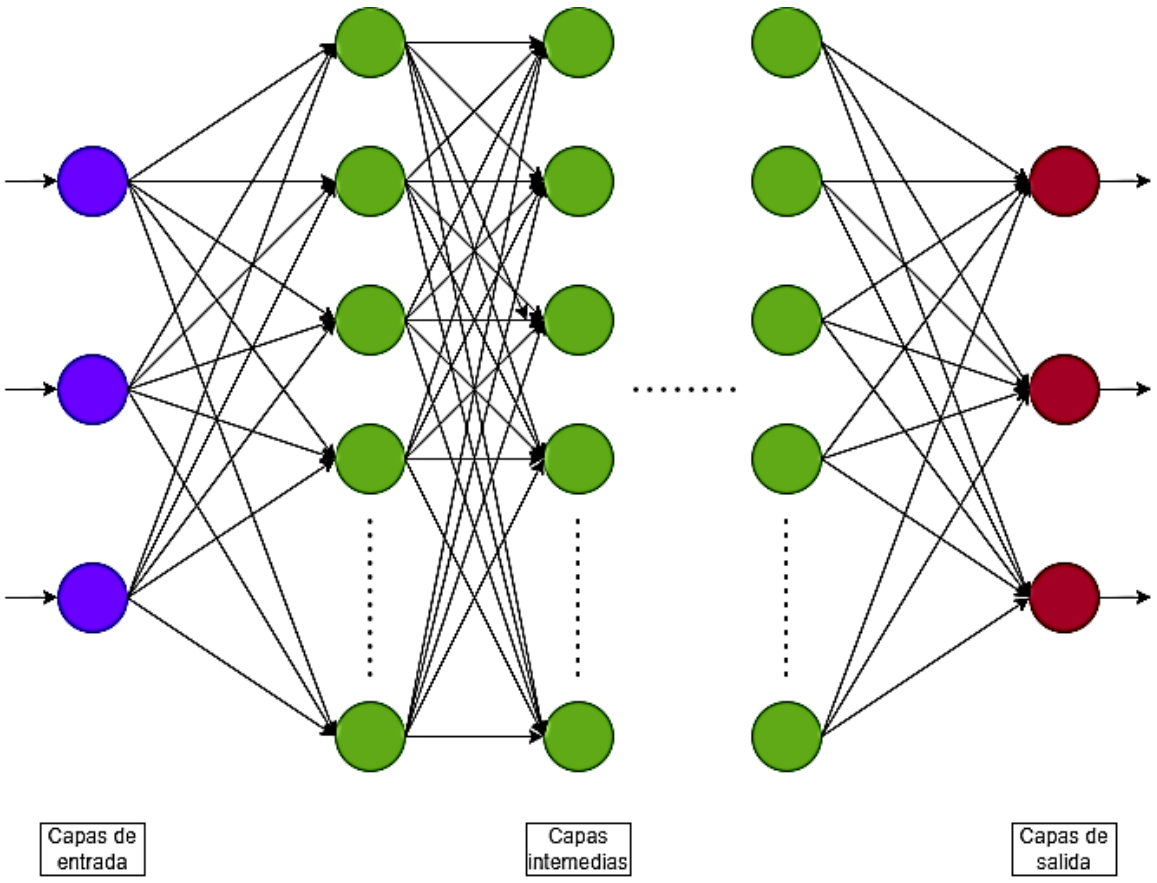
Una vez culminada la parte de convolución, se contará con matrices de entrada de dimensiones más pequeñas que las imágenes iniciales que contienen características relevantes de estas. Esto se vuelve la nueva entrada para la parte de clasificación, la cual se muestra en la Figura 29 y son las llamadas redes neuronales artificiales.

Estas redes artificiales aprenden de forma ordenada y progresiva, es decir que entre cada salto entre capas de neuronas va obteniendo resultados más complejos hasta llegar a una capa de salida, cuyo tamaño dependerá de las categorías que se desee clasificar (Bárcena Rodríguez, 2022).

Las neuronas que se mencionan en la definición anterior se refieren a unidades de procesamiento de las redes neuronales y son las encargadas de realizar el procesamiento de los datos de entrada, ejecutando un cálculo o una operación matemática.

Figura 29

Representación gráfica de una red neuronal



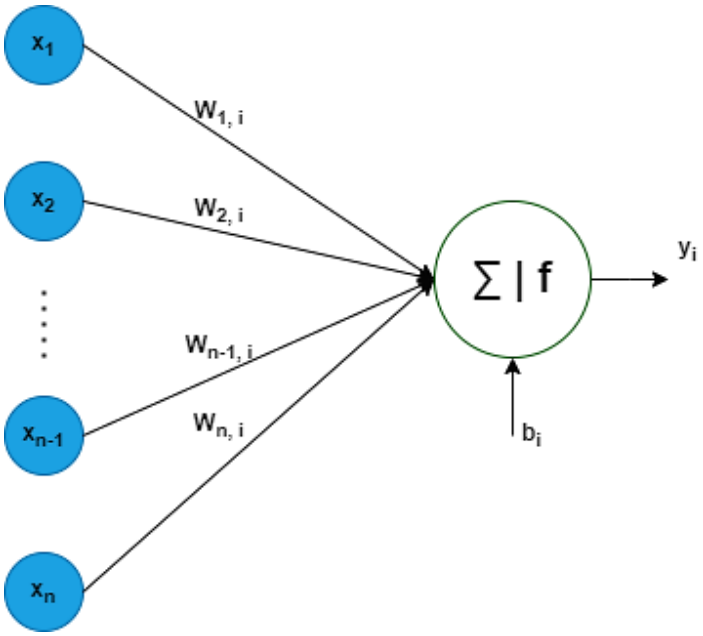
La operación matemática que realiza cada neurona está descrita en la ecuación (13), la cual inicia con una combinación lineal de las N entradas de cada capa anterior con sus respectivos parámetros denominados pesos (w_i), y a este resultado se le suma el bias. Finalmente se le aplica la función de activación f (Rey, 2022).

$$y_i = f\left(b_i + \sum_{j=1}^{N} w_{j,i}\, x_j\right) = f(\bar{x}) \qquad (13)$$

Donde i es la posición de la neurona en una respectiva capa.

Figura 30

Representación gráfica de la estructura matemática de una red neuronal



La Figura 30 muestra un ejemplo para la neurona i de una capa, repitiéndose el mismo análisis para las demás neuronas hasta terminar con todas.

Inicialmente, los parámetros de la CNN como los pesos, bias y filtros se eligen aleatoriamente para la primera iteración de la red. Se recorre la red con estos primeros valores y en cuanto se obtengan los valores de salida, hace aparición una técnica esencial conocida como Backpropagation (Singh, 2020).

El Backpropagation es un proceso que ajusta los parámetros de la red luego de comparar sus salidas con los valores reales. Esta técnica tiene un papel crucial en Deep Learning, debido a que le da la capacidad de aprender a la red.

Adicionalmente, existen dos elementos útiles que influyen en la operación matemática de cada neurona descrita en la ecuación (13).

El primero es **Dropout**, una técnica eficaz para evitar el overfitting. La idea detrás de Dropout es simple: durante el entrenamiento, aleatoriamente se apaga un porcentaje de las neuronas en una capa, lo que significa que estas neuronas no contribuyen por un período a la propagación de la información en la red ni de la retropropagación de errores. Esto ocurre en cada iteración de la red. En consecuencia, evita que las neuronas se vuelvan demasiado especializadas en ciertos patrones de datos, forzando el aprendizaje de características más robustas y generales (Barmettler, 20 C.E.).

El segundo es **Batch Normalization**, un algoritmo que acelera el entrenamiento y mejorar la estabilidad de las redes neuronales (Garbin et al., 2020). Este algoritmo consiste en tres pasos (Peccia, 2018):

- **Primer paso**: Cálculo de la media y la varianza.

$$\mu_B = \frac{1}{m}\sum_{j=1}^{m} x_j \qquad (14)$$

$$\sigma_B^2 = \frac{1}{m}\sum_{j=1}^{m}\left(x_j - \mu_B\right)^2 \qquad (15)$$

- **Segundo paso**: Normalización.

$$\widehat{x_j} = \frac{x_j - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \qquad (16)$$

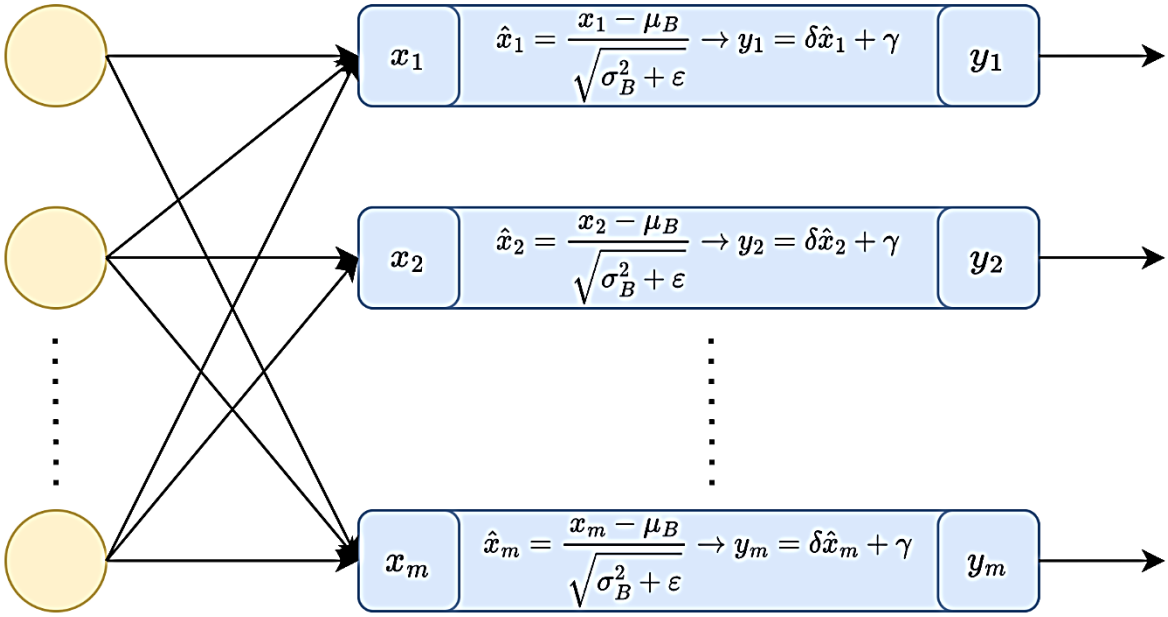
- **Tercer paso**: Ajuste de la normalización, donde aparecen dos parámetros obtenidos en el entrenamiento.

$$y_j = \delta\widehat{x_j} + \gamma \qquad (17)$$

En la Figura 31 se puede observar estos pasos en un diagrama.

Figura 31

Batch Normalization



a) Función de activación

En las redes neuronales se utilizan comúnmente varias funciones de activación, entre las cuales se incluyen la función rectificadora lineal unitario (ReLU), la función escalón, la función sigmoide, etc. (Maisueche Cuadrado, 2019).

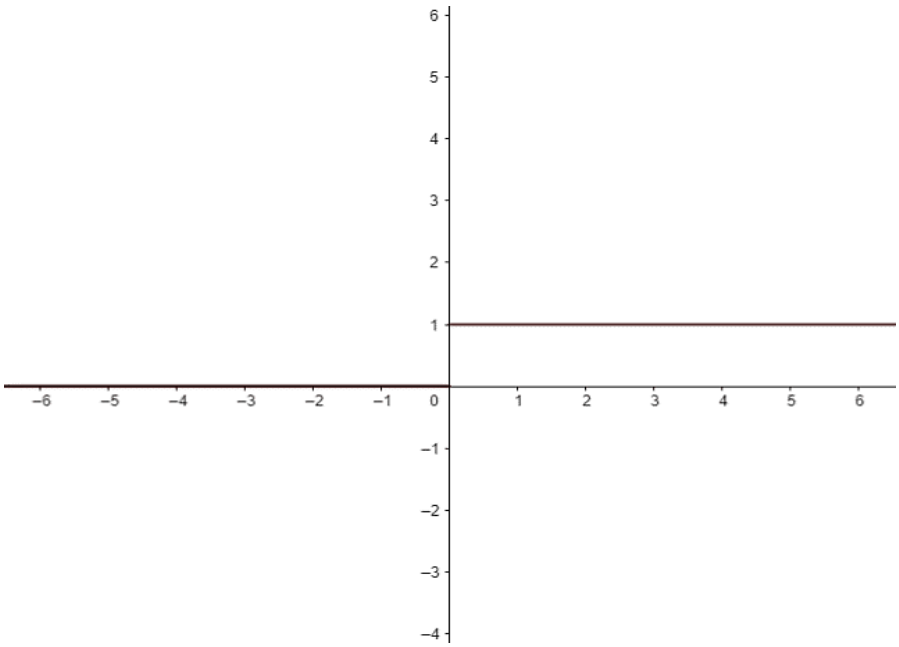
i. Función escalón

Esta función es de las más simples, ya que en su lógica, activa la neurona si el valor de entrada es mayor a un umbral, y, caso contrario, la desactiva (Sharma et al., 2020). Se define de la siguiente manera:

$$f(\bar{x}) = \begin{cases} 0 \; si \; \bar{x} < 0 \\ 1 \; si \; \bar{x} \geq 0 \end{cases} \qquad (18)$$

Figura 32

Función escalón



La función escalón, mostrada en la Figura 32, se utiliza en casos en que se quiera clasificar en 2 categorías.

Si bien es una función simple, la función escalón cuenta con varias limitantes, siendo la más significativa la referente a la actualización de los pesos. Dado que la función tiene un gradiente igual a cero en todo su dominio, impide que los pesos se actualicen (Gupta, 2023).

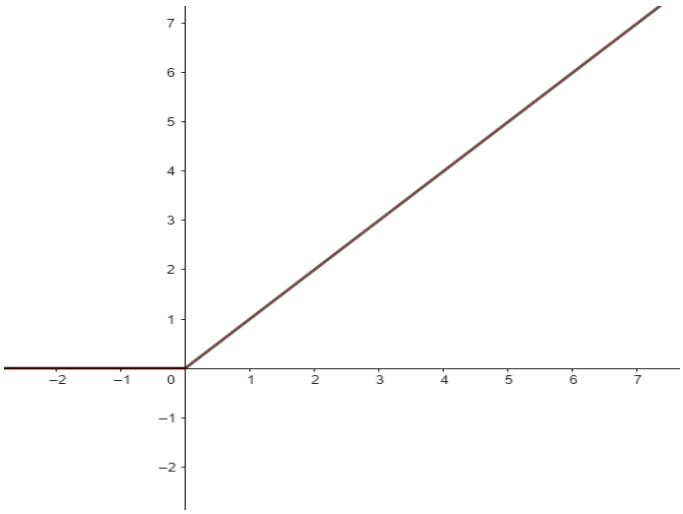
ii. Función rectificadora lineal unitario (ReLU):

La función ReLu es de la más populares debido a su forma simple y a su ventaja principal de que no todas las neuronas se mantienen activas al mismo tiempo ante una determinada entrada (Sharma et al., 2020). Matemáticamente, se representa de la siguiente manera:

$$f(\bar{x}) = \max(0, \bar{x}) \quad (19)$$

Figura 33

Función ReLu



La simplicidad de la función ReLu, mostrada en la Figura 33, hace que sea computacionalmente eficiente. Sin embargo, pese a estos beneficios de eficiencia y simplicidad, esta función de activación presenta problemas como los siguientes:

- Existe la posibilidad de que algunas neuronas permanezcan inactivas durante el entrenamiento, lo cual resultaría en una falta de actualización de sus pesos.
- Si la región negativa contiene información muy importante, se perderán, ya que la función ReLu les asigna el valor de 0.

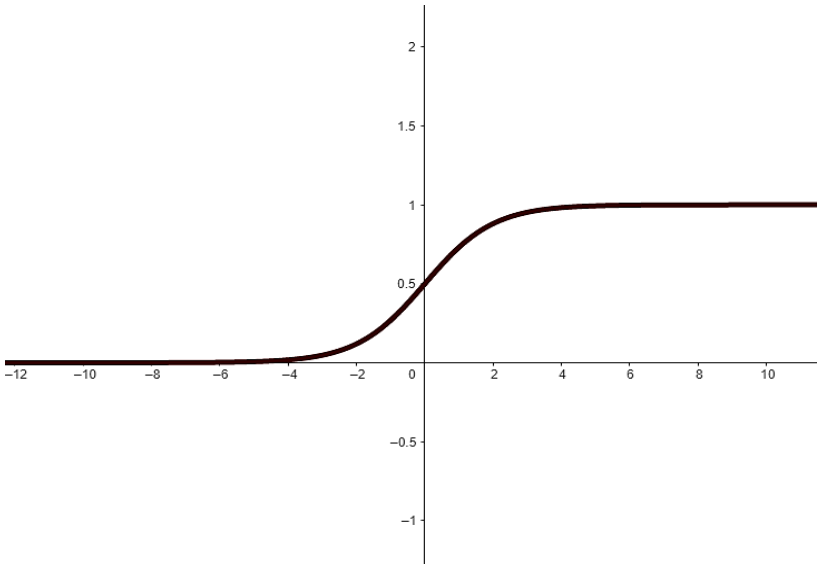
iii. Función sigmoide

La función Sigmoide también es ampliamente empleada. Cualquier dato de entrada es transformado a un valor dentro del intervalo de 0 a 1. Es una función no lineal y la definición matemática es la siguiente:

$$f(\bar{x}) = \frac{1}{1 + e^{-\bar{x}}} \qquad (20)$$

Figura 34

Función sigmoide



Como se aprecia en la Figura 34, esta función tiene una curva suave y diferenciable en todo su rango, facilitando el cálculo de los gradientes y, por ende, mejora la precisión en el ajuste de los pesos (Sharma, 2017).

Debido a que el valor de su salida está entre 0 y 1, es una función muy usada cuando se requiere una salida en forma de probabilidad.

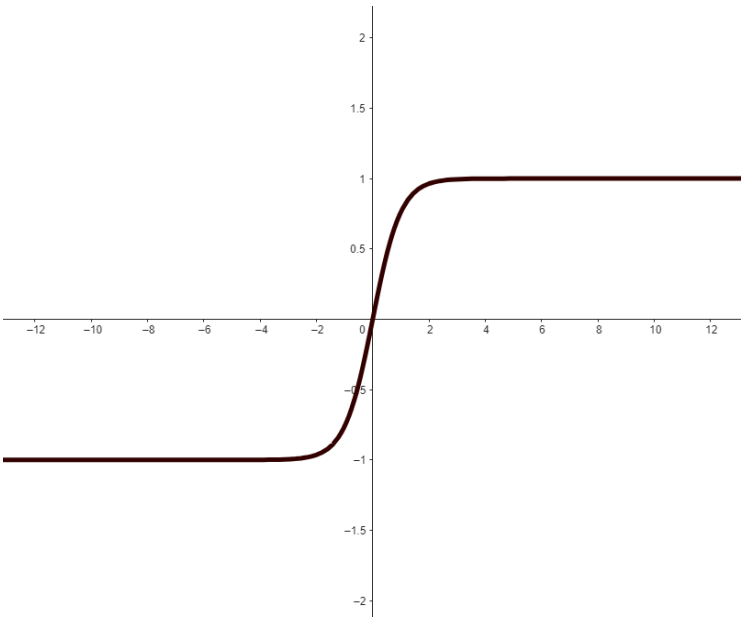
iv. Función tangente hiperbólica

La tangente hiperbólica es simétrica en el origen, tal y como muestra la Figura 35, mapeando los valores de salida en el rango de -1 a 1 (Sharma et al., 2020). Se define matemáticamente como:

$$f(\bar{x}) = \frac{1 - e^{-\bar{x}}}{1 + e^{-\bar{x}}} \qquad (21)$$

Figura 35

Función tangente hiperbólica



La función Sigmoide es útil cuando se necesiten salidas que abarquen valores tanto positivos como negativos o centradas en cero.

v. Función softmax

A diferencia de las funciones anteriores, Softmax está diseñada especialmente para clasificaciones de datos en varias categorías, y es utilizada comúnmente en la capa final de la red neuronal (Sharma et al., 2020). Se expresa matemáticamente como:

$$softmax(\bar{x})_j = \frac{e^{\bar{x}_j}}{\sum_{i=1}^{M} e^{\bar{x}_i}} \quad (22)$$

La función Softmax calcula el valor exponencial de cada valor de entrada para luego normalizar cada uno de estos.

b) Función de pérdida

En el entrenamiento de una red neuronal, una función denominada función de pérdida se hace presente, la cual compara la predicción de una característica con la original. Estas funciones permiten evaluar la calidad de la modelización de la red neuronal (Zhao et al., 2017).

De la gran cantidad de funciones de pérdida que existen, se suelen utilizar las siguientes:

i. Mean Squared Error (MSE)

Esta función realiza el cálculo de la media del cuadrado de la diferencia de la característica objetivo (y) y la predicha ($\hat{y}$) (Khan et al., 2020).

$$MSE = \frac{1}{n}\sum_{j=0}^{n}\left(y_j - \hat{y_j}\right)^2 \quad (23)$$

ii. Mean Absolute Error (MAE)

Como se menciona en su nombre, esta función encuentra la media del valor absoluto de la diferencia entre y e $\hat{y}$ (Khan et al., 2020).

$$MAE = \frac{1}{n}\sum_{j=0}^{n}\left|y_j - \hat{y_j}\right| \quad (24)$$

Esta función al igual que el MSE son utilizadas en redes neuronales de regresión.

iii. Binary Cross-Entropy

Dada una distribución o probabilidad P, la entropía se define con la siguiente ecuación (Zhang et al., 2021):

$$H(P) = \sum_{i} -P_i * logP_i \quad (25)$$

En redes neuronales, P hace referencia a la probabilidad de que una característica objetivo sea predicha correctamente. En el caso de un modelo de clasificación binaria, solo se tendrán 2 categorías como salida y las probabilidades p serán representadas con un 0 o un 1 (Ho & Wookey, 2020).

$$CE\ Loss = \frac{1}{n}\sum_{k=0}^{n} -(y_k * \log(p_k) + (1 - y_k) * \log(1 - p_k)) \quad (26)$$

Donde n es la cantidad de datos de entrenamiento.

iv. Categorical Cross-Entropy

A diferencia de la función anterior, Categorical Cross-Entropy se utiliza cuando se desea clasificar en más de 2 categorías (Ho & Wookey, 2020).

$$CE\ Loss = -\frac{1}{n}\sum_{k=0}^{L}\sum_{i=0}^{n} y_{ik} * \log\left(p_{ik}\right) \quad (27)$$

Donde L es la cantidad de categorías.

c) Algoritmos de optimización

Los algoritmos de optimización son métodos que se utilizan con la intención de minimizar la función de pérdida, mejorando el rendimiento de los modelos (Liquet et al., 2022). Estos algoritmos desempeñan un papel decisivo en el entrenamiento de las CNN, ya que permiten ajustar los pesos y sesgos de la red para encontrar el óptimo global o local de la función de pérdida, superando obstáculos como mínimos locales y puntos de silla.

Antes de adentrarnos en la discusión de estos algoritmos, es esencial comprender dos conceptos clave mencionados anteriormente:

- **Mínimo local**: Este término se refiere a los puntos donde la gradiente de la función de pérdida es igual a 0: $\nabla f(\theta) = 0$ y la matriz Hessiana es semidefinida positiva.
- **Punto de silla**: Estos puntos aparecen cuando la gradiente de la función de pérdida es 0 pero no es un extremo local (mínimo o máximo). Además, la matriz Hessiana es indefinida.

i. Descenso de gradiente (GD)

Este método de optimización es considerado uno de los más básicos con el cual se utiliza la información de la gradiente de la función de pérdida f para actualizar sus parámetros en el sentido opuesto al gradiente, y de esta manera, disminuir el valor de f (Hallén, 2017).

Sea $f(\theta)$ la función de pérdida y que representa la medida del error entre las salidas predichas por el modelo y las salidas reales, y θ es la representación de los parámetros de la red neuronal, entonces:

$$\theta_{i+1} = \theta_i - \varphi_i \nabla f(\theta_i) \quad (28)$$

La tasa de aprendizaje, representada por φ_i, determina cuán grandes serán los pasos de actualización, y su valor se irá tanteando hasta encontrar el mejor número para llegar a un mínimo de la función f. Sin embargo, es importante tener en cuenta que este algoritmo está limitado a funciones diferenciables en todos sus puntos, y, además, solo es capaz de encontrar un punto mínimo, incluso si no es el mínimo global (IBM, 2023).

A manera de ejemplo se tiene la función de Himmelblau:

$$f(\theta) = (\theta_1^2 + \theta_2 - 11)^2 + (\theta_1 + \theta_2^2 - 7)^2 \quad (29)$$

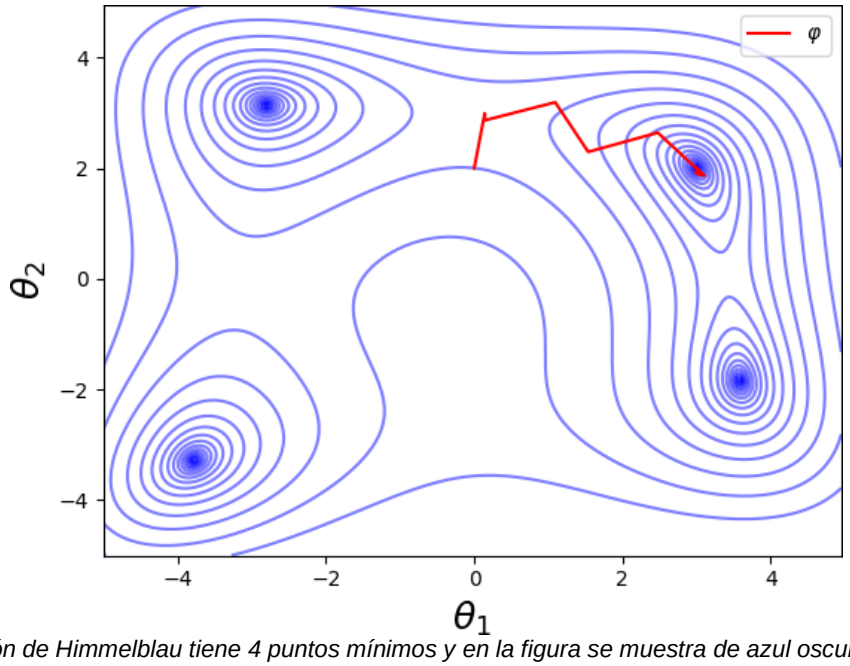
Y sus derivadas parciales son:

$$\frac{\partial f(\theta)}{\partial \theta_1} = 4\theta_1(\theta_1^2 + \theta_2 - 11) + 2(\theta_1 + \theta_2^2 - 7) \quad (30)$$

$$\frac{\partial f(\theta)}{\partial \theta_2} = 2(\theta_1^2 + \theta_2 - 11) + 4\theta_2(\theta_1 + \theta_2^2 - 7) \quad (31)$$

Figura 36

Comportamiento de Descenso de gradiente

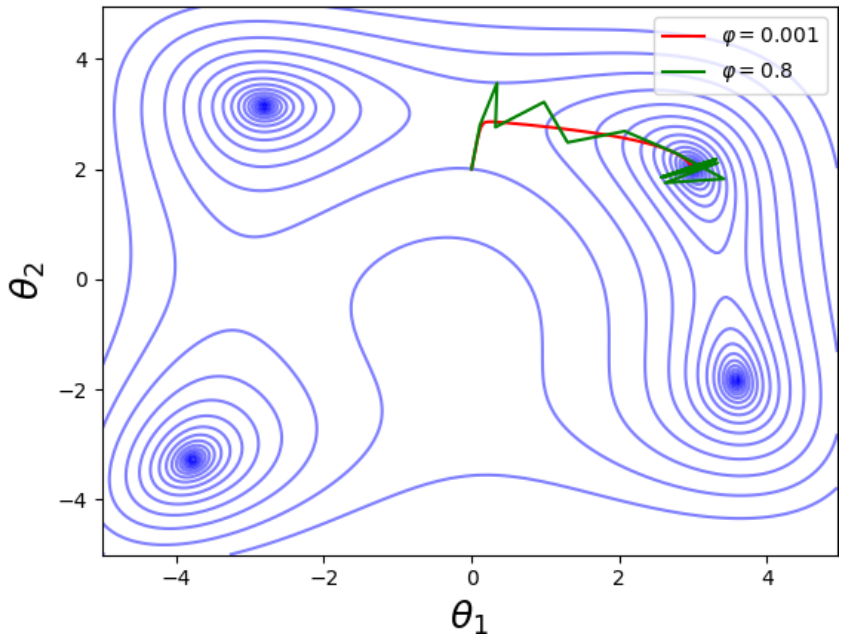


Nota: La función de Himmelblau tiene 4 puntos mínimos y en la figura se muestra de azul oscuro

Se observa en la Figura 36 cómo varía φ hasta encontrar uno de los puntos mínimos de la función de Himmelblau. Ahora observemos cómo sería la búsqueda de este punto mínimo si φ es fijado en un valor.

Figura 37

Comportamiento de GD con valores fijados de ***φ***



La Figura 37 muestra el comportamiento del algoritmo con valores fijos de taza de aprendizaje. Si φ tiene un valor muy pequeño, eventualmente alcanzará un punto mínimo, pero le tomaría mucho tiempo y requeriría numerosas iteraciones. Por otro lado, si le asignamos un valor demasiado grande, su comportamiento será caótico, ya que estará dando saltos amplios y no logrará converger a un punto mínimo de la función.

ii. Gradiente conjugado (CGD)

Al igual que el método de GD, el gradiente conjugado también utiliza la gradiente de la función de pérdida para hallar su mínimo. La diferencia recae en la dirección de descenso (d), donde en el GD se escoge $d_i = -\nabla f(\theta_i)$, mientras que para el CGD se le suma la dirección previa multiplicada por un escalar (Lindfield & Penny, 2019).

$$\theta_{i+1} = \theta_i + \varphi_i d_i \tag{32}$$

$$d_{i+1} = \delta d_i - \nabla f(\theta_{i+1}) \tag{33}$$

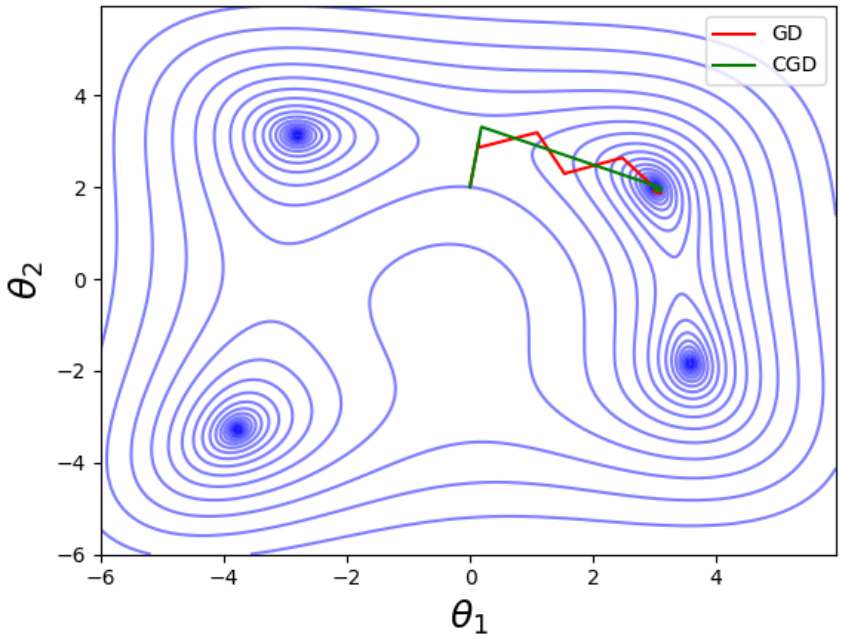
Como en su nombre lo indica, la dirección de descenso escogida d_{i+1} y la dirección previa d_i, son mutuamente conjugadas respecto a una matriz simétrica, positiva y real H, y se expresa de la siguiente manera.

$$(d_{i+1})^T H d_i = 0 \tag{34}$$

CGD converge a un punto mínimo mucho más rápido que GD (Barnard & Cole, 1989), y esto se visualiza en la Figura 38.

Figura 38

Descenso de gradiente vs Gradiente conjugado



iii. Descenso de gradiente estocástico (SGD)

SGD es similar al descenso de gradiente, con una fórmula que lo define también similar.

$$\theta_{i+1} = \theta_i - \varphi_i \nabla f(\theta_i, x_i) \qquad (35)$$

Si comparamos las ecuaciones (28) y (35), notaremos que la única diferencia es el término x_i, el cual representa a una muestra aleatoria sobre la que se calculan la predicción, el error y el backpropagation. Entonces, SGD es el descenso de gradiente aplicado a una muestra aleatoria (Hallén, 2017).

SGD es más caótico que el descenso de gradiente debido a su aleatoriedad, pero reduce bastante el costo computacional y aumenta la velocidad de convergencia.

iv. Momentum

Cuando se presenten funciones de pérdida que exhiban superficies extensas y casi planas (mesetas) alrededor de puntos de silla, el valor de la gradiente en ese contexto es cercana a cero. Esto ocasiona que, utilizando los métodos de GD, las actualizaciones de los parámetros y la convergencia sean muy lentas (Dauphin et al., 2014).

En solución a estos problemas, aparece el algoritmo de optimización Momentum, que se utiliza junto con el GD, agregando un parámetro iterable más denominado parámetro de impulso (β) (Liquet et al., 2022).

$$v_{i+1} = \beta v_i + \nabla f(\theta_i) \quad (36)$$

$$\theta_{i+1} = \theta_i - \varphi_i v_{i+1} \quad (37)$$

Este nuevo parámetro debe ser menor a 1 y mayor a 0. Notar que si $\beta = 0$, entonces se tiene el método de GD.

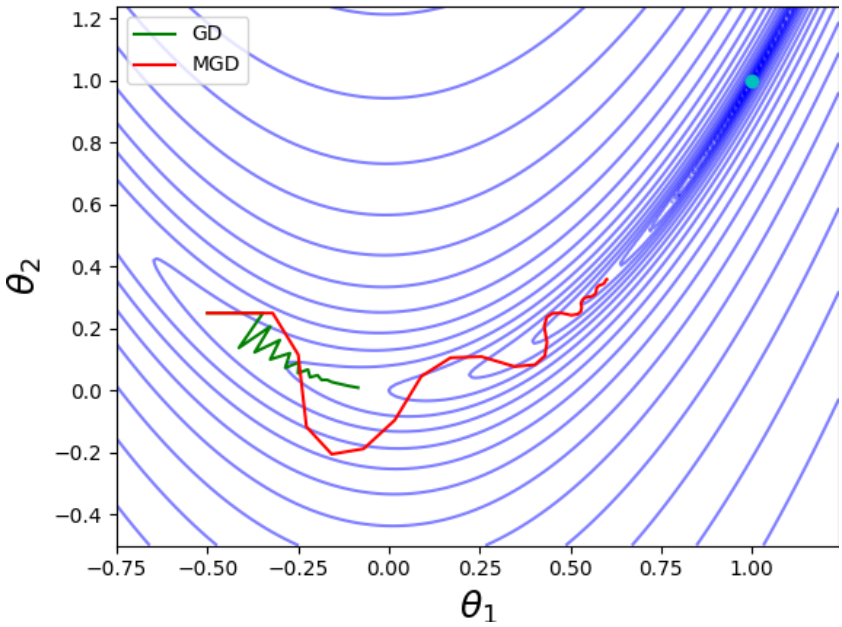
Este método de optimización mejora aún más la velocidad de convergencia al mínimo, ya que le provee “memoria” al SDG al acumular las gradientes de los pasos anteriores en una variable de velocidad acumulada v_{i+1} (Ding et al., 2019). Además, ayuda a superar los mínimos locales y a evitar oscilaciones en el proceso (Haji & Abdulazeez, 2021).

Una muy buena manera de visualizar las ventajas de este algoritmo es con un ejemplo, y, para este caso, se utilizará la función de Rosenbrock como función de pérdida.

$$f(\theta) = (1 - \theta_1)^2 + 100(\theta_2 - \theta_1^2)^2 \quad (38)$$

Figura 39

GD sin Momentum y con Momentum



En la Figura 39 se observa que ambos algoritmos no llegan al punto mínimo marcado de color cian (1,1). Sin embargo, GD con Momentum se acerca significativamente más al mínimo en comparación con GD utilizando la misma cantidad de iteraciones.

v. Adaptive Gradient Algorithm (AdaGrad)

Es un método de optimización adaptativo cuya característica es de ajustar la tasa de aprendizaje individualmente para cada parámetro según sus ajustes anteriores al acumular la suma de los cuadrados de las gradientes pasados y actuales (Atici et al., 2022). Esto lo realiza mediante el siguiente cálculo:

$$Q_{i,k} = \sum_{j=0}^{i} \left(\frac{\partial f(\theta_j)}{\partial \theta_k} \right)^2 \quad (39)$$

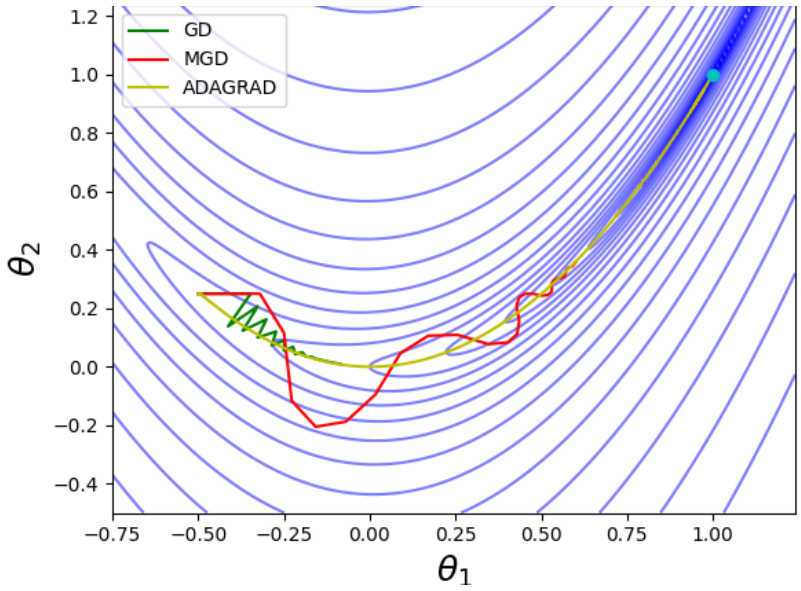
$$\theta_{i+1,k} = \theta_{i,k} - \frac{\varphi_i}{\sqrt{Q_{i,k} + \varepsilon}} \nabla f(\theta_{i,k}) \quad (40)$$

De esta manera, mientras el gradiente acumulado sea mayor, menor será la tasa de aprendizaje y mejor control se tendrá sobre esta y viceversa (Liquet et al., 2022).

En la Figura 40 se observa como con Adagrad se consigue alcanzar el punto mínimo marcado de color cian, un comportamiento que no se presentaba con los otros dos algoritmos presentados en la Figura 39.

Figura 40

GD, GD con Momentum y Adagrad



Como todo algoritmo, Adagrad también tiene sus limitaciones. Dado que se acumula los valores de los cuadrados de las gradientes, y en caso este valor acumulado sea elevado en los inicios del entrenamiento, las actualizaciones se harán muy pequeñas provocando, talvez, una temprana interrupción del proceso (Sun et al., 2019).

vi. Root Mean Square Propagation (RMSprop)

Este algoritmo de optimización también adapta la tasa de aprendizaje de manera específica para cada parámetro. Sin embargo, a diferencia de AdaGrad, donde la suma de los cuadrados de las gradientes va aumentando, con RMSprop disminuye (Liquet et al., 2022).

$$Q_{i+1,k} = \vartheta Q_{i,k} + (1-\vartheta)\left(\nabla f\left(\theta_{i,k}\right)\right)^2 \quad (41)$$

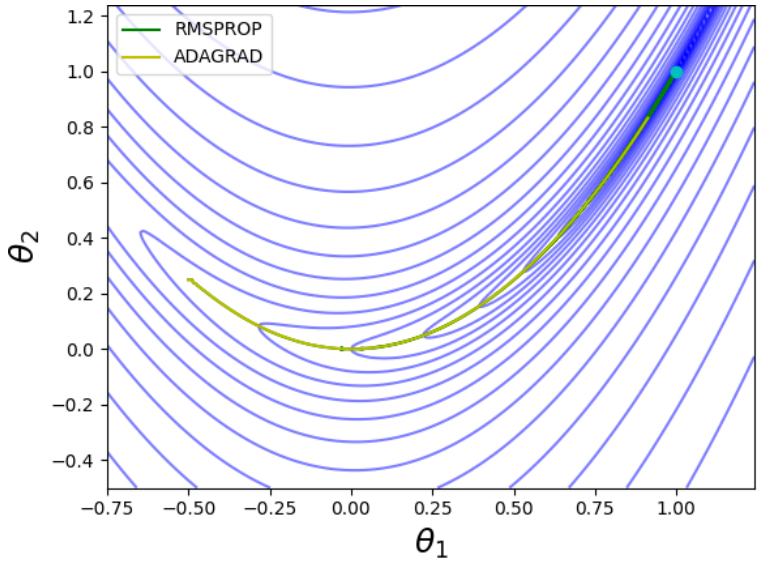
$$\theta_{i+1,k} = \theta_{i,k} - \frac{\varphi_i}{\sqrt{Q_{i+1,k}+\varepsilon}}\nabla f\left(\theta_{i,k}\right) \quad (42)$$

Donde ϑ es un parámetro de decaimiento o factor de amortiguación cuyo valor se encuentra entre 0 y 1.

Al introducir este nuevo factor de decaimiento, RMSprop se vuelve una excelente herramienta para abordar el problema de las disminución de las tasas de aprendizaje que se presentaba con Adagrad (Atici et al., 2022).

Figura 41

RMSprop vs Adagrad



En la Figura 41 se presenta el mismo ejemplo que el de la Figura 40, pero con menos iteraciones. Se observa que con RMSprop la convergencia es más rápida que Adagrad.

vii. Adaptative Moment Estimation (ADAM)

Es la combinación de los métodos anteriores, es decir, que utiliza el descenso de gradiente con momentum y RMSprop, por lo que sus reglas seguirán las reglas de los algoritmos mencionados (Liquet et al., 2022; Sun et al., 2019).

$$v_{i+1,k} = \beta v_{i,k} + (1-\beta)\nabla f(\theta_{i,k}) \quad (43)$$

$$Q_{i+1,k} = \vartheta Q_{i,k} + (1-\vartheta)\left(\nabla f\left(\theta_{i,k}\right)\right)^2 \quad (44)$$

$$\theta_{i+1,k} = \theta_{i,k} - \frac{\varphi_i}{\sqrt{\frac{Q_{i+1,k}}{1-\vartheta_i}+\varepsilon}} \cdot \frac{v_{i+1,k}}{1-\beta_i} \quad (45)$$

3. Data augmentation

Para que un modelo de CNN sea efectivo, es importante entrenarlo con un conjunto de datos grande y balanceado que represente adecuadamente la diversidad y complejidad de las situaciones reales. Sin embargo, en la práctica, esta situación no siempre es posible y puede ser un desafío significativo (Iglesias et al., 2023).

En este contexto, entra en juego la técnica de Data Augmentation, herramienta con la cual se crean ejemplos sintéticos a partir de la manipulación y transformación de los datos existentes. Además, con la generación de estos nuevos datos, el modelo se vuelve más robusto frente a las variaciones que se aplicaron, y que pueden presentarse en una situación real como el ruido ambiental y otras fuentes de interferencia (Madhu & Kumaraswamy, 2019).

En particular, para los audios existen técnicas comunes para aumento de datos como la incorporación de ruido (Noise Injection), cambio en el tono (Pitch Shifting) y la modificación de la velocidad del audio (Time Stretching) entre otras cosas (Mushtaq & Su, 2020). Además, se pueden generar datos sintéticos a partir de los espectrogramas de los audios utilizando herramientas como el enmascaramiento en tiempo y frecuencia.

a) Pitch Shifting

Consiste en cambiar positiva o negativamente el tono de la señal de audio sin variar la duración de esta (Mushtaq & Su, 2020).

b) Noise Injection

Consiste en agregar ruido de manera controlada al audio original. Existen varios tipos de ruido, pero los más comunes son el ruido blanco y el ruido gaussiano (Wei et al., 2020).

c) Time Stretching

La velocidad del audio se aumenta o disminuye sin alterar el tono y la frecuencia del mismo (Wei et al., 2020).

d) Enmascaramiento en tiempo y frecuencia

Como el nombre indica, se coge un tramo continuo en el dominio del tiempo o frecuencia, y se le reemplaza por silencio (Zhou et al., 2022). Esto se aplica aleatoriamente sobre el espectrograma, realizándose las veces que se encuentre necesaria.

Figura 42

Enmascaramiento en tiempo y frecuencia

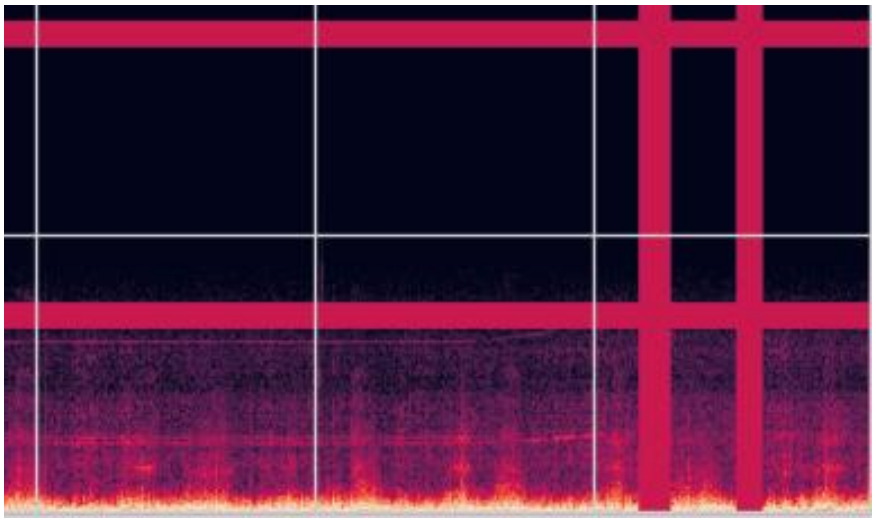


Figura 42 muestra un doble enmascaramiento vertical y un doble enmascaramiento horizontal en el espectrograma de Mel.

4. Transfer Learning

Una técnica muy importante en el campo del aprendizaje automático es el Transfer Learning. Este enfoque se fundamenta en la decisión de aprovechar el conocimiento previamente obtenido en el entrenamiento de un modelo con un conjunto de datos amplio y diverso, y adaptarlos a la tarea de nuestro interés (Tsalera et al., 2021). Esto compensa la limitación del conjunto de datos disponible para el entrenamiento de un modelo, mejorando el rendimiento de esta nueva tarea de clasificación (Becherer et al., 2019).

El Transfer Learning se puede implementar de dos maneras. La primera es con Feature Extraction o extracción de características, que tiene como objetivo aprovechar la arquitectura de la red convolucional y los valores de los pesos de las neuronas previamente entrenadas. En este enfoque, la única parte que se ajusta es la capa de salida, la cual representa la cantidad de categorías que deseamos clasificar en nuestra tarea específica (Lopez-Betancur et al., 2021).

La segunda manera de aplicar el Transfer Learning es conocida como Fine-tuning y difiere de la primera técnica en que busca aprovechar únicamente la arquitectura de la red convolucional, actualizando los valores de los pesos de todas las neuronas durante el entrenamiento (Lopez-Betancur et al., 2021). La elección entre los 2 enfoques dependerá de la relación que tenga nuestro tipo de datos con el de la red pre entrenada, eligiendo el segundo si no existe mucha relación.

Con el transcurrir del tiempo, se han desarrollado modelos de redes neuronales convolucionales (CNN) cada vez más complejos. Estos modelos avanzados se han convertido en puntos de partida ideales para nuestras tareas de clasificación y, a continuación, se explicarán los que serán utilizados en este trabajo.

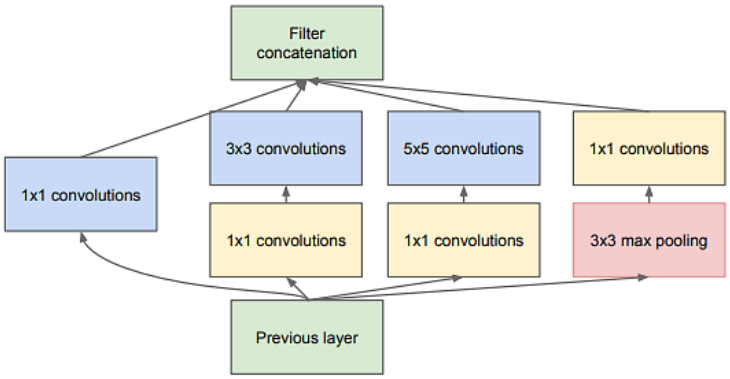
a) Inception

Inception es una arquitectura de CNN desarrollada por un equipo de Google, la cual fue presentada para abordar el desafío de capturar características a diferente escalas en imágenes (Raj, 2018).

En (Szegedy et al., 2015), los autores presentaron la novedosa arquitectura la cual está compuesta por un banco de filtros denominado módulo Inception, que consiste en múltiples ramas que contienen tamaños diferentes de filtros y max pooling, y que se concatenan para ser entrada de la siguiente capa. En la Figura 43 se muestra el módulo Inception para la primera versión de esta arquitectura, llamada GoogleNet.

Figura 43

Módulo Inception para GoogleNet

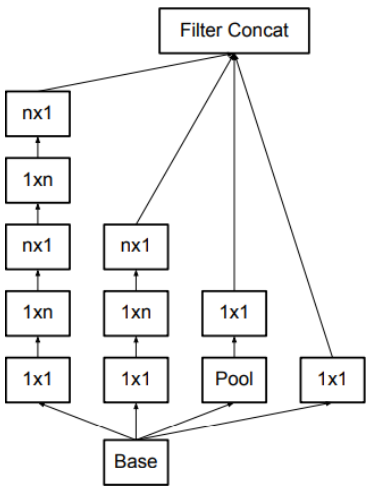


Nota: Fuente (Szegedy et al., 2015)

Esta arquitectura demostró ser muy efectiva en el concurso ImageNet y tuvo diferentes versiones como InceptionV2 e InceptionV3, que presentan una mejora en el módulo Inception, como se muestra en la Figura 44, obteniendo la arquitectura de la Figura 45

Figura 44

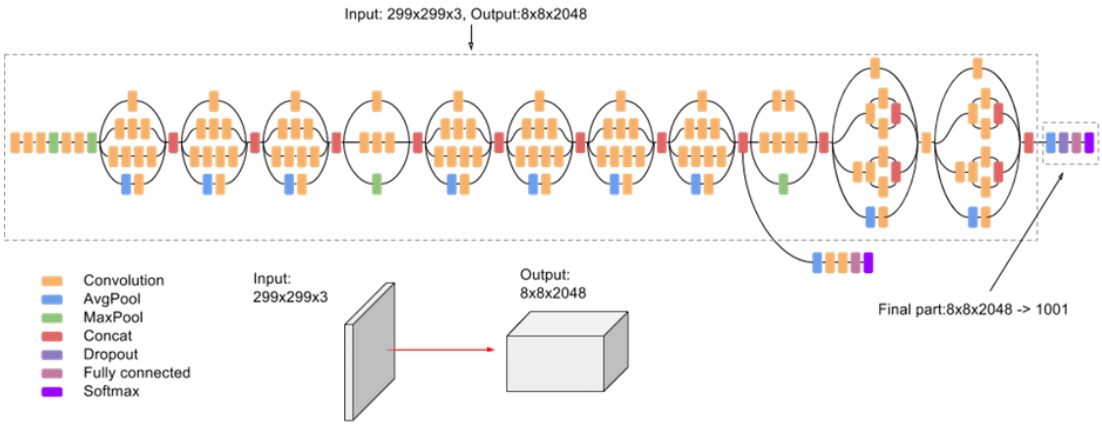
Módulo Inception para las versiones InceptionV2 e V3



Nota: fuente (Szegedy et al., 2016)

Figura 45

Arquitectura de Inception V3



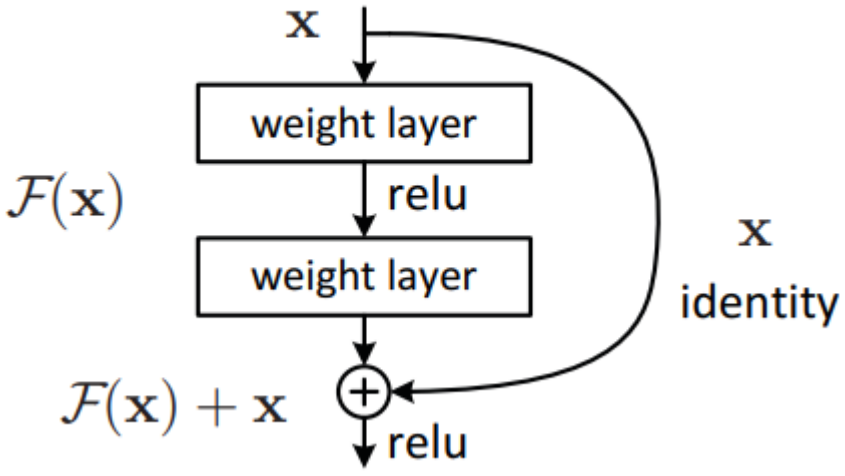
Nota: fuente (Google, 2024)

b) ResNet

ResNet o Residual Networks, es una arquitectura de CNN desarrollada por el grupo de Microsoft Research y fue ganadora del concurso ImageNet en 2015 (Chacón Chamorro, 2023). Esta arquitectura introdujo el concepto de bloque residual (Figura 46), el cual, en vez de realizar conexiones directas entre capas, utilizaba conexiones de salto, evitando de esta manera el desvanecimiento de gradientes en redes muy profundas (He et al., 2016).

Figura 46

Diagrama del bloque residual



Nota: fuente (He et al., 2016)

Mediante estas conexiones de salto, ResNet garantiza que la salida de una capa sea al menos igual de buena que la de la capa anterior o incluso mejor. La inclusión de bloques residuales mejoró significativamente el rendimiento de ResNet en el entrenamiento para tareas de clasificación de imágenes y visión por computadora en comparación con otros modelos existentes en ese momento (He et al., 2016).

2.2 Marco Conceptual

1) Algoritmo

Un algoritmo es un conjunto de instrucciones o procedimientos detallados que deben ser seguidos secuencialmente con el objetivo de completar o resolver una tarea o un problema en particular (Rao, 2023).

2) Overfitting

El overfitting en Deep Learning es un concepto que surge cuando un modelo aprende en exceso los detalles y ruido específicos de los datos de entrenamiento (IBM, 2023). Esto ocurre en situaciones donde el modelo es muy complejo, cuando hay insuficientes datos, o cuando se sobreentrena el modelo (Amazon Web Services [AWS], 2023). Como resultado, el modelo encuentra dificultades para generalizar y aplicarse correctamente a nuevos datos.

3) GPU

Por sus siglas es Unidad de Procesamiento Gráfico y en Deep Learning permite a los usuarios acelerar de manera significativa el entrenamiento de sus modelos (*Deep Learning*, 2015).

4) Dataset

Es un conjunto de datos utilizados para el entrenamiento y validación de un modelo de aprendizaje automático.

5) Matriz de confusión

Es una herramienta esencial que evalúa el desempeño de un modelo respecto a un conjunto de datos de validación (Shultz et al., 2011). Esta matriz proporciona información detallada acerca del número de aciertos en las predicciones realizadas por el modelo en cada categoría de interés.

6) Falso positivo (FP)

Es una clasificación errónea, es decir, cuando el modelo entrenado clasifica una muestra a una clase A cuando en realidad no lo es.

7) Falso negativo (FN)

Es una clasificación errónea, es decir, cuando el modelo entrenado clasifica una muestra como no perteneciente a la clase A cuando en realidad sí lo es.

8) Verdadero positivo (TP)

Es una clasificación acertada, es decir, cuando el modelo entrenado clasifica una muestra a una clase A y acierta.

9) Verdadero negativo (TN)

Es una clasificación acertada, es decir, cuando el modelo entrenado clasifica una muestra como no perteneciente a una clase A y acierta.

10) Sensibilidad

También conocida como Recall, que representa la proporción de verdaderos positivos y matemáticamente se define como:

$$Sens = \frac{TP}{TP + FN} \quad (46)$$

Esta métrica evalúa la habilidad del modelo para reconocer adecuadamente los verdaderos positivos.

11) Especifidad

Representa la proporción de verdaderos negativos y matemáticamente se define como:

$$Spec = \frac{TN}{TN + FP} \quad (47)$$

Esta métrica evalúa la habilidad del modelo para reconocer adecuadamente los verdaderos negativos.

12) Precisión

Esta métrica mide la relación entre las predicciones positivas correctas y el total de estas.

$$Prec = \frac{TP}{TP + FP} \quad (48)$$

13) Accuracy

Esta métrica evalúa el número de aciertos en la predicción que realizó el modelo, y se define por la siguiente ecuación:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (49)$$

14) F1-Score

Esta última métrica utiliza la precisión y la sensibilidad, y su fórmula es:

$$F1_{sc} = 2 * \frac{Prec * Sens}{Prec + Sens} \quad (50)$$

2.3 Marco filosófico

Este trabajo se enmarca en el humanismo ya que la detección de la EPOC desde los sonidos pulmonares utilizando algoritmos de clasificación de Deep Learning representa una aplicación tecnológica con un impacto de modo significante en la vida de las personas, buscando contribuir a la salud y bienestar de estas.

2.3.1 Humanismo y tecnología

El humanismo se fundamenta en la valoración del ser humano y su bienestar en el contexto del progreso tecnológico (Edis, 2016). En el ámbito de la detección de la EPOC, así como en la aplicación de algoritmos de clasificación de Deep Learning, se persigue el objetivo de proporcionar una mejora en el bienestar y la salud de las personas, priorizando siempre la utilización responsable de la tecnología.

2.3.2 Ética y responsabilidad

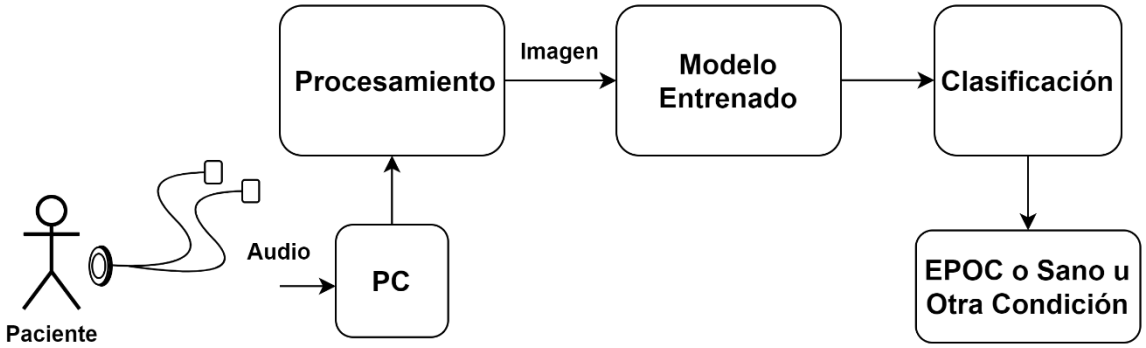
En la aplicación de algoritmos de clasificación de Deep Learning, se plantea la importancia de abordar aspectos éticos y de responsabilidad (Russell, 2021). Es esencial tener en cuenta las implicaciones potenciales en la sociedad, salvaguardar la privacidad de los datos y garantizar la transparencia en el uso de la tecnología, con el objetivo de generar confianza y brindar beneficios tanto a los pacientes como a los especialistas de la salud, buscando un equilibrio tecnológico y la protección de los derechos y valores humanos.

Capítulo III. Desarrollo del trabajo de investigación

En esta sección, la metodología aplicada en este trabajo sobre la implementación de un sistema inteligente mediante algoritmos de Deep Learning para la detección de EPOC a través de sonidos pulmonares se exhibe. La metodología se diseñó cuidadosamente dividiendo el trabajo en etapas, mostrada en la Figura 47, lo cual permitió abordar el tema de una manera estructurada y ordenada, garantizando así la eficacia y coherencia de todo el proceso.

Figura 47

Etapas del Sistema inteligente basado en Deep Learning para la detección de EPOC

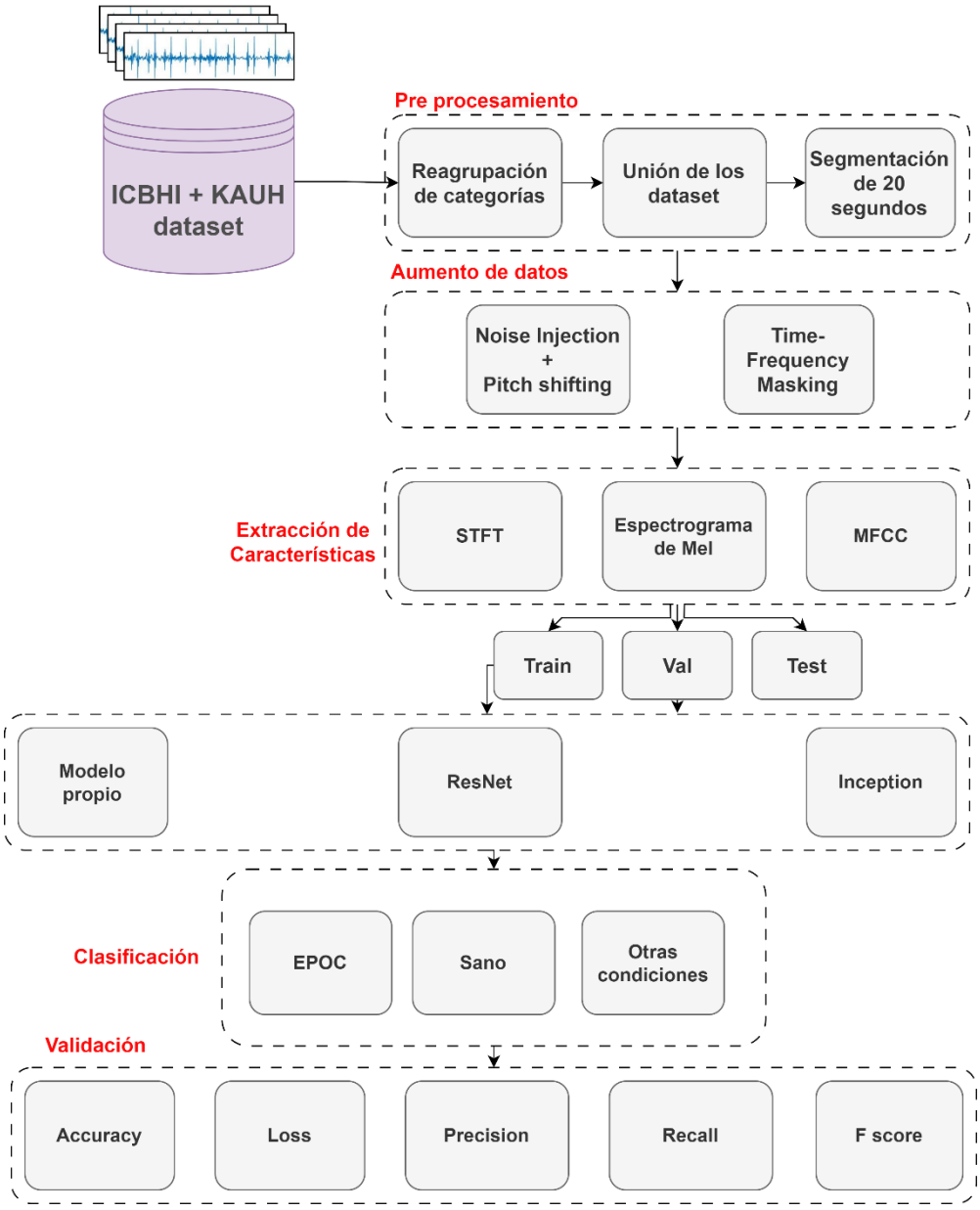


Los audios pulmonares recopilados de un paciente, se almacenarán en una computadora para que el algoritmo de Deep Learning se encargue de clasificarlos en una de las 3 categorías.

El proceso de entrenamiento del algoritmo también tuvo un conjunto de etapas que facilitaron el progreso del trabajo de investigación, y este esquema está detallado en la Figura 48.

Figura 48

Esquema de principio para entrenamiento de los algoritmos a usar en el sistema inteligente basado en Deep Learning.



El inicio de la investigación se centra en la recopilación y el estudio de los datos de sonidos pulmonares o respiratorios de pacientes, abarcando tanto aquellos diagnosticados con EPOC como aquellos que no.

En la siguiente etapa, se realiza el preprocesamiento de los datos para prepararlos de manera óptima para el entrenamiento de los modelos. En este paso, se unen los dataset escogidos y se reagrupan los audios en tres categorías distintas. Además, se estandariza

la duración de los audios a 20 segundos y se aplica un filtrado para eliminar ruido no deseado.

Posteriormente, se aplican métodos de aumento de datos con el fin de balancear el dataset. Una vez preparados los datos, se extraen las características de los audios utilizando sus representaciones espectrales.

En la etapa de diseño del modelo, se plantea y diseña una arquitectura de CNN con el objetivo de entrenarla para la clasificación de los sonidos pulmonares. Además, se emplean dos redes pre entrenadas para clasificar estos sonidos respiratorios.

Finalmente, se analizan y evalúan los modelos utilizando las métricas que se describen en el marco conceptual, permitiendo una comprensión precisa del rendimiento del sistema.

3.1 Recopilación y análisis de dataset de sonidos pulmonares

Se utilizaron dos bases de datos para este trabajo de investigación. Una de estas fue creada por dos grupos de investigadores de Grecia y Portugal, respectivamente, para el desafío en la Conferencia Internacional de Informática Biomédica y de Salud (ICBHI) organizada en 2017 (Rocha et al., 2018).

Este dataset incluye 920 grabaciones de sonidos pulmonares adquiridas de 126 pacientes que presentaban diferentes complicaciones respiratorias, entre ellas la EPOC. Los dispositivos para obtener estas grabaciones fueron:

a) “AKG C417L Microphone” representado en la base de datos como AKGV417L

b) “3M Littmann Classic II SE Stethoscope” representado como LittC2SE

c) “3M Litmmann 3200 Electronic Stethoscope” representado como Litt3200

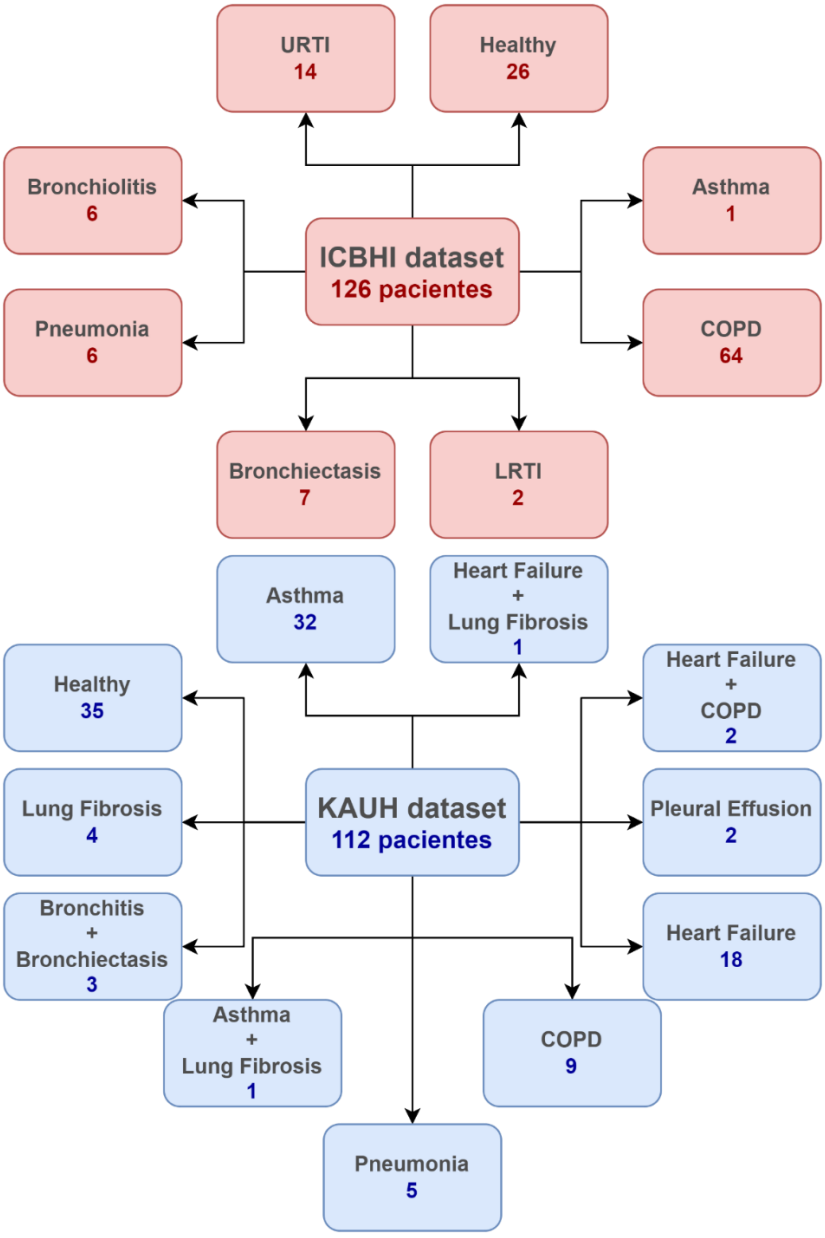
d) “WelchAllyn Meditron Master Elite Electronic Stethoscope” representado como Meditron

La segunda base de datos fue creada por un grupo de investigadores del King Abdullah University Hospital (KAUH) (M. Fraiwan et al., 2021), la cual presenta 336 audios respiratorios adquiridos de 112 pacientes utilizando el estetoscopio electrónico Litt3200.

En la Figura 49 se proporciona más información sobre las bases datos mencionadas.

Figura 49

Información de las bases de datos ICBHI y KAUH



3.2 Preprocesamiento de los datos

3.2.1 Unión de los datasets y reagrupación de categorías

Dado que el objetivo principal en este trabajo es la detección de la EPOC, se ha estructurado la clasificación en tres categorías fundamentales: EPOC, sano y otras

condiciones, como muestra la Figura 50 y Figura 51. La categoría "sano" desempeña un papel crucial en este estudio al servir como punto de referencia para la normalidad en los sonidos pulmonares.

Por otra parte, la categoría "Otras condiciones" engloba otras afecciones respiratorias que no son EPOC. La inclusión de esta categoría es vital para establecer un marco comparativo que permita al sistema inteligente discernir entre los patrones sonoros específicos de la EPOC y aquellos asociados con otras afecciones respiratorias. Este enfoque comparativo es esencial para el desarrollo de un sistema inteligente preciso y confiable.

Figura 50

Número de pacientes por categoría

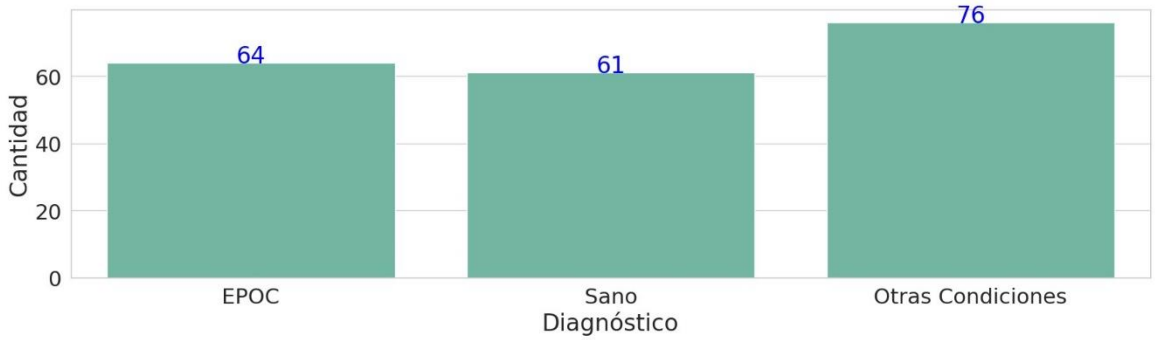
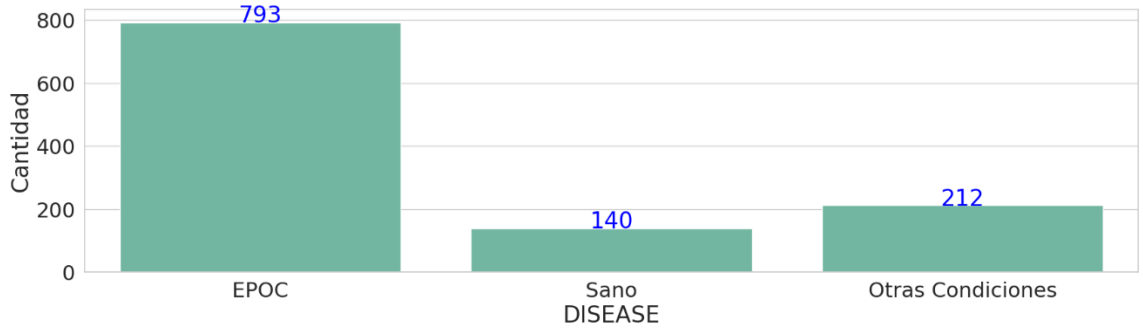


Figura 51

Número de audios respiratorios por categoría



3.2.2 Segmentación

El conjunto de datos contiene audios con diversos tiempos de grabación, desde 4 segundos hasta periodos más extensos. Con el fin de estandarizar estos periodos y simplificar el proceso de entrenamiento de los modelos de CNN, se realizó una estandarización uniforme a 20 segundos, tiempo en el que se llevan a cabo, en promedio,

unas 6 respiraciones. El margen de tiempo mencionado ofrece una ventana adecuada para capturar anomalías respiratorias en los pacientes.

Para dicha estandarización, se siguieron los siguientes puntos:

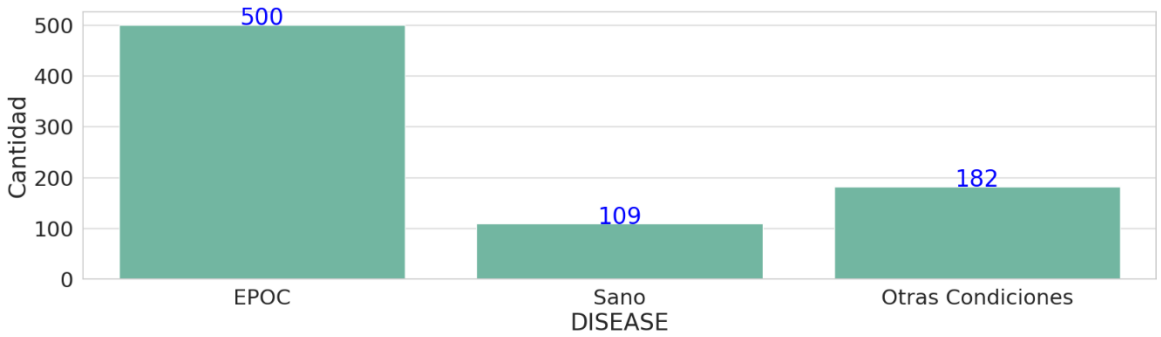
A. Se descartaron los audios con una duración menor a 15 segundos.

B. A los audios con una duración mayor a 15 segundos, pero menor a 20 segundos, se les redujo su velocidad para ajustarlos a la duración estándar de 20 segundos. Sin embargo, es importante destacar que estos audios no serán utilizados para el aumento de datos.

C. En los audios con una duración mayor a 20 segundos de duración se realizó una segmentación desplazada. Esto implicó truncar el audio en intervalos de 20 segundos, con un desplazamiento de 5 segundos. Por ejemplo, para uno de los audios de duración de 37 segundos, se obtuvieron cuatro segmentaciones: de 0 a 20 segundos, de 5 a 25 segundos, de 10 a 30 segundos y de 15 a 35 segundos.

Figura 52

Número de audios respiratorios segmentados por categoría



3.2.3 Filtrado

De acuerdo con la revisión realizada en el marco teórico, los sonidos pulmonares suelen manifestarse en el rango de frecuencias entre 50Hz y 2500Hz. En concordancia con esta información, se ha implementado un filtro Butterworth de orden 6 que actuó como un filtro pasa banda, específicamente ajustado para operar dentro del rango de frecuencias mencionado. La elección de este tipo de filtro se debe a su respuesta plana en la banda de

paso, minimizando las variaciones en la amplitud de las señales de audio y manteniendo su fidelidad.

3.3 Aumento de datos

En la Figura 52 se visualiza un desbalance marcado en el conjunto de datos, donde predomina la categoría EPOC con 500 muestras, mientras que las otras dos categorías cuentan con una cantidad inferior. Con el propósito de equilibrar este conjunto de datos, se utilizaron métodos de aumento de datos, y sirvieron para observar con cual método se obtuvo un mejor desempeño durante el proceso de entrenamiento de los modelos de CNN.

3.3.1 Noise Injection + Pitch Shifting

Se ha implementado una función que aplica dos técnicas de aumento de datos: inyección de ruido y cambio de tono. En la Figura 53 se presenta el código en Python utilizando la librería “**librosa**”:

Figura 53

Extracto del código de aumento de datos con Noise Injection y Pitch Shifting

```
def data_aug_1(f, k, _path):
  audio, sr = librosa.load(f)
  if(k == 0):
    noise_level              = 0.005
    audio_noisy_005          = audio + noise_level * np.random.randn(len(audio))
    sf.write(_path+str(aux)+'.wav', audio_noisy_005, sr)
  elif(k == 1):
    noise_level              = 0.001
    audio_noisy_001          = audio + noise_level * np.random.randn(len(audio))
    sf.write(_path+str(aux)+'.wav', audio_noisy_001, sr)
  elif(k == 2):
    n_steps                  = 2
    audio_pitch_shifted_up1  = librosa.effects.pitch_shift(y=audio, sr=sr, n_steps=n_steps)
    sf.write(_path+str(aux)+'.wav', audio_pitch_shifted_up1, sr)
```

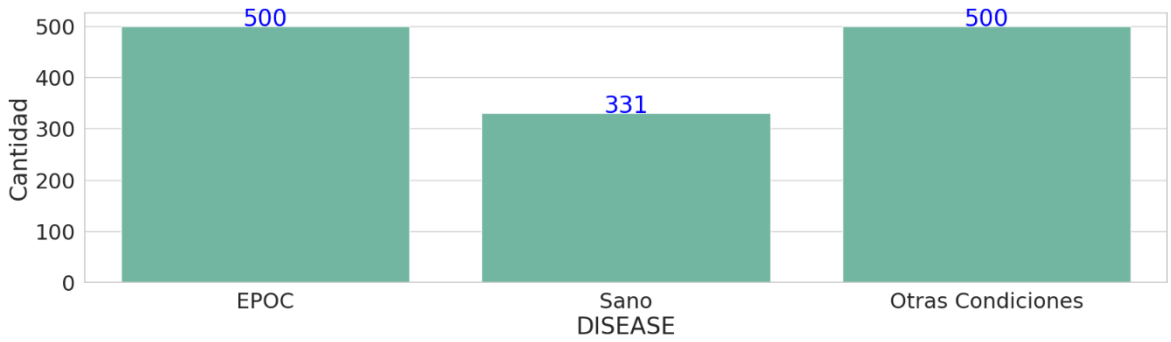
Esta función toma como entrada el archivo de audio **f**, un indicador **k** que especifica el tipo de técnica y un parámetro **aux** para nombrar los audios resultantes.

Este proceso de aumento de datos se ha aplicado de manera iterativa para cada audio dentro de las categorías "Sano" y "Otras condiciones". Esta iteración continúa hasta

que las categorías alcancen las 500 muestras o no existan más audios disponibles para aumentar.

Figura 54

Número de audios respiratorios luego de la primera técnica de aumento de datos



3.3.2 Enmascaramiento en frecuencia y tiempo

En un principio, se utilizó la librería “**Tensorflow_io”** para llevar a cabo estos 2 métodos. Esta librería permite realizar un enmascaramiento de ancho aleatorio desde 0 hasta un parámetro f, que se estableció en 20. Sin embargo, en algunas ocasiones, este proceso genera un enmascaramiento de ancho igual a 0, en otras palabras, no realiza ningún cambio en la imagen.

En este sentido, se modificaron las funciones de esta librería para permitir un ancho mínimo y estas modificaciones se muestran en el extracto de código presentado en la Figura 55.

Con estas modificaciones, se resolvió el problema de obtener un enmascaramiento de ancho igual a 0. Posteriormente, se utilizaron estas funciones modificadas para aplicar tres enmascaramientos en cada dominio con un ancho aleatorio cada uno.

Figura 55

Extracto del código para el aumento de datos con enmascaramiento en tiempo y frecuencia

```
def frequency_masking(input, param = [0, 1], name=None):
    """
    Apply masking to a spectrogram in the freq domain.

    Args:
      input: An audio spectogram.
      param: Parameter of freq masking.
      name: A name for the operation (optional).
    Returns:
      A tensor of spectrogram.
    """
    input = tf.convert_to_tensor(input)
    # TODO: Support audio with channel > 1.
    freq_max = tf.shape(input)[1]
    f = tf.random.uniform(shape=(), minval=param[0], maxval=param[1], dtype=tf.dtypes.int32)
    f0 = tf.random.uniform(
        shape=(), minval=0, maxval=freq_max - f, dtype=tf.dtypes.int32
    )
    indices = tf.reshape(tf.range(freq_max), (1, -1))
    condition = tf.math.logical_and(
        tf.math.greater_equal(indices, f0), tf.math.less(indices, f0 + f)
    )
    return tf.where(condition, tf.cast(0, input.dtype), input)

def time_masking(input, param = [0, 1], name=None):
    """
    Apply masking to a spectrogram in the time domain.

    Args:
      input: An audio spectogram.
      param: Parameter of time masking.
      name: A name for the operation (optional).
    Returns:
      A tensor of spectrogram.
    """
    input = tf.convert_to_tensor(input)
    # TODO: Support audio with channel > 1.
    time_max = tf.shape(input)[0]
    t = tf.random.uniform(shape=(), minval=param[0], maxval=param[1], dtype=tf.dtypes.int32)
    t0 = tf.random.uniform(
        shape=(), minval=0, maxval=time_max - t, dtype=tf.dtypes.int32
    )
    indices = tf.reshape(tf.range(time_max), (-1, 1))
    condition = tf.math.logical_and(
        tf.math.greater_equal(indices, t0), tf.math.less(indices, t0 + t)
    )
    return tf.where(condition, tf.cast(0, input.dtype), input)

def save_data_aug2(_path_, image, N, aux):
    matrix_path = os.path.join(_path_, f"{str(N+aux)}_mfcc.npy")
    image = np.expand_dims(image, axis=0)
    print(image.shape)
    np.save(matrix_path, image)
```

Con esta segunda técnica de aumento de datos se obtuvo la misma distribución de muestras por categoría que en la Figura 54.

3.4 Extracción de características

La transformación de las señales de audio a representaciones espectrales es esencial para iniciar con éxito el proceso de entrenamiento de los modelos de Deep Learning. En este contexto, se procedió a extraer características clave de los audios mediante el empleo de dos técnicas: espectrograma de Mel y los MFCC.

La elección del tamaño de estas matrices de características es de suma importancia. Un tamaño demasiado grande, puede volver pesados y lentos a los procesos subsiguientes, mientras que un tamaño muy pequeño podría no proporcionar suficientes características relevantes para el entrenamiento de los modelos de CNN. Por ello, fue importante hallar un equilibrio apropiado en el tamaño de estas matrices para garantizar un óptimo rendimiento en las etapas posteriores.

3.4.1 Espectrograma de Mel

En el caso del espectrograma de Mel, se ajustaron tres parámetros: la longitud de la ventana (Frame_size = 2048), el desplazamiento de la ventana (Hop_size = 512) y la cantidad de coeficientes de mel (nmels=40). A medida que se aumentaba este último valor, también lo hacía el tamaño de la matriz de características. Así, en un ajuste final, el tamaño de cada matriz de características de Mel resultó en (40, 626).

3.4.2 MFCC

En la extracción de los MFCC, se tuvo en cuenta el número de coeficientes MFCC (nmfcc = 40). El tamaño final de cada matriz de características MFCC resultó en (40, 626).

La obtención de estas características se consiguió utilizando la librería librosa y a través de las funciones mostradas en la Figura 56.

Figura 56

Extracto del código para el preprocesamiento de los audios

```
def get_MEL(self, audio):
    audio_mel   = librosa.feature.melspectrogram(y = audio, sr = self.sample_rate,
                                                 n_fft = self.FRAME_SIZE, hop_length = self.HOP_SIZE,
                                                 n_mels = self.N_mels)
    Y_audio     = librosa.power_to_db(audio_mel)
    Y_audio_n   = self.Normalization_MEL(Y_audio)

    return Y_audio_n

def get_MFCC(self, audio):

    audio_mfcc        = librosa.feature.mfcc(y = audio, sr = self.sample_rate, n_mfcc = self.N_mfcc)
    Y_audio_n         = self.Normalization_MFCC(audio_mfcc)

    return Y_audio_n
```

3.5 Entrenamiento y validación de los modelos CNN

3.5.1 Modelo propio

Para el modelo propio, sSe optó por un modelo CNN de doble rama para abordar de manera efectiva la complejidad inherente de los datos de entrada. Esta arquitectura, tiene la capacidad de procesar y extraer características de forma independiente en dos caminos paralelos. Las ventajas de este tipo de redes se exploraron en detalle en la parte teórica y la arquitectura en cuestión se encuentra representada visualmente en la Figura 57 y Figura 58.

Figura 57

Arquitectura diseñada del modelo propio

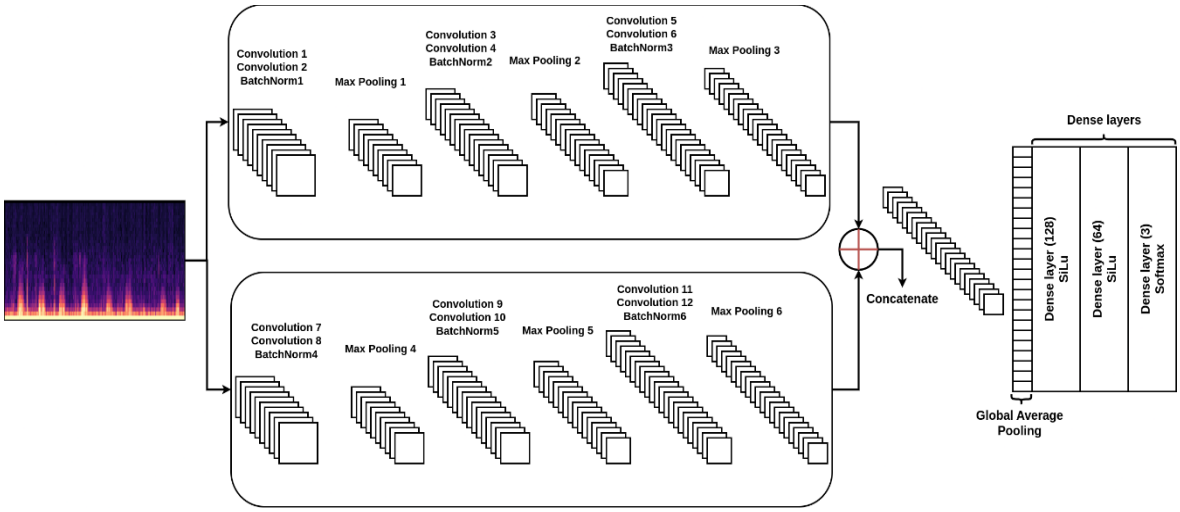
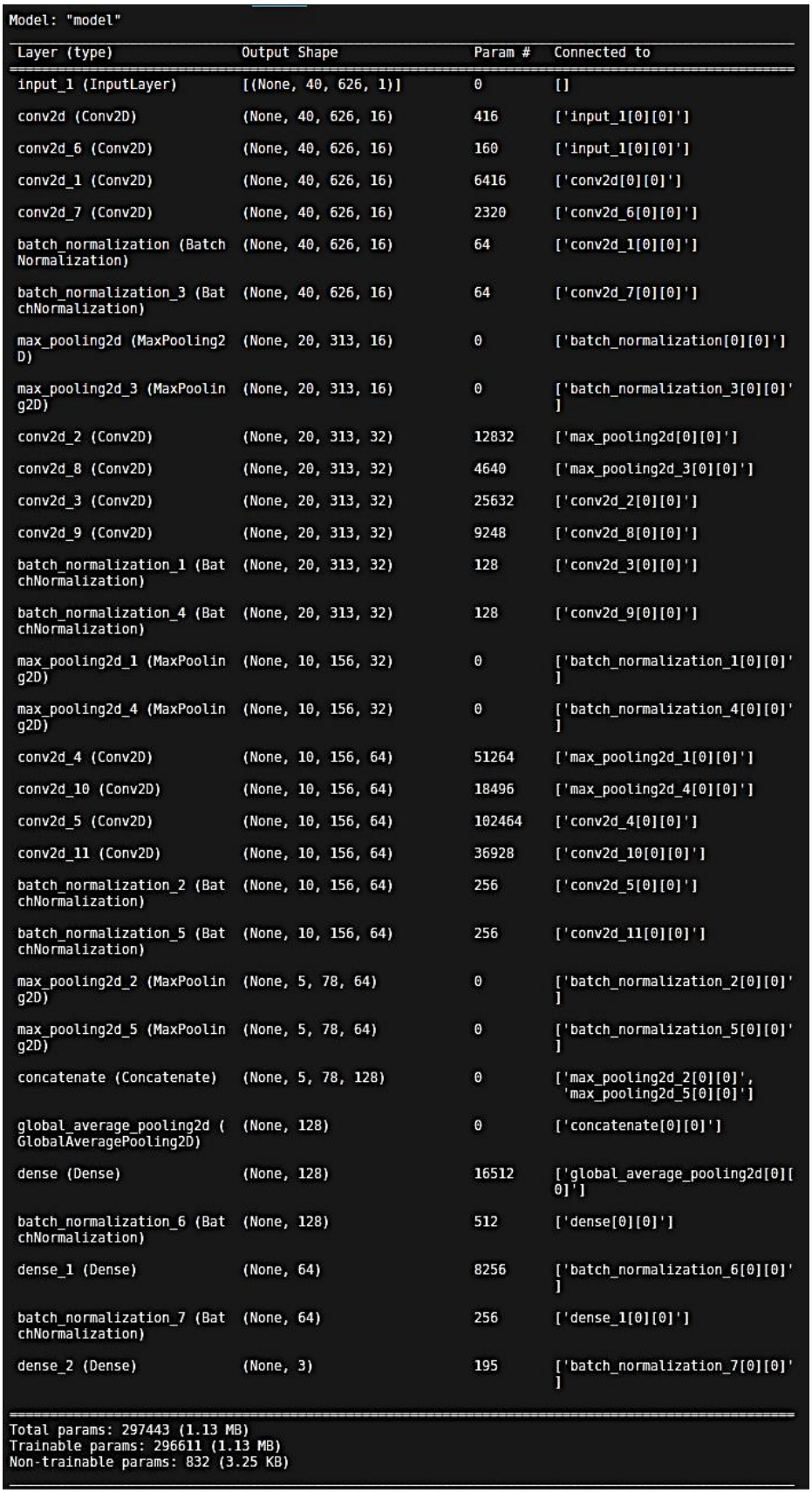


Figura 58

Arquitectura del modelo propio y cantidad de parámetros



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================
 input_1 (InputLayer)        [(None, 40, 626, 1)]      0         []

 conv2d (Conv2D)             (None, 40, 626, 16)       416       ['input_1[0][0]']

 conv2d_6 (Conv2D)           (None, 40, 626, 16)       160       ['input_1[0][0]']

 conv2d_1 (Conv2D)           (None, 40, 626, 16)       6416      ['conv2d[0][0]']

 conv2d_7 (Conv2D)           (None, 40, 626, 16)       2320      ['conv2d_6[0][0]']

 batch_normalization (Batch  (None, 40, 626, 16)       64        ['conv2d_1[0][0]']
 Normalization)

 batch_normalization_3 (Bat  (None, 40, 626, 16)       64        ['conv2d_7[0][0]']
 chNormalization)

 max_pooling2d (MaxPooling2  (None, 20, 313, 16)       0         ['batch_normalization[0][0]']
 D)

 max_pooling2d_3 (MaxPoolin  (None, 20, 313, 16)       0         ['batch_normalization_3[0][0]'
 g2D)                                                            ]

 conv2d_2 (Conv2D)           (None, 20, 313, 32)       12832     ['max_pooling2d[0][0]']

 conv2d_8 (Conv2D)           (None, 20, 313, 32)       4640      ['max_pooling2d_3[0][0]']

 conv2d_3 (Conv2D)           (None, 20, 313, 32)       25632     ['conv2d_2[0][0]']

 conv2d_9 (Conv2D)           (None, 20, 313, 32)       9248      ['conv2d_8[0][0]']

 batch_normalization_1 (Bat  (None, 20, 313, 32)       128       ['conv2d_3[0][0]']
 chNormalization)

 batch_normalization_4 (Bat  (None, 20, 313, 32)       128       ['conv2d_9[0][0]']
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 10, 156, 32)       0         ['batch_normalization_1[0][0]'
 g2D)                                                            ]

 max_pooling2d_4 (MaxPoolin  (None, 10, 156, 32)       0         ['batch_normalization_4[0][0]'
 g2D)                                                            ]

 conv2d_4 (Conv2D)           (None, 10, 156, 64)       51264     ['max_pooling2d_1[0][0]']

 conv2d_10 (Conv2D)          (None, 10, 156, 64)       18496     ['max_pooling2d_4[0][0]']

 conv2d_5 (Conv2D)           (None, 10, 156, 64)       102464    ['conv2d_4[0][0]']

 conv2d_11 (Conv2D)          (None, 10, 156, 64)       36928     ['conv2d_10[0][0]']

 batch_normalization_2 (Bat  (None, 10, 156, 64)       256       ['conv2d_5[0][0]']
 chNormalization)

 batch_normalization_5 (Bat  (None, 10, 156, 64)       256       ['conv2d_11[0][0]']
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 5, 78, 64)         0         ['batch_normalization_2[0][0]'
 g2D)                                                            ]

 max_pooling2d_5 (MaxPoolin  (None, 5, 78, 64)         0         ['batch_normalization_5[0][0]'
 g2D)                                                            ]

 concatenate (Concatenate)   (None, 5, 78, 128)        0         ['max_pooling2d_2[0][0]',
                                                                  'max_pooling2d_5[0][0]']

 global_average_pooling2d (  (None, 128)               0         ['concatenate[0][0]']
 GlobalAveragePooling2D)

 dense (Dense)               (None, 128)               16512     ['global_average_pooling2d[0][
                                                                 0]']

 batch_normalization_6 (Bat  (None, 128)               512       ['dense[0][0]']
 chNormalization)

 dense_1 (Dense)             (None, 64)                8256      ['batch_normalization_6[0][0]'
                                                                 ]

 batch_normalization_7 (Bat  (None, 64)                256       ['dense_1[0][0]']
 chNormalization)

 dense_2 (Dense)             (None, 3)                 195       ['batch_normalization_7[0][0]'
                                                                 ]

=================================================================
Total params: 297443 (1.13 MB)
Trainable params: 296611 (1.13 MB)
Non-trainable params: 832 (3.25 KB)
_________________________________________________________________
```

En la primera rama, filtros de tamaño 3x3 fueron empleados con la intención de extraer características a una escala pequeña. Por otro lado, el tamaño de los filtros de la segunda rama fueron de tamaño 5x5, permitiendo la extracción de características a una escala más grande.

Además, la función de pérdida SiLu fue utilizada como alternativa al ReLu, ya que, durante la pruebas y ajustes experimentales, se observaron resultados ligeramente superiores con esta función.

El modelo se entrenó probando con cuatro optimizadores durante 150 épocas y utilizando la clase “ReduceLROnPlateau”, de Keras, para reducir el learning rate cuando no se obtengan mejoras en el entrenamiento.

Figura 59

Función de error del modelo propio entrenado con el primer aumento de datos

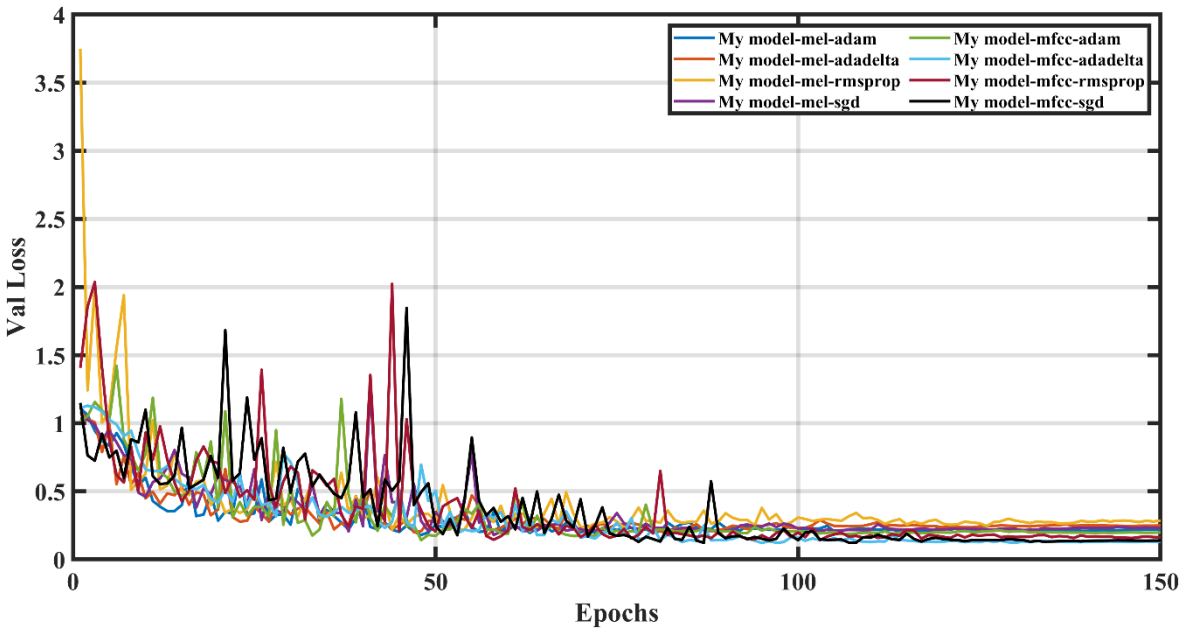


Figura 60

Métrica de exactitud del modelo propio entrenado con el primer aumento de datos

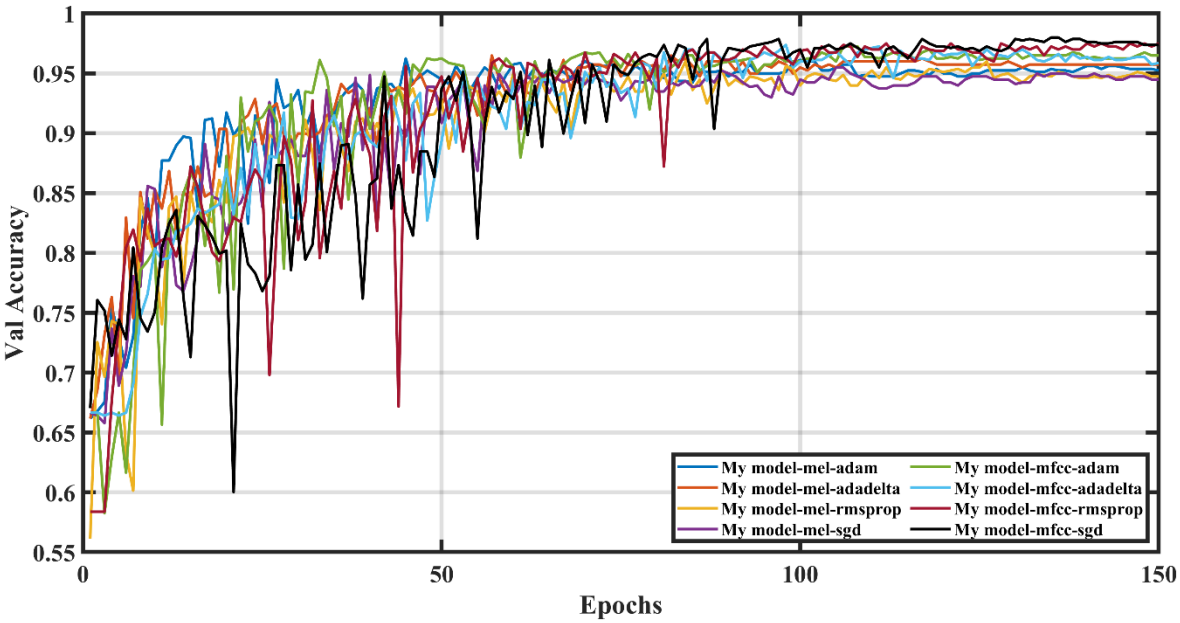


Figura 61

Métrica de precisión del modelo propio entrenado con el primer aumento de datos

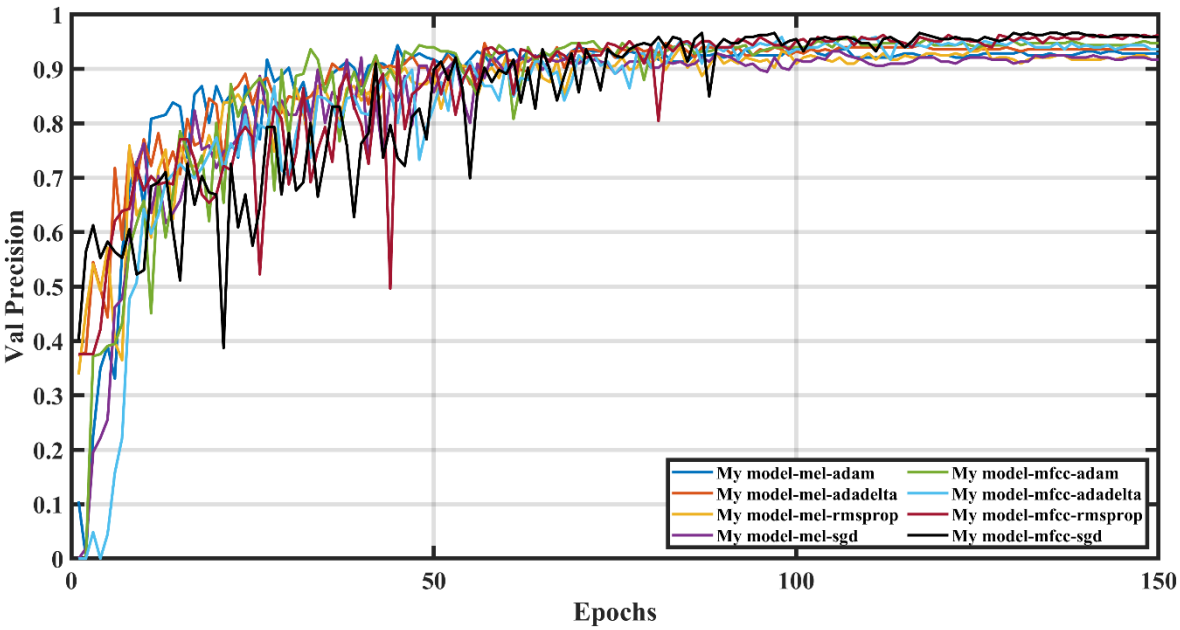


Figura 62

Métrica de sensibilidad del modelo propio entrenado con el primer aumento de datos

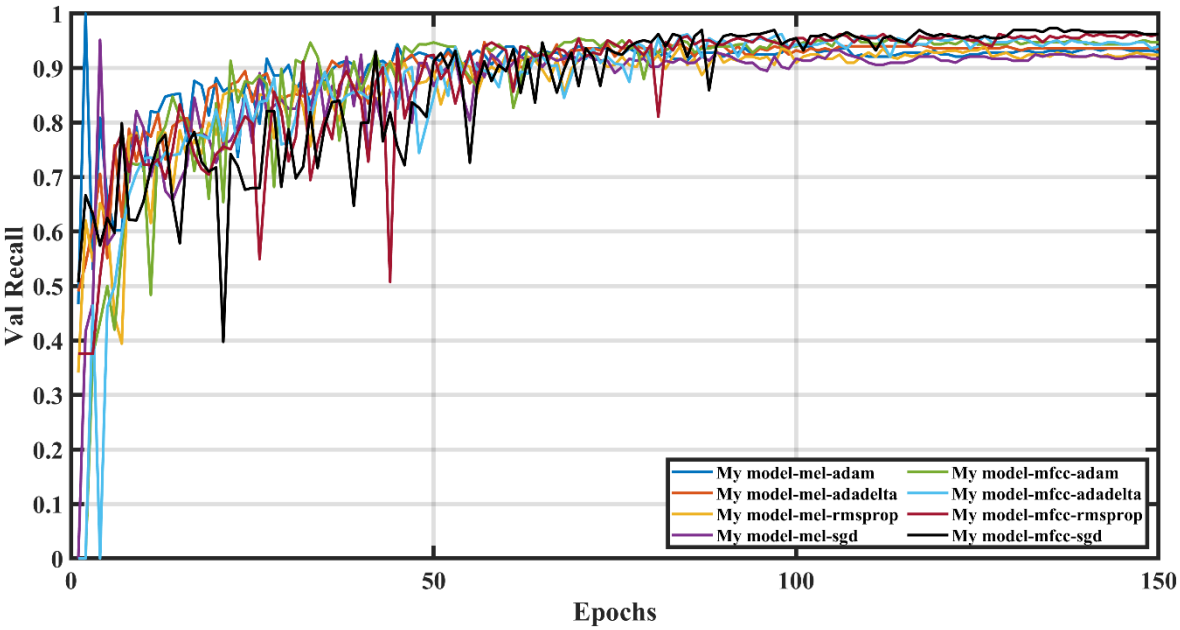


Tabla 3

Resumen de resultados del entrenamiento del modelo propio con el primer aumento de datos

Característica	Optimizador	Datos	Accuracy	Loss	Precision	Recall	F score
MEL	Adam	Train	0.9975	0.0154	0.9962	0.9962	0.9958
		Val	**0.9624**	**0.2032**	**0.9436**	**0.9436**	**0.9388**
	Rmsprop	Train	0.9983	0.0098	0.9975	0.9975	0.9978
		Val	**0.9599**	**0.2690**	**0.9398**	**0.9398**	**0.9372**
	Adadelta	Train	1.0000	0.0032	1.0000	1.0000	1.0000
		Val	**0.9649**	**0.1715**	**0.9474**	**0.9474**	**0.9451**
	Sgd	Train	1.0000	0.0003	1.0000	1.0000	1.0000
		Val	**0.9549**	**0.2133**	**0.9323**	**0.9323**	**0.9293**
MFCC	Adam	Train	1.0000	0.0001	1.0000	1.0000	1.0000
		Val	**0.9724**	**0.1913**	**0.9586**	**0.9586**	**0.9581**
	Rmsprop	Train	1.0000	0.0018	1.0000	1.0000	1.0000
		Val	**0.9749**	**0.1656**	**0.9624**	**0.9624**	**0.9642**
	Adadelta	Train	0.9992	0.0132	0.9987	0.9987	0.9989
		Val	**0.9737**	**0.1284**	**0.9623**	**0.9586**	**0.9600**
	Sgd	Train	1.0000	0.0007	1.0000	1.0000	1.0000
		Val	**0.9787**	**0.1230**	**0.9698**	**0.9662**	**0.9647**

Figura 63

Función de error del modelo propio entrenado con el segundo aumento de datos

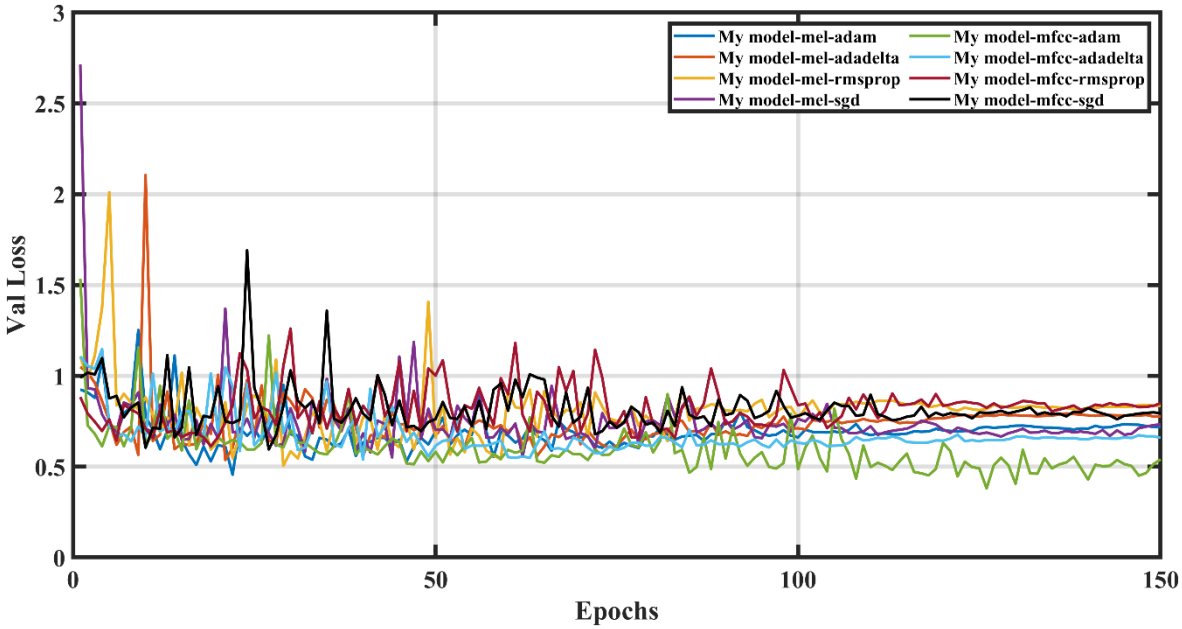


Figura 64

Métrica de exactitud del modelo propio entrenado con el segundo aumento de datos

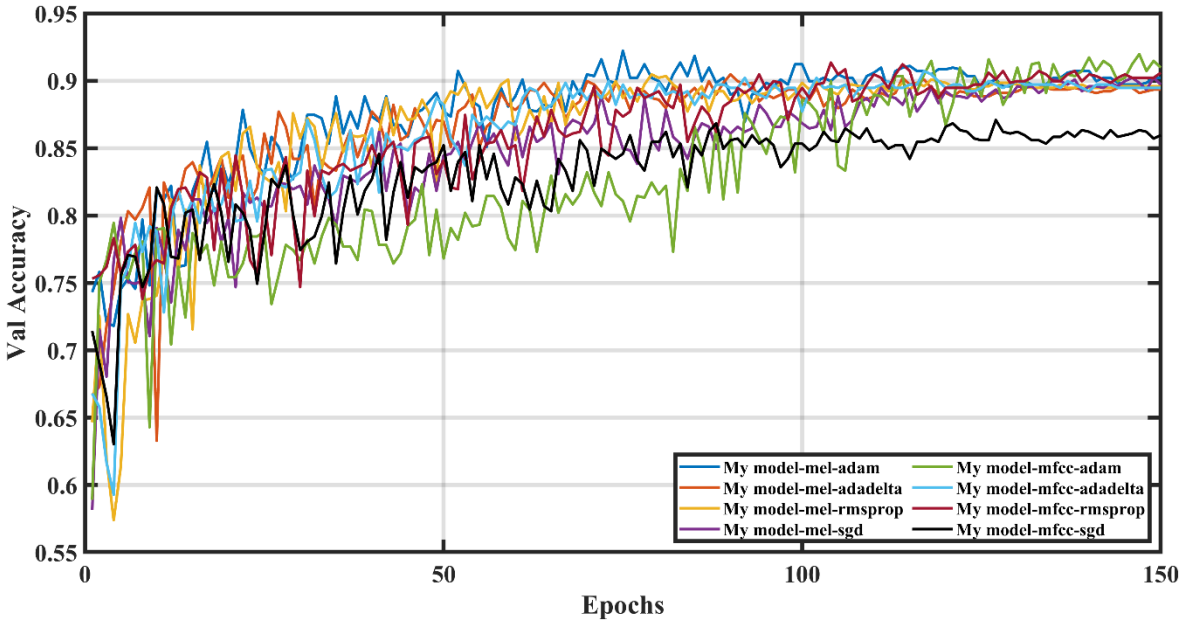


Figura 65

Métrica de precisión del modelo propio entrenado con el segundo aumento de datos

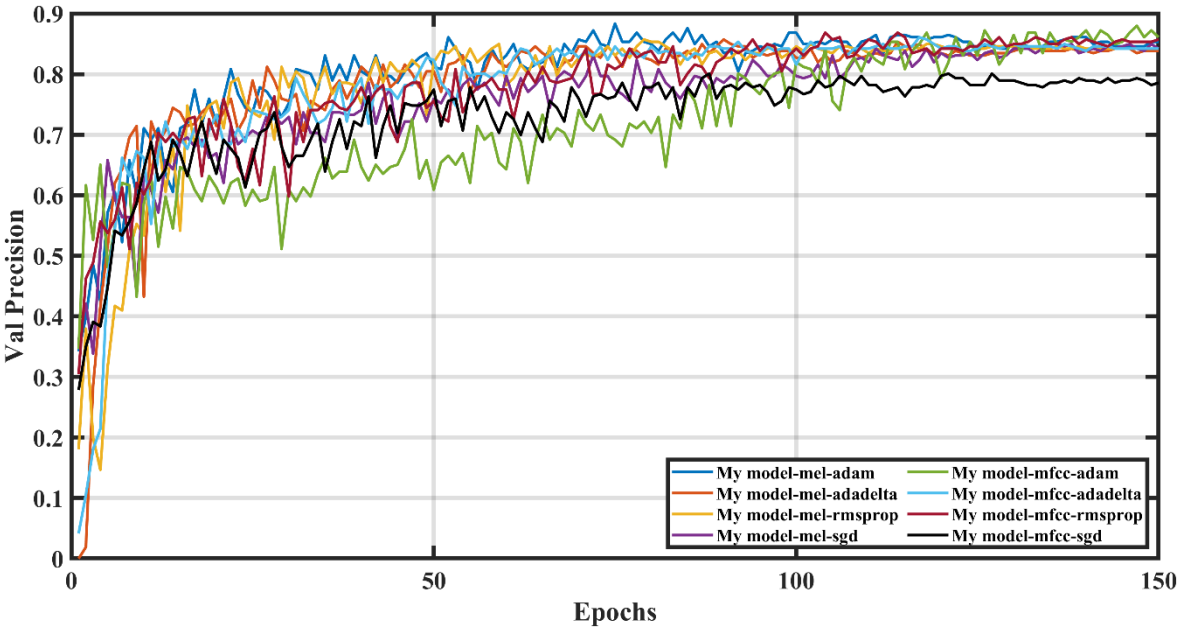


Figura 66

Métr*ica de sensibilidad del modelo propio entrenado con el segundo aumento de datos*

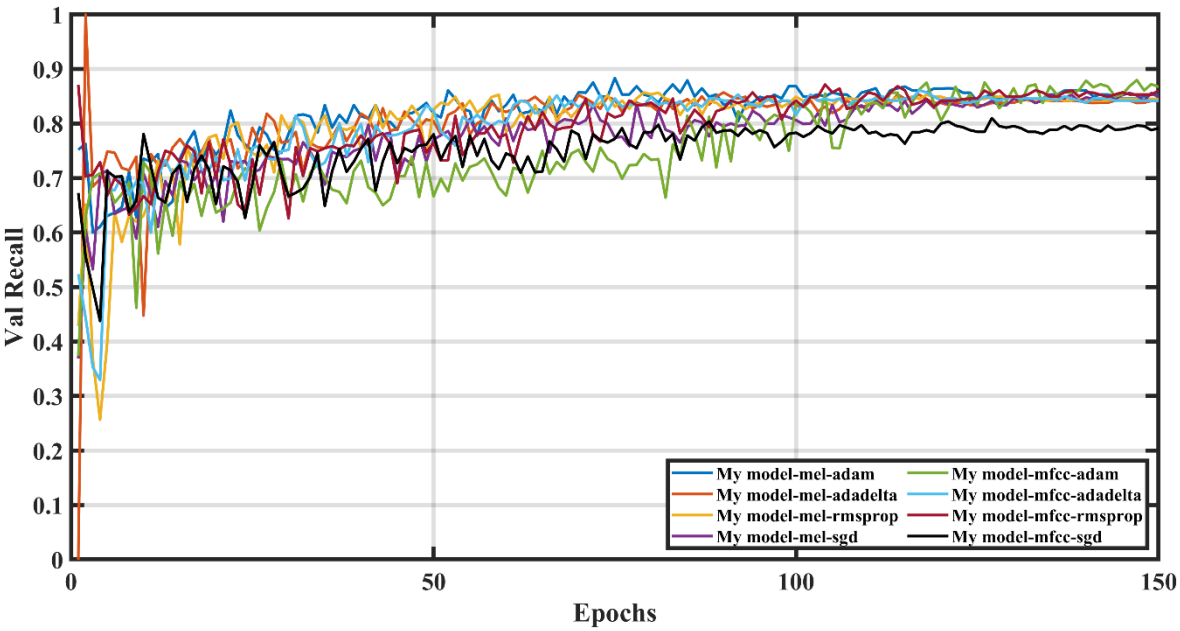


Tabla 4

Resumen de resultados del entrenamiento del modelo propio con el segundo aumento de datos

Característica	Optimizador	Datos	Accuracy	Loss	Precision	Recall	F score
MEL	Adam	Train	1.0000	0.0014	1.0000	1.0000	1.0000
		Val	**0.9223**	**0.6051**	**0.8835**	**0.8835**	**0.8699**
	Rmsprop	Train	1.0000	2.4e-5	1.0000	1.0000	1.0000
		Val	**0.9048**	**0.7815**	**0.8571**	**0.8571**	**0.8412**
	Adadelta	Train	1.0000	0.0001	1.0000	1.0000	1.0000
		Val	**0.9048**	**0.6946**	**0.8571**	**0.8571**	**0.8403**
	Sgd	Train	0.9992	0.0042	0.9987	0.9987	0.9986
		Val	**0.9010**	**0.6850**	**0.8528**	**0.8496**	**0.8345**
MFCC	Adam	Train	0.9975	0.0088	0.9962	0.9962	0.9967
		Val	**0.9198**	**0.4510**	**0.8797**	**0.8797**	**0.8711**
	Rmsprop	Train	1.0000	0.0002	1.0000	1.0000	1.0000
		Val	**0.9135**	**0.7125**	**0.8717**	**0.8684**	**0.8550**
	Adadelta	Train	1.0000	0.0016	1.0000	1.0000	1.0000
		Val	**0.9073**	**0.6330**	**0.8609**	**0.8609**	**0.8455**
	Sgd	Train	0.9983	0.0150	0.9975	0.9975	0.9978
		Val	**0.8684**	**0.7229**	**0.8038**	**0.8008**	**0.7785**

3.5.2 ResNet

Durante la pruebas y ajustes experimentales, se evaluaron las versiones de Resnet50V2 y Resnet101V2 de Keras, los resultados fueron desfavorables, y algunos casos, se experimentó un agotamiento de la memoria RAM de la GPU, que fue una RTX 3060 de 8GB. Este problema se atribuyó principalmente a la profundidad y cantidad de parámetros de las redes.

En este sentido, fue necesario reducir tanto la profundidad de la red como la cantidad de parámetros de esta, y, tomando como base el Resnet50V2 y lo expuesto en la parte teórica, se construyó Resnet20.

Se tomaron en cuenta estas 2 funciones específicas de Keras en la construcción del Resnet20, mostradas en la Figura 67.

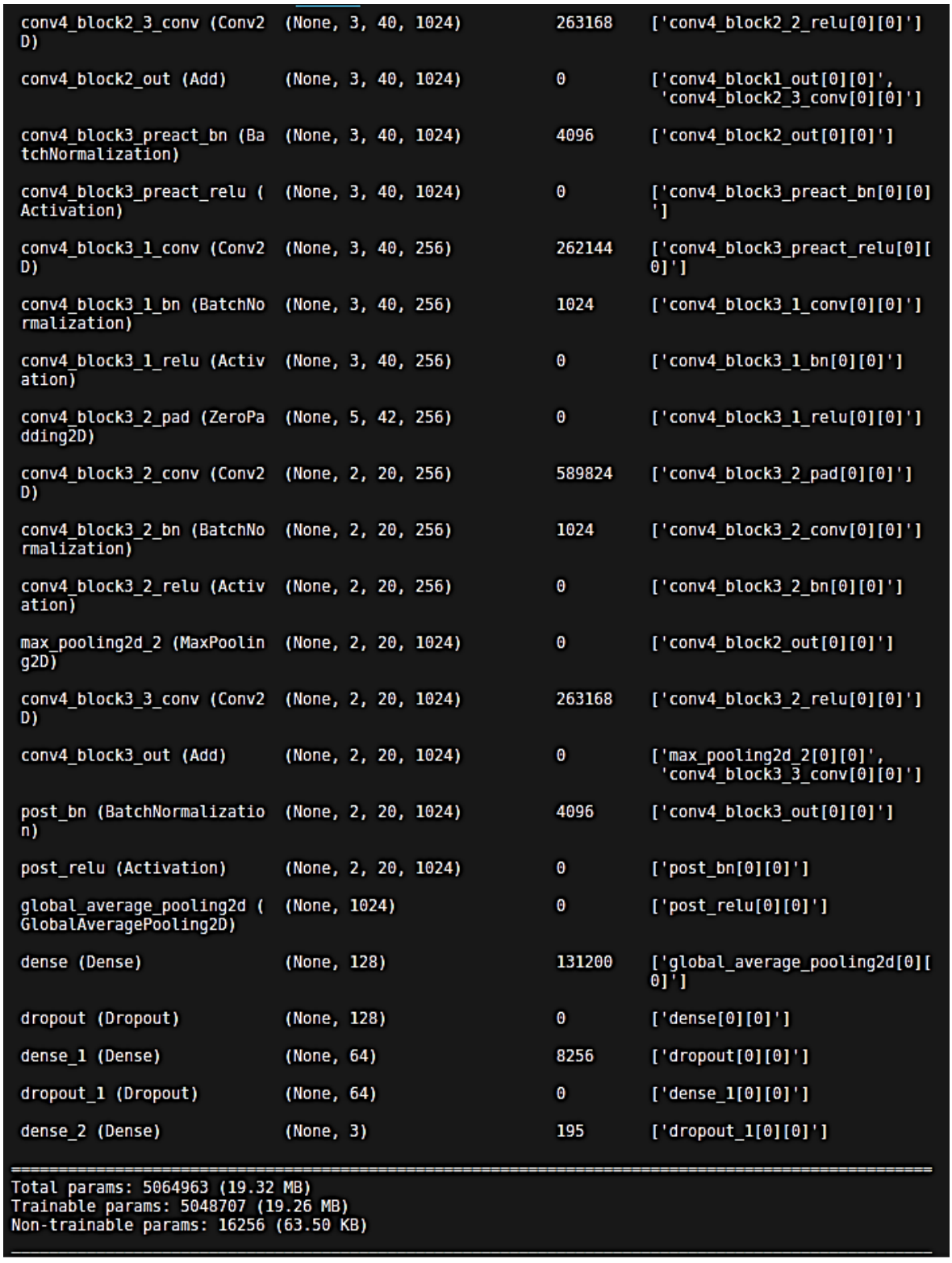
Figura 67

Funciones de Keras para la construcción de la arquitectura ResNet

```
def block2(x, filters, kernel_size=3, stride=1, conv_shortcut=False, name=None):
    """
    bn_axis = 3 if backend.image_data_format() == "channels_last" else 1

    preact = layers.BatchNormalization(
        axis=bn_axis, epsilon=1.001e-5, name=name + "_preact_bn"
    )(x)
    preact = layers.Activation("relu", name=name + "_preact_relu")(preact)

    if conv_shortcut:
        shortcut = layers.Conv2D(
            4 * filters, 1, strides=stride, name=name + "_0_conv"
        )(preact)
    else:
        shortcut = (
            layers.MaxPooling2D(1, strides=stride)(x) if stride > 1 else x
        )

    x = layers.Conv2D(
        filters, 1, strides=1, use_bias=False, name=name + "_1_conv"
    )(preact)
    x = layers.BatchNormalization(
        axis=bn_axis, epsilon=1.001e-5, name=name + "_1_bn"
    )(x)
    x = layers.Activation("relu", name=name + "_1_relu")(x)

    x = layers.ZeroPadding2D(padding=((1, 1), (1, 1)), name=name + "_2_pad")(x)
    x = layers.Conv2D(
        filters,
        kernel_size,
        strides=stride,
        use_bias=False,
        name=name + "_2_conv",
    )(x)
    x = layers.BatchNormalization(
        axis=bn_axis, epsilon=1.001e-5, name=name + "_2_bn"
    )(x)
    x = layers.Activation("relu", name=name + "_2_relu")(x)

    x = layers.Conv2D(4 * filters, 1, name=name + "_3_conv")(x)
    x = layers.Add(name=name + "_out")([shortcut, x])
    return x


def stack2(x, filters, blocks, stride1=2, name=None):
    """A set of stacked residual blocks.

    Args:
        x: input tensor.
        filters: integer, filters of the bottleneck layer in a block.
        blocks: integer, blocks in the stacked blocks.
        stride1: default 2, stride of the first layer in the first block.
        name: string, stack label.

    Returns:
        Output tensor for the stacked blocks.
    """
    x = block2(x, filters, conv_shortcut=True, name=name + "_block1")
    for i in range(2, blocks):
        x = block2(x, filters, name=name + "_block" + str(i))
    x = block2(x, filters, stride=stride1, name=name + "_block" + str(blocks))
    return x
```

Figura 68

Últimas capas del modelo Resnet20 y cantidad de parámetros



```
conv4_block2_3_conv (Conv2  (None, 3, 40, 1024)         263168    ['conv4_block2_2_relu[0][0]']
D)

conv4_block2_out (Add)      (None, 3, 40, 1024)         0         ['conv4_block1_out[0][0]',
                                                                   'conv4_block2_3_conv[0][0]']

conv4_block3_preact_bn (Ba  (None, 3, 40, 1024)         4096      ['conv4_block2_out[0][0]']
tchNormalization)

conv4_block3_preact_relu (  (None, 3, 40, 1024)         0         ['conv4_block3_preact_bn[0][0]
Activation)                                                        ']

conv4_block3_1_conv (Conv2  (None, 3, 40, 256)          262144    ['conv4_block3_preact_relu[0][
D)                                                                 0]']

conv4_block3_1_bn (BatchNo  (None, 3, 40, 256)          1024      ['conv4_block3_1_conv[0][0]']
rmalization)

conv4_block3_1_relu (Activ  (None, 3, 40, 256)          0         ['conv4_block3_1_bn[0][0]']
ation)

conv4_block3_2_pad (ZeroPa  (None, 5, 42, 256)          0         ['conv4_block3_1_relu[0][0]']
dding2D)

conv4_block3_2_conv (Conv2  (None, 2, 20, 256)          589824    ['conv4_block3_2_pad[0][0]']
D)

conv4_block3_2_bn (BatchNo  (None, 2, 20, 256)          1024      ['conv4_block3_2_conv[0][0]']
rmalization)

conv4_block3_2_relu (Activ  (None, 2, 20, 256)          0         ['conv4_block3_2_bn[0][0]']
ation)

max_pooling2d_2 (MaxPoolin  (None, 2, 20, 1024)         0         ['conv4_block2_out[0][0]']
g2D)

conv4_block3_3_conv (Conv2  (None, 2, 20, 1024)         263168    ['conv4_block3_2_relu[0][0]']
D)

conv4_block3_out (Add)      (None, 2, 20, 1024)         0         ['max_pooling2d_2[0][0]',
                                                                   'conv4_block3_3_conv[0][0]']

post_bn (BatchNormalizatio  (None, 2, 20, 1024)         4096      ['conv4_block3_out[0][0]']
n)

post_relu (Activation)      (None, 2, 20, 1024)         0         ['post_bn[0][0]']

global_average_pooling2d (  (None, 1024)                0         ['post_relu[0][0]']
GlobalAveragePooling2D)

dense (Dense)               (None, 128)                 131200    ['global_average_pooling2d[0][
                                                                   0]']

dropout (Dropout)           (None, 128)                 0         ['dense[0][0]']

dense_1 (Dense)             (None, 64)                  8256      ['dropout[0][0]']

dropout_1 (Dropout)         (None, 64)                  0         ['dense_1[0][0]']

dense_2 (Dense)             (None, 3)                   195       ['dropout_1[0][0]']

==================================================================================================
Total params: 5064963 (19.32 MB)
Trainable params: 5048707 (19.26 MB)
Non-trainable params: 16256 (63.50 KB)
__________________________________________________________________________________________________
```

Como se puede ver en la Figura 68, se redujo significativamente la cantidad de parámetros a entrenar y solucionó el problema de falta de memoria.

Al igual que con el modelo propio, se entrenó el modelo probando con cuatro optimizadores durante 150 épocas y utilizando la clase “ReduceLROnPlateau”, de Keras.

Figura 69

Función de error del modelo ResNet entrenado con el primer aumento de datos

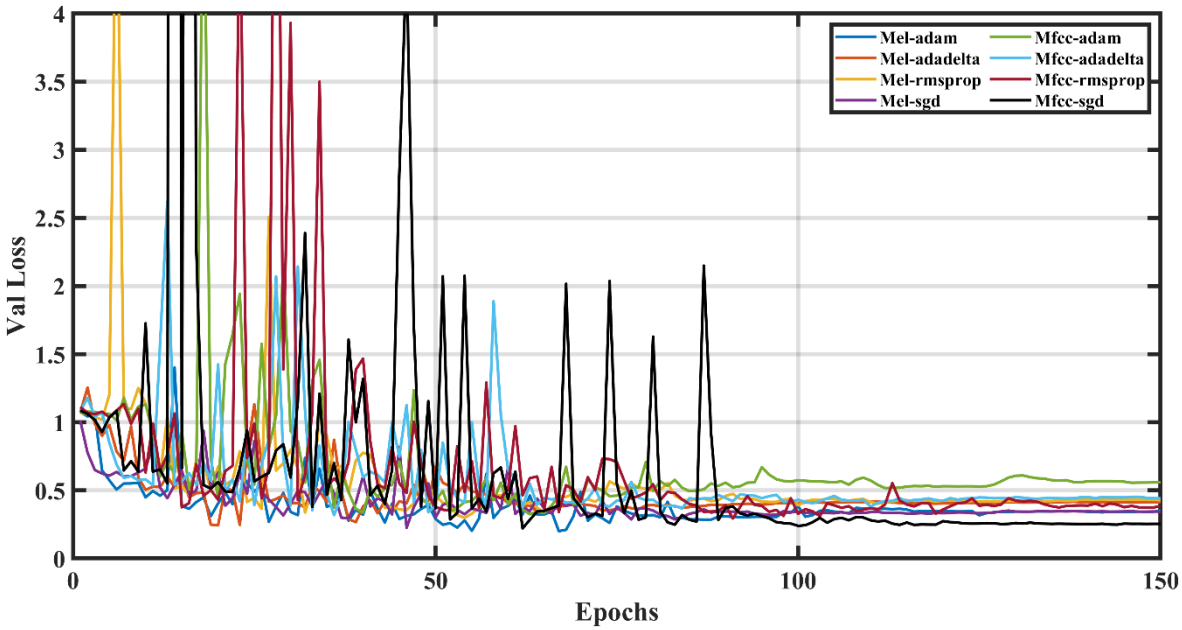


Figura 70

Métrica de exactitud del modelo ResNet entrenado con el primer aumento de datos

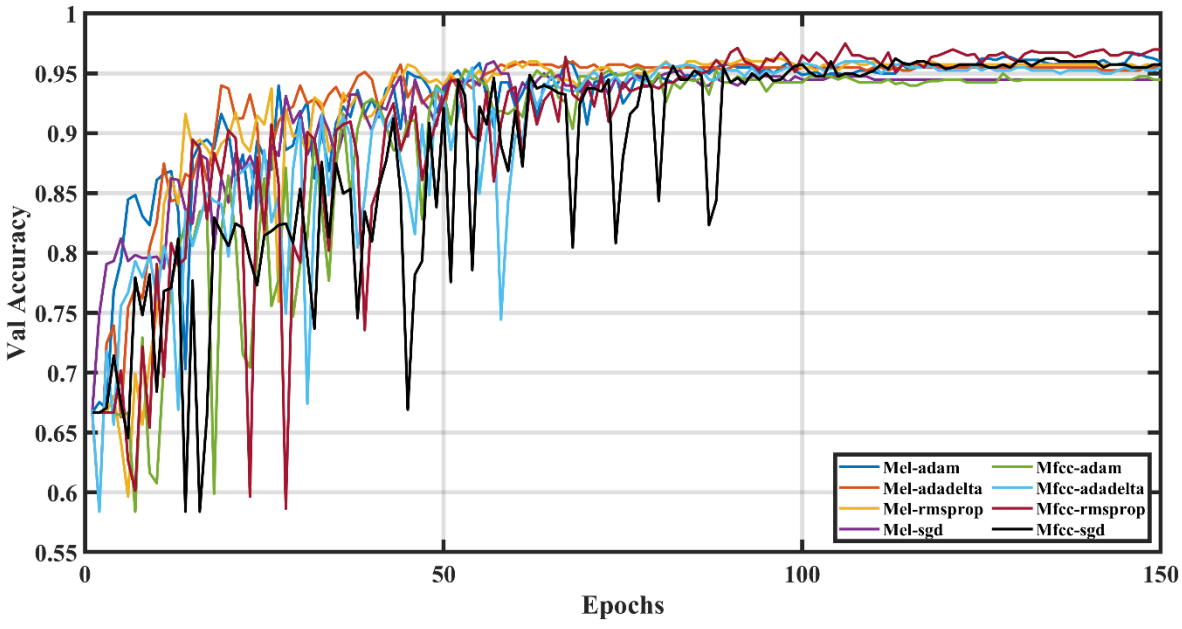


Figura 71

Métrica de precisión del modelo ResNet entrenado con el primer aumento de datos

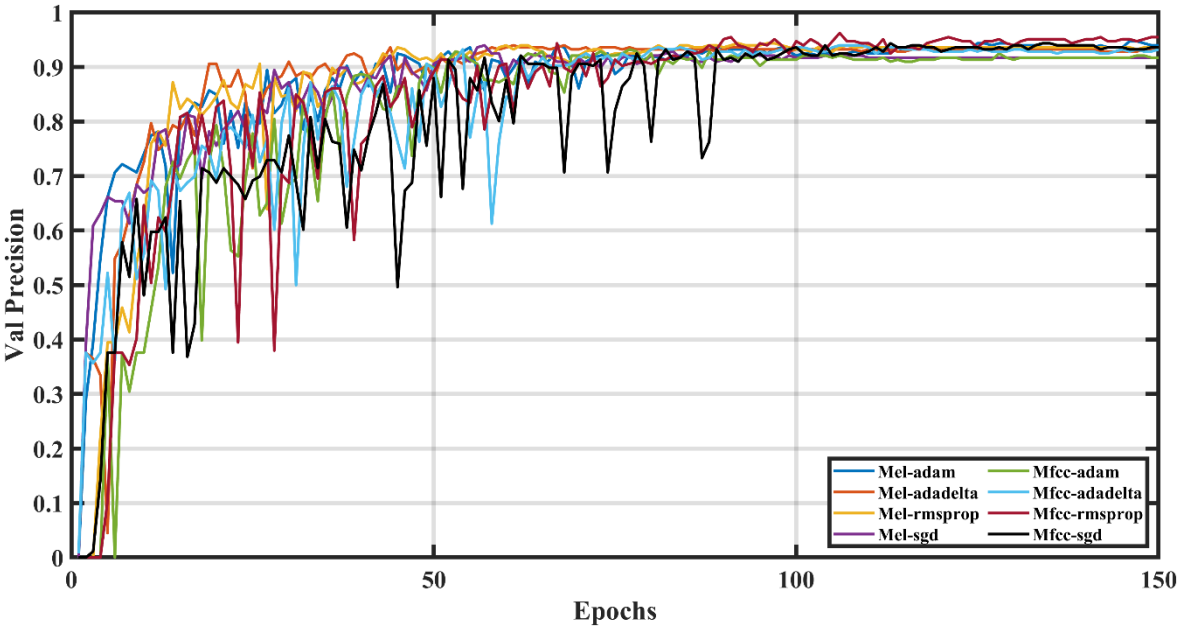


Figura 72

Métrica de sensibilidad del modelo ResNet entrenado con el primer aumento de datos

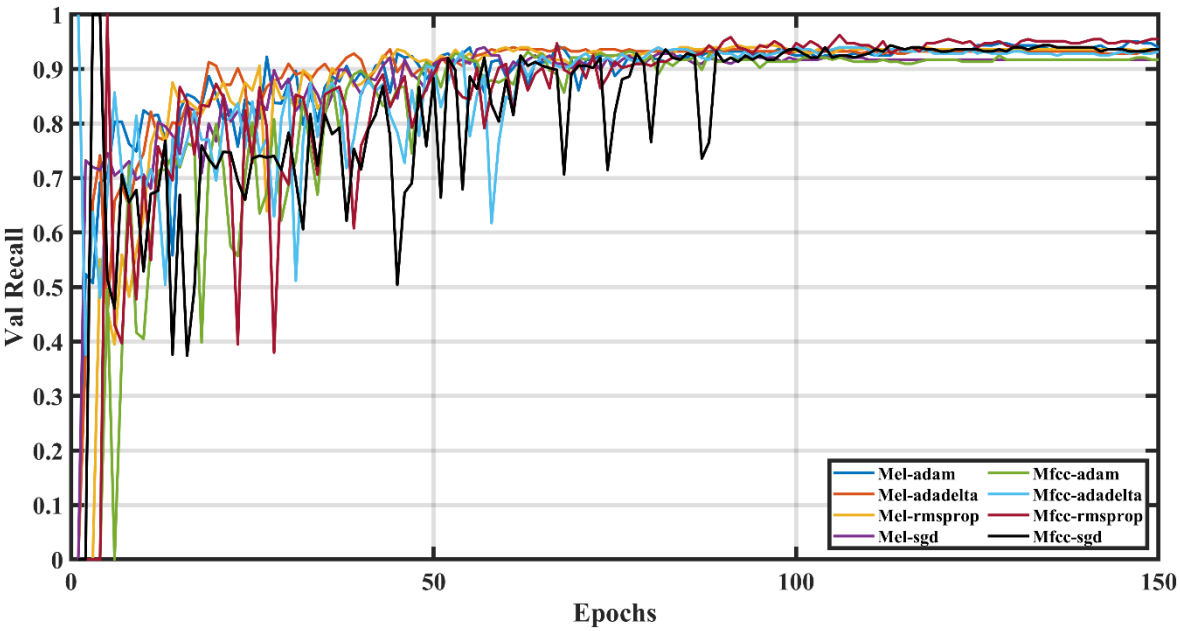


Tabla 5

Resumen de resultados del entrenamiento del modelo ResNet con el primer aumento de datos

Característica	Optimizador	Datos	Accuracy	Loss	Precision	Recall	F score
MEL	Adam	Train	1.0000	4.3e-6	1.0000	1.0000	1.0000
		Val	**0.9662**	**0.3419**	**0.9509**	**0.9474**	**0.9488**
	Rmsprop	Train	1.0000	9.4e-8	1.0000	1.0000	1.0000
		Val	**0.9624**	**0.4380**	**0.9436**	**0.9436**	**0.9426**
	Adadelta	Train	1.0000	1.6e-6	1.0000	1.0000	1.0000
		Val	**0.9599**	**0.4075**	**0.9398**	**0.9398**	**0.9369**
	Sgd	Train	1.0000	0.0001	1.0000	1.0000	1.0000
		Val	**0.9599**	**0.3356**	**0.9398**	**0.9398**	**0.9371**
MFCC	Adam	Train	1.0000	0.0001	1.0000	1.0000	1.0000
		Val	**0.9549**	**0.5218**	**0.9323**	**0.9323**	**0.9350**
	Rmsprop	Train	1.0000	8.6e-7	1.0000	1.0000	1.0000
		Val	**0.9749**	**0.3725**	**0.9624**	**0.9624**	**0.9641**
	Adadelta	Train	1.0000	1.2e-6	1.0000	1.0000	1.0000
		Val	**0.9599**	**0.4315**	**0.9398**	**0.9398**	**0.9382**
	Sgd	Train	1.0000	9.9e-5	1.0000	1.0000	1.0000
		Val	**0.9624**	**0.2567**	**0.9436**	**0.9436**	**0.9431**

Figura 73

Función de error del modelo ResNet entrenado con el segundo aumento de datos

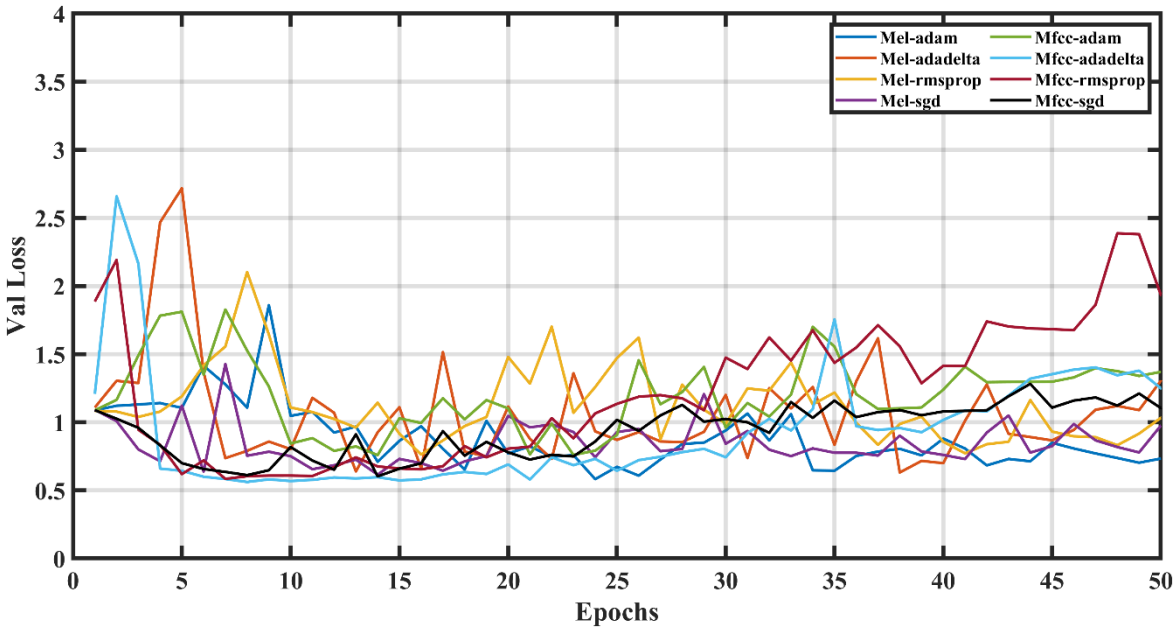


Figura 74

Métrica de exactitud del modelo ResNet entrenado con el segundo aumento de datos

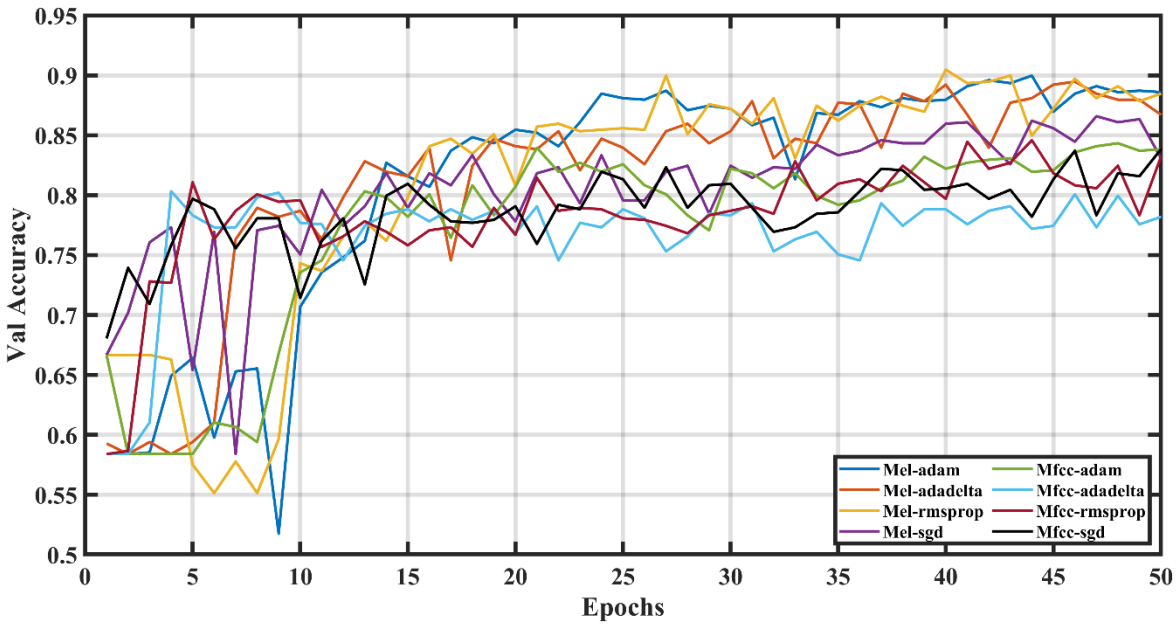


Figura 75

Métrica de precisión del modelo ResNet entrenado con el segundo aumento de datos

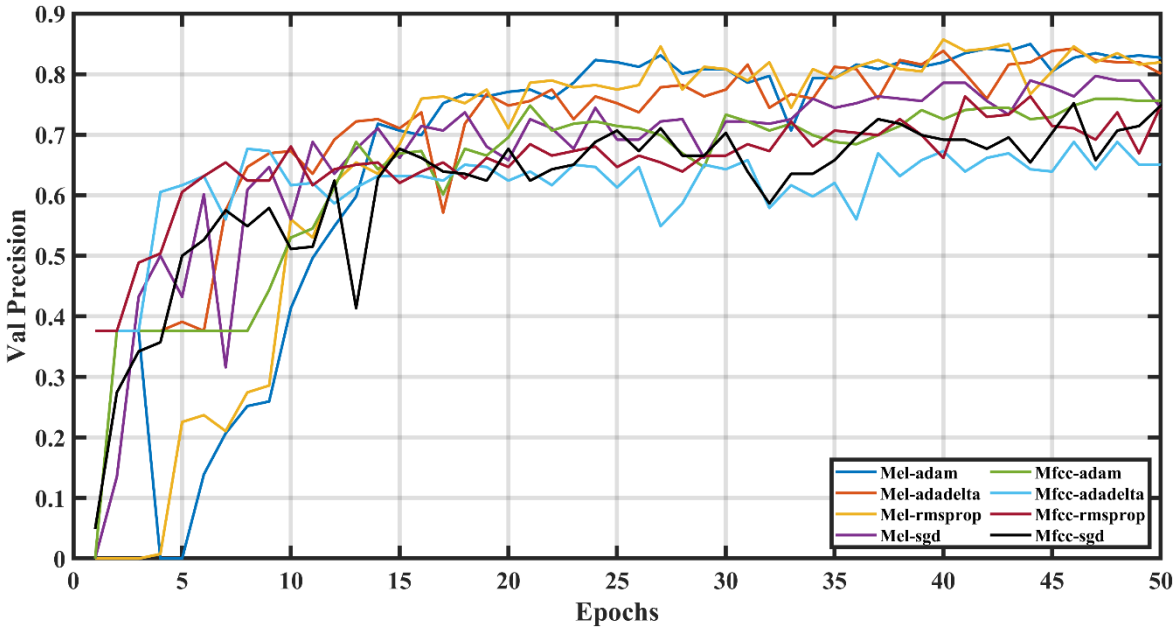


Figura 76

Métrica de sensibilidad del modelo ResNet entrenado con el segundo aumento de datos

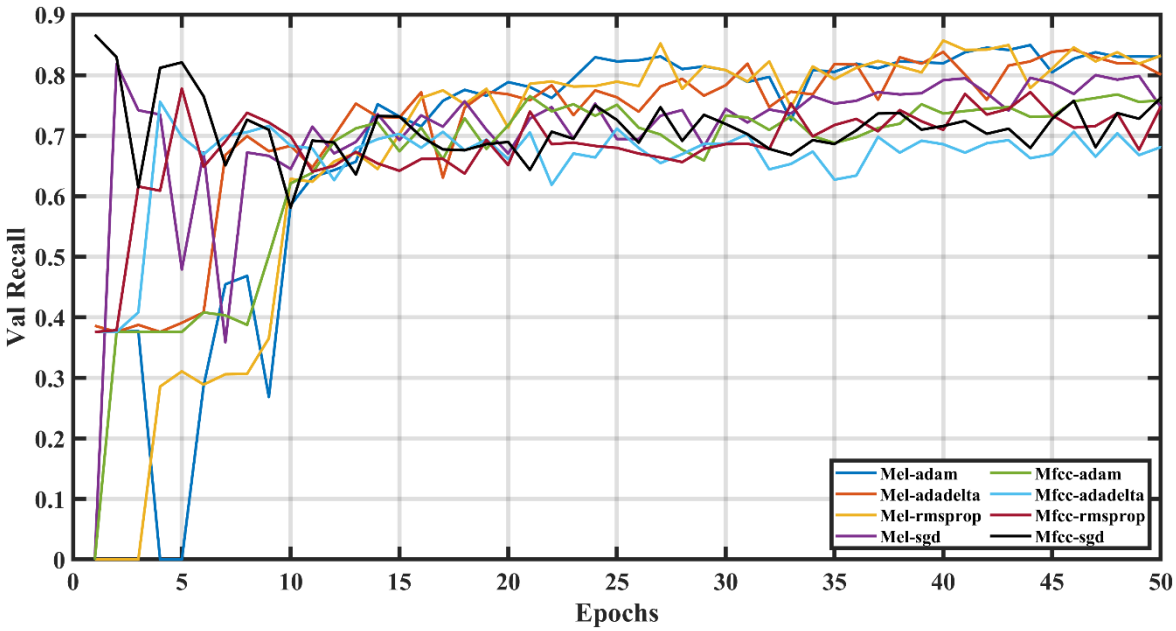


Tabla 6

Resumen de resultados del entrenamiento del modelo ResNet con el segundo aumento de datos

Característica	Optimizador	Datos	Accuracy	Loss	Precision	Recall	F score
MEL	Adam	Train	1.0000	6.3e-6	1.0000	1.0000	1.0000
		Val	**0.9023**	**0.7817**	**0.8534**	**0.8534**	**0.8388**
	Rmsprop	Train	1.0000	2.6e-5	1.0000	1.0000	1.0000
		Val	**0.9048**	**0.8577**	**0.8571**	**0.8571**	**08397**
	Adadelta	Train	1.0000	0.0008	1.0000	1.0000	1.0000
		Val	**0.9048**	**0.9896**	**0.8571**	**0.8571**	**0.8453**
	Sgd	Train	1.0000	3.2e-5	1.0000	1.0000	1.0000
		Val	**0.9048**	**0.6754**	**0.8571**	**0.8571**	**0.8443**
MFCC	Adam	Train	1.0000	0.0001	1.0000	1.0000	1.0000
		Val	**0.8471**	**1.4218**	**0.7707**	**0.7707**	**0.7360**
	Rmsprop	Train	1.0000	9.0e-6	1.0000	1.0000	1.0000
		Val	**0.8521**	**2.9425**	**0.7782**	**0.7782**	**0.7646**
	Adadelta	Train	0.9891	0.0437	0.9849	0.9825	0.9814
		Val	**0.8120**	**1.8775**	**0.7197**	**0.7143**	**0.6988**
	Sgd	Train	1.0000	0.0035	1.0000	1.0000	1.0000
		Val	**0.8459**	**1.4539**	**0.7698**	**0.7669**	**0.7448**

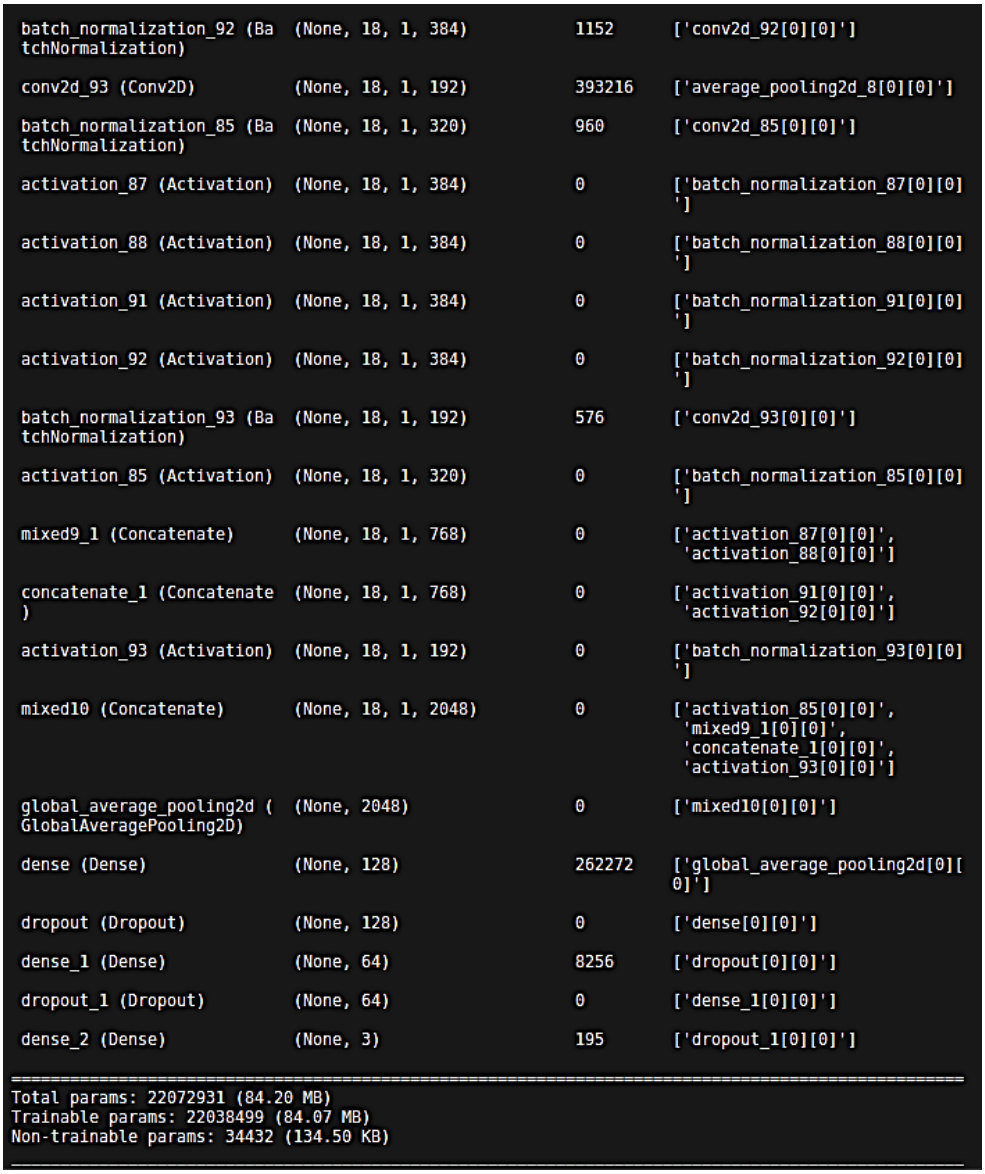
3.5.3 Inception

La librería Keras ofrece la arquitectura InceptionV3, que se utilizó como otro punto de comparación con el modelo propio de doble rama. Una vez extraída la arquitectura, se eliminó la última capa y se incorporaron dos capas densas de tamaño 128 y 64 respectivamente, ambas utilizando ReLu como función de activación.

Finalmente, una capa densa final con tamaño 3 fue añadida, con una función de activación Softmax. Entre capas densas se agregaron capas de Dropout para la regularización del modelo.

Figura 77

Últimas capas del modelo InceptionV3 y cantidad de parámetros



Otro detalle importante al utilizar InceptionV3 es el tamaño de las entradas. Este permitía imágenes de tamaño mayor o igual a 75x75, y como las características mel y mfcc tenían tamaño de 40x626, fue necesario realizar un ajuste.

Se utilizó la librería cv2 para este trabajo y el extracto del código para la tarea en cuestión se presenta en la Figura 78.

Figura 78

Extracto del código para el ajuste del tamaño de las entradas para InceptionV3

```
if(Params._feature == 'stft'):
    features, labels = STFT_EXTRACTOR.load_STFT(path_images_stft=Params.stft_features_path)
elif(Params._feature == 'mel'):
    features, labels = MEL_EXTRACTOR.load_MEL(path_images_mel=Params.mel_features_path)
elif(Params._feature == 'mfcc'):
    features, labels = MFCC_EXTRACTOR.load_MFCC(path_images_mfcc=Params.mfcc_features_path)
else:
    print("Características no existentes, prueba con stft, mel o mfcc")

# print(len(features))
# print(features[0].shape)

for i in range(len(labels)):
    labels[i] = label_numerical[labels[i]]

features      = np.array(features)
y_categorical = np.array(labels)

resized_features = [cv2.resize(img, (75, 626)) for img in features] # Ajuste del tamaño de las entradas
resized_features = np.array(resized_features)
```

Similar a los otros dos modelos, se entrenó InceptionV3 con 4 diferentes optimizadores y durante 150 épocas.

Figura 79

Función de error del modelo InceptionV3 entrenado con el primer aumento de datos

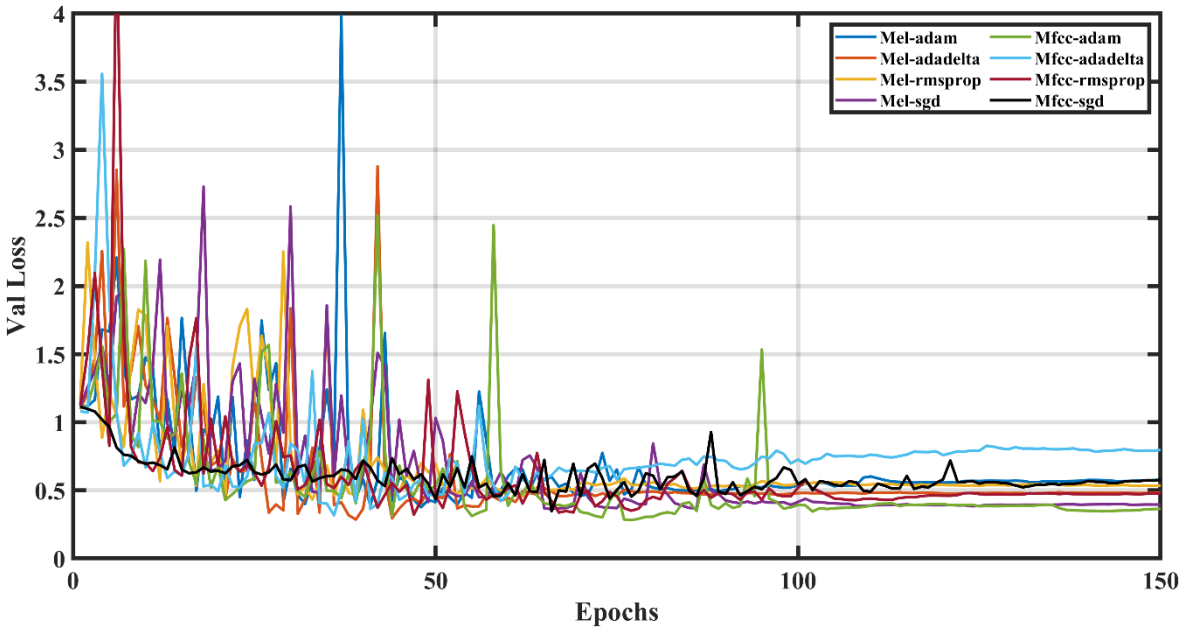


Figura 80

Métrica de exactitud del modelo InceptionV3 entrenado con el primer aumento de datos

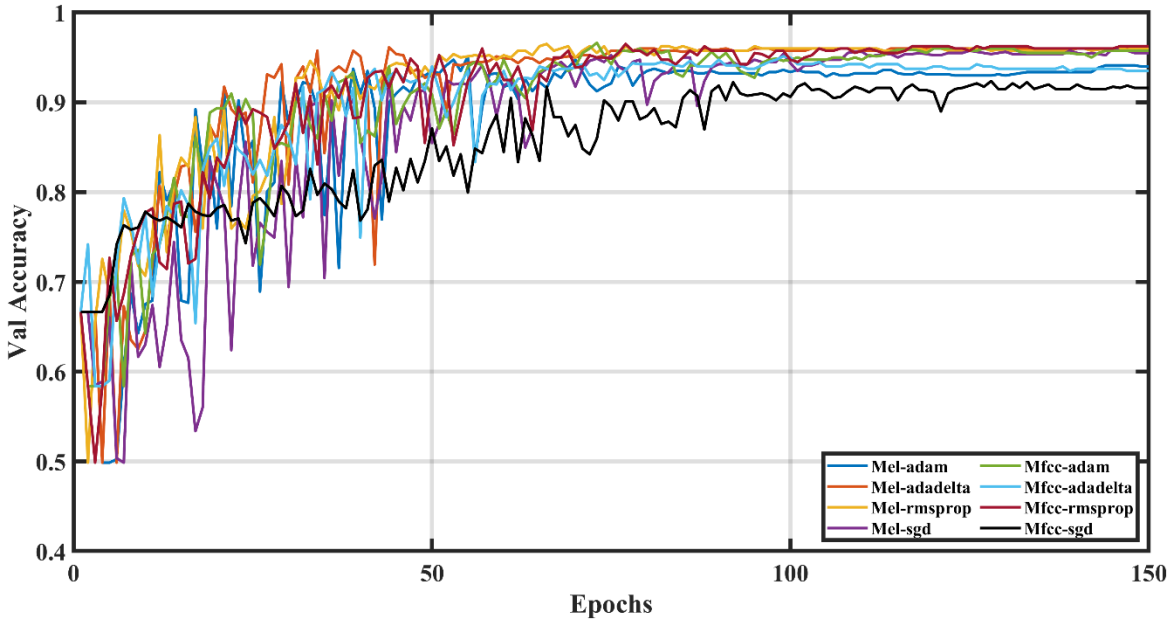


Figura 81

Métrica de precisión del modelo InceptionV3 entrenado con el primer aumento de dato

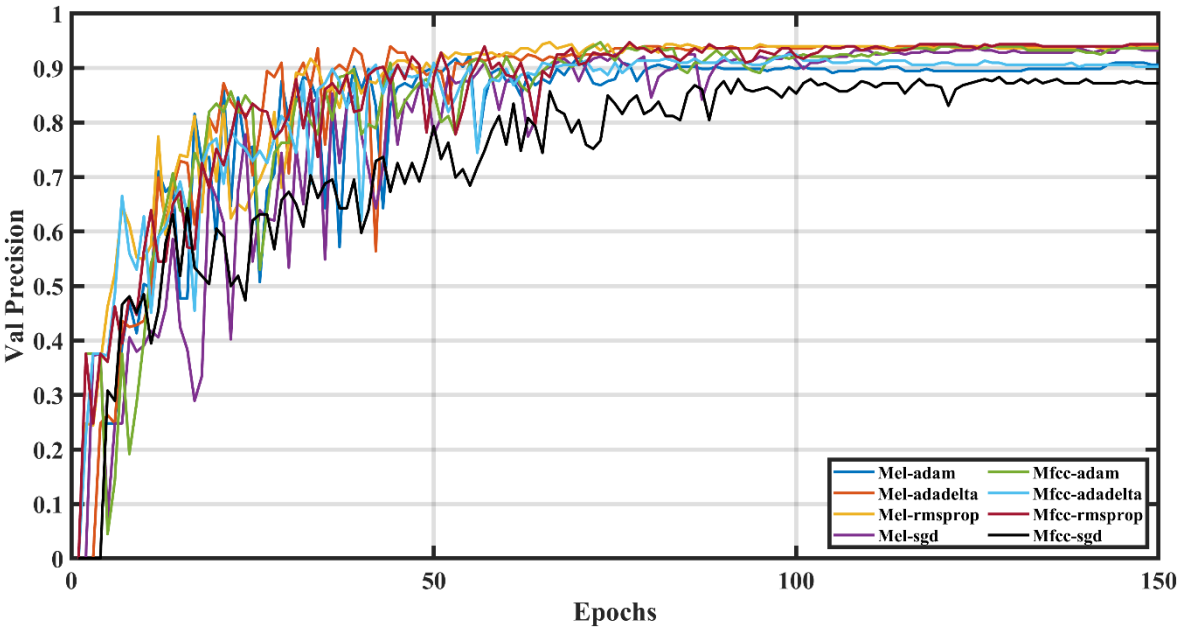


Figura 82

Métrica de sensibilidad del modelo InceptionV3 entrenado con el primer aumento de dato

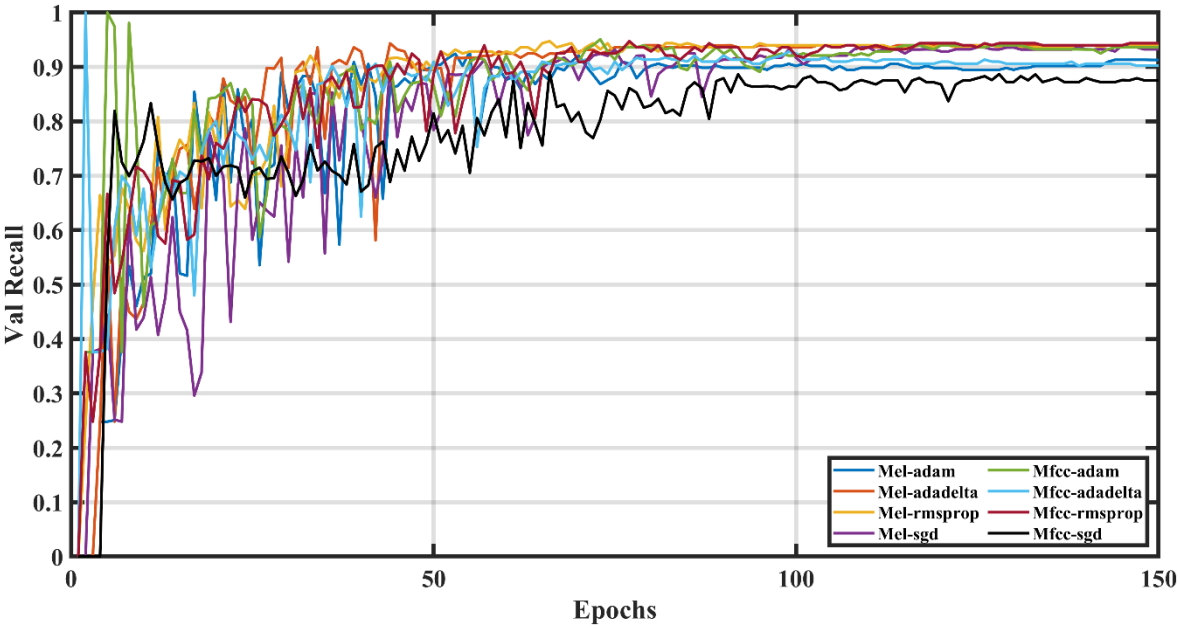


Tabla 7

Resumen de resultados del entrenamiento del modelo InceptionV3 con el primer aumento de datos

Característica	Optimizador	Datos	Accuracy	Loss	Precision	Recall	F score
MEL	Adam	Train	1.0000	0.0003	1.0000	1.0000	1.0000
		Val	**0.9511**	**0.4468**	**0.9283**	**0.9248**	**0.9239**
	Rmsprop	Train	1.0000	8.4e-7	1.0000	1.0000	1.0000
		Val	**0.9649**	**0.5046**	**0.9474**	**0.9474**	**0.9472**
	Adadelta	Train	1.0000	0.0019	1.0000	1.0000	1.0000
		Val	**0.9612**	**0.2958**	**0.9434**	**0.9398**	**0.9440**
	Sgd	Train	1.0000	2.6e-5	1.0000	1.0000	1.0000
		Val	**0.9599**	**0.3988**	**0.9398**	**0.9398**	**0.9407**
MFCC	Adam	Train	1.0000	0.0003	1.0000	1.0000	1.0000
		Val	**0.9662**	**0.3023**	**0.9509**	**0.9474**	**0.9495**
	Rmsprop	Train	1.0000	1.5e-6	1.0000	1.0000	1.0000
		Val	**0.9649**	**0.3513**	**0.9474**	**0.9474**	**0.9465**
	Adadelta	Train	1.0000	2.1e-6	1.0000	1.0000	1.0000
		Val	**0.9511**	**0.6981**	**0.9283**	**0.9248**	**0.9285**
	Sgd	Train	1.0000	0.0008	1.0000	1.0000	1.0000
		Val	**0.9236**	**0.5553**	**0.8868**	**0.8835**	**0.8816**

Figura 83

Función de error del modelo InceptionV3 entrenado con el segundo aumento de datos

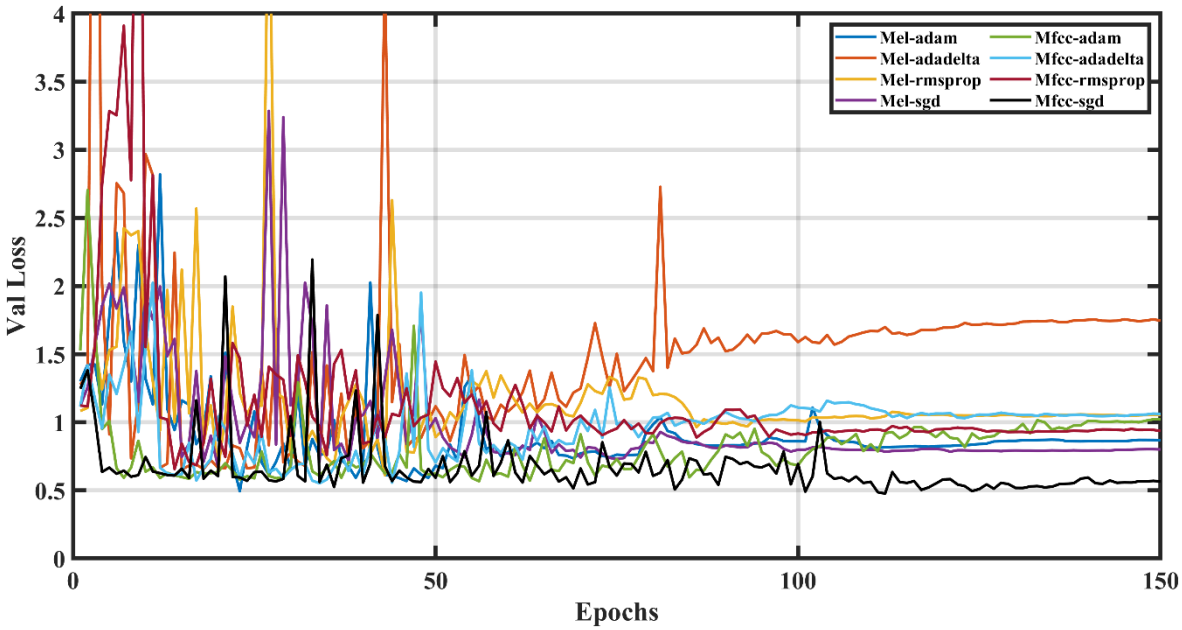


Figura 84

Métrica de exactitud del modelo InceptionV3 entrenado con el segundo aumento de datos

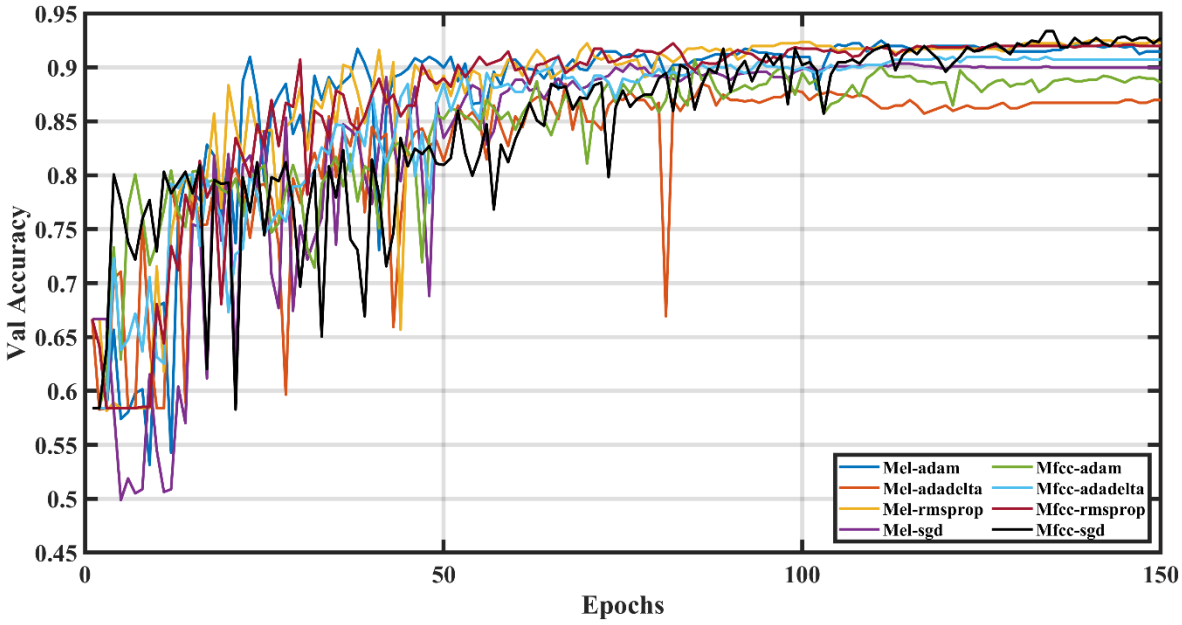


Figura 85

Métrica de precisión del modelo InceptionV3 entrenado con el segundo aumento de datos

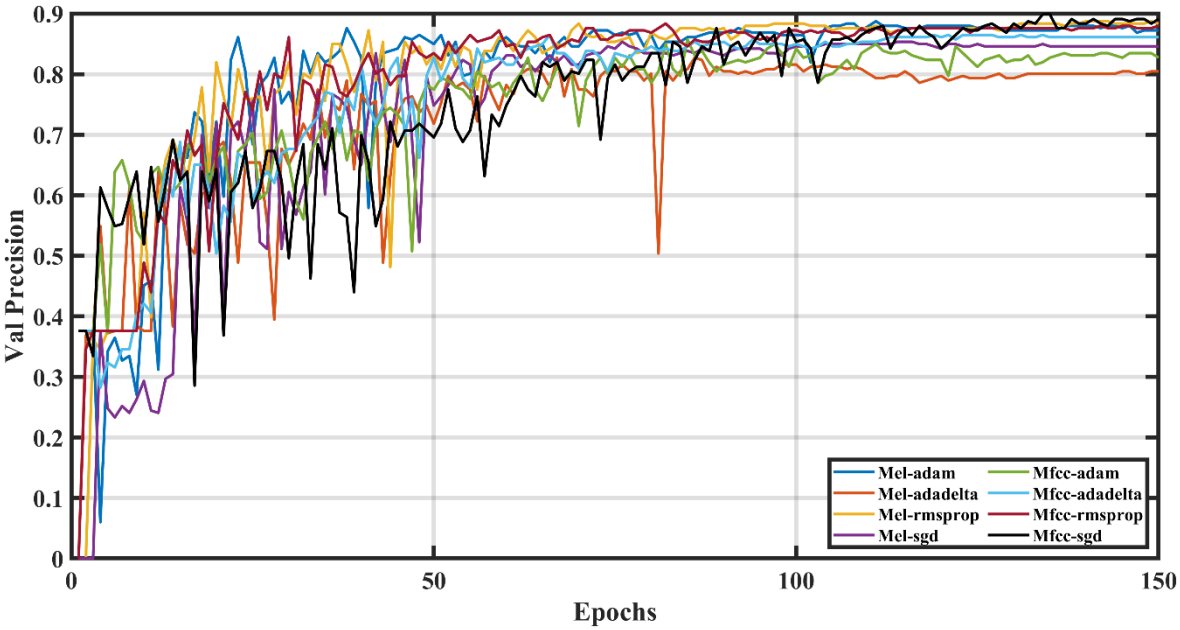


Figura 86

Métrica de sensibilidad del modelo InceptionV3 entrenado con el segundo aumento de datos

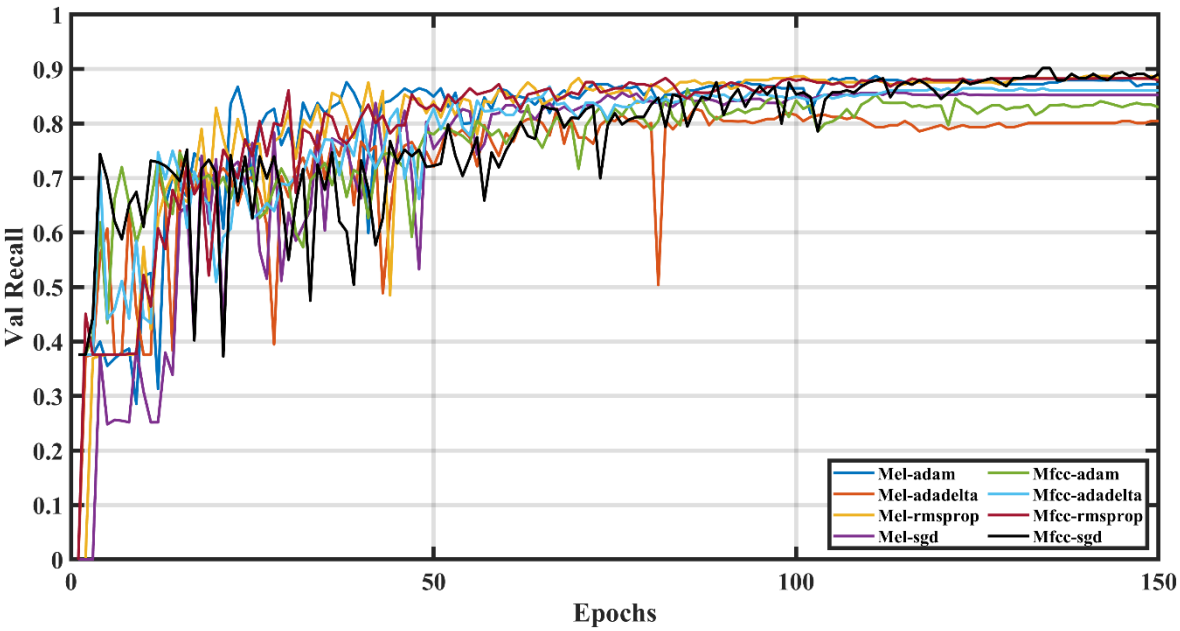


Tabla 8

Resumen de resultados del entrenamiento del modelo InceptionV3 con el segundo aumento de datos

Característica	Optimizador	Datos	Accuracy	Loss	Precision	Recall	F score
MEL	Adam	Train	1.0000	1.1e-5	1.0000	1.0000	1.0000
		Val	**0.9248**	**0.8135**	**0.8872**	**0.8872**	**0.8838**
	Rmsprop	Train	1.0000	4.3e-7	1.0000	1.0000	1.0000
		Val	**0.9248**	**1.0566**	**0.8872**	**0.8872**	**0.8799**
	Adadelta	Train	1.0000	0.0013	1.0000	1.0000	1.0000
		Val	**0.8847**	**1.1643**	**0.8271**	**0.8271**	**0.8217**
	Sgd	Train	1.0000	0.0002	1.0000	1.0000	1.0000
		Val	**0.9035**	**0.7381**	**0.8566**	**0.8534**	**0.8483**
MFCC	Adam	Train	0.9975	0.0085	0.9962	0.9962	0.9958
		Val	**0.9073**	**0.5962**	**0.8636**	**0.8571**	**0.8469**
	Rmsprop	Train	1.0000	2.9e-5	1.0000	1.0000	1.0000
		Val	**0.9223**	**1.0261**	**0.8835**	**0.8835**	**0.8760**
	Adadelta	Train	0.9992	0.0029	0.9987	0.9987	0.9989
		Val	**0.9110**	**0.8179**	**0.8679**	**0.8647**	**0.8577**
	Sgd	Train	0.9992	0.0070	0.9987	0.9987	0.9989
		Val	**0.9336**	**0.5230**	**0.9019**	**0.8985**	**0.8957**

Capítulo IV. Análisis y discusión de resultados

En esta sección, los resultados obtenidos al evaluar diferentes optimizadores durante el entrenamiento de los tres modelos CNN para la clasificación de sonidos pulmonares (de la Figura 59 a la Figura 86) son analizados. Aunque los detalles pueden observarse de la Tabla 3 a la Tabla 8, en resumen, la Tabla 9 presenta una síntesis de los mejores resultados obtenidos por cada modelo y técnica aplicada.

Tabla 9

Comparativa entre resultados de los modelos entrenados en este trabajo

Modelo	Accuracy	Loss	Precision	Recall	F score
CNN doble rama-MFCC-Noise Injection + Pitch Shifting-SGD	**97.87%**	**0.1230**	**96.98%**	**96.62%**	**96.47%**
CNN doble rama-MEL-Noise Injection + Pitch Shifting-Adadelta	96.49%	0.1715	94.74%	94.74%	94.51%
CNN doble rama-MFCC-Time and frequency masking-Adam	91.98%	0.4510	87.97%	87.97%	87.11%
CNN doble rama-MEL-Time and frequency masking-Adam	92.23%	0.6051	88.35%	88.35%	86.99%
ResNet20-MFCC-Noise Injection + Pitch Shifting-Rmsprop	97.49%	0.3725	96.24%	96.24%	96.41%
ResNet20-MEL-Noise Injection + Pitch Shifting-Adam	96.62%	0.3419	95.09%	94.74%	94.88%
ResNet20-MFCC- Time and frequency masking-Rmsprop	85.21%	2.9425	77.82%	77.82%	76.46%
ResNet20-MEL- Time and frequency masking-Adadelta	90.48%	0.9896	85.71%	85.71%	84.53%
InceptionV3-MFCC- Time and frequency masking-Adam	96.62%	0.3023	95.09%	94.74%	94.95%
InceptionV3-MEL-Noise Injection + Pitch Shifting-Rmsprop	96.49%	0.5046	94.74%	94.74%	94.72%
InceptionV3-MFCC- Time and frequency masking-SGD	93.36%	0.5230	90.19%	89.85%	89.57%
InceptionV3-MEL- Time and frequency masking-Adam	92.48%	0.8135	88.72%	88.72%	88.38%

Los mejores resultados se consiguieron al aplicar el primer proceso de aumento de datos, que involucra el desplazamiento de tono y la inyección de ruido.

Cabe señalar que las técnicas de enmascaramiento en frecuencia y tiempo dieron los resultados pocos significativos, posiblemente porque eliminaron información relevante de los espectrogramas.

En particular el mejor resultado se logró con el modelo CNN de doble rama, que utilizó las características MFCC, el aumento de datos 1 y el optimizador SGD.

En la Figura 87, Figura 88 y Figura 89 se muestran las matrices de confusión correspondientes a cada grupo de datos (entrenamiento, validación y prueba).

Figura 87

Matriz de confusión al evaluar los datos de entrenamiento

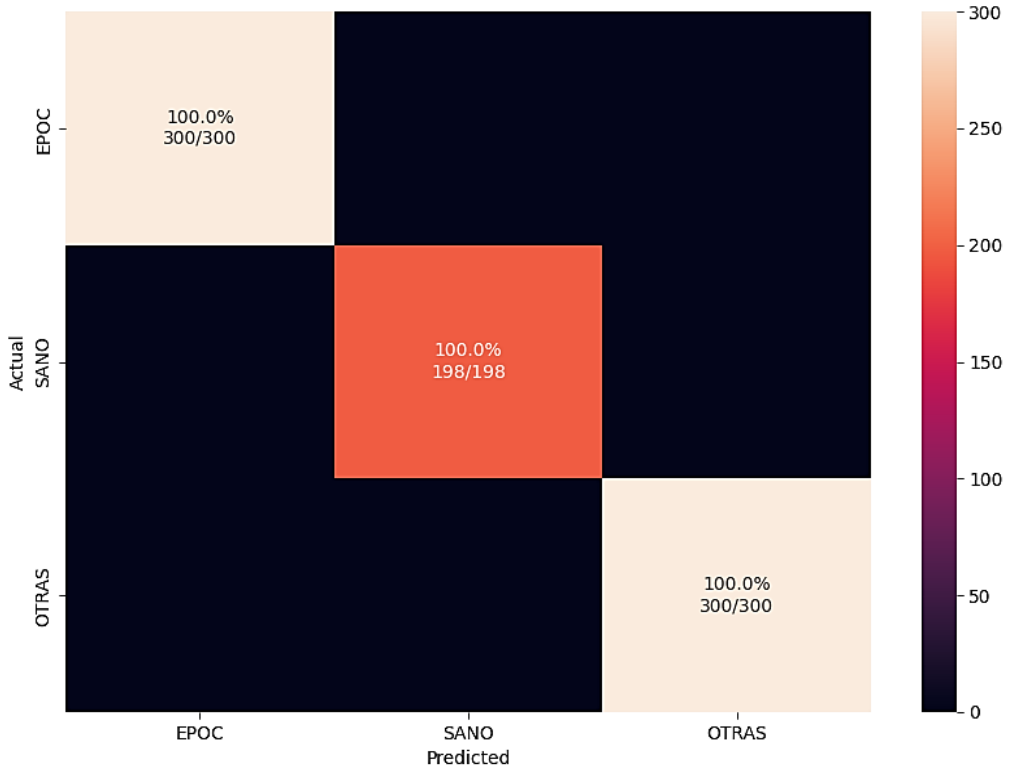


Figura 88

Matriz de confusión al evaluar los datos de validación

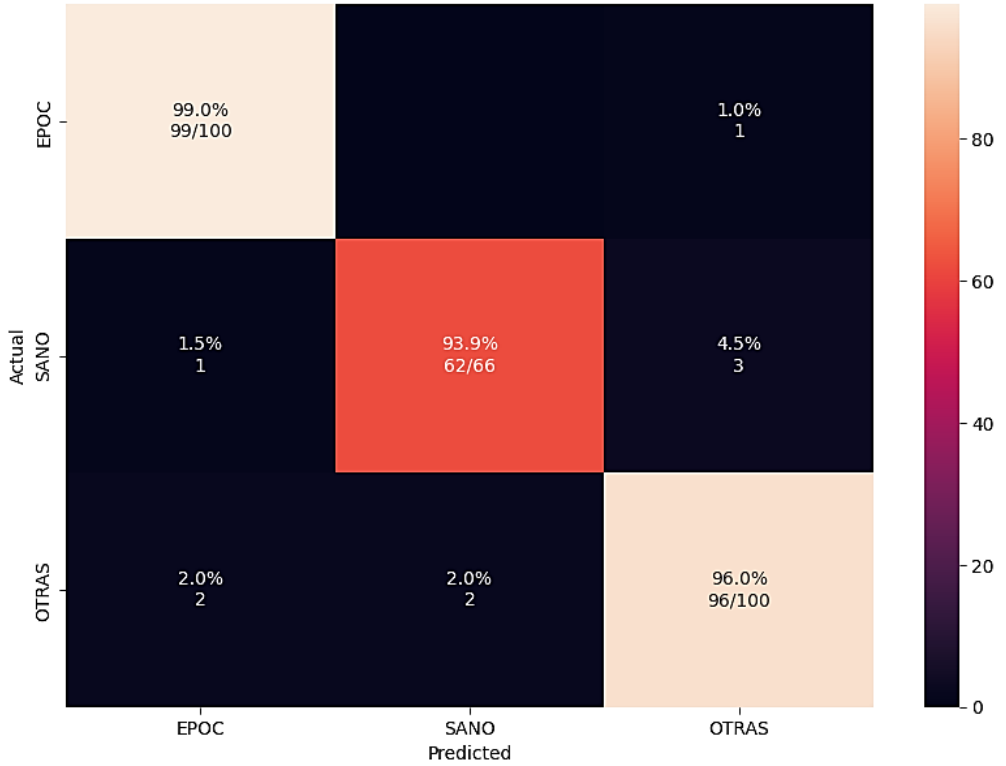
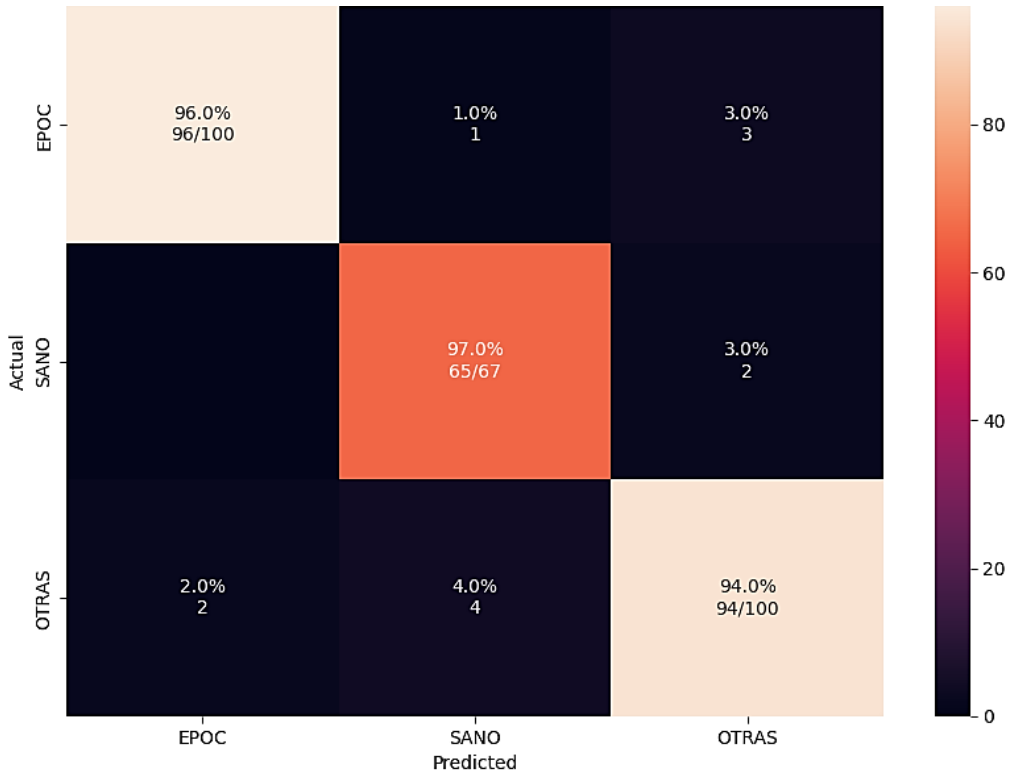


Figura 89

Matriz de confusión al evaluar los datos de prueba



Cabe resaltar que los datos de prueba son independientes a los del proceso de entrenamiento y validación de los modelos.

Además, para evaluar la eficiencia del modelo, es crucial que este pueda realizar las predicciones minimizando el tiempo de máquina. Esto se puede verificar al observar la Figura 90 y Figura 91.

Figura 90

Tiempo de respuesta en la predicción de un paciente con EPOC

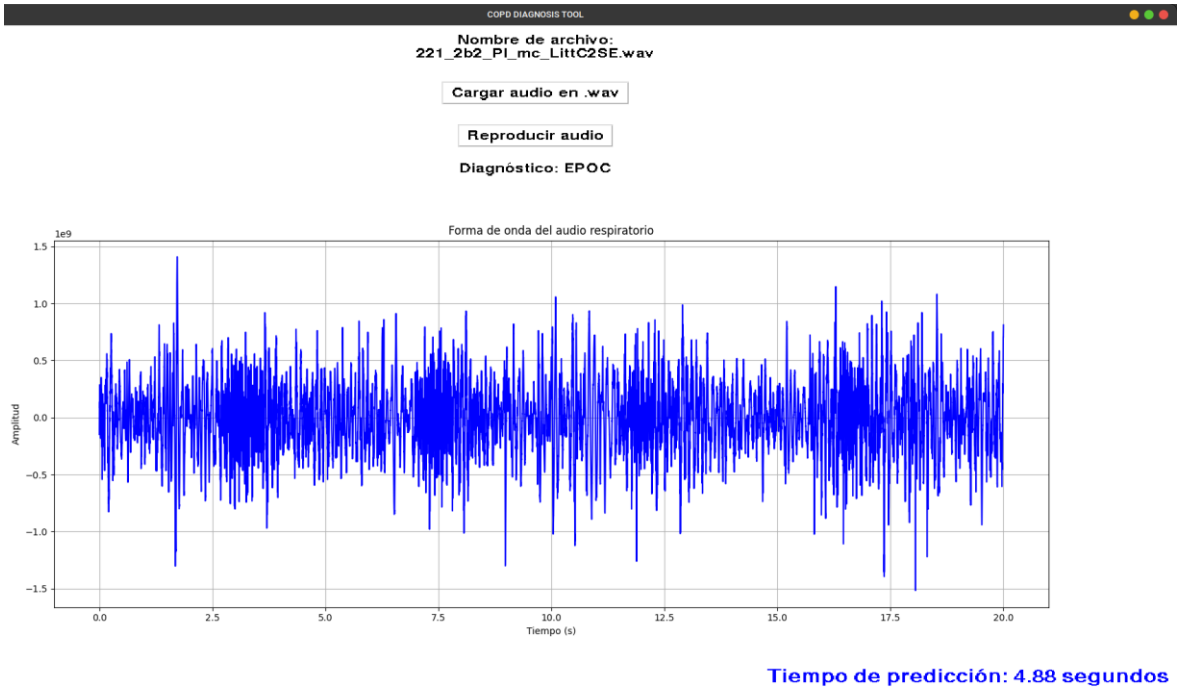
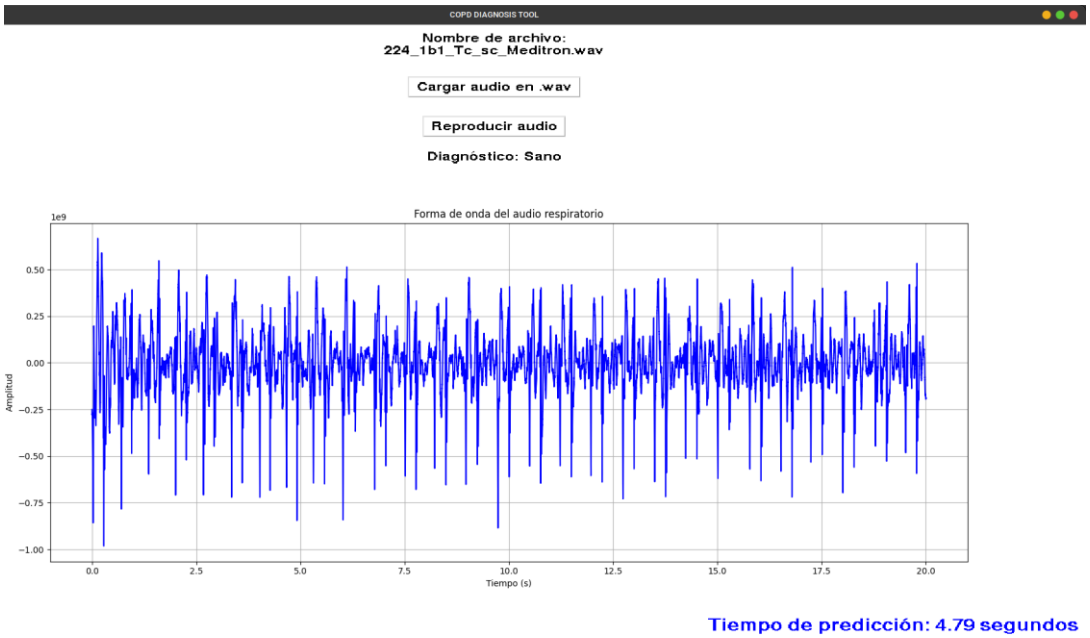


Figura 91

Tiempo de respuesta en la predicción de un paciente sano



4.1 Discusión de resultados

El modelo CNN de doble rama mencionado en la Tabla 9, ha logrado resultados satisfactorios en la clasificación de los sonidos pulmonares. Sin embargo, es fundamental comparar estos logros con lo obtenidos en las investigaciones previas mencionadas en la sección 1. Es importante señalar que la comparación directa difiere debido a que no se están clasificando las mismas categorías en todos los estudios.

Por una parte, Fraiwan et al. (2020), en su estudio, obtuvieron un 95.28% de sensibilidad y 93.61% de F1score, clasificando condiciones respiratorias en seis categorías, tres más que en este trabajo. Usualmente a mayor número de categorías, mayor tarea de clasificación.

Adicionalmente, en el estudio Hazra & Majhi (2020), también se clasificaron audios respiratorios en 6 categorías. Obtuvieron un 71.83% de precisión, 50.83% de sensibilidad y 54.83% de F1score. Sin embargo, sus resultados no son muy favorables a comparación de este trabajo, y esto se puede atribuir a que su conjunto de datos es muy desbalanceado, lo que conlleva a un aprendizaje deficiente de las categorías con menos muestras.

Por otra parte, con una clasificación más general de ruidos anormales en la respiración, Petmezas et al. (2022) obtuvieron valores en sus métricas menores que en este trabajo. Sin embargo, ambos hemos coincidido en que el uso de STFT o sus variantes como el espectrograma de Mel y MFCC, para extraer las características de los audios respiratorios, es efectivo.

Finalmente, el estudio de Tessema et al. (2022) logró los mejores resultados en comparación con este trabajo y los anteriores mencionados. Esto es atribuible al uso de una base de datos balanceada.

Con el análisis realizado, procedemos a contrastar las hipótesis planteadas al inicio de este trabajo.

4.2 Contrastación de las hipótesis

4.2.1 Hipótesis 1: Con la implementación de redes neuronales convolucionales, se detectará la EPOC mediante sonidos pulmonares.

Para verificar esta hipótesis, se diseñó y entrenó una CNN de doble rama específicamente adaptada para la clasificación de sonidos pulmonares. Los resultados obtenidos indican un impacto positivo en el entrenamiento y la detección de la EPOC, mostrando una precisión destacable por encima del 90% en la clasificación de sonidos pulmonares. Estos logros han sido resumidos en la Tabla 3.

4.2.2 Hipótesis 2: Con la implementación de arquitecturas de redes pre entrenadas, se identificará la óptima para la detección de la EPOC mediante sonidos pulmonares.

Al implementar arquitecturas pre entrenadas, se ha identificado que la adaptación de estas arquitecturas, como ResNet20 e InceptionV3, han contribuido significativamente a la detección de la EPOC mediante sonidos pulmonares. Se evaluaron y compararon estas arquitecturas para determinar cuál se ajusta de manera óptima para la tarea en cuestión. Los resultados obtenidos, como muestran la Tabla 5 y Tabla 7, sugiere que el uso de redes pre entrenadas tuvo un impacto positivo en la detección de la enfermedad.

4.2.3 Hipótesis 3: Con la aplicación de diversos algoritmos de optimización se logra determinar la arquitectura más significativa en términos de precisión.

Cuatro optimizadores se evaluaron por cada modelo utilizado en este trabajo. El resumen de los mejores resultados conseguidos se presenta en la Tabla 9, alcanzando métricas de precisión por encima del 90% al clasificar los sonidos pulmonares. Estos resultados respaldan la hipótesis tres, y sugieren que el uso de algoritmos de optimización mejora significativamente los modelos para la detección de la EPOC.

4.2.4 Hipótesis 4: Con la verificación del modelo entrenado aplicando sonidos pulmonares de pacientes con sospecha de EPOC, se confirmará la eficacia del sistema inteligente.

Como método de verificación del mejor modelo entrenado, se guardaron 267 audios respiratorios, 100 de estos audios de pacientes con EPOC, que no fueron utilizados durante el entrenamiento y validación del modelo.

La matriz de confusión presentada en la Figura 89 indica que el modelo entrenado es efectivo para la clasificación de sonidos pulmonares, especialmente en la detección de la EPOC. Además, se destaca la rapidez de las predicciones, con un tiempo de 5 segundos aproximadamente, lo que refuerza la eficacia del sistema inteligente.

Conclusiones

1. Con la implementación de la arquitectura CNN de doble rama de clasificación de sonidos pulmonares se han obtenido resultados positivos con una precisión de 96.98%, sensibilidad de 96.62%, y F1-score de 96.47% en los datos de validación y una precisión de 95.49%, sensibilidad de 95.13%, y F1-score de 95.44% en los datos de prueba.
2. Con los espectrogramas de Mel y los MFCC, se han obtenido características relevantes en los sonidos pulmonares, permitiendo una mejora significativa en la capacidad de clasificación de los modelos, alcanzando métricas de precisión y exactitud superiores a 90%.
3. La aplicación de técnicas de aumento de datos, como la inyección de ruido y desplazamiento de tono, ha permitido mejorar el rendimiento de los modelos, superando a las técnicas de enmascaramiento en tiempo y frecuencia, alcanzando valores superiores en aproximadamente 5% en exactitud, 8% en precisión, 8% en sensibilidad y 7% en F1-score.
4. Con el sistema inteligente desarrollado se ha obtenido una herramienta para diagnóstico temprano de la EPOC, empleando un tiempo de cómputo rápido de 4 a 5 segundos aproximadamente por predicción.
5. La arquitectura CNN de doble rama diseñada es más ligera que ResNet20 en 17%, y que InceptionV3 en 73%, con lo cual se ha reducido el uso de recursos computacionales en el entrenamiento.

Recomendaciones

1. Se recomienda utilizar audios respiratorios que tengan una duración mínima de 20 segundos.
2. Es importante asegurar la calidad de los audios, auscultando en una habitación con niveles de ruido mínimos.
3. Es importante realizar pruebas más exhaustivas en entornos médicos reales para evaluar la precisión y fiabilidad del diagnóstico.

Referencias Bibliográficas

Abdullah, A., Ali, M., & Jawad, H. (2019). Evaluation of the Performance of Kirsch and Robinson Methods on Different Color Spaces. *International Journal of Simulation: Systems, Science & Technology*, 1–9. https://doi.org/10.5013/IJSSST.a.20.04.05

Amazon Web Services [AWS]. (2023). *What is overfitting?* AWS. https://aws.amazon.com/es/what-is/overfitting/

American Lung Association. (2023). *Learn about COPD*. American Lung Association. https://www.lung.org/lung-health-diseases/lung-disease-lookup/copd/learn-about-copd

Atici, S., Pan, H., & Cetin, A. E. (2022). *Input Normalized Stochastic Gradient Descent Training of Deep Neural Networks*. http://arxiv.org/abs/2212.09921

Barai, B., Das, D., Das, N., Basu, S., & Nasipuri, M. (2019). VQ/GMM-based speaker identification with emphasis on language dependency. *Advances in Intelligent Systems and Computing*, *883*, 125–141. https://doi.org/10.1007/978-981-13-3702-4_8

Baraniuk, R. (n.d.). *Signals and systems*. LibreTexts. https://eng.libretexts.org/Bookshelves/Electrical_Engineering/Signal_Processing_and_Modeling/Signals_and_Systems_(Baraniuk_et_al.)

Bárcena Rodríguez, M. (2022). Desarrollo y validación de algoritmos de Deep Learning para la clasificación de fundiciones de hierro [Universidad de Cantabria]. In *Repositorio abierto de la Universidad de Cantabria*. https://repositorio.unican.es/xmlui/handle/10902/26246

Barmettler, J. (20 C.E.). *The Full Story behind Convolutional Neural Networks and the Math Behind it*. Hackernoon. https://hackernoon.com/the-full-story-behind-convolutional-neural-networks-and-the-math-behind-it-2j4fk3zu2

Barnard, E., & Cole, R. (1989). A neural-net training program based on conjugate-gradient optimization. In *Technical Report CSE* (Issues 89–014).

Becherer, N., Pecarina, J., Nykl, S., & Hopkinson, K. (2019). Improving optimization of convolutional neural networks through parameter fine-tuning. *Neural Computing and Applications*, *31*(8), 3469–3479. https://doi.org/10.1007/s00521-017-3285-0

Bertrand, F., Segall, D., Sánchez, I., & Bertrand, P. (2020). La auscultación pulmonar en el siglo 21. *Revista Chilena de Pediatria*, *91*(4), 500–506. https://doi.org/10.32641/rchped.v91i4.1465

Braun, S. (2001). Windows. *Encyclopedia of Vibration*, 1587–1595. https://doi.org/10.1006/rwvb.2001.0052

Centers for Disease Control and Prevention. (2022). *Smoking and COPD*. CDC. https://www.cdc.gov/tobacco/campaign/tips/diseases/copd.html

Chacón-Chaves, R. A., Sibaja-Campos, M., Dávila-Haas, J. A., Gutiérrez-Pimentel, R., Gutiérrez-Sanabria, A., Rocha-Contreras, B., & Sánchez-Romero, G. (2003). Enfermedad Pulmonar Obstructiva Crónica (EPOC). *Acta Médica Costarricense*, *45*(1), 23–28. http://www.scielo.sa.cr/scielo.php?script=sci_arttext&pid=S0001-60022003000500003

Chacón Chamorro, M. V. (2023). Estudio de la reducción del sobreajuste en arquitecturas de redes neuronales residuales ResNet en un escenario de clasificación de patrones [Universidad Nacional de Colombia]. In *Repositorio UNAL*. https://repositorio.unal.edu.co/handle/unal/84211

Choi, J. Y., & Rhee, C. K. (2020). Diagnosis and treatment of early chronic obstructive lung disease (COPD). *Journal of Clinical Medicine*, *9*(11), 3426. https://doi.org/10.3390/jcm9113426

Correa, S., Gonzaléz, M., De Betolaza, S., Spiess, C., Perera, P., Algorta, S., & Goñi, M. (2019). Estudio descriptivo de pacientes con EPOC asistidos en medicina interna del Hospital Pasteur de Montevideo: características demográficas y comorbilidades. *Revista Uruguaya de Medicina Interna*, *4*(1), 5–15. https://doi.org/10.26445/04.01.1

D. McCaffrey, J. (2018). *Convolution Image Size, Filter Size, Padding and Stride*. James D. McCaffrey. https://jamesmccaffrey.wordpress.com/2018/05/30/convolution-image-

size-filter-size-padding-and-stride/

Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., & Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems*, *4*(January), 2933–2941. http://arxiv.org/abs/1406.2572

Deep Learning. (2015). NVIDIA Developer. https://developer.nvidia.com/deep-learning

Del Coco, M., Carcagnì, P., Leo, M., Spagnolo, P., Mazzeo, P. L., & Distante, C. (2017). Multi-branch CNN for Multi-scale Age Estimation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10485*, 234–244. https://doi.org/10.1007/978-3-319-68548-9_22

Ding, X., Ding, G., Zhou, X., Guo, Y., Han, J., & Liu, J. (2019). Global Sparse Momentum SGD for Pruning Very Deep Neural Networks. *Advances in Neural Information Processing Systems*, *32*(NeurIPS), 1–13. http://arxiv.org/abs/1909.12778

Edis, T. (2016). *Humanism and Technology* (A. B. Pinn (Ed.)). Springer International Publishing. https://doi.org/10.1007/978-3-319-31714-4

Elsevier. (2020). *EPOC: factores de riesgo y alteraciones anatomopatológicas*. Elsevier. https://www.elsevier.com/es-es/connect/medicina/edu-epoc-factores-de-riesgo-y-alteraciones-anatomopatologicas

Espinoza Villafuerte, N. (2019). Diagnóstico automatizado para la clasificación de café trillado con broca mediante procesamiento de imágenes con Deep Learning [Universidad Peruana Union]. In *Repositorio de Tesis - Universidad Peruana Union*. https://repositorio.upeu.edu.pe/handle/20.500.12840/1961

Flórez, E., Cardona, S., & Jordi, L. (2009). Selección de la ventana temporal en la transformada de Fourier en tiempos cortos utilizada en el análisis de señales de vibración para determinar planos en las ruedas de un tren Selecting. *Revista Facultad de Ingenieria Univ. Antioquia*, *50*, 145–158.

Fraiwan, L., Hassanin, O., Fraiwan, M., Khassawneh, B., Ibnian, A. M., & Alkhodari, M. (2021). Automatic identification of respiratory diseases from stethoscopic lung sound

signals using ensemble classifiers. *Biocybernetics and Biomedical Engineering*, *41*(1), 1–14. https://doi.org/10.1016/j.bbe.2020.11.003

Fraiwan, M., Fraiwan, L., Khassawneh, B., & Ibnian, A. (2021). A dataset of lung sounds recorded from the chest wall using an electronic stethoscope. *Data in Brief*, *35*, 106913. https://doi.org/10.1016/j.dib.2021.106913

Fuchslocher Courard, J. (2021). Desarrollo e implementación de un algoritmo de verificación de rostros en la empresa GeoVictoria [Universidad de Chile]. In *Repositorio académico de la Universidad de Chile*. http://repositorio.uchile.cl/handle/2250/180508

Garbin, C., Zhu, X., & Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, *79*(19–20), 12777–12815. https://doi.org/10.1007/s11042-019-08453-9

Global Initiative for Chronic Obstructive Lung Disease [GOLD]. (2023). *Global strategy for the diagnosis, management, and prevention of chronic obstructive pulmonary disease*.

Gómez Ayala, A. E. (2016). Enfermedad pulmonar obstructiva crónica (EPOC) y alimentación. *Farmacia Profesional*, *30*(1), 26–29. https://www.elsevier.es/es-revista-farmacia-profesional-3-articulo-enfermedad-pulmonar-obstructiva-cronica-epoc--X0213932416474622

Google. (2024). *Guía avanzada de Inception v3*. Google Cloud. https://cloud.google.com/tpu/docs/inception-v3-advanced?hl=es-419

Guha, S., Das, A., Singh, P. K., Ahmadian, A., Senu, N., & Sarkar, R. (2020). Hybrid feature selection method based on harmony search and naked mole-rat algorithms for spoken language identification from audio signals. *IEEE Access*, *8*(March 2021), 182868–182887. https://doi.org/10.1109/ACCESS.2020.3028121

Gupta, D. (2023). *Fundamentals of Deep Learning – Activation Functions and When to Use Them?* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-

them/?utm_source=blog&utm_medium=mathematics-convolutional-neural-network#h-1-binary-step-function

Haji, S. H., & Abdulazeez, A. M. (2021). Comparison of Optimization Techniques Based on Gradient Descent Algorithm: a Review. *PalArch's Journal of Archaeology of Egypt / Egyptology*, *18*(4), 2715–2743. https://archives.palarch.nl/index.php/jae/article/view/6705

Hallén, R. (2017). *A Study of Gradient-Based Algorithms*. https://api.semanticscholar.org/CorpusID:67711237

Hatice Çatal, Dursun Zafer Seker, & Bulent Bayram. (2016). Bone tumor segmentation by gradient operators from CT images. *International Scientific Conference on Applied Sciences*, *September*.

Hazra, R., & Majhi, S. (2020). Detecting respiratory diseases from recorded lung sounds by 2D CNN. *2020 5th International Conference on Computing, Communication and Security (ICCCS)*. https://doi.org/10.1109/ICCCS49678.2020.9277101

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, *2016-Decem*, 770–778. https://doi.org/10.1109/CVPR.2016.90

Ho, Y., & Wookey, S. (2020). The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling. *IEEE Access*, *8*, 4806–4813. https://doi.org/10.1109/ACCESS.2019.2962617

Holt, S. (2021). *Charting Lung Sounds: Normal Findings*. Lecturio Nursing. https://www.lecturio.com/nursing/free-cheat-sheet/charting-lung-sounds-normal-findings/

Huang, J., Chen, B., Yao, B., & He, W. (2019). ECG Arrhythmia Classification Using STFT-Based Spectrogram and Convolutional Neural Network. *IEEE Access*, *7*, 92871–92880. https://doi.org/10.1109/ACCESS.2019.2928017

IBM. (2023). *What is overfitting?* IBM.

Iglesias, G., Talavera, E., González-Prieto, Á., Mozo, A., & Gómez-Canaval, S. (2023). Data

Augmentation techniques in time series domain: A survey and taxonomy. *Neural Computing and Applications*, *35*(14), 10123–10145. https://doi.org/10.1007/s00521-023-08459-3

Intel. (2023). *Convolutional Neural Networks (CNNs), Deep Learning, and Computer Vision*. Intel. https://www.intel.la/content/www/xl/es/internet-of-things/computer-vision/convolutional-neural-networks.html

International Business Machines Corporation (IBM). (2023a). *¿Qué es Deep Learning?* IBM. https://www.ibm.com/es-es/topics/deep-learning

International Business Machines Corporation (IBM). (2023b). *What is gradient descent?* IBM. https://www.ibm.com/topics/gradient-descent#:~:text=Gradient descent is an optimization,each iteration of parameter updates.

Jeon, H., Jung, Y., Lee, S., & Jung, Y. (2020). Area-efficient short-time fourier transform processor for time–frequency analysis of non-stationary signals. *Applied Sciences (Switzerland)*, *10*(20), 1–10. https://doi.org/10.3390/app10207208

Jung, S. Y., Liao, C. H., Wu, Y. S., Yuan, S. M., & Sun, C. T. (2021). Efficiently classifying lung sounds through depthwise separable cnn models with fused stft and mfcc features. *Diagnostics*, *11*(4), 732. https://doi.org/10.3390/DIAGNOSTICS11040732

Khan, Z. A., Hussain, T., Ullah, A., Rho, S., Lee, M., & Baik, S. W. (2020). Towards efficient electricity forecasting in residential and commercial buildings: A novel hybrid CNN with a LSTM-AE based framework. *Sensors*, *20*(5), 1–16. https://doi.org/10.3390/s20051399

Kim, Y., Hyon, Y. K., Jung, S. S., Lee, S., Yoo, G., Chung, C., & Ha, T. (2021). Respiratory sound classification for crackles, wheezes, and rhonchi in the clinical field using deep learning. *Scientific Reports*, *11*(1), 17186. https://doi.org/10.1038/s41598-021-96724-7

Laucho-Contreras, M. E., & Cohen-Todd, M. (2020). Early diagnosis of COPD: Myth or a true perspective. *European Respiratory Review: An Official Journal of the European Respiratory Society*, *29*(158), 200131. https://doi.org/10.1183/16000617.0131-2020

Liberato Bernal, D. E., & Quilcat Pesantes, R. M. (2021). Sistema Informático móvil inteligente para la detección temprana y control de enfermedades respiratorias en pacientes del Sector Privado de Salud en la ciudad de Trujillo [Universidad Privada Antenor Orrego]. In *Repositorio de Tesis - Universidad Privada Antenor Orrego*. https://repositorio.upao.edu.pe/handle/20.500.12759/8368

Lindfield, G., & Penny, J. (2019). Optimization Methods. In *Numerical Methods* (pp. 433–483). Elsevier. https://doi.org/10.1016/B978-0-12-812256-3.00018-X

Liquet, B., Moka, S., & Nazarathy, Y. (2022). *The Mathematical Engineering of Deep Learning*. https://deeplearningmath.org

Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modeling. *ISMIR*, *270*, 11.

Lopez-Betancur, D., Bosco Duran, R., Guerrero-Mendez, C., Zambrano Rodríguez, R., & Saucedo Anaya, T. (2021). Comparación de arquitecturas de redes neuronales convolucionales para el diagnóstico de COVID-19. *Computación y Sistemas*, *25*(3), 601–615. https://doi.org/10.13053/cys-25-3-3453

López-Giraldo, A., Rodriguez-Roisin, R., & Agustí, A. (2014). Enfermedad pulmonar obstructiva crónica. *Medicina Clinica*, *29*(1), 43–48. https://doi.org/10.1157/13076464

Lores Gutiérrez, M. V. (2006). Valoración de la actividad física cotidiana en la enfermeda pulmonar obstructiva crónica meidante un acelerómetro. Relación con parámetros clínicos y funcionales. [UNIVERSIDAD AUTÓNOMA DE MADRID]. In *Fundación Dialnet*. https://dialnet.unirioja.es/servlet/tesis?codigo=2318

Madhu, A., & Kumaraswamy, S. (2019). Data Augmentation Using Generative Adversarial Network for Environmental Sound Classification. *2019 27th European Signal Processing Conference (EUSIPCO)*, 1–5. https://doi.org/10.23919/EUSIPCO.2019.8902819

Maisueche Cuadrado, A. (2019). Utilización del Machine Learning en la industria 4.0. [Universidad de Valladolid]. In *Universidad de Valladolid - Repositorio Documental*. https://uvadoc.uva.es/handle/10324/37908

Martínez Luna, M., Rojas Granados, A., Lázaro Pacheco, R. I., Meza Alvarado, J. E.,

Ubaldo Reyes, L., & Ángeles Castellanos, M. (2020). Enfermedad pulmonar obstructiva crónica (EPOC) Bases para el médico general. *Revista de La Facultad de Medicina*, *63*(3), 28–35. https://doi.org/10.22201/fm.24484865e.2020.63.3.06

Mayo Clinic. (2021). *EPOC*. Mayo Clinic. https://www.mayoclinic.org/es/diseases-conditions/copd/symptoms-causes/syc-20353679

McIlhagga, W. (2018). Estimates of edge detection filters in human vision. *Vision Research*, *153*(September), 30–36. https://doi.org/10.1016/j.visres.2018.09.007

Ministro de Salud [Minsa]. (2023). *Ministro de Salud expuso plan estratégico para fortalecer la atención primaria en regiones*. https://www.gob.pe/institucion/minsa/noticias/790622-ministro-de-salud-expuso-plan-estrategico-para-fortalecer-la-atencion-primaria-en-regiones

Mushtaq, Z., & Su, S.-F. (2020). Environmental sound classification using a regularized deep convolutional neural network with data augmentation. *Applied Acoustics*, *167*(October 2020), 107389. https://doi.org/10.1016/j.apacoust.2020.107389

Myers, E. (2011). Notas de enfermería. In *Mc Graw Hill: Vol. 3era edici*.

Naqvi, S. Z. H., & Choudhry, M. A. (2020). An automated system for classification of chronic obstructive pulmonary disease and pneumonia patients using lung sound analysis. *Sensors*, *20*(22), 6512. https://doi.org/10.3390/s20226512

National Heart Lung and Blood Insitute [NHLBI]. (2022). *COPD*. NHLBI. https://www.nhlbi.nih.gov/health/copd

Olu Ipinlaye, O. (2023). *Filters In Convolutional Neural Networks*. Paper Space. httpsblog.paperspace.comfilters-in-convolutional-neural-networks#images-on-edge

Ortiz de Landaluce, M. P. (2021). Clasificación de imágenes mediante algoritmos de Deep Learning: Mascarillas de COVID-19 [Universidad de Sevilla]. In *idUS - Depósito de Investigación Universidad de Sevilla*. https://idus.us.es/handle/11441/127041

Osgood, B. (2019). Lectures on the Fourier Transform and Its Applications. In *American Mathematical Society*.

Parra-Sánchez, J. S., Oviedo-Carrascal, A. I., & Amaya-Fernández, F. O. (2020). Data

Analytics: incidence of air pollution on public health in Medellin, Colombia. *Revista de Salud Publica*, *22*(6), 609–617. https://doi.org/10.15446/RSAP.V22N6.78985

Pazos Ruiz, F. (2020). *The convolution formula and the central limit theorem*. AWS. https://rstudio-pubs-static.s3.amazonaws.com/704352_9c3df08935a74f2191e22af879b1fd7c.html

Peccia, F. (2018). *Batch normalization: theory and how to use it with Tensorflow*. Medium. https://towardsdatascience.com/batch-normalization-theory-and-how-to-use-it-with-tensorflow-1892ca0173ad

Petmezas, G., Cheimariotis, G. A., Stefanopoulos, L., Rocha, B., Paiva, R. P., Katsaggelos, A. K., & Maglaveras, N. (2022). Automated Lung Sound Classification Using a Hybrid CNN-LSTM Network and Focal Loss Function. *Sensors*, *22*(3), 1232. https://doi.org/10.3390/s22031232

Raj, B. (2018). *A Simple Guide to the Versions of the Inception Network*. Towards Data Science. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

Rao, A. (2023). *What Are Algorithms? A Guide to Algorithms for Children*. Juni. https://junilearning.com/blog/guide/what-are-algorithms/

Rey, L. (2022). Clasificación de tumores renales sólidos a partir de imágenes tomográficas, utilizando algoritmos de Deep Learning [Instituto Tecnológico de Buenos Aires]. In *Repositorio Institucional - Instituto Tecnológico de Buenos Aires (ITBA)*. http://ri.itba.edu.ar/handle/123456789/3771

Rocha, B. M., Filos, D., Mendes, L., Vogiatzis, I., Perantoni, E., Kaimakamis, E., Natsiavas, P., Oliveira, A., Jácome, C., Marques, A., Paiva, R. P., Chouvarda, I., Carvalho, P., & Maglaveras, N. (2018). A Respiratory Sound Database for the Development of Automated Classification. In *Physiological Measurement* (Vol. 40, Issue 035001, pp. 33–37). https://doi.org/10.1007/978-981-10-7419-6_6

Russell, S. (2021). Human-Compatible Artificial Intelligence. In *Human-Like Machine Intelligence* (pp. 3–23). Oxford University Press.

https://doi.org/10.1093/oso/9780198862536.003.0001

Salvi, S., & Barnes, P. J. (2010). Is exposure to biomass smoke the biggest risk factor for COPD globally? *Chest*, *138*(1), 3–6. https://doi.org/10.1378/chest.10-0645

Sarkar, M., Madabhavi, I., Niranjan, N., & Dogra, M. (2015). Auscultation of the respiratory system. *Annals of Thoracic Medicine*, *10*(3), 158–168. https://doi.org/10.4103/1817-1737.160831

Sharma, S. (2017). *Activation Functions in Neural Networks*. Medium. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Sharma, S., Sharma, S., & Athaiya, A. (2020). Activation Functions in Neural Networks. *International Journal of Engineering Applied Sciences and Technology*, *4*(12), 310–316. https://doi.org/10.33564/ijeast.2020.v04i12.054

Shultz, T. R., Fahlman, S. E., Craw, S., Andritsos, P., Tsaparas, P., Silva, R., Drummond, C., Ling, C. X., Sheng, V. S., Drummond, C., Lanzi, P. L., Gama, J., Wiegand, R. P., Sen, P., Namata, G., Bilgic, M., Getoor, L., He, J., Jain, S., … Mueen, A. (2011). Confusion Matrix. In *Encyclopedia of Machine Learning* (pp. 209–209). Springer US. https://doi.org/10.1007/978-0-387-30164-8_157

Singh, A. (2020). *Demystifying the Mathematics Behind Convolutional Neural Networks (CNNs)*. Analytics Vidhya.

Sun, Y., Zhang, W., Gu, H., Liu, C., Hong, S., Xu, W., Yang, J., & Gui, G. (2019). Convolutional Neural Network Based Models for Improving Super-Resolution Imaging. *IEEE Access*, *7*, 43042–43051. https://doi.org/10.1109/ACCESS.2019.2908501

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2818–2826.

https://doi.org/10.1109/CVPR.2016.308

Tak, R. N., Agrawal, D. M., & Patil, H. A. (2017). Novel Phase Encoded Mel Filterbank Energies for Environmental Sound Classification. *Lecture Notes in Computer Science*, *10597*(November), 317–325. https://doi.org/10.1007/978-3-319-69900-4_40

Tejero Caballo, E. (2020). *Aplicaciones de Machine Learning a la bioacústica marina* [Universidad de Salamanca]. https://www.researchgate.net/publication/349536480_Aplicaciones_de_Machine_Learning_a_la_Bioacustica_Marina_Msc_Thesis

Tessema, B. A., Nemomssa, H. D., & Simegn, G. L. (2022). Acquisition and Classification of Lung Sounds for Improving the Efficacy of Auscultation Diagnosis of Pulmonary Diseases. *Medical Devices: Evidence and Research*, *15*, 89–102. https://doi.org/10.2147/MDER.S362407

Tsalera, E., Papadakis, A., & Samarakou, M. (2021). Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning. *Journal of Sensor and Actuator Networks*, *10*(4), 72. https://doi.org/10.3390/jsan10040072

Vargas, R. (2015). Auscultación de Ruidos Respiratorios - Medical League. In *Medical League*. https://www.youtube.com/watch?v=C-Jc_peO3Eg

Wang, S., Jiang, Y., Hou, X., Cheng, H., & Du, S. (2017). Cerebral Micro-Bleed Detection Based on the Convolution Neural Network with Rank Based Average Pooling. *IEEE Access*, *5*, 16576–16583. https://doi.org/10.1109/ACCESS.2017.2736558

Wei, S., Zou, S., Liao, F., & Lang, W. (2020). A Comparison on Data Augmentation Methods Based on Deep Learning for Audio Classification. *Journal of Physics: Conference Series*, *1453*(1), 012085. https://doi.org/10.1088/1742-6596/1453/1/012085

World Health Organization [WHO]. (2021). *World health statistics 2021: Monitoring health for the SDGs, sustainable development goals*.

World Health Organization [WHO]. (2023). *Chronic obstructive pulmonary disease (COPD)*. WHO. https://www.who.int/news-room/fact-sheets/detail/chronic-obstructive-pulmonary-disease-(copd)

Yusnita, M. A., Paulraj, M. P., Yaacob, S., Yusuf, R., & Shahriman, A. B. (2013). Analysis of accent-sensitive words in multi-resolution mel-frequency cepstral coefficients for classification of accents in Malaysian english. *International Journal of Automotive and Mechanical Engineering*, *7*(1), 1053–1073. https://doi.org/10.15282/ijame.7.2012.21.0086

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2021). *Dive Into Deep Learning* (Vol. 1). https://d2l.ai/#

Zhao, H., Gallo, O., Frosio, I., & Kautz, J. (2017). Loss Functions for Image Restoration With Neural Networks. *IEEE Transactions on Computational Imaging*, *3*(1), 47–57. https://doi.org/10.1109/TCI.2016.2644865

Zhou, G., Chen, Y., & Chien, C. (2022). On the analysis of data augmentation methods for spectral imaged based heart sound classification using convolutional neural networks. *BMC Medical Informatics and Decision Making*, *22*(1), 226. https://doi.org/10.1186/s12911-022-01942-2

Anexos

Anexo 1: Matriz de consistencia del proyecto “Implementación de un sistema inteligente aplicando algoritmos de Deep Learning para la detección de Enfermedad Pulmonar Obstructiva Crónica mediante sonidos pulmonares” 1

Anexo 2: Código principal para el entrenamiento de los modelos CNN 2

Anexo 1: Matriz de consistencia del proyecto "Implementación de un sistema inteligente aplicando algoritmos de Deep Learning para la detección de Enfermedad Pulmonar Obstructiva Crónica mediante sonidos pulmonares"

PROBLEMA GENERAL	OBJETIVO GENERAL	HIPÓTESIS GENERAL	VARIALBES
¿De qué manera se implementa un sistema inteligente aplicando algoritmos de Deep Learning que permita detectar la EPOC mediante sonidos pulmonares?	Implementar un sistema inteligente aplicando algoritmos de Deep Learning que permita detectar la EPOC mediante sonidos pulmonares.	Con la implementación de un sistema inteligente aplicando algoritmos de Deep Learning, se detectará la EPOC mediante sonidos pulmonares.	VD. Enfermedad pulmonar obstructiva crónica. VI. Sistema inteligente aplicando algoritmos de Deep Learning.
PROBLEMAS ESPECÍFICOS		**OBJETIVOS ESPECÍFICOS**	**HIPÓTESIS ESPECÍFICAS**
¿Cómo se implementan las arquitecturas de las redes neuronales convolucionales que permitan detectar la EPOC mediante sonidos pulmonares?		Implementar las arquitecturas de redes neuronales convolucionales para detectar la EPOC mediante sonidos pulmonares.	Con la implementación de redes neuronales convolucionales, se detectará la EPOC mediante sonidos pulmonares.
¿De qué modo se implementan las arquitecturas de redes pre entrenadas para detectar la EPOC mediante sonidos pulmonares?		Implementar arquitecturas de redes pre entrenadas para identificar la óptima para detectar la EPOC mediante sonidos pulmonares.	Con la implementación de arquitecturas de redes pre entrenadas, se identificará la óptima para la detección de la EPOC mediante sonidos pulmonares.
¿De qué forma se aplican los algoritmos de optimización que permitan elegir la arquitectura más significativa para detectar la EPOC mediante sonidos pulmonares?		Aplicar algoritmos de optimización que permitan determinar la arquitectura más significativa en términos de precisión para detectar la EPOC mediante sonidos pulmonares.	Con la aplicación de diversos algoritmos de optimización se logra determinar la arquitectura más significativa en términos de precisión.
¿De qué modo se verifica el modelo entrenado para detectar la EPOC mediante sonidos pulmonares y permita confirmar la eficacia del sistema inteligente?		Verificar el modelo entrenado aplicando sonidos pulmonares de pacientes con sospecha de EPOC para confirmar la eficacia del sistema inteligente.	Con la verificación del modelo entrenado aplicando sonidos pulmonares de pacientes con sospecha de EPOC, se confirmará la eficacia del sistema inteligente.

Anexo 2: Código principal para el entrenamiento de los modelos CNN

```python
from Models import *
from params import Params
from Processing import STFT_EXTRACTOR, MEL_EXTRACTOR, MFCC_EXTRACTOR
from sklearn.model_selection import train_test_split
import cv2
from keras.utils import to_categorical

label_numerical = {"EPOC": 0, "Sano": 1, "Otras Condiciones": 2}

if __name__ == "__main__":

    if(Params._feature == 'stft'):
        features, labels = STFT_EXTRACTOR.load_STFT(path_images_stft=Params.stft_features_path)
    elif(Params._feature == 'mel'):
        features, labels = MEL_EXTRACTOR.load_MEL(path_images_mel=Params.mel_features_path)
    elif(Params._feature == 'mfcc'):
        features, labels = MFCC_EXTRACTOR.load_MFCC(path_images_mfcc=Params.mfcc_features_path)
    else:
        print("Características no existentes, prueba con stft, mel o mfcc")

    for i in range(len(labels)):
        labels[i] = label_numerical[labels[i]]

    features      = np.array(features)
    y_categorical = np.array(labels)

    if(Params._model == 'Inception'):
        features = [cv2.resize(img, (75, 626)) for img in features] # Ajuste del tamaño de las entradas si se usa Inception
        features = np.array(features)

    X_train, X_val, y_train, y_val = train_test_split(features, y_categorical, test_size = 0.40, random_state = Params._Seed_value, stratify=y_categorical)
    X_val, X_test, y_val, y_test   = train_test_split(X_val, y_val, test_size = 0.50, random_state = Params._Seed_value, stratify=y_val)

    y_train = to_categorical(y_train, num_classes=3)
    y_test = to_categorical(y_test, num_classes=3)
    y_val = to_categorical(y_val, num_classes=3)

    My_model   (X_train, X_val, y_train, y_val,
                learning_rate = Params._learning_rate_0,
                metric_condi= Params._metric_condition,
                metric_monit=Params._metric_monitoring,
                lr_decay_factor=Params._lr_decay_factor,
                minimum_lr=Params._minimum_lr,
                batch_size = Params._batch_size,
                epochs = Params._epochs,
                dropout = Params.Dropout_value,
                optimizer = Params._optimizer,
                loss = Params._loss,
                name_model = Params._name_best_model,
                name_history = Params._name_best_model_hist,
                model_path = Params.best_model_path)
```