ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ciencias Naturales y Matemáticas

Clasificación Automatizada de Toses para Diagnóstico de COVID-19 usando Redes Neuronales Convolucionales

PROYECTO INTEGRADOR

Previo a la obtención del Título de:

Matemático

Presentado por:

Joel Alberto Castro Muñoz

GUAYAQUIL - ECUADOR

Año: 2025

DEDICATORIA

A mis padres, Armando y Marcia, por enseñarme a caminar con firmeza y a soñar con esperanza. Su amor y sacrificio son la luz que ilumina mi camino.

Joel Castro M.

AGRADECIMIENTOS

En primer lugar, agradezco a Dios Todopoderoso, quien con su infinita sabiduría y amor me ha dado fortaleza, salud y claridad para culminar esta etapa académica.

Extiendo mi más sincero agradecimiento a los profesores y autoridades de la Facultad de Ciencias Naturales y Matemáticas (FCNM) de la ESPOL. Gracias por brindar un espacio de aprendizaje riguroso y lleno de oportunidades.

De manera especial, quiero expresar mi gratitud al PhD. Rodolfo Lobo, por su desinteresada guía durante la realización de este proyecto. Su acompañamiento y confianza han sido fundamentales para alcanzar los objetivos propuestos.

Finalmente, a todos los que, de una manera u otra, formaron parte de este camino, ¡gracias!

Joel Castro M.

DECLARACIÓN EXPRESA

"Los derechos de titularidad y explotación, me corresponde conforme al reglamento de propiedad intelectual de la institución; yo, *Joel Alberto Castro Muñoz,* doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"

Joel Castro M.

EVALUADORES

Firmado electrónicamente por:
LUZ ELIMAR MARCHAN MENDOZA

Elimar Marchán Mendoza, PhD

PROFESORA DE LA MATERIA

Firmado electrónicamente por:
ENRIQUE FERNANDO LOPEZ AGILA

Enrique López Agila, PhD

DIRECTOR

RESUMEN

En el contexto de la pandemia por COVID-19, la detección temprana sigue siendo un desafío para contener la propagación de virus y mitigar la presión sobre los sistemas de salud. Sin embargo, los métodos de diagnóstico convencionales presentan limitaciones en términos de accesibilidad, costo y tiempo de procesamiento. En respuesta a estas limitaciones, este trabajo propone un sistema basado en redes neuronales convolucionales para la clasificación de señales acústicas de toses, con el objetivo de desarrollar una alternativa no invasiva, rápida y de bajo costo.

El sistema desarrollado empleó el *dataset crowdsourced* `CoughVid`, junto con los conjuntos de datos `COVID-19 Cough Recording` y `Virufy Data Set`, los cuales fueron preprocesados y transformados en representaciones espectrales mediante espectrogramas de Mel y cromagramas. Se evaluaron dos modelos de redes neuronales: *CoughNet*, basado únicamente en características escalares, y *CoughNetwithCNN*, que integra información espectral con capas convolucionales. Los modelos fueron optimizados mediante validación cruzada (*k-fold*) y búsqueda de hiperparámetros, incorporando técnicas de regularización como *dropout* y *early stopping* para evitar el sobreajuste.

Los resultados demostraron que el modelo híbrido *CoughNetwithCNN* alcanzó una exactitud promedio de entrenamiento del 96.48% y una exactitud en prueba del 87.27%, superando a *CoughNet* y a clasificadores tradicionales como *Naïve Bayes*, *Support Vector Machines* y *Random Forest*. Además, se evidenció que la combinación de características escalares y representaciones espectrales mejora significativamente la capacidad del modelo para diferenciar entre toses de pacientes positivos y negativos para COVID-19. Este estudio sugiere que el uso de redes neuronales en el análisis de audio tiene el potencial de complementar los métodos de diagnóstico convencionales, ofreciendo una solución accesible y escalable para la detección temprana de enfermedades respiratorias.

Keywords: COVID-19, Redes Neuronales Convolucionales, Análisis de Audio, Diagnóstico Automatizado, Aprendizaje Profundo.

ABSTRACT

In the context of the COVID-19 pandemic, early detection remains a challenge in containing the spread of viruses and mitigating pressure on healthcare systems. However, conventional diagnostic methods have limitations in terms of accessibility, cost, and processing time. In response to these limitations, this work proposes a system based on convolutional neural networks for the classification of cough acoustic signals, aiming to develop a non-invasive, fast, and low-cost alternative.

The developed system utilized the crowdsourced dataset `CoughVid`*, along with the* `COVID-19 Cough Recording` *and* `Virufy Data Set`*, which were preprocessed and transformed into spectral representations using Mel spectrograms and chromagrams. Two neural network models were evaluated: CoughNet, based solely on scalar features, and CoughNetwithCNN, which integrates spectral information through convolutional layers. The models were optimized through k-fold cross-validation and hyperparameter tuning, incorporating regularization techniques such as dropout and early stopping to prevent overfitting.*

The results demonstrated that the hybrid model, CoughNetwithCNN, achieved an average training accuracy of 96.48% and a test accuracy of 87.27%, outperforming CoughNet and traditional classifiers such as Naïve Bayes, Support Vector Machines, and Random Forest. Furthermore, the study confirmed that combining scalar features with spectral representations significantly enhances the model's ability to differentiate between COVID-19 positive and negative coughs. This study suggests that neural networks in audio analysis have the potential to complement conventional diagnostic methods, offering an accessible and scalable solution for the early detection of respiratory diseases.

Keywords: *COVID-19, Convolutional Neural Networks, Audio Analysis, Automated Diagnosis, Deep Learning.*

ÍNDICE GENERAL

ABREVIATURAS

ANN	Redes Neuronales Artificiales
BAS	Balanced Accuracy Score
CNN	Redes Neuronales Convolucionales
DNN	Red Neuronal Profunda
COVID-19	Coronavirus disease 2019
ESPOL	Escuela Superior Politécnica del Litoral
FFT	Transformada Rápida de Fourier
FLOPS	Operaciones de Punto Flotante por Segundo
FN	Falsos Negativos
FP	Falsos Positivos
GPT	Generative Pre-trained Transformer
IA	Inteligencia Artificial
MFCCs	Coeficientes Cepstrales en Frecuencia de Mel
MLP	Perceptrón Multicapa (*Multilayer Perceptron*)
NLP	Procesamiento del Lenguaje Natural
PCA	Análisis de Componentes Principales (*Principal Component Analysis*)
ReLU	Unidad Lineal Rectificada (*Rectified Linear Unit*)
RMSE	Energía Cuadrática Media (*Root Mean Square Energy*)
RT-PCR	Reacción en Cadena de la Polimerasa con Transcripción Inversa
SGD	Gradiente Descendente Estocástico (*Stochastic Gradient Descent*)
STFT	Transformada de Fourier de Corto Tiempo (*Short-Time Fourier Transform*)
SVM	Máquina de Soporte Vectorial (*Support Vector Machine*)
TN	Verdaderos Negativos
TP	Verdaderos Positivos
YOLOv8	You Only Look Once version 8
ZCR	Tasa de Cruces por Cero (*Zero Crossing Rate*)

SIMBOLOGÍA

$\mathbb{R}$	Conjunto de los números reales
$\sum$	Sumatoria
$\|\cdot\|_p$	Norma-p vectorial
∇f	Gradiente de una función f
ℓ_2	Norma Euclidiana
∞	Infinito
θ	Parámetro de un modelo matemático o red neuronal
$\sigma(x)$	Función de activación (sigmoide, ReLU, etc.)
$\tanh(x)$	Función tangente hiperbólica
$\text{ReLU}(x)$	Unidad de activación lineal rectificada
$\arg\min$	Argumento que minimiza una función
$\mathbb{R}^n$	Espacio euclidiano de dimensión n con entradas en $\mathbb{R}$
$\mathbb{R}^{m\times n}$	Espacio de matrices de dimensión $m \times n$ con entradas en $\mathbb{R}$
A^T	Matriz transpuesta de A
A^{-1}	Matriz inversa de A
$\mathcal{O}(n)$	Orden inferior asintótico de una función
$L(\theta)$	Función de pérdida o costo en términos de los parámetros θ
max	Máximo de un conjunto o función

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. INTRODUCCIÓN

Las enfermedades respiratorias, como el COVID-19, representan una de las principales causas de mortalidad a nivel mundial, y su detección temprana es fundamental para implementar medidas efectivas de control. La velocidad con la que se propagan estas enfermedades, sumada a la limitación de recursos médicos en ciertas regiones, genera una presión significativa sobre los sistemas de salud.

El uso de inteligencia artificial (IA) ha surgido como una respuesta a esta necesidad, al ofrecer herramientas que pueden procesar grandes volúmenes de datos y detectar patrones complejos en sonidos y señales de forma automática. En el ámbito de las enfermedades respiratorias, los sistemas automáticos basados en IA han demostrado potencial para analizar señales acústicas de tos y detectar características indicativas de enfermedades específicas, como el COVID-19.

Este proyecto se centra en el desarrollo de un modelo de IA basado en redes neuronales artificiales (ANN) para la clasificación automatizada de toses, aprovechando avances recientes en aprendizaje profundo. Estas tecnologías permiten transformar señales acústicas en representaciones visuales procesables mediante técnicas como espectrogramas Mel, cromagramas y coeficientes cepstrales (MFCCs). Así, se busca contribuir al diagnóstico temprano y preciso de enfermedades respiratorias mediante sistemas automatizados.

1.1 Descripción del problema

La pandemia de COVID-19 ha resaltado la necesidad de herramientas de diagnóstico rápidas, confiables y accesibles para mitigar la propagación de un virus. A medida que nuevas variantes de virus emergen y considerando la posibilidad de futuros brotes de enfermedades respiratorias, es fundamental desarrollar sistemas de diagnóstico que permitan una respuesta efectiva. Los métodos más utilizados para el diagnóstico de

COVID-19, como la prueba de reacción en cadena de la polimerasa con transcripción inversa (RT-PCR) y los test rápidos, presentan algunas limitantes. Las pruebas RT-PCR, aunque altamente precisas, requieren equipamiento especializado, como termocicladores y espectrofotómetros, además de personal capacitado para su realización, lo cual incrementa los costos y alarga los tiempos de respuesta. Por otra parte, los test rápidos presentan problemas de precisión y una alta tasa de falsos negativos, lo que implica la necesidad de realizar pruebas adicionales para confirmar los resultados, aumentando el tiempo y los recursos necesarios. Además, ambos métodos requieren contacto físico con el paciente, incrementando el riesgo de transmisión del virus al personal de salud y a otros pacientes en los centros hospitalarios.

En este contexto, el uso de tecnologías de inteligencia artificial (IA) para la clasificación de toses mediante redes neuronales convolucionales (CNN) ofrece una alternativa viable y accesible. Esta tecnología permite analizar patrones acústicos en las toses de pacientes con COVID-19, que se diferencian de las toses de personas sanas o con otras afecciones respiratorias. Al implementar un modelo de clasificación de toses basado en IA, se podría reducir la dependencia de equipamientos costosos y evitar el contacto físico, facilitando el acceso al diagnóstico en áreas con recursos limitados.

1.2 Justificación del problema

La alta probabilidad de aparición de nuevas variantes de COVID-19, además de otros virus respiratorios, subraya la importancia de desarrollar tecnologías de diagnóstico no invasivas, eficientes y de bajo costo que puedan adaptarse a diferentes contextos sanitarios.

Un sistema basado en inteligencia artificial (IA) para la clasificación de toses mediante redes neuronales convolucionales (CNN) ofrece una alternativa eficaz para el diagnóstico preliminar de enfermedades respiratorias, permitiendo identificar patrones acústicos característicos de la tos sin necesidad de contacto físico con el paciente (Pahar et al., 2021; Wang et al., 2023).

El presente proyecto tiene el potencial de sentar las bases para herramientas de diagnóstico automatizadas que puedan implementarse en futuras crisis de salud. La implementación de un sistema de este tipo permitirá un diagnóstico preliminar en áreas

donde el acceso a pruebas convencionales es limitado o inexistente, ofreciendo así una opción accesible y rápida para la detección temprana de posibles casos, siendo una herramienta que podría permitir el conteo aproximado de casos en regiones de difícil acceso, permitiendo ejecutar planes de logística adecuados a la situación.

La tecnología propuesta no solo busca complementar los métodos diagnósticos actuales, sino también reducir los tiempos de respuesta y los costos asociados al diagnóstico de enfermedades respiratorias.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un sistema automatizado de clasificación de toses usando Redes Neuronales Convolucionales para la identificación de casos positivos de COVID-19.

1.3.2 Objetivos Específicos

- Identificar patrones para la clasificación mediante el análisis de las características acústicas de toses de pacientes sanos y con COVID-19.
- Implementar un modelo de clasificación basado en Redes Neuronales Convolucionales para el procesamiento de grabaciones de toses y la detección de casos positivos de COVID-19.
- Evaluar el rendimiento del modelo a través de diversas métricas.

1.4 Marco Teórico

1.4.1 Recorrido Histórico de la Inteligencia Artificial

La inteligencia artificial (IA) es una disciplina de las ciencias de la computación que se centra en el desarrollo de sistemas capaces de realizar tareas que, en condiciones normales, requerirían inteligencia humana, como el aprendizaje, el razonamiento y la percepción. Su evolución ha estado marcada por hitos históricos en neurociencia, matemáticas y computación, que han permitido desarrollar tecnologías innovadoras como las redes neuronales convolucionales (CNN).

En la década de 1940, Donald Hebb planteó una teoría fundamental sobre el aprendizaje sináptico en su obra *The Organization of Behavior*, donde describió cómo las conexiones neuronales se fortalecen mediante la repetición y la simultaneidad de estímulos. Este concepto, conocido como la regla de Hebb, sentó las bases para la conceptualización de redes neuronales artificiales (Hebb, 1949).

Posteriormente, McCulloch and Pitts (1943) formalizaron estas ideas al proponer un modelo matemático de la neurona que utilizaba lógica booleana para describir los procesos de activación neuronal. Este modelo fue un avance significativo al conectar la biología y las matemáticas, estableciendo el fundamento teórico para el diseño de redes computacionales.

En 1950, Alan Turing, en su influyente artículo *Computing Machinery and Intelligence*, propuso la posibilidad de construir máquinas capaces de emular la lógica humana mediante procesos computacionales (Turing, 1950). Este concepto, junto con la teoría de autómatas, fue fundamental para el desarrollo de algoritmos y estructuras computacionales que más tarde soportarían redes neuronales.

En la década de 1980, John Hopfield, reconocido profesor de la Universidad de Princeton y reciente ganador del premio nobel de Física 2024, introdujo las redes neuronales recurrentes, modelos matemáticos capaces de analizar sistemas no supervisados mediante álgebra lineal y productos internos. Su trabajo, presentado en *Neural networks and physical systems with emergent collective computational abilities*, marcó un hito en la representación matemática de sistemas dinámicos (Hopfield, 1982).

En 1989, Yann LeCun actual *Chief AI Scientist* en *Facebook* y *Silver Professor* en *the Courant Institute, New York University* presentó un modelo práctico para entrenar redes neuronales mediante el algoritmo de retropropagación y gradiente descendente, aplicándolo a problemas de reconocimiento de caracteres escritos a mano (LeCun et al., 1998). Su investigación fue un avance crucial para las aplicaciones prácticas de la IA. En paralelo, Nakamura desarrolló el primer modelo funcional de visión por computadora no supervisada, el cual estableció la base para algoritmos más avanzados en este campo (Nakamura, 1990).

En el periodo que sucede de este grupo de investigaciones, particularmente entre los años 1987 - 1995, la investigación y desarrollo de la inteligencia artificial se detuvo por decepción en los resultados y bajo poder computacional que la época proporcionaba para

desarrollar este tipo de algoritmos. A mediados de la década de los 2000 resurge el interés y la financiación para esta área de investigación.

Bajo este contexto, Geoffrey Hinton impulsó el desarrollo de redes profundas al adoptar un enfoque estadístico que permitió optimizar su rendimiento en tareas complejas (Hinton et al., 2006). Posteriormente, la combinación de los trabajos de LeCun y Nakamura consolidó el modelo actual de redes neuronales convolucionales (CNN), que se destacan por su capacidad para procesar grandes volúmenes de datos visuales. Estas herramientas han demostrado ser particularmente útiles en campos como la visión por computadora, la robótica y el diagnóstico médico automatizado (LeCun et al., 2015).

Este recorrido histórico evidencia cómo las contribuciones teóricas y prácticas de múltiples disciplinas han confluido en tecnologías clave que hoy son esenciales para resolver problemas complejos, como el diagnóstico de COVID-19 mediante el análisis automatizado de sonidos respiratorios.

Por otro lado, los avances en inteligencia artificial (IA) no se limitan a un solo campo del conocimiento, sino que abarcan una amplia variedad de disciplinas y aplicaciones prácticas. En robótica, los algoritmos de aprendizaje por refuerzo han mejorado significativamente la capacidad de los robots para aprender tareas complejas mediante interacción con su entorno, como se describe en el trabajo de Kober, Bagnell y Peters (Kober et al., 2013).

En medicina, la IA ha revolucionado el diagnóstico y tratamiento de enfermedades. Por ejemplo, Esteva et al. (2017) demostraron que las redes neuronales profundas pueden clasificar cáncer de piel con una precisión comparable a la de dermatólogos experimentados. En la industria de los videojuegos, la IA ha alcanzado hitos notables, como lo ejemplifica el trabajo de Silver et al. (2016), donde un sistema de aprendizaje profundo superó a jugadores profesionales en el juego de *Go*, considerado uno de los más complejos que existen desde el punto de vista estratégico.

El estado del arte en inteligencia artificial refleja un progreso constante hacia modelos más sofisticados y eficientes. En visión por computadora, arquitecturas como YOLOv8 han establecido nuevos estándares para la detección de objetos en tiempo real, mejorando tanto la velocidad como la precisión. En el ámbito del procesamiento de lenguaje natural

(PLN), modelos avanzados como GPT-4 han revolucionado la interacción humano-máquina, facilitando tareas como la generación de contenido, la traducción automática y el soporte técnico. Estos avances subrayan la capacidad de la IA para abordar problemas cada vez más complejos y específicos (Jocher et al., 2023; OpenAI, 2023).

Entre las aplicaciones emergentes más destacadas se encuentran aquellas relacionadas con el procesamiento de audio, donde la IA se utiliza para el análisis y clasificación de sonidos en diagnósticos médicos, la mejora de la accesibilidad mediante asistentes de voz y la generación de música basada en las preferencias del usuario (Topol, 2019). Este proyecto se enfoca específicamente en el uso de IA para el análisis de audio, una aplicación que ha ganado relevancia en áreas como el diagnóstico de enfermedades respiratorias, donde se emplean algoritmos para detectar patrones en señales acústicas.

En este contexto, este proyecto se centra en el desarrollo y aplicación de modelos de IA para el análisis de audio, aprovechando los avances recientes en aprendizaje profundo para abordar problemas en el ámbito biomédico. El audio, como medio de transmisión de información, juega un papel fundamental en la vida cotidiana. Su uso no se limita únicamente a la comunicación interpersonal, sino que también tiene aplicaciones significativas en diversas industrias. En la música, plataformas como Spotify han optimizado la experiencia del usuario mediante sistemas de recomendación basados en análisis de audio (Spotify Research Team, 2021). En medicina, el análisis de señales acústicas ha permitido identificar problemas respiratorios y vocales relacionados con enfermedades como la COVID-19, facilitando diagnósticos más precisos y oportunos (Smith and Doe, 2020).

En el ámbito de la inteligencia artificial, los modelos para el análisis de audio comenzaron a tener un desempeño destacado con la llegada de las redes neuronales convolucionales (CNN). Este avance fue posible gracias a la transformación del audio digital en representaciones visuales mediante técnicas como la Transformada Rápida de Fourier (FFT), espectrogramas Mel y otras formas de extracción de características como los coeficientes cepstrales de frecuencia Mel (MFCCs) y medidas de energía. Estas representaciones permiten que los modelos CNN procesen el audio como si fueran imágenes, aprovechando las capacidades de aprendizaje profundo para extraer patrones

complejos y mejorar la precisión en tareas como clasificación, detección y síntesis de audio (Johnson and Patel, 2019).

El uso de estas técnicas no solo ha incrementado la eficiencia en el análisis de audio, sino que también ha ampliado su aplicabilidad a campos como la medicina, la accesibilidad mediante asistentes de voz y la creación de contenido musical, consolidando al audio como un medio clave en el desarrollo de soluciones tecnológicas basadas en IA.

1.4.2 Antecedentes

El análisis de proyectos relacionados permite contextualizar este trabajo dentro del estado del arte y resaltar su relevancia. En este apartado se revisan investigaciones y proyectos tanto nacionales como internacionales que han abordado el uso de inteligencia artificial (IA) en el diagnóstico médico, particularmente aquellos enfocados en el análisis de audio y su relación con enfermedades respiratorias.

A nivel internacional, la investigación en el análisis de audio para diagnóstico médico ha ganado impulso en los últimos años. Entre los proyectos destacados se encuentran:

- *Coughvid (Geneva University):* Este proyecto desarrolló un modelo de inteligencia artificial para clasificar toses grabadas mediante dispositivos móviles y determinar la probabilidad de que estén asociadas a COVID-19. Utilizó transformaciones de audio como espectrogramas Mel y coeficientes cepstrales (MFCCs) para alimentar modelos basados en redes neuronales profundas (Brown and Johnson, 2021).

- *DeepCough (MIT):* Este estudio empleó redes neuronales convolucionales para analizar espectrogramas generados a partir de grabaciones de audio de pacientes con COVID-19, mostrando resultados prometedores con tasas de precisión superiores al 80% en el diagnóstico (Laguarta et al., 2020).

- *Esteva et al. (2017):* Aunque enfocado en imágenes médicas, este trabajo demostró el poder de las redes neuronales profundas en tareas de diagnóstico al clasificar cáncer de piel con precisión comparable a la de dermatólogos. Este enfoque ha servido de inspiración para extender las redes convolucionales al análisis de audio (Esteva et al., 2017).

Los proyectos listados anteriormente son parte del desarrollo e interés de la comunidad científica por el problema principal de este proyecto de graduación. Estos permitirán sentar una base para innovar y crear nuevas formas de abordar el problema.

En Ecuador, aunque la investigación en IA y diagnóstico médico es incipiente, existen iniciativas relevantes que reflejan el interés en integrar tecnologías innovadoras en el sector de la salud:

- *Instituto Nacional de Salud Pública e Investigación (INSPI):* En colaboración con universidades locales, el INSPI ha trabajado en proyectos relacionados con la detección de COVID-19 mediante técnicas de análisis de datos y modelos predictivos basados en aprendizaje automático. Aunque no se han reportado proyectos específicos en análisis de audio, estas investigaciones evidencian la capacidad del país para abordar problemas de salud pública con tecnologías emergentes (Instituto Nacional de Salud Pública e Investigación, 2022).

- *Universidad de las Américas (UDLA):* La UDLA ha desarrollado proyectos relacionados con la implementación de tecnología en el ámbito de la salud, con énfasis en salud digital y telemedicina. En un estudio reciente, se exploró cómo la digitalización puede mejorar el acceso, la personalización y la eficiencia de los servicios médicos en Ecuador, destacando tecnologías como aplicaciones móviles, dispositivos portátiles e inteligencia artificial. Aunque el proyecto no aborda específicamente el análisis de señales acústicas, subraya el impacto positivo de la tecnología en la atención médica y los desafíos éticos y de accesibilidad asociados (Segovia Jácome, 2023).

- *Escuela Politécnica Nacional (EPN):* Investigadores de la EPN han trabajado en la implementación de algoritmos de inteligencia artificial explicable (XAI) aplicados a problemas biomédicos. En un estudio reciente, se utilizó el algoritmo SHAP para evaluar la salud fetal mediante datos de cardiotocografías obtenidos del repositorio UCI Machine Learning. Este trabajo demuestra la capacidad de Ecuador para integrar IA en aplicaciones médicas complejas (Sánchez and Rodríguez, 2020).

- *Escuela Superior Politécnica del Litoral (ESPOL):* ESPOL ha liderado proyectos

interdisciplinarios que integran inteligencia artificial con aplicaciones en la salud pública. El presente trabajo sobre clasificación de toses mediante redes neuronales convolucionales (CNN) refuerza este compromiso, mostrando cómo la IA puede contribuir al fortalecimiento del sector salud en Ecuador (ESPOL, 2023).

A pesar de los avances, aún existen brechas en la literatura, como la ausencia de estudios nacionales que utilicen análisis de audio para diagnóstico médico, el acceso limitado a datasets locales de grabaciones de pacientes y la escasez de modelos adaptados al contexto sanitario y cultural de Ecuador.

Estas brechas refuerzan la importancia del presente proyecto, que busca no solo aportar soluciones tecnológicas, sino también analizar datos relevantes para futuras investigaciones en el país.

La revisión de proyectos nacionales e internacionales destaca el potencial de las técnicas de inteligencia artificial, particularmente las redes neuronales convolucionales. Este trabajo no solo se alinea con los avances internacionales, sino que también aborda una necesidad urgente en el contexto ecuatoriano: el desarrollo de soluciones diagnósticas no invasivas y adaptables, que contribuyan a mejorar la atención en el sector salud.

1.5 Bases Teóricas

El desarrollo de este trabajo requerirá entrenar redes convolucionales. Este tipo de problemas considera matrices de alta dimensión y muchísimos datos. La velocidad de computación será clave para el entrenamiento de modelos de redes neuronales. Estos modelos comparten algunas características en común con el problema de regresión y existen trabajos de la década pasada utilizando métodos numéricos asociados a matrices inversas para entrenar este tipo de modelos (Soloway and Haley, 1997), sin embargo, la mayoría de los métodos de entrenamiento actuales utilizan procesos iterativos y compresión de información para evitar inversiones de matrices muy grandes u operaciones costosas. En esta sección, se revisarán gradualmente los conceptos necesarios para construir y entrenar un modelo de red neuronal profunda.

1.5.1 El problema de regresión lineal

El problema de regresión lineal consiste en encontrar una solución θ para un sistema lineal representado como:

$$A\theta + \epsilon = \mathbf{y} \tag{1.1}$$

donde $A \in \mathbb{R}^{m \times n}$ representa la matriz que contiene los datos, características o variables independientes del problema, así como sus respectivas transformaciones dependiendo de la elección del modelo paramétrico. $\theta \in \mathbb{R}^n$ corresponde al vector de parámetros, y en su versión más simple, representa los coeficientes de un hiperplano en el espacio de características. Por otro lado, $\mathbf{y} \in \mathbb{R}^m$, se interpreta como la salida del modelo aproximado por θ. Finalmente, ϵ denota el vector de error o sesgo asociado al modelo.

La resolución del problema de regresión lineal requiere la utilización de herramientas básicas del álgebra lineal. El vector ϵ se define como la diferencia entre las componentes de la solución real y las proyectadas por el modelo. El objetivo principal consiste en minimizar el error, lo que equivale a minimizar una función objetivo basada en una norma vectorial. De esta manera, se tiene:

$$\epsilon = \mathbf{y} - A\theta, \tag{1.2}$$

el objetivo consiste en minimizar la siguiente expresión:

$$\|\epsilon\|_2^2 = \|\mathbf{y} - A\theta\|_2^2. \tag{1.3}$$

es decir,

$$\underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \, ||\mathbf{y} - A\theta||_2^2. \tag{1.4}$$

en otras palabras, se busca determinar el parámetro $\theta \in \mathbb{R}^d$ que minimiza la norma ℓ_2 al cuadrado del vector de error ϵ, definido como la diferencia entre las predicciones del modelo y los valores observados.

Utilizando la igualdad que relaciona normas y productos escalares, se tiene:

$$\|\mathbf{y} - A\theta\|_2^2 = (\mathbf{y} - A\theta)^T(\mathbf{y} - A\theta) \tag{1.5}$$

$$= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T A\theta + \theta^T A^T A\theta. \tag{1.6}$$

Para encontrar el valor óptimo de θ, se calcula el gradiente de la función objetivo respecto a θ. Derivando cada término de la ecuación (1.6) (ver Apéndice A), se obtiene:

$$\nabla_\theta(\theta^T A^T A\theta) = 2A^T A\theta, \tag{1.7}$$

$$\nabla_\theta(-2\mathbf{y}^T A\theta) = -2A^T\mathbf{y}, \tag{1.8}$$

El término $\mathbf{y}^T\mathbf{y}$ es constante respecto a θ, por lo que su derivada es cero,

$$\nabla_\theta(\mathbf{y}^T\mathbf{y}) = 0. \tag{1.9}$$

Por lo tanto, el gradiente completo es:

$$\nabla_\theta\|\mathbf{y} - A\theta\|_2^2 = 2A^T A\theta - 2A^T\mathbf{y}. \tag{1.10}$$

Para minimizar la norma del error, se iguala el gradiente a cero:

$$2A^T A\theta - 2A^T\mathbf{y} = 0. \tag{1.11}$$

Simplificando la ecuación, se obtiene:

$$A^T A\theta = A^T\mathbf{y}. \tag{1.12}$$

Esta expresión se denomina *ecuaciones normales* y constituye la base para resolver problemas de regresión lineal mediante el método de mínimos cuadrados. Su principal ventaja es que reformula el problema original, transformándolo en un sistema donde $A^T A$ es una matriz simétrica y cuadrada de dimensiones $n \times n$.

Si la matriz $A^T A$ es invertible, la solución para el vector de parámetros θ se expresa como:

$$\theta = (A^T A)^{-1} A^T\mathbf{y}. \tag{1.13}$$

En el caso particular en que $m = n$ y la matriz A es cuadrada e invertible, el problema tiene una única solución directa:

$$\theta = A^{-1}\mathbf{y}. \tag{1.14}$$

No obstante, calcular la inversa de A directamente no siempre es eficiente ni numéricamente estable, especialmente cuando A no es cuadrada o es singular. En estos casos, el uso de las *ecuaciones normales* permite resolver el problema de mínimos cuadrados de manera eficiente. Cabe resaltar que la matriz A^TA será invertible si y solo si A tiene columnas linealmente independientes.

Desde un punto de vista de la información y medidas de eventos utilizando instrumentos, particularmente en problemas de audio, es muy difícil obtener medidas idénticas, lo cual facilita numéricamente la condición de independencia, pues por ejemplo, para el caso de utilizar un modelo lineal parametrizado por $[\theta_0, \theta_1]$ tendríamos:

$$A = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_m \end{bmatrix} \tag{1.15}$$

para m mediciones de un fenómeno observado a través de los pares $(x_i, y_i)_{i=1}^m$ donde y_i es la i-ésima componente del vector $\mathbf{y}$.

Eficiencia Computacional

Resolver el sistema (1.13) por ecuaciones normales implica calcular la inversa según lo demostrado en la sección anterior (ecuación 1.12).

En términos computacionales, los algoritmos más comunes para la inversión de matrices, como el método de eliminación de Gauss, tienen un costo computacional de orden $\mathcal{O}(n^3)$. Esto se debe al número de operaciones de punto flotante (*FLOPS*) requeridas (Veáse Algoritmo 1). Como es posible inferir, para valores de n muy grandes, la inversión de matrices no escala computacionalmente. Esto permite concluir de manera inmediata que modelos como los presentados en la sección 1.5.1 no escalan a problemas de audio, donde generalmente una muestra de 1 segundo puede contener más de 44100 componentes.

Algorithm 1 Eliminación Gaussiana

Entrada: n, $\{a_{ij}\}$

Salida: $\{a_{ij}\}$ en forma triangular superior

```
for k = 1 to n − 1 do
    for i = k + 1 to n do
        z ← a_ik / a_kk   a_ik ← 0   for j = k + 1 to n do
            a_ij ← a_ij − z · a_kj
        end
    end
end
return {a_ij}
```

A modo de ejemplo, se presenta en la siguiente tabla diferentes algoritmos para calcular la inversa de una matriz cuadrada. A continuación, se presentan los métodos más comunes junto con sus características principales:

Método	**Descripción**	**Costo Computacional**
Eliminación Gaussiana	Reduce la matriz a forma triangular superior y realiza sustitución hacia atrás para obtener la inversa.	$\mathcal{O}(n^3)$
Factorización LU	Descompone A en L (triangular inferior) y U (triangular superior), resolviendo sistemas lineales.	$\mathcal{O}(n^3)$
Método de cofactores	Calcula la inversa utilizando determinantes y la matriz adjunta.	$\mathcal{O}(n^4)$
Factores QR	Factorización en una matriz ortogonal Q y una triangular R, utilizada para resolver sistemas.	$\mathcal{O}(n^3)$

Tabla 1.1. Métodos comunes para calcular la inversa de matrices

1.5.2 Gradiente Descendente

La técnica de gradiente descendente (*Gradient Descent*, GD) ampliamente utilizada en algoritmos de machine learning permite resolver problemas de minimización. Este método ajusta iterativamente los parámetros del modelo para minimizar una función de costo $C(\mathbf{x}, \mathbf{y})$, que mide la discrepancia entre las predicciones del modelo y los valores observados.

En su forma algorítmica, el gradiente descendente se expresa mediante la siguiente expresión:

$$\theta^{k+1} = \theta^k - \alpha \nabla C(\mathbf{x}, \mathbf{y}), \tag{1.16}$$

donde θ^k representa los parámetros del modelo en la iteración k, α es la tasa de aprendizaje, un escalar positivo que controla el tamaño del paso en cada iteración y $\nabla C(\mathbf{x}, \mathbf{y})$ es el gradiente de la función de costo respecto a los parámetros del modelo, que indica la dirección de mayor incremento de C.

El gradiente descendente busca iterativamente un conjunto de parámetros θ que minimicen C, avanzando en la dirección opuesta al gradiente. Esta dirección garantiza una disminución en el valor de la función de costo en cada paso, siempre que α sea apropiadamente pequeño.

Además, el gradiente descendente ha demostrado ser efectivo tanto para funciones convexas como no convexas (Shewchuk, 1994; Arora, 2013). En problemas convexos, garantiza convergencia al mínimo global, mientras que en problemas no convexos, converge a un mínimo local. Por ejemplo, en problemas de regresión lineal, C suele ser la función de error cuadrático medio. En otros casos como el de aprendizaje profundo, C puede representar funciones de pérdida más complejas, como la entropía cruzada.

Versión Estocástica

El gradiente descendente estocástico (*Stochastic Gradient Descent*, SGD) es una variante del gradiente descendente que busca mejorar la eficiencia en problemas de optimización con grandes conjuntos de datos. En lugar de calcular el gradiente completo de la función de costo, SGD utiliza una estimación basada en un único ejemplo o un pequeño subconjunto de datos en cada iteración.

Dado un conjunto de datos $(\mathbf{x}, \mathbf{y})$, la función de costo se define como:

$$C(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \ell(h_\theta(\mathbf{x}), \mathbf{y}), \tag{1.17}$$

donde $h_\theta(\mathbf{x})$ es la hipótesis parametrizada por θ y $\ell(h_\theta(\mathbf{x}), \mathbf{y})$ es la pérdida para el conjunto de datos $(\mathbf{x}, \mathbf{y})$

El gradiente descendente estocástico reemplaza el gradiente total $\nabla C(\mathbf{x}, \mathbf{y})$ con una estimación basada en un único ejemplo aleatorio en cada iteración t:

$$\theta^{k+1} = \theta^k - \alpha \nabla \ell(h_\theta(\mathbf{x}), \mathbf{y}), \tag{1.18}$$

donde θ^k representa los parámetros del modelo en la iteración k, α es la tasa de aprendizaje, un escalar positivo que controla el tamaño del paso en cada iteración y $\nabla \ell(h_\theta(\mathbf{x}), \mathbf{y})$ es el gradiente de la pérdida respecto a los parámetros del modelo para el ejemplo seleccionado aleatoriamente.

Esta aproximación reduce el costo computacional de cada iteración de $\mathcal{O}(n)$, como ocurre en el gradiente descendente estándar, a $\mathcal{O}(1)$ al operar con un único ejemplo.

1.5.3 Neurona Artificial

Una *neurona artificial (artificial neuron)* es una unidad básica de las redes neuronales artificiales, inspirada en la neurona biológica. Matemáticamente, toma un conjunto de entradas, aplica una transformación lineal con pesos asociados y un sesgo, y luego pasa el resultado a través de una función de activación no lineal para producir una salida.

Dado $\mathbf{x} \in \mathbb{R}^n$ un vector de características y $\theta \in \mathbb{R}^d$ el vector de pesos sinápticos (parámetros) y una función generalmente no lineal σ llamada de función de activación, se define una neurona artificial como:

$$\mathbf{z} = \sigma\left(\theta_0 + \theta_1 x^{(i)} + \cdots + \theta_{d-1} x^{(i)}\right) \tag{1.19}$$

Y $\mathbf{z}$ a veces se denomina como potencial de activación o simplemente activación.

Modelo de McCulloch y Pitts

El modelo de McCulloch y Pitts, introducido en 1943, fue uno de los primeros modelos matemáticos de una neurona artificial. Este modelo conceptualiza la neurona como una unidad de procesamiento que realiza una operación de suma ponderada de las entradas, seguida de una decisión binaria mediante una función de activación (umbral).

La ecuación matemática del perceptrón básico puede escribirse como:

$$y = \sigma\left(\sum_{i=1}^{n} \theta_i x_i + b\right), \tag{1.20}$$

donde y es la salida de la neurona, θ_i es el peso asociado a la entrada x_i, x_i es la componente i-ésima del vector de características o entrada, b es el sesgo o *bias* , parámetro adicional y $\sigma(\cdot)$ es la función de activación.

El modelo puede representarse matemáticamente como una suma ponderada y visualmente como un nodo que recibe múltiples entradas con pesos conectados, como se muestra en la Figura 1.1.

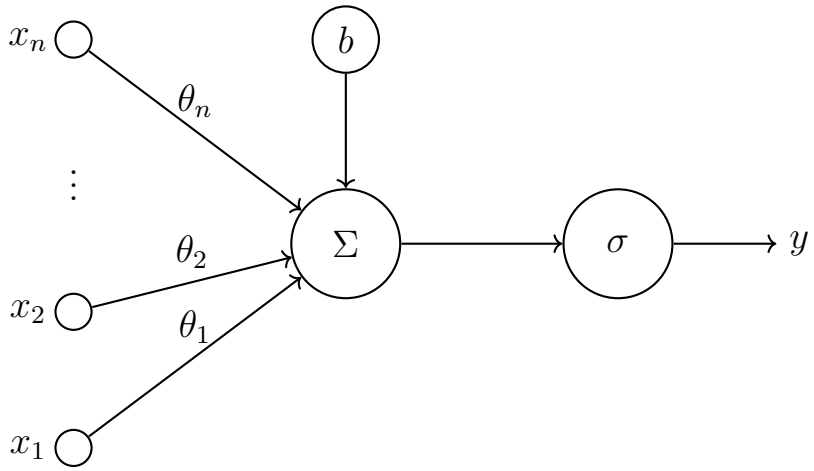


Figura 1.1. Representación visual de una neurona artificial según el modelo de McCulloch y Pitts. Donde θ y b son parámetros entrenables del modelo de neurona artificial, Σ representa la sumartoria de los productos ponderados y σ una función de activación no lineal.

1.5.4 Funciones de Activación

Las funciones de activación son componentes clave en las redes neuronales, ya que introducen no linealidad en los modelos y permiten aprender relaciones complejas en los datos. A continuación, se describen las principales funciones de activación utilizadas en aprendizaje profundo:

Sigmoide

La función sigmoide se define como

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.21}$$

donde su rango se encuentra en $(0, 1)$. Esta función comprime las salidas en este intervalo, haciéndola ideal para problemas de clasificación binaria. Entre sus ventajas está que resulta interpretable como una probabilidad, pero presenta desventajas como el problema de gradientes pequeños (saturación) para valores extremos de x, además de ser computacionalmente costosa debido a la operación exponencial.

La Figura 1.2 muestra la representación gráfica de la función sigmoide.

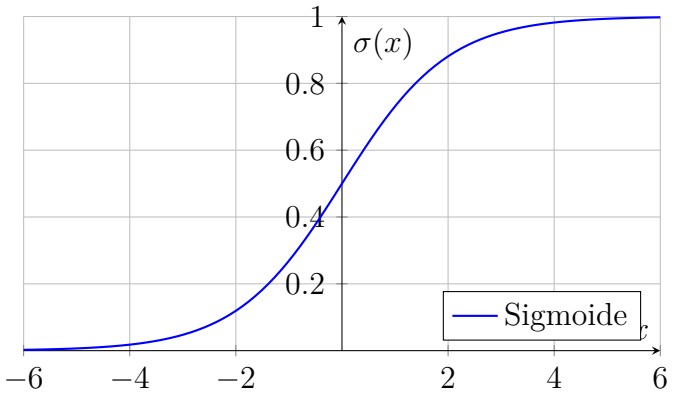


Figura 1.2. Gráfico de la función sigmoide.

Tangente Hiperbólica (tanh)

La tangente hiperbólica se define como

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1.22}$$

y tiene un rango en $(-1, 1)$. Similar a la sigmoide, pero centrada en cero, esta función es mejor para datos que requieren dicha centralización. Su ventaja principal es que presenta un mejor comportamiento en convergencia que la sigmoide, pero también puede saturarse para valores extremos de x.

La Figura 1.3 ilustra la función tangente hiperbólica.

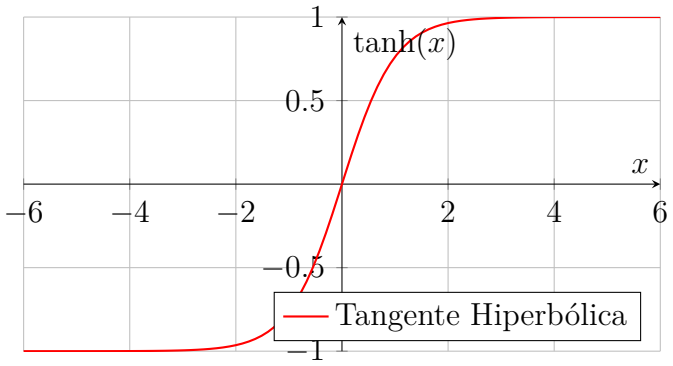


Figura 1.3. Gráfico de la función tangente hiperbólica.

ReLU (Rectified Linear Unit)

La función ReLU se define como

$$\text{ReLU}(x) = \max(0, x) \tag{1.23}$$

con un rango en $[0, \infty)$. Esta función introduce *sparsity* en las activaciones, lo que significa que muchas de las salidas de las neuronas son iguales a cero, mejorando así la eficiencia del modelo. Es computacionalmente eficiente y no sufre saturación para valores positivos. Sin embargo, tiene el problema conocido como ”muerte de ReLU”, donde las neuronas pueden quedar inactivas si $x \leqslant 0$ durante todo el entrenamiento.

La Figura 1.4 muestra la gráfica de la función ReLU.

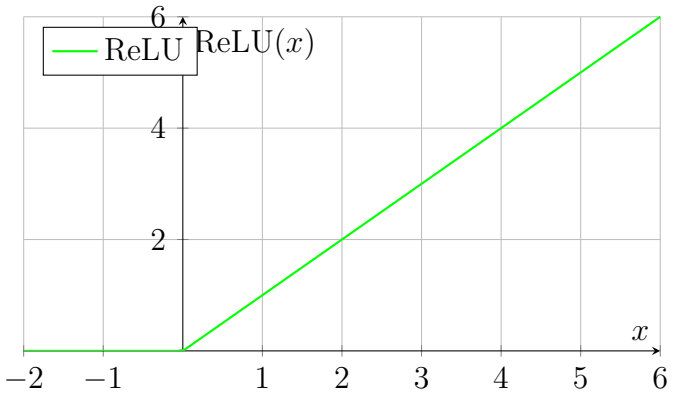


Figura 1.4. Gráfico de la función ReLU.

Leaky ReLU

La función Leaky ReLU se define como

$$\text{Leaky ReLU}(x) = \begin{cases} x & \text{si } x > 0, \\ \alpha x & \text{si } x \leqslant 0, \end{cases} \tag{1.24}$$

donde su rango es $(-\infty, \infty)$. Modifica ReLU al permitir un gradiente pequeño ($\alpha > 0$) para valores negativos, solucionando el problema de muerte de ReLU. Su principal desventaja es que introduce un hiperparámetro adicional (α).

La Figura 1.5 representa la función Leaky ReLU con $\alpha = 0.1$.

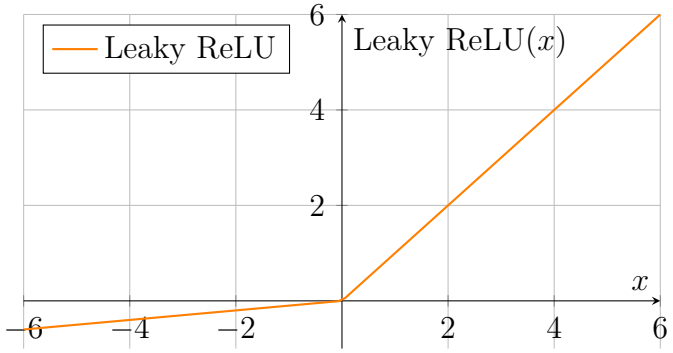


Figura 1.5. Gráfico de la función Leaky ReLU con $\alpha = 0.1$.

1.5.5 Red Neuronal

Una *red neuronal* es una estructura compuesta por múltiples neuronas artificiales conectadas en capas. Estas redes están diseñadas para modelar relaciones complejas entre las entradas y salidas, aprendiendo patrones no triviales a través del entrenamiento.

Matemáticamente, una red neuronal es una composición de conjuntos de neuronas artificiales, organizadas en capas. La salida de una red neuronal de L capas con funciones de activación σ puede expresarse mediante la siguiente ecuación matemática:

$$\mathbf{y} = \sigma\left(\boldsymbol{\theta}^{[L]}\sigma\left(\boldsymbol{\theta}^{[L-1]}\sigma\left(\cdots\sigma\left(\boldsymbol{\theta}^{[1]}\mathbf{x}+\mathbf{b}^{[1]}\right)+\mathbf{b}^{[2]}\cdots\right)+\mathbf{b}^{[L-1]}\right)+\mathbf{b}^{[L]}\right) \tag{1.25}$$

Donde $\mathbf{x}$ representa las entradas al modelo, $\boldsymbol{\theta}^{[l]}$ es la matriz de pesos de la capa l, donde $l = 1, \ldots, L$, $\mathbf{b}^{[l]}$ es el vector de sesgos de la capa l, y $\mathbf{y}$ es la salida final de la red neuronal.

Cada capa aplica una transformación lineal a los datos seguido de una transformación no lineal componente a componente (por ejemplo, ReLU o sigmoide). Esta estructura permite a las redes neuronales aprender relaciones complejas en los datos.

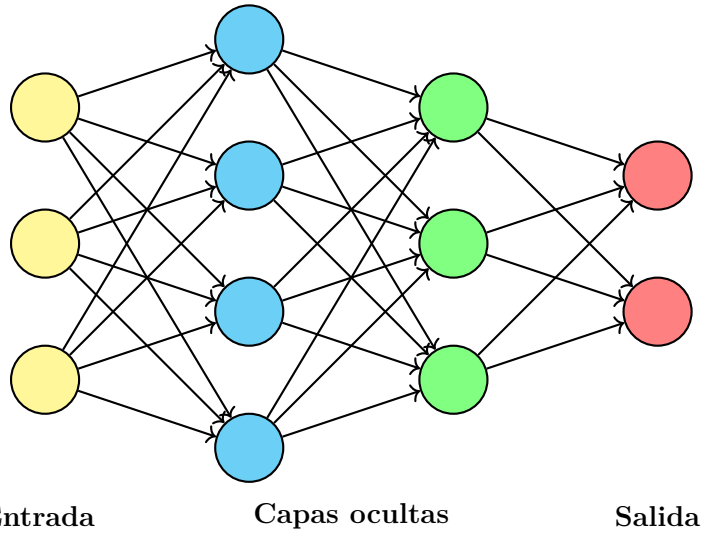


Figura 1.6. Esquema de una red neuronal artificial multicapa con una capa de entrada (amarilla), dos capas ocultas (azul y verde) y una capa de salida (roja). Cada neurona está conectada a todas las neuronas de la capa siguiente. La capa de salida cuenta con dos neuronas, permitiendo tareas de clasificación con múltiples categorías.

En la Figura 1.6 se ilustra una red neuronal de múltiples capas, donde la capa de

entrada recibe los datos y los transmite a las capas ocultas, donde se aplican transformaciones a través de funciones de activación. Finalmente, la capa de salida proporciona el resultado de la red, ya sea una clasificación, una predicción numérica o cualquier otra tarea para la que la red haya sido entrenada. La conectividad entre neuronas permite que el modelo capture relaciones complejas dentro de los datos, facilitando su capacidad de generalización en problemas de aprendizaje automático.

A modo de ejemplo, y utilizando notación matricial, es posible apreciar que la red neuronal con arquitectura Figura 1.6 puede representarse de la siguiente forma:

Ejemplo 1.5.1. Vector de entrada o también llamado vector de características:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Pesos asociados a la primera capa, donde las filas representan la cantidad de salidas de esta primera capa, y las columnas la cantidad de neuronas de entrada:

$$W_1 = \begin{pmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \\ w_{31}^1 & w_{32}^1 & w_{33}^1 \\ w_{41}^1 & w_{42}^1 & w_{43}^1 \end{pmatrix}$$

b es el vector de sesgos de la primera capa:

$$b_1 = \begin{pmatrix} b_1^1 \\ b_2^1 \\ b_3^1 \\ b_4^1 \end{pmatrix}$$

Las activaciones de esta primera transformación se representan utilizando la letra h, donde la función de activación σ se aplica componente a componente.

$$h_1 = \sigma(W_1 x + b_1) = \begin{pmatrix} h_1^1 \\ h_2^1 \\ h_3^1 \\ h_4^1 \end{pmatrix}$$

La segunda capa realizaría explícitamente la siguiente transformación:

$$\begin{pmatrix} h_1^2 \\ h_2^2 \\ h_3^2 \end{pmatrix} = \sigma \left(\begin{pmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 & w_{14}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 & w_{24}^2 \\ w_{31}^2 & w_{32}^2 & w_{33}^2 & w_{34}^2 \end{pmatrix} \begin{pmatrix} h_1^1 \\ h_2^1 \\ h_3^1 \\ h_4^1 \end{pmatrix} + \begin{pmatrix} b_1^2 \\ b_2^2 \\ b_3^2 \end{pmatrix} \right)$$

que en su notación reducida es dada por:

$$h_2 = \sigma(W_2 h_1 + b_2)$$

finalmente:

$$W_3 = \begin{pmatrix} w_{11}^3 & w_{12}^3 & w_{13}^3 \\ w_{21}^3 & w_{22}^3 & w_{23}^3 \end{pmatrix}$$

$$b_3 = \begin{pmatrix} b_1^3 \\ b_2^3 \end{pmatrix} \tag{1.26}$$

$$y = \sigma(W_3 h_2 + b_3) = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

Es decir, hemos aplicado en (1.27) la forma de composición definida en la ecuación (1.25), exponiendo los cálculos de manera explícita para mejor entendimiento.

$$y = \sigma(W_3 \sigma(W_2 \sigma(W_1 x + b_1) + b_2) + b_3) \tag{1.27}$$

1.5.6 Red Neuronal Convolucional (CNN)

Las redes neuronales convolucionales, conocidas como *CNN* por sus siglas en inglés, son un tipo especializado de redes neuronales diseñadas para procesar datos estructurados en forma de cuadrículas, como imágenes o señales de audio. Su arquitectura se inspira en la organización del sistema visual biológico, donde las neuronas responden a regiones específicas del campo visual y extraen información relevante de manera jerárquica.

Una bloque convolucional en el contexto de *deep learning* es una capa neuronal similar a las capas definidas en la sección 1.5.5 pero *no son completamente densas*. Una convolución es una operación que aplica un filtro sobre las componentes de una matriz (pixeles en el caso de imágenes) punto a punto, barriendo sobre toda la matriz la misma operación. La serie de imágenes 1.7

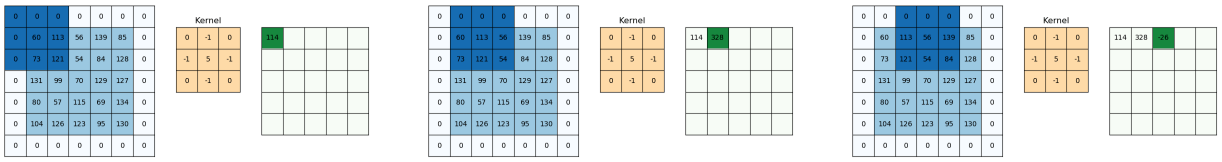


Figura 1.7. Aplicando un kernel convolucional sobre una matriz. De izquierda a derecha es posible apreciar como el kernel se mueve a través de la imágen para poder generar la salida o activación. En este ejemplo particular, el kernel fue aplicado tres veces, dando pasos de tamaño uno en el eje horizontal.

Como fue posible apreciar en el ejemplo anterior, el funcionamiento de una *CNN* se basa en la *capa convolucional*, la cual aplica filtros o *kernels* sobre la entrada para detectar patrones espaciales y locales en la imagen. Cada neurona en esta capa realiza la operación de convolución, definida matemáticamente como

$$y_{ij}^{k} = \sum_{m} \sum_{n} x_{(i+m)(j+n)} \theta_{mn}^{k} + b^{k} \tag{1.28}$$

donde y_{ij}^{k} representa el valor del píxel en la posición (i, j) del mapa de características k, $x_{(i+m)(j+n)}$ es la entrada en la posición $(i + m, j + n)$, θ_{mn}^{k} es el filtro aplicado y b^{k} es el sesgo asociado. Esta operación permite resaltar características esenciales de la imagen, como bordes, texturas y estructuras geométricas.

De manera sencilla es posible observar a través de un ejemplo pequeño, que la capa convolucional es equivalente a un red neuronal pequeña no densa.

Ejemplo 1.5.2. Asumamos que tenemos un filtro de tamaño 2×2 dado por:

$$\theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix}$$

Y queremos aplicarlo a la imagen:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}$$

En su primera aplicación debemos multiplicar el filtro para la esquina superior izquierda de X

$$\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} * \begin{bmatrix} k_{11} & k_{12} & - \\ k_{21} & k_{22} & - \\ - & - & - \end{bmatrix}$$

Con esto obtenemos una componente de salida z que formará parte de la activación luego de aplicar la función no lineal:

$$z = x_{11} * k_{11} + x_{12} * k_{12} + x_{21} * k_{21} + x_{22} * k_{22}$$
$$a = \sigma(z)$$

Si continuamos moviendo el filtro con paso 1 tanto horizontal como verticalmente, es posible encontrar una forma de reescribir la operación utilizando a penas una multiplicación matriz vector. El producto entre la primera linea de esta matriz aplicada al vector representando a X (en una representación estirada o *flatten* de la matrix X):

$$\begin{bmatrix} k_{11} & k_{12} & 0 & k_{21} & k_{22} & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = x_{11} * k_{11} + x_{12} * k_{12} + x_{21} * k_{21} + x_{22} * k_{22} = z$$

Es decir, la operación completa de convolución se puede representar como un producto matriz vector, donde la matriz tendrá parámetros repetidos y muchos ceros (esparcidad).

La capa anterior se combina con otras estrategias de reducción de dimensión con tal de reducir el tamaño del modelo, en la medida que se hace más profundo. Para reducir la dimensionalidad de los datos y mejorar la eficiencia computacional, se introduce la *capa de*

pooling, que selecciona los valores más representativos de una región determinada. Existen dos tipos principales de *pooling*. El primero es *max-pooling*, que selecciona el valor máximo dentro de una ventana deslizante sobre la imagen, conservando la información más relevante de cada región. Matemáticamente, esta operación se expresa como

$$y_{ij} = \max\{x_{(i+m)(j+n)} : m, n \in \text{ventana de pooling}\} \tag{1.29}$$

El segundo tipo es *average-pooling*, que en lugar de seleccionar el valor máximo dentro de la ventana, calcula el valor promedio de los píxeles en dicha región, lo que permite suavizar la representación de la imagen y retener información de contexto más general. Su expresión matemática es

$$y_{ij} = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} x_{(i+m)(j+n)} \tag{1.30}$$

donde M y N representan las dimensiones de la ventana de *pooling*. Mientras que *max-pooling* es útil para preservar características clave y mejorar la invariancia a pequeñas traslaciones en la imagen, *average-pooling* ayuda a reducir el ruido y capturar información de fondo de manera más difusa.

Para introducir no linealidad en el modelo y permitir la detección de relaciones más complejas, se emplea una *capa de activación*. Entre las más utilizadas se encuentra la función $ReLU$, que consiste en una transformación matemática definida como

$$\text{ReLU}(x) = \max(0, x) \tag{1.31}$$

Esta función reemplaza los valores negativos por cero, lo que facilita el entrenamiento de redes profundas al evitar la saturación de los gradientes y mejorar la eficiencia del aprendizaje.

Luego de la extracción de características, las redes convolucionales incorporan una *capa completamente conectada*, también conocida como *fully connected layer*. En esta etapa, las características obtenidas en las capas anteriores se transforman en una representación plana para su posterior clasificación o regresión. El funcionamiento de esta capa se rige por la ecuación general de una red neuronal densa,

$$\mathbf{y} = \sigma(\boldsymbol{\theta} \cdot \mathbf{x} + \mathbf{b}) \tag{1.32}$$

donde $\boldsymbol{\theta}$ representa los pesos de la red, $\mathbf{x}$ es la entrada y $\mathbf{b}$ es el sesgo. La salida final de la *CNN* se interpreta según la tarea específica, como la probabilidad de clasificación en problemas supervisados.

La Figura 1.8 ilustra el esquema general de una red neuronal convolucional, donde se observa el flujo de procesamiento desde la imagen de entrada hasta la capa final de clasificación. En este proceso, las capas convolucionales extraen características fundamentales de la imagen, mientras que las capas de *pooling* reducen la dimensionalidad para mejorar la eficiencia computacional. Finalmente, las capas totalmente conectadas procesan la información extraída y generan una predicción.

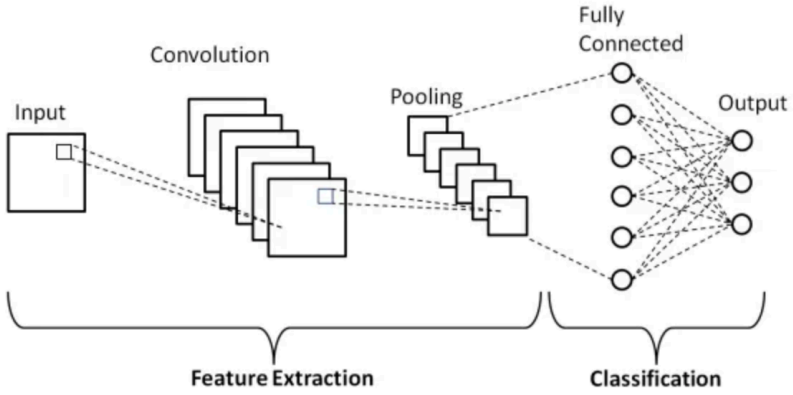


Figura 1.8. Esquema de una red neuronal convolucional. La imagen de entrada pasa por capas convolucionales donde se extraen características esenciales, seguidas por capas de *max-pooling* y *average-pooling* que reducen la dimensionalidad. Finalmente, las características se introducen en capas densamente conectadas para la clasificación final.

Las redes neuronales convolucionales han demostrado ser altamente efectivas en tareas de visión por computadora, como el reconocimiento de imágenes, la detección de objetos y el análisis biomédico. Un caso ampliamente estudiado es el de la base de datos *MNIST*, donde una *CNN* puede identificar con gran precisión los dígitos manuscritos del 0 al 9. Su capacidad para extraer automáticamente características relevantes sin necesidad de intervención manual

las convierte en herramientas poderosas en el campo del aprendizaje profundo.

Para mejorar el rendimiento y la adaptabilidad de estas redes, se pueden emplear diversas técnicas de optimización. Algunas de las más utilizadas incluyen *Batch Normalization*, que estabiliza el entrenamiento al normalizar la activación de cada capa, y *Dropout*, que reduce el sobreajuste eliminando aleatoriamente algunas conexiones durante el entrenamiento. Además, el uso de modelos preentrenados, como *ResNet*, ha permitido mejorar significativamente el reconocimiento de imágenes en grandes bases de datos.

En muchos casos, es posible combinar redes convolucionales con otros tipos de arquitecturas neuronales para capturar dependencias espaciales y temporales simultáneamente. Un ejemplo de ello es la integración de redes convolucionales con redes recurrentes, como las *Long Short-Term Memory* (LSTM), que permiten analizar datos secuenciales en tareas como el reconocimiento de voz y la traducción automática.

1.5.7 Complejización y Personalización

Las redes neuronales pueden personalizarse para abordar problemas más complejos mediante el aumento de su profundidad, es decir, incrementando la cantidad de parámetros del modelo a través de capas adicionales o combinando distintos tipos de arquitecturas. Existen diferentes enfoques para mejorar la capacidad de aprendizaje y generalización de los modelos, incluyendo el uso de distintas arquitecturas, la optimización de hiperparámetros y la aplicación de técnicas de regularización.

Uno de los enfoques más comunes para aumentar la expresividad de los modelos es la combinación de redes neuronales convolucionales (*CNN*) y redes recurrentes (*RNN*). Mientras que las *CNN* son altamente eficaces en la extracción de características espaciales en imágenes (Kaiming He and Sun, 2016), las *RNN* están diseñadas para capturar dependencias temporales en secuencias, lo que las hace ideales para tareas de modelado de audio y lenguaje natural (Hochreiter and Schmidhuber, 1997).

Además, la personalización de las redes neuronales también implica la modificación de hiperparámetros críticos del proceso de optimización, como la tasa de aprendizaje, el tamaño del lote y la elección del optimizador. Técnicas avanzadas para mitigar problemas como la desaparición o explosión del gradiente incluyen la normalización por lotes (*Batch Normalization*) (Ioffe and Szegedy, 2015) y la regularización mediante *Dropout*

(Nitish Srivastava and Salakhutdinov, 2014), que previenen el sobreajuste en redes profundas.

Otro enfoque ampliamente utilizado es el empleo de modelos preentrenados, que aprovechan arquitecturas avanzadas adaptadas a problemas específicos en diversas áreas como procesamiento de lenguaje natural, visión por computadora y análisis de audio. Ejemplos de modelos avanzados incluyen los *Transformers* (Ashish Vaswani and Polosukhin, 2017), que han revolucionado el aprendizaje automático y son la base de modelos de lenguaje como GPT (et al., 2020); los *Autoencoders*, empleados en reducción de dimensionalidad y generación de datos (Kingma and Welling, 2014); y las arquitecturas profundas de clasificación como *ResNet* (Kaiming He and Sun, 2016), que han demostrado mejoras significativas en el reconocimiento de imágenes a gran escala.

Estos métodos permiten la adaptación de modelos neuronales a aplicaciones altamente especializadas, optimizando su rendimiento y eficiencia en diferentes dominios del aprendizaje automático.

1.5.8 Equilibrio en Modelos de Aprendizaje Automático

El desempeño de un modelo de aprendizaje automático depende en gran medida de su capacidad para generalizar a nuevos datos. En este contexto, los conceptos de *sesgo* y *varianza* permiten analizar los errores cometidos durante el entrenamiento y la evaluación del modelo. La Figura 1.9 ilustra tres escenarios característicos: *subajuste* (*underfitting*), ajuste óptimo y *sobreajuste* (*overfitting*).

El *subajuste* ocurre cuando el modelo tiene una capacidad de representación insuficiente para capturar la estructura de los datos. En este caso, la función de decisión es demasiado simple y no logra separar adecuadamente las clases, lo que se traduce en un *error elevado tanto en los datos de entrenamiento como en los datos de prueba.* Generalmente, este problema está asociado a modelos con alta restricción, como los modelos lineales en tareas que requieren relaciones no lineales.

Por otro lado, el *sobreajuste* se produce cuando el modelo se adapta excesivamente a las características específicas del conjunto de entrenamiento, incluyendo ruido y patrones irrelevantes. Como resultado, el modelo muestra un *error de entrenamiento bajo*, pero un *error elevado en los datos de prueba*, lo que indica una pobre capacidad de generalización.

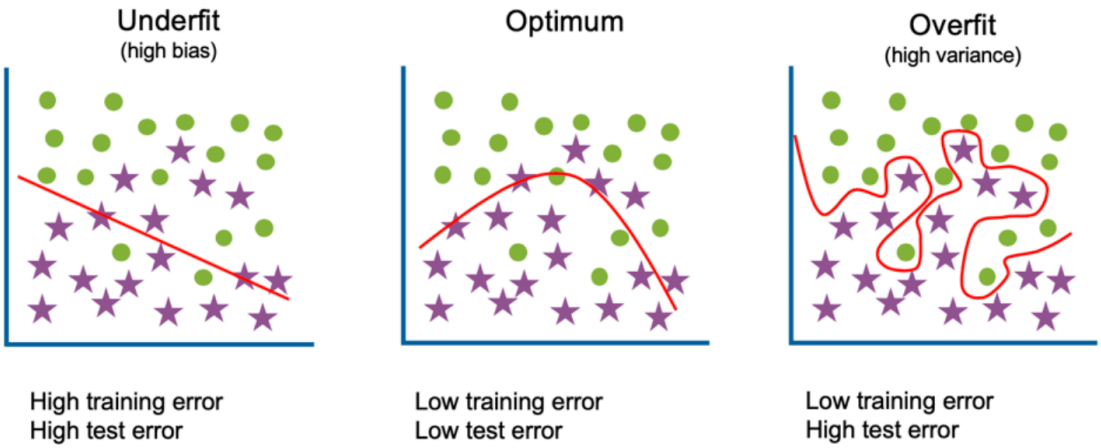


Figura 1.9. Relación entre sesgo y varianza en modelos de aprendizaje automático. El *subajuste* (izquierda) ocurre cuando el modelo no logra capturar la estructura de los datos, el ajuste óptimo (centro) representa el balance adecuado entre sesgo y varianza, mientras que el *sobreajuste* (derecha) se produce cuando el modelo se ajusta excesivamente a los datos de entrenamiento, perdiendo capacidad de generalización.

Este fenómeno es característico de modelos con alta complejidad, como redes neuronales profundas sin regularización o modelos de alta dimensionalidad con un número reducido de muestras.

El *equilibrio entre sesgo y varianza* se alcanza cuando el modelo logra capturar los patrones generales de los datos sin ajustarse en exceso a variaciones específicas del conjunto de entrenamiento. En este escenario, el modelo obtiene un *error bajo tanto en los datos de entrenamiento como en los datos de prueba*, asegurando una adecuada capacidad de generalización. La selección del modelo, el ajuste de hiperparámetros y el uso de técnicas de regularización juegan un papel clave en la prevención del *sobreajuste* y el *subajuste*, promoviendo un aprendizaje efectivo y estable.

1.5.9 Métricas en problemas de clasificación

Matriz de Confusión

La matriz de confusión es una herramienta fundamental para evaluar el rendimiento de un modelo de clasificación. Proporciona un resumen detallado del desempeño del modelo al comparar las predicciones con las etiquetas reales. En términos generales, una matriz de

confusión para un problema de clasificación binaria tiene la siguiente estructura:

	Predicción Positiva	**Predicción Negativa**
Clase Positiva	True Positives (TP)	False Negatives (FN)
Clase Negativa	False Positives (FP)	True Negatives (TN)

Tabla 1.2. Estructura general de una matriz de confusión en un problema de clasificación binaria.

Cada celda de la matriz representa una categoría específica: los *True Positives (TP)* corresponden a los ejemplos positivos correctamente clasificados, mientras que los *False Negatives (FN)* representan los ejemplos positivos que fueron incorrectamente predichos como negativos. Por otro lado, los *False Positives (FP)* indican ejemplos negativos que el modelo clasificó erróneamente como positivos, y los *True Negatives (TN)* corresponden a los ejemplos negativos correctamente identificados.

Cálculo de Métricas a partir de la Matriz de Confusión

A partir de la matriz de confusión, se pueden derivar diversas métricas de evaluación:

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{1.33}$$

$$\text{Sensibilidad (Recall)} = \frac{TP}{TP + FN} \tag{1.34}$$

$$\text{Especificidad} = \frac{TN}{TN + FP} \tag{1.35}$$

$$\text{Balanced Accuracy Score (BAS)} = \frac{\text{Sensibilidad} + \text{Especificidad}}{2} \tag{1.36}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \tag{1.37}$$

La precisión mide cuántos de los ejemplos clasificados como positivos son realmente positivos, mientras que la sensibilidad evalúa la capacidad del modelo para identificar correctamente los casos positivos. La especificidad, por otro lado, indica la capacidad del modelo para evitar falsos positivos.

Visualización de la Matriz de Confusión

Para analizar mejor el desempeño del modelo, se genera una representación gráfica de la matriz de confusión. La Figura 1.10 muestra un ejemplo de una matriz de confusión.

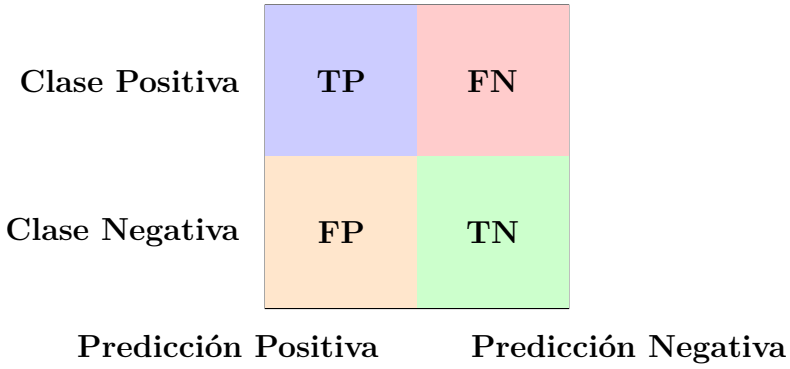


Figura 1.10. Matriz de confusión con representación visual de las categorías: True Positives (TP), False Negatives (FN), False Positives (FP) y True Negatives (TN).

Esta matriz permite detectar patrones de error del modelo y ajustar estrategias de entrenamiento para mejorar su desempeño. Por ejemplo, si el modelo presenta un alto número de falsos negativos, podría ser necesario ajustar el umbral de decisión o emplear técnicas de balanceo de clases.

En un modelo de clasificación de toses para COVID-19, un falso negativo (FN) representaría un caso de COVID-19 que no fue detectado, lo que podría tener graves implicaciones médicas. En contraste, un falso positivo (FP) implicaría que una persona sana fue clasificada erróneamente como infectada, lo que podría generar alarmas innecesarias. Por ello, es crucial encontrar un balance entre sensibilidad y especificidad dependiendo de la aplicación práctica del modelo.

1.5.10 Curvas de Pérdida y Estrategias de Regularización

Durante el entrenamiento de modelos de aprendizaje automático, es fundamental analizar la evolución de la función de pérdida para evaluar la convergencia del modelo y detectar posibles problemas de *subajuste* o *sobreajuste*. La Figura 1.11 muestra la evolución típica de la pérdida en función del número de épocas, diferenciando la pérdida en el conjunto de entrenamiento y el conjunto de prueba.

Al inicio del entrenamiento, el modelo se encuentra en una fase de *subajuste*, donde

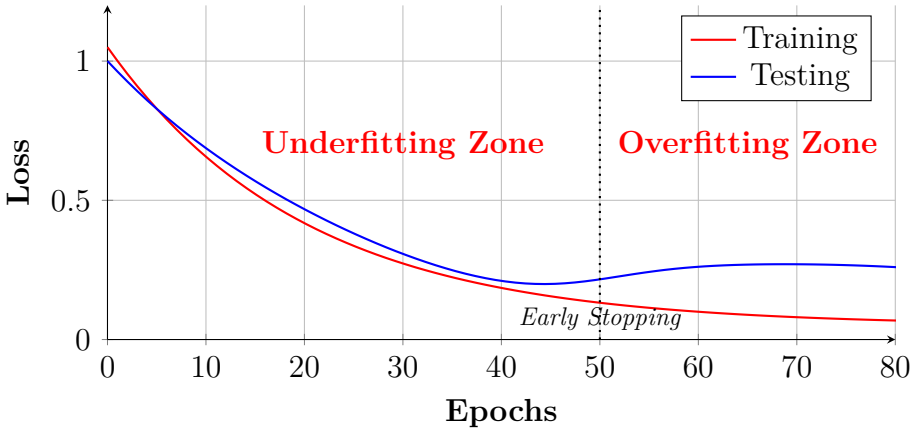


Figura 1.11. Curvas de pérdida durante el entrenamiento de un modelo de aprendizaje automático. La zona de *underfitting* se caracteriza por pérdidas altas en ambos conjuntos. En la zona de *overfitting*, la pérdida en prueba comienza a aumentar mientras la de entrenamiento sigue disminuyendo. La línea de *early stopping* indica el punto donde se debería detener el entrenamiento para optimizar la generalización.

tanto la pérdida de entrenamiento como la de prueba son elevadas, ya que el modelo aún no ha aprendido patrones significativos de los datos. Conforme el entrenamiento avanza, la pérdida en ambos conjuntos disminuye progresivamente, lo que indica que el modelo está mejorando su capacidad para predecir correctamente.

El punto de convergencia óptimo se alcanza cuando la pérdida en el conjunto de prueba es mínima. Más allá de este punto, si el entrenamiento continúa sin medidas de control, la pérdida en el conjunto de prueba comienza a aumentar mientras que la pérdida en el conjunto de entrenamiento sigue disminuyendo. Este comportamiento es característico del *sobreajuste*, donde el modelo memoriza patrones específicos del conjunto de entrenamiento, perdiendo su capacidad de generalización.

Para mitigar el *sobreajuste*, se implementan estrategias de regularización como la detención temprana (*early stopping*), indicada en la Figura 1.11. Esta técnica consiste en monitorear la pérdida en el conjunto de validación y detener el entrenamiento cuando se detecta un aumento sostenido, evitando que el modelo continúe ajustándose excesivamente a los datos de entrenamiento. Otras estrategias incluyen el uso de *dropout*, la penalización mediante normas L_1 y L_2, y la expansión del conjunto de datos mediante técnicas de aumentación.

El análisis de las curvas de pérdida es una herramienta fundamental en el entrenamiento de modelos de aprendizaje automático, ya que permite ajustar hiperparámetros y detectar anomalías.

CAPÍTULO 2

2. METODOLOGÍA

2.1 Flujo de Trabajo para la Clasificación Automatizada de Toses para el Diagnóstico de COVID-19

El desarrollo del proyecto de clasificación automatizada de toses utilizando redes neuronales convolucionales siguió un flujo de trabajo ampliamente utilizado en los modelos de *Machine Learning*, representado en la Figura 2.1. (Islam et al., 2024)

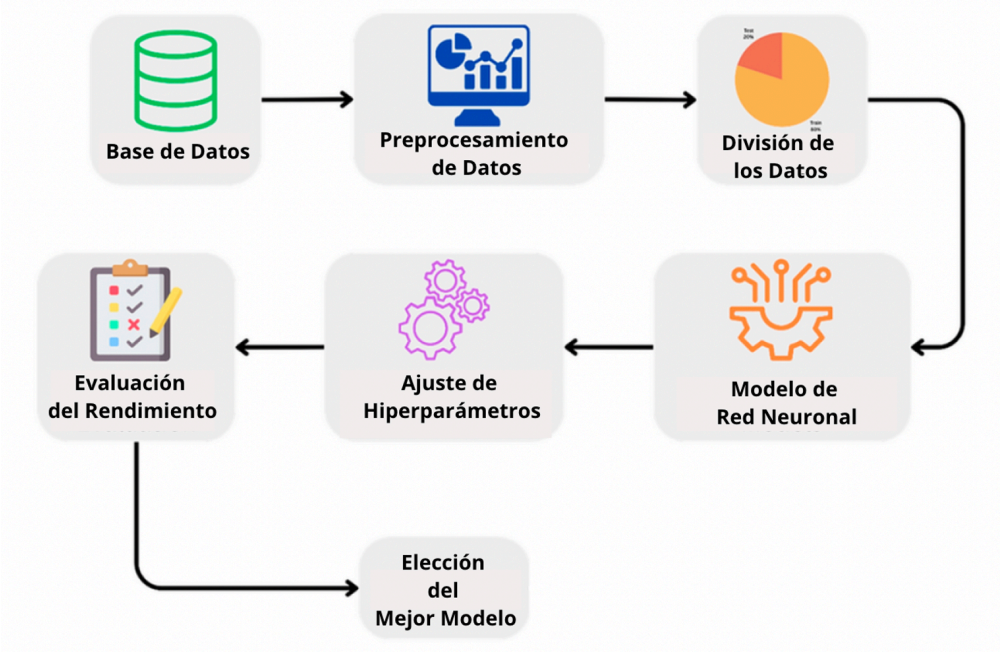


Figura 2.1. Flujo de trabajo general adaptado al proyecto de clasificación automatizada de toses.

Este flujo de trabajo tuvo un enfoque iterativo, permitiendo volver a etapas previas (como el preprocesamiento o el ajuste de hiperparámetros) si los resultados no cumplían con los objetivos esperados. Esto permitió la obtención de un modelo eficiente y confiable.

2.1.1 Dataset

Esta etapa del proyecto se basó en la utilización de múltiples conjuntos de datos especializados en la recopilación de toses de pacientes con COVID-19 y otras afecciones respiratorias. Se emplearon tres bases de datos principales: *COUGHVID*, *COVID-19 Cough Recordings* y *Virufy Data Set*. La combinación de estos datasets permitió mejorar la calidad y representatividad de las muestras utilizadas en la clasificación automática mediante redes neuronales convolucionales.

Fuentes de Datos

El conjunto de datos *COUGHVID* contenía más de 25,000 grabaciones de toses recopiladas de manera colaborativa (*crowdsourced*), abarcando una amplia diversidad de edades, géneros, ubicaciones geográficas y estados de salud de los participantes. Un subconjunto de 2,800 grabaciones fue etiquetado por médicos experimentados, proporcionando una referencia de alta calidad para la clasificación de señales de tos.

El dataset *COVID-19 Cough Recordings* incluyó 170 grabaciones de tos, de las cuales 19 correspondían a pacientes con COVID-19 y 151 a pacientes sin la enfermedad. Este conjunto de datos fue empleado en diversas investigaciones enfocadas en la identificación automática de toses asociadas al COVID-19.

Por su parte, *Virufy Data Set* proporcionó un total de 121 grabaciones de toses obtenidas en un entorno clínico, donde cada muestra fue etiquetada con el estado de COVID-19 basado en pruebas PCR. Se identificaron 48 grabaciones positivas y 73 negativas. Este dataset resultó relevante debido a la rigurosidad en su recopilación, ya que las grabaciones fueron obtenidas en hospitales bajo supervisión médica, asegurando datos confiables para su análisis.

Calidad de los Datos

Si bien el dataset *COUGHVID* fue ampliamente utilizado, su calidad presentó algunas limitaciones debido a la variabilidad en las condiciones de grabación y la presencia de ruido en muchas de las muestras recopiladas de manera colaborativa. Para mitigar este problema, se incorporaron los datasets *COVID-19 Cough Recordings* y *Virufy Data Set*, los

cuales ofrecieron un mayor control en la calidad de las grabaciones y una mejor precisión en las etiquetas asignadas a cada muestra.

En particular, *Virufy Data Set* se destacó por haber sido recopilado en hospitales, siguiendo protocolos estándar y asegurando que cada grabación estuviera correctamente segmentada y libre de interferencias de ruido. Esto permitió mejorar la precisión del modelo al contar con datos más limpios y confiables.

Diversidad y Representatividad

Las bases de datos utilizadas presentaron una diversidad considerable en términos de características demográficas de los participantes, incluyendo edad, género y ubicación geográfica. Mientras que *COUGHVID* y *Virufy Data Set* contaban con grabaciones obtenidas de múltiples países, *Virufy* se distinguió por incluir información detallada sobre el historial médico de los participantes, lo que permitió una evaluación más precisa de los factores que afectaban la clasificación de la tos.

Además, *Virufy* incorporó tanto grabaciones completas como segmentos de tos preprocesados, lo que facilitó el entrenamiento del modelo en diferentes niveles de granularidad. La combinación de estos datasets aseguró que el modelo pudiera generalizar mejor en diferentes contextos clínicos y poblaciones.

Formato de los Datos

El conjunto de datos original de *COUGHVID* estaba compuesto por grabaciones en formatos `webm`, `ogg` y `WAV`, mientras que los datasets *COVID-19 Cough Recordings* y *Virufy Data Set* utilizaron principalmente el formato `WAV`. Para garantizar compatibilidad con las bibliotecas de procesamiento y análisis de audio, como `librosa` y `torchaudio`, todas las grabaciones fueron convertidas a formato `WAV` con una frecuencia de muestreo estandarizada de 48 kHz.

El uso combinado de estos datasets permitió mejorar la calidad de los datos de entrenamiento y reducir el impacto de grabaciones de baja calidad presentes en *COUGHVID*. Con ello, se buscó entrenar un modelo de clasificación de toses más robusto, capaz de detectar patrones acústicos distintivos de COVID-19 y otras afecciones respiratorias con mayor precisión.

2.1.2 Preprocesamiento de Datos

El preprocesamiento de datos consistió en transformar las grabaciones crudas de audio en un formato estandarizado y adecuado para su análisis y entrenamiento. Para garantizar la consistencia en la preparación de los datos, se aplicaron diversos procedimientos de limpieza, normalización y transformación a todas las grabaciones.

Se procesaron todas las grabaciones del dataset, asegurando que cada archivo pasara por un tratamiento uniforme. Este enfoque permitió estandarizar los datos y reducir la variabilidad causada por diferencias en la duración, el formato y la calidad de las grabaciones.

Cada audio fue ajustado a una duración uniforme basada en el percentil 95 de las duraciones del conjunto de datos. En primer lugar, si la grabación contenía múltiples canales, se convirtió a una señal mono al calcular el promedio de los canales disponibles. Luego, si la frecuencia de muestreo difería de 48,000 Hz, se realizó un resampleo utilizando la función `torchaudio.transforms.Resample` para asegurar la compatibilidad con el resto del procesamiento.

Posteriormente, si la duración de la grabación excedía el percentil 95, se recortó para ajustarse al umbral definido. En cambio, si la duración era inferior al valor establecido, se rellenó con ceros al final de la señal para mantener una longitud uniforme en todas las muestras. Esta estandarización permitió que el modelo trabajara con entradas de tamaño fijo, simplificando el procesamiento posterior y asegurando una estructura homogénea en los datos de entrenamiento.

Conversión de Audios a Mel-Espectrogramas

Tras la limpieza y normalización, las grabaciones de audio fueron convertidas a mel-espectrogramas, una representación en el dominio tiempo-frecuencia ampliamente utilizada en tareas de aprendizaje automático. Este proceso permitió transformar las señales acústicas en un formato más adecuado para la extracción de características relevantes.

Para ello, se aplicó una Transformada Rápida de Fourier (FFT) con parámetros específicos, definiendo la longitud de ventana (`n_fft`), la longitud de salto (`hop_length`) y el número de bandas mel (`n_mels`). Posteriormente, se convirtió la amplitud de los

espectrogramas a una escala logarítmica utilizando `torchaudio.transforms.AmplitudeToDB`, lo que mejoró la representación acústica al reducir la sensibilidad a variaciones de energía en la señal.

Los parámetros utilizados para la generación de los mel-espectrogramas se detallan en la Tabla 2.1:

Parámetro	**Valor**
Frecuencia de muestreo (`sr`)	48000 Hz
Longitud de ventana (`n_fft`)	1024
Longitud de ventana efectiva (`win_length`)	512
Desplazamiento (`hop_length`)	256
Número de bandas mel (`n_mels`)	128
Escala de potencia (`power`)	2.0
Semilla (`np.seed`)	42

Tabla 2.1. Parámetros utilizados en la generación de mel-espectrogramas.

Cada mel-espectrograma generado fue almacenado en formato `.pt` utilizando la biblioteca `torch`, optimizando el acceso a los datos durante el entrenamiento del modelo. Los archivos fueron guardados con nombres únicos basados en un índice secuencial (`mel_spec_{i}.pt`) y organizados en un directorio de salida predefinido.

Posteriormente, se realizó una verificación visual de los mel-espectrogramas para evaluar la calidad de las características extraídas. La Figura 2.2 muestra un ejemplo de mel-espectrograma generado, representado en una escala de colores mediante la biblioteca `matplotlib`.

Conversión de Audios a Cromagramas

El cromagrama es una representación espectral que mapea la energía de la señal de audio en 12 semitonos por octava, permitiendo capturar información tonal relevante para la clasificación de toses. Para generar esta representación, se re-muestrean todas las grabaciones a una frecuencia estándar de 48 kHz, asegurando uniformidad en el procesamiento. Posteriormente, se aplica la Transformada de Fourier de Corto Tiempo

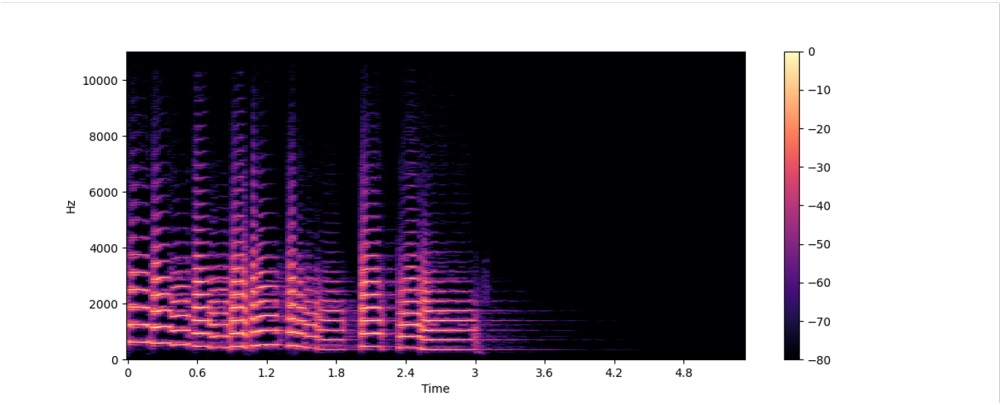


Figura 2.2. Ejemplo de un mel-espectrograma generado, donde se observa la distribución de la energía espectral en función del tiempo.

(STFT) con una ventana de 2048 muestras, un desplazamiento de 512 muestras y una ventana de Hann de 2048 muestras para obtener la distribución espectral en el tiempo.

A partir del espectrograma obtenido, se extrae el cromagrama mediante un banco de filtros cromáticos de 12 bandas, donde se consideran frecuencias a partir de 20 Hz con ajuste tonal neutral. Esta conversión resalta la estructura armónica del audio sin depender de la frecuencia absoluta. Para mejorar la interpretación de los valores obtenidos, se convierte la energía cromática a una escala logarítmica en decibeles, aplicando la transformación:

$$\text{Chroma-dB} = 10 \cdot \log_{10}(\text{Chroma Energy}) \tag{2.1}$$

La Figura 2.3 muestra un ejemplo de cromagrama generado, donde el eje vertical representa las 12 clases de notas musicales, el eje horizontal indica el tiempo en segundos y la intensidad del color refleja la energía asociada a cada nota en distintos momentos del audio.

Los cromagramas se almacenan posteriormente en formato `.pt` como tensores de dimensiones $(12, T)$, donde T representa el número de frames temporales del audio procesado. Esto permite su uso eficiente durante el entrenamiento del modelo de clasificación.

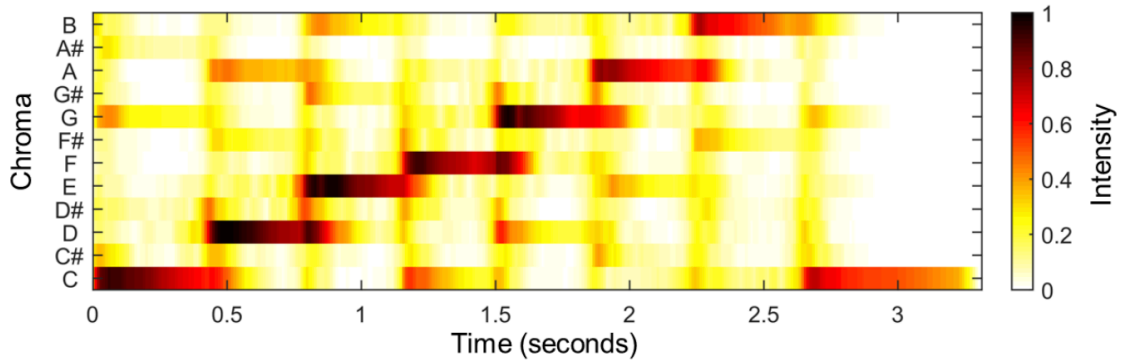


Figura 2.3. Ejemplo de cromagrama de un audio de tos. Se observa la distribución de la energía en función de los 12 semitonos a lo largo del tiempo.

Extracción de Características Acústicas para el Análisis de Audio

Para analizar las grabaciones de tos, se han extraído diversas características acústicas que permiten representar la señal en términos de su contenido espectral y temporal. Estas características proporcionan información clave sobre la estructura del sonido y su variabilidad, facilitando la clasificación automática mediante modelos de aprendizaje profundo.

El *Chroma STFT* representa la energía distribuida en los 12 tonos de la escala musical y resulta útil para detectar patrones tonales en la señal. La *energía cuadrática media* o *Root Mean Square Energy* (RMSE) mide la intensidad promedio de la señal, permitiendo diferenciar sonidos fuertes de débiles. El *centroide espectral* indica la frecuencia en la que se concentra la mayor parte de la energía, lo que se relaciona con la ”brillantez” del sonido.

Además, el *ancho de banda espectral* mide la dispersión de las frecuencias en torno al centroide espectral, ayudando a identificar la variedad de frecuencias presentes en la tos. El *rolloff espectral* determina la frecuencia por debajo de la cual se encuentra la mayor parte de la energía espectral, lo que resulta útil para distinguir entre sonidos agudos y graves. El *tasa de cruces por cero* (*Zero Crossing Rate, ZCR*) mide la cantidad de veces que la señal cambia de signo, proporcionando información sobre la naturaleza ruidosa o suave del sonido.

Por último, los *Coeficientes Cepstrales en Frecuencia de Mel* (MFCCs) son un conjunto de 20 coeficientes que capturan la envolvente espectral de la señal en una escala perceptual, imitando cómo el oído humano percibe los sonidos. Estos coeficientes son

ampliamente utilizados en reconocimiento de voz y en el análisis biomédico de la tos, ya que permiten extraer información relevante sobre la modulación de frecuencia y la estructura espectral de la señal.

La combinación de estas características permite modelar las señales de tos y extraer patrones relevantes para su posterior análisis y clasificación mediante técnicas de aprendizaje automático. Este enfoque facilita la identificación de diferencias sutiles en la estructura del sonido, contribuyendo a la detección y diagnóstico basado en señales acústicas.

Aumentación de Datos

Para mejorar la variabilidad del conjunto de datos y aumentar la robustez del modelo, se aplicaron diversas técnicas de aumentación. En primer lugar, se modificó la velocidad de las grabaciones mediante un factor de escala (`speed_factor = 1.2`), lo que permitió simular diferentes características acústicas en la señal. Además, se generó ruido aleatorio con una distribución normal utilizando `np.random.normal` y se incorporó a las grabaciones originales, con el objetivo de reproducir condiciones de grabación más realistas. Por último, se aplicó un desplazamiento temporal en la señal, introduciendo variabilidad adicional en los datos de entrada.

Reproducibilidad

Para garantizar resultados consistentes y replicables, todas las operaciones de preprocesamiento fueron realizadas utilizando una semilla aleatoria fija (`np.random.seed(42)`). Esto aseguró que las transformaciones aplicadas mantuvieran coherencia en cada ejecución del proceso, permitiendo reproducir los resultados obtenidos en futuras iteraciones.

Tras completar esta etapa, las grabaciones fueron transformadas en un formato uniforme, con duración fija y características acústicas enriquecidas mediante su conversión a mel-espectrogramas. Estas representaciones fueron almacenadas como archivos en formato `.pt`, listos para su utilización en el entrenamiento de modelos de clasificación de toses.

2.1.3 Data Split

Esta etapa consistió en dividir el conjunto de datos preprocesado en subconjuntos separados para entrenamiento, prueba y validación, lo que permitió evaluar el desempeño del modelo en datos no vistos y garantizar su capacidad de generalización. Para ello, se utilizó una división inicial junto con un esquema de validación cruzada basado en el algoritmo *k-fold cross-validation*. A continuación, se describen los pasos realizados en esta etapa:

División de Datos

El conjunto de datos de mel-espectrogramas fue inicialmente dividido en dos subconjuntos principales. El *conjunto de entrenamiento* representó el 80% del total de datos y se utilizó para ajustar los parámetros del modelo durante el proceso de entrenamiento. Por otro lado, el *conjunto de prueba* representó el 20% restante y se empleó para evaluar el desempeño del modelo en datos no utilizados durante la fase de entrenamiento.

La división se realizó de manera aleatoria utilizando la función `torch.utils.data.random_split`, asegurando que ambas particiones fueran mutuamente exclusivas y que conservaran la distribución original de las etiquetas. Sin embargo, para mejorar la estabilidad de la evaluación y reducir la dependencia en una sola división de datos, se implementó validación cruzada.

Validación Cruzada con K-Fold Cross Validation

Para mejorar la estimación del desempeño del modelo, se utilizó el algoritmo *k-fold cross-validation* con $k = 8$. Este método permitió dividir el conjunto de entrenamiento en ocho subconjuntos de igual tamaño, de manera que en cada iteración, uno de los subconjuntos fue utilizado como conjunto de validación, mientras que los siete restantes fueron empleados para entrenamiento. Al finalizar, el modelo fue evaluado en cada uno de los ocho subconjuntos, obteniendo un promedio de las métricas de desempeño.

El procedimiento con $k = 8$ permitió garantizar que cada observación participara en el entrenamiento y en la validación en diferentes momentos, reduciendo el impacto de una selección sesgada de datos. Este enfoque también contribuyó a mejorar la estabilidad de las métricas y a reducir la varianza de los resultados.

Uso de DataLoaders

Para facilitar la entrega de los datos al modelo durante el entrenamiento y la evaluación, se emplearon `DataLoaders`, que permitieron crear lotes de datos y manejarlos de manera eficiente en memoria. Durante el entrenamiento, se definió un tamaño de lote de 16, procesando simultáneamente dieciséis ejemplos en cada iteración. Además, en el conjunto de entrenamiento, se activó el parámetro `shuffle=True` para asegurar que los datos fueran mezclados en cada época, lo que ayudó a mejorar la generalización del modelo. En contraste, en el conjunto de prueba y en las iteraciones de validación cruzada, se desactivó la mezcla de datos (`shuffle=False`) para garantizar la consistencia en la evaluación.

Importancia de una Correcta División de Datos

La correcta separación de los datos fue vital para evitar el sobreajuste y garantizar que el modelo pudiera generalizar adecuadamente. La combinación de un conjunto de prueba independiente y el uso de validación cruzada aseguró que los resultados reflejaran el desempeño real del modelo y no estuvieran sesgados por una única partición de los datos. Este enfoque permitió obtener un análisis más confiable de la capacidad de generalización del modelo y facilitó la comparación con otros métodos de clasificación de señales de tos.

2.1.4 Modelos de Aprendizaje Automático

En este estudio, se emplearon dos arquitecturas de redes neuronales para la clasificación de toses: *CoughNet*, una red completamente conectada basada en características escalares, y *CoughNetwithCNN*, un modelo híbrido que combina espectrogramas y cromagramas con capas convolucionales. Ambas arquitecturas fueron diseñadas para evaluar el impacto de diferentes representaciones acústicas en la clasificación de toses.

Arquitectura de *CoughNet*

La arquitectura *CoughNet* detallada en el Apéndice B1, consiste en una red neuronal completamente conectada que recibe como entrada características escalares extraídas de la señal de tos. Está compuesta por múltiples capas densas con activaciones `ReLU`, organizadas en una estructura decreciente `512-256-128-64-16-2`, finalizando con una capa

lineal de salida para la clasificación binaria. Este modelo se basa únicamente en parámetros estadísticos, sin emplear representaciones espectrales.

Arquitectura de *CoughNetwithCNN*

El modelo *CoughNetwithCNN* detallada en el Apéndice B2, fusiona características escalares con representaciones espectrales mediante dos ramas convolucionales independientes para espectrogramas y cromagramas. La rama del espectrograma emplea capas `Conv2d`+`ReLU`+`MaxPooling`, seguidas de `AdaptiveAvgPool2d` para reducir la dimensionalidad antes de la fusión con la rama del cromagrama, que sigue un esquema similar. La información combinada se procesa en capas densas que reducen progresivamente la dimensionalidad antes de la clasificación.

Configuración del Entrenamiento

Los datos se dividieron en 80% para entrenamiento y 20% para prueba. Se implementó validación cruzada con *k-fold* para mejorar la generalización. Se utilizó la función de pérdida `CrossEntropyLoss` y el optimizador Adam (*Adaptive Moment Estimation*), una variante mejorada del Gradiente Descendente Estocástico (SGD), con tasa de aprendizaje de 1×10^{-3}. El entrenamiento se realizó con 15 épocas, tamaño de lote 16, activación `ReLU` y regularización con `Dropout` (0.5 en las capas densas). Este esquema permitió evaluar el rendimiento de cada modelo en múltiples particiones de los datos, asegurando que no estuvieran sobreajustados y mejorando su capacidad de generalización.

Comparación con Otros Clasificadores

Para evaluar la efectividad del modelo basado en redes neuronales convolucionales, se comparó su desempeño con tres clasificadores tradicionales: *Naïve Bayes*, *Support Vector Machine (SVM)* y *Random Forest*. Cada modelo fue entrenado utilizando las mismas características acústicas extraídas de las señales de tos y se validó mediante *k-fold cross-validation* con $k = 4$. Se aplicó normalización con `StandardScaler` y se utilizó la función de pérdida `CrossEntropyLoss`, con una configuración de 15 épocas, un tamaño de lote de 16 y una tasa de aprendizaje de 1×10^{-3}.

El clasificador *Naïve Bayes* asumió una distribución gaussiana de los datos, mientras que el *SVM* se configuró con un núcleo polinomial. Por su parte, el modelo *Random Forest* utilizó 100 árboles de decisión con una profundidad máxima de 10. La comparación con estos modelos permitió establecer un punto de referencia para medir la efectividad del enfoque basado en aprendizaje profundo, considerando diferentes estrategias de clasificación. Esta evaluación validó la relevancia del uso de redes neuronales convolucionales en la tarea de clasificación de toses, destacando su capacidad para capturar información representativa a partir de representaciones espectrales de audio.

2.1.5 Ajuste de Hiperparámetros

El ajuste de hiperparámetros se realizó con el objetivo de optimizar el desempeño del modelo, explorando combinaciones que mejoraran su capacidad de generalización. En este proyecto, se ajustaron la tasa de aprendizaje, el momento, el tamaño de lote, la cantidad de filtros en capas convolucionales y la tasa de `dropout`. Se probaron valores como 0.001, 0.01 y 0.1 para la tasa de aprendizaje, momentos entre 0.8 y 0.95, tamaños de lote de 8 a 32, configuraciones de filtros $16 \rightarrow 32 \rightarrow 64$ y $32 \rightarrow 64 \rightarrow 128$, y tasas de `dropout` entre 0.2 y 0.5.

Para la optimización, se emplearon dos estrategias: *Grid Search*, que evaluó combinaciones predefinidas de hiperparámetros, y *Random Search*, que seleccionó configuraciones aleatorias dentro de un rango, reduciendo costos computacionales. Cada combinación se evaluó mediante validación cruzada con $k = 8$, utilizando `CrossEntropyLoss` y el optimizador Adam. Se implementó *early stopping* para evitar sobreajuste, deteniendo el entrenamiento si la pérdida en validación no mejoraba en iteraciones consecutivas.

El modelo entrenado con la mejor configuración se seleccionó según métricas como precisión y pérdida, asegurando un balance óptimo entre rendimiento y estabilidad. Este proceso fue clave para maximizar la efectividad del modelo en la clasificación de toses.

2.1.6 Evaluación del Rendimiento

La evaluación del rendimiento del modelo se llevó a cabo utilizando un conjunto de prueba compuesto por ejemplos no vistos durante el entrenamiento ni en el ajuste de hiperparámetros. Esta estrategia aseguró una evaluación imparcial y proporcionó una

medida objetiva de la capacidad de generalización del modelo. Para mejorar la estabilidad del proceso de evaluación, se aplicó validación cruzada con $k = 8$, lo que permitió obtener métricas más robustas y reducir la variabilidad en los resultados.

El modelo fue evaluado en términos de su capacidad de clasificación, utilizando la función de pérdida `CrossEntropyLoss` y métricas de desempeño obtenidas a partir de la matriz de confusión. Se implementó un procedimiento sistemático donde el modelo fue sometido a múltiples iteraciones de prueba y se registraron los valores promedio de desempeño en cada subconjunto de la validación cruzada.

Además, se verificó el comportamiento del modelo frente a datos ruidosos y variaciones en la señal de entrada, analizando su robustez en escenarios con condiciones acústicas distintas. Finalmente, los resultados fueron comparados con otros clasificadores para contextualizar el desempeño de la red neuronal convolucional en relación con enfoques tradicionales de aprendizaje automático.

2.1.7 Mejor Modelo

La selección del mejor modelo se basó en la evaluación comparativa de distintas arquitecturas a través de validación cruzada y análisis de métricas de clasificación. Se consideraron aspectos como precisión, sensibilidad, especificidad para garantizar un desempeño equilibrado en todas las clases.

Para analizar el comportamiento del modelo, se utilizó la matriz de confusión, lo que permitió identificar patrones de error y ajustar la estrategia de entrenamiento. Además, se monitorearon las curvas de pérdida en función del número de épocas, asegurando una convergencia estable y evitando el sobreajuste mediante la implementación de *early stopping*.

El modelo final fue seleccionado con base en su desempeño en validación cruzada con $k = 8$, priorizando la estabilidad y capacidad de generalización. Se aplicaron técnicas de regularización, como `Dropout`, y optimización con el algoritmo `Adam` para mejorar la robustez del entrenamiento.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

En este capítulo se presentan los resultados obtenidos con el modelo *CoughNet* durante la validación cruzada con $k = 8$. Se incluyen curvas de pérdida durante el entrenamiento y evaluación, matrices de confusión y un análisis descriptivo de los resultados obtenidos.

3.1 Curvas de Pérdida y Validación

Las Figuras 3.1 a 3.8 muestran las curvas de pérdida para cada fold durante el proceso de entrenamiento y validación. En estas gráficas, la curva azul representa la pérdida en el conjunto de entrenamiento, mientras que la curva naranja representa la pérdida en el conjunto de validación.

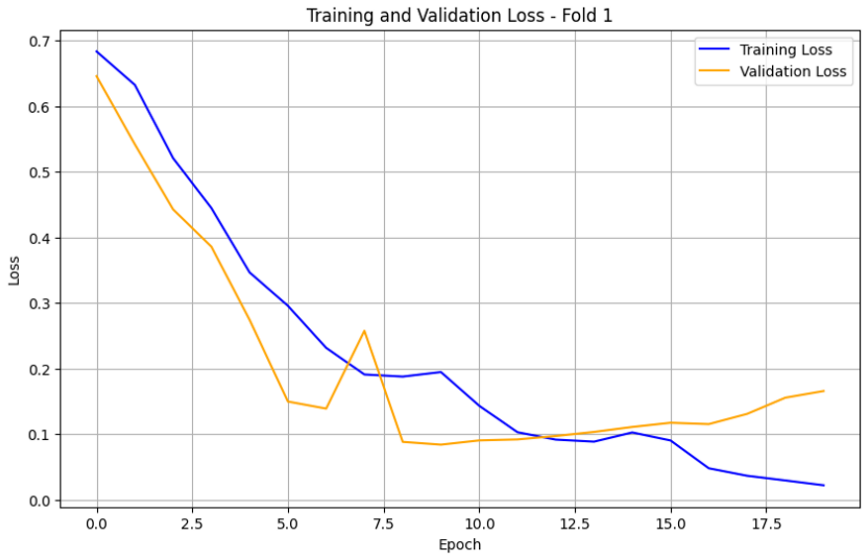


Figura 3.1. Curva de pérdida y validación para el Fold 1 del modelo *CoughNet*. Se observa una disminución progresiva de la pérdida en el conjunto de entrenamiento, mientras que la pérdida en el conjunto de validación se estabiliza.

Figura 3.2. Curva de pérdida y validación para el Fold 2 del modelo *CoughNet*. Se aprecia una convergencia adecuada entre las curvas de entrenamiento y validación.

Figura 3.3. Curva de pérdida y validación para el Fold 3 del modelo *CoughNet*. Se observa estabilidad en la curva de validación, aunque con ligeros picos que indican fluctuaciones en la generalización.

Figura 3.4. Curva de pérdida y validación para el Fold 4 del modelo *CoughNet*. Se observa una disminución significativa de la pérdida en el entrenamiento, mientras que la validación presenta estabilidad.

Figura 3.5. Curva de pérdida y validación para el Fold 5 del modelo *CoughNet*. La validación muestra ligeras fluctuaciones pero mantiene una tendencia estable.

Figura 3.6. Curva de pérdida y validación para el Fold 6 del modelo *CoughNet*. Se observa una buena convergencia entre entrenamiento y validación.

Figura 3.7. Curva de pérdida y validación para el Fold 7 del modelo *CoughNet*. Presenta un descenso pronunciado en la pérdida de entrenamiento, mientras que la validación mantiene estabilidad.

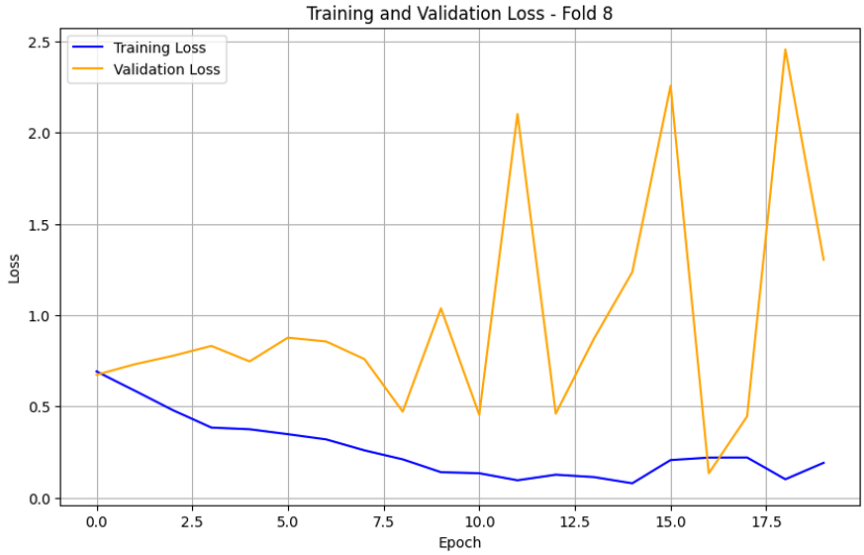


Figura 3.8. Curva de pérdida y validación para el Fold 8 del modelo *CoughNet*. Se identifican fluctuaciones en la validación que pueden indicar sobreajuste.

3.2 Matrices de Confusión

Las matrices de confusión de los diferentes folds permiten visualizar el desempeño del modelo en términos de predicciones correctas y errores de clasificación. Las Figuras 3.9 a 3.16 presentan las matrices de confusión correspondientes a cada fold.

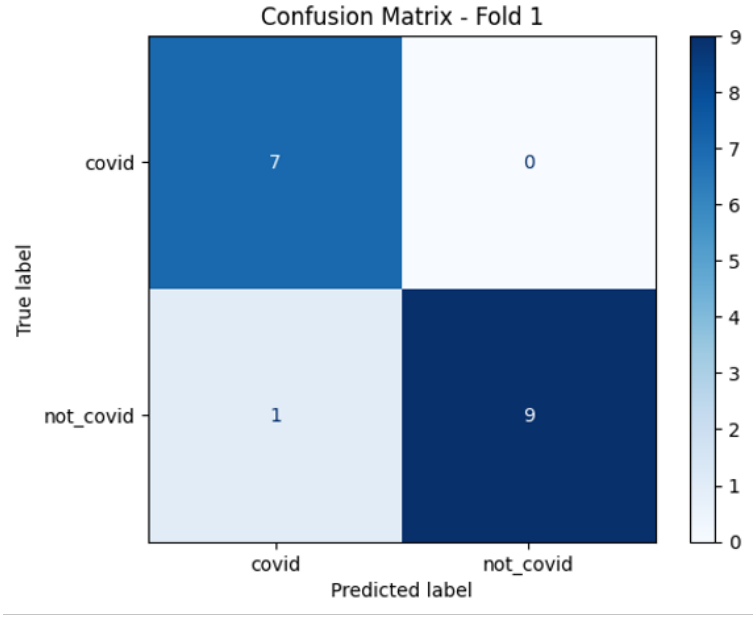


Figura 3.9. Matriz de confusión para el Fold 1 del modelo *CoughNet*. Se observa un alto nivel de precisión en la clasificación de ambas clases.

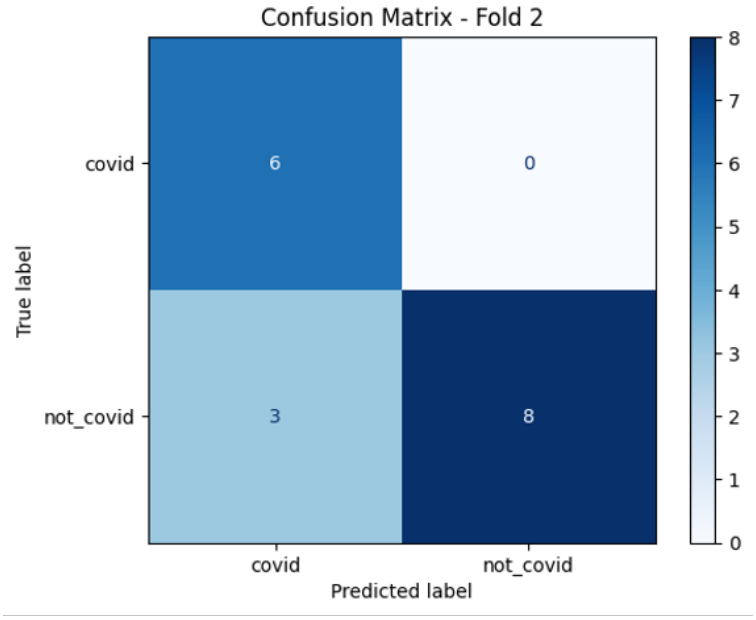


Figura 3.10. Matriz de confusión para el Fold 2 del modelo *CoughNet*. Se observan algunos errores de clasificación en la clase negativa, pero con un desempeño general aceptable.

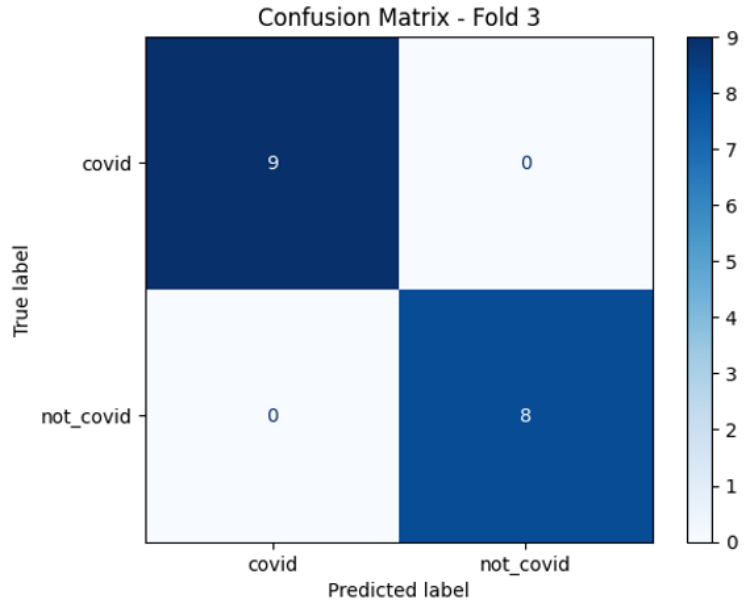


Figura 3.11. Matriz de confusión para el Fold 3 del modelo *CoughNet*. Se observa una alta precisión en ambas clases.

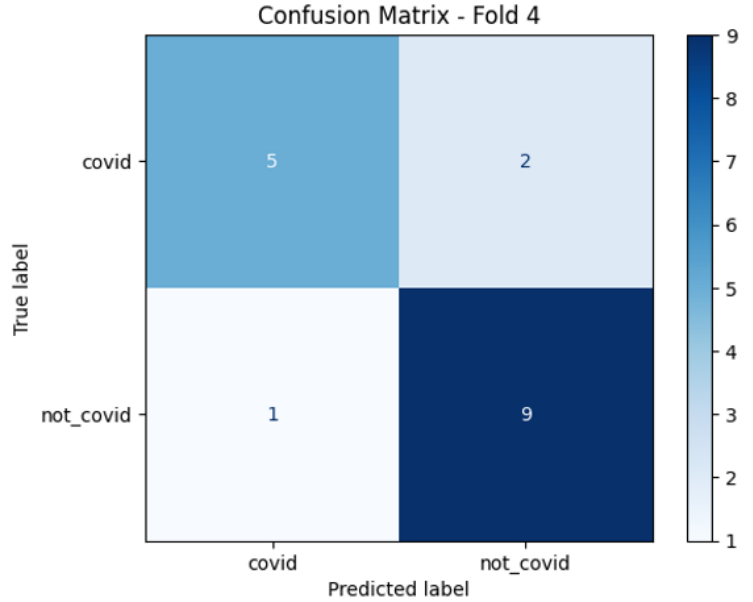


Figura 3.12. Matriz de confusión para el Fold 4 del modelo *CoughNet*. Se identifican algunos errores en la clasificación de la clase positiva.

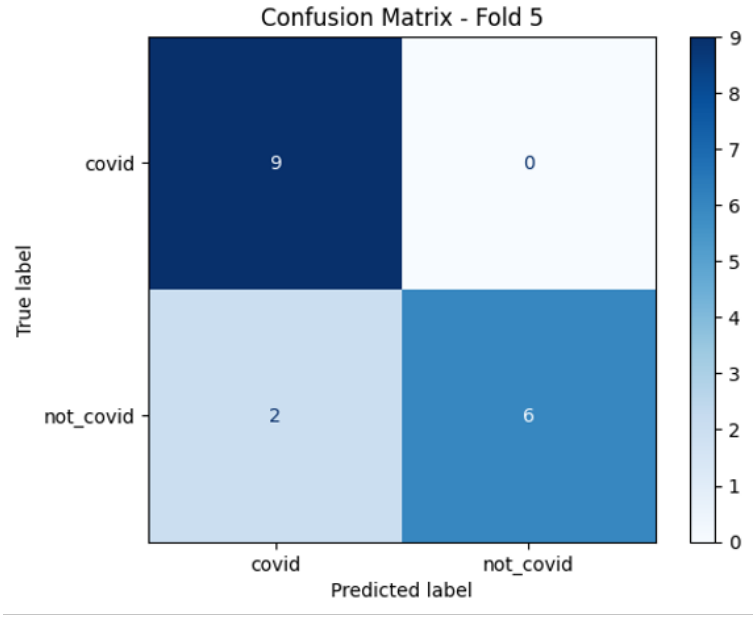


Figura 3.13. Matriz de confusión para el Fold 5 del modelo *CoughNet*. Se observa un buen balance en la predicción de ambas clases.

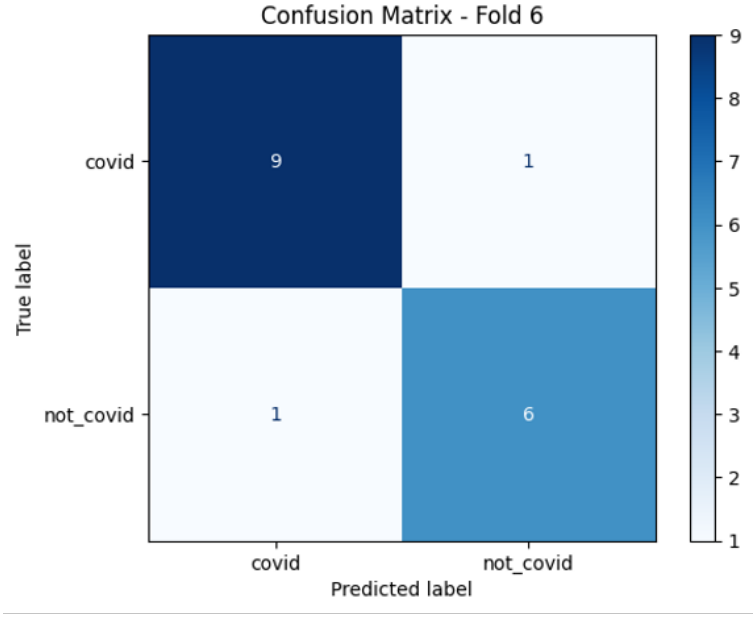


Figura 3.14. Matriz de confusión para el Fold 6 del modelo *CoughNet*. Se observa un desempeño consistente con errores mínimos.

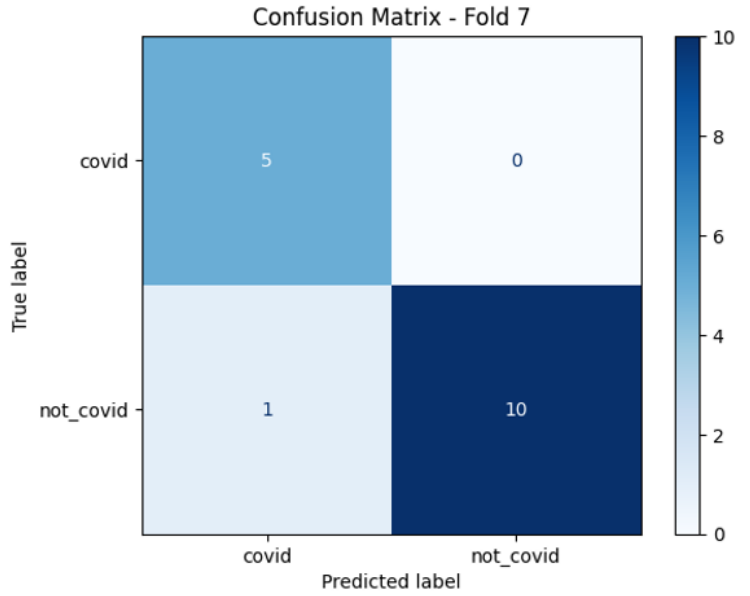


Figura 3.15. Matriz de confusión para el Fold 7 del modelo *CoughNet*. Se mantiene un nivel adecuado de predicciones correctas.

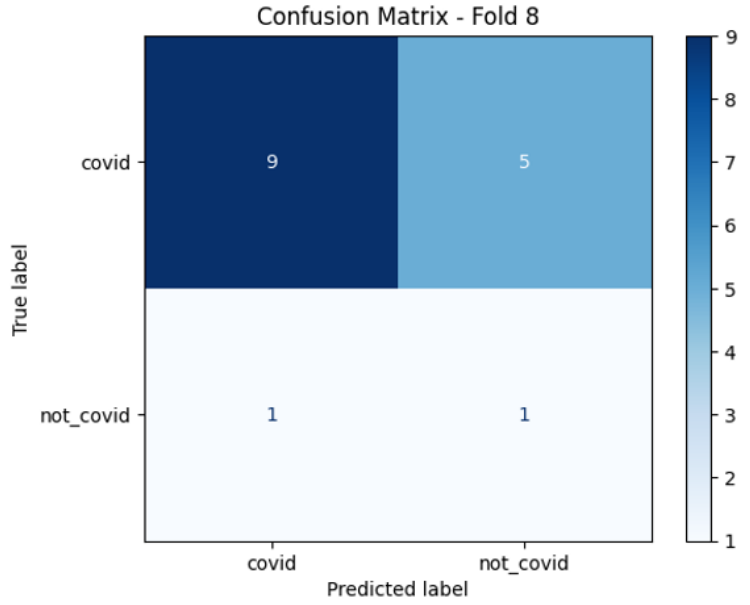


Figura 3.16. Matriz de confusión para el Fold 8 del modelo *CoughNet*. Se presentan errores en la clase positiva.

3.3 Análisis Preliminar de Resultados

El modelo *CoughNet* fue evaluado mediante validación cruzada $k = 8$, obteniendo resultados variables en los diferentes folds. La Tabla 3.1 resume la exactitud de entrenamiento y prueba para cada fold, lo que permite evaluar la estabilidad del modelo en distintas particiones de los datos.

Fold	**Exactitud en Entrenamiento**	**Exactitud en Prueba**
1	99.15%	94.12%
2	99.15%	82.35%
3	96.58%	100.00%
4	98.29%	82.35%
5	98.29%	88.24%
6	98.29%	88.24%
7	97.46%	93.75%
8	92.37%	62.50%
Average	97.45	86.44

Tabla 3.1. Resultados de exactitud en entrenamiento y prueba para cada fold en la validación cruzada. Se observa variabilidad en el rendimiento, indicando diferencias en la distribución de datos en los subconjuntos.

Este análisis preliminar permite identificar patrones en el comportamiento del modelo, como su capacidad de generalización y la presencia de posibles casos de sobreajuste en algunos folds. En la siguiente sección, se comparará el rendimiento de *CoughNet* con otros modelos de referencia.

3.4 Comparación con Otros Clasificadores

Se comparó el desempeño del modelo *CoughNet* con otros clasificadores tradicionales, incluyendo Naive Bayes, Support Vector Machines y Random Forest. La Tabla 3.2 resume los resultados obtenidos.

Modelo	Train Accuracy	Test Accuracy
Naive Bayes	77.63%	75.51%
SVM	81.34%	75.45%
Random Forest	100.00%	87.30%

Tabla 3.2. Comparación de desempeño entre *CoughNet* y otros clasificadores. Se presentan los valores promedio de exactitud en entrenamiento y prueba.

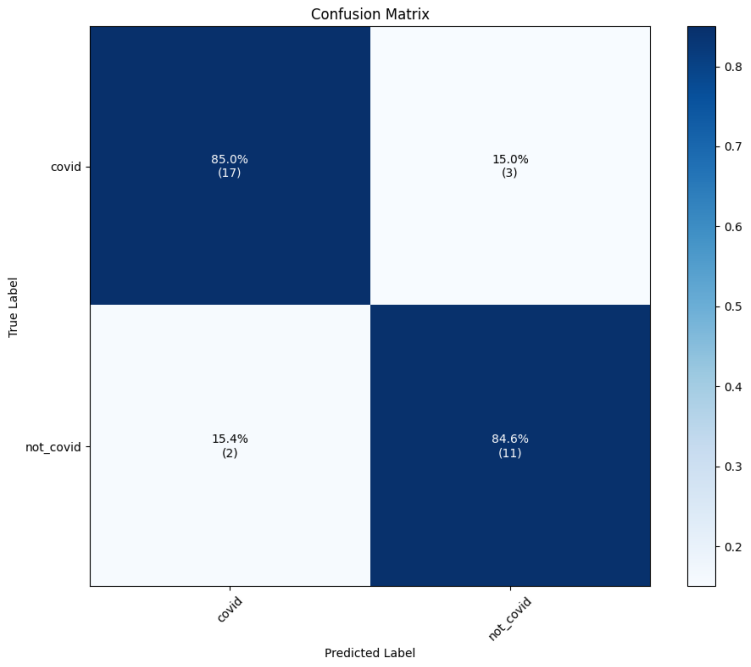


Figura 3.17. Resultados de validación cruzada para Naive Bayes. Se observó un rendimiento inferior en comparación con *CoughNet*.

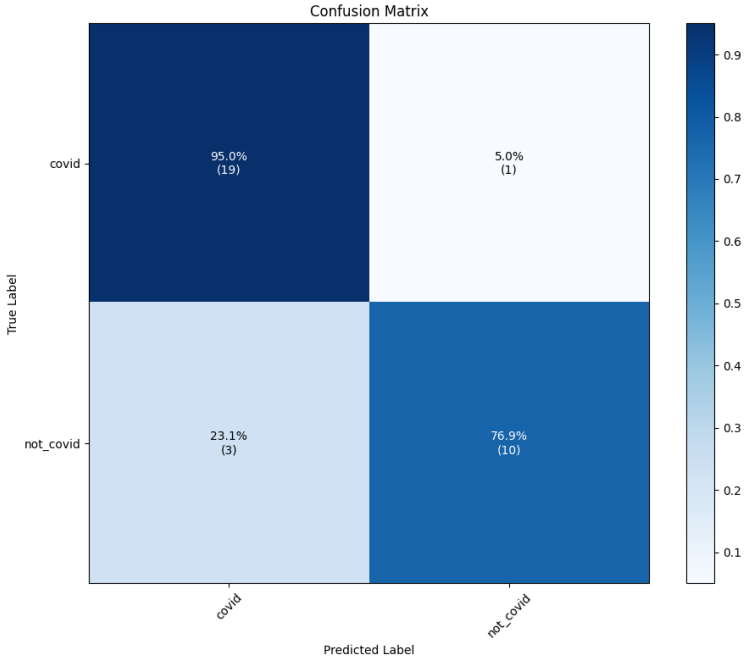


Figura 3.18. Resultados de validación cruzada para Support Vector Machines. Se obtuvo un desempeño moderado en la clasificación.

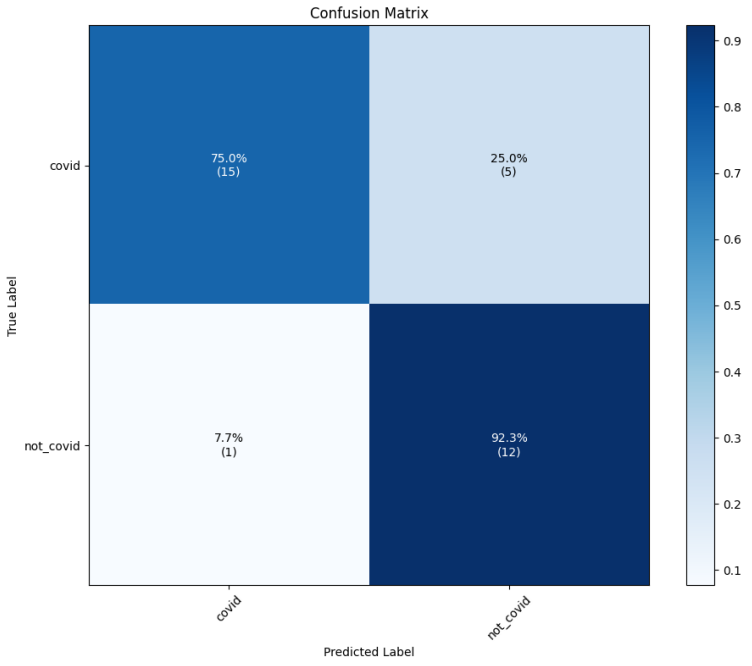


Figura 3.19. Resultados de validación cruzada para Random Forest. Se logró una exactitud alta en entrenamiento y prueba.

3.5 Evaluación del Modelo 2: CoughNetwithCNN

El modelo *CoughNetwithCNN* combinó características escalares, espectrogramas de Mel y cromagramas para mejorar la clasificación de toses. Su arquitectura integró tres ramas de procesamiento: una red completamente conectada para características escalares, una CNN para espectrogramas y otra CNN para cromagramas. Tras la extracción de características, las salidas se fusionaron en capas densas para la clasificación final.

El modelo fue evaluado con validación cruzada ($k = 8$). La Tabla 3.3 resume los resultados obtenidos.

Fold	**Exactitud en Entrenamiento**	**Exactitud en Prueba**
0	94.02%	88.24%
1	98.29%	88.24%
2	98.29%	100.00%
3	96.58%	82.35%
4	96.58%	88.24%
5	95.73%	82.35%
6	95.76%	93.75%
7	96.61%	75.00%
Promedio	96.48%	87.27%

Tabla 3.3. Exactitud en entrenamiento y prueba para cada fold en la validación cruzada del modelo *CoughNetwithCNN*.

El modelo mostró una exactitud promedio de prueba del 87.27%, superior a *CoughNet*, lo que sugiere que la combinación de representaciones acústicas mejoró la discriminación de clases.

CAPÍTULO 4

4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

El presente trabajo ha desarrollado un sistema de clasificación de toses basado en redes neuronales, evaluando distintas arquitecturas y comparándolas con modelos tradicionales de aprendizaje automático. Los resultados han permitido extraer las siguientes conclusiones:

- Se demostró que el modelo híbrido *CoughNetwithCNN*, que combina características escalares, espectrogramas de Mel y cromagramas, superó en rendimiento al modelo *CoughNet* basado únicamente en características escalares. El modelo híbrido alcanzó una exactitud promedio de entrenamiento del 96.48% y una exactitud en prueba del 87.27%, mientras que *CoughNet* obtuvo una exactitud de entrenamiento del 97.45% y una exactitud en prueba del 86.44%. Esto indica que la incorporación de representaciones espectrales aporta información relevante para la clasificación de toses, mejorando la capacidad de generalización del modelo;

- En la comparación con clasificadores tradicionales como *Naïve Bayes*, *Support Vector Machines (SVM)* y *Random Forest*, se evidenció que las redes neuronales convolucionales fueron más efectivas en la extracción de patrones acústicos complejos. En particular, el modelo *Random Forest* obtuvo una exactitud de entrenamiento del 100.00% y una exactitud en prueba del 87.30%, similar a *CoughNetwithCNN*, lo que sugiere que para ciertos conjuntos de datos, métodos más simples pueden ser competitivos si se emplean características bien seleccionadas;

- El análisis de las matrices de confusión reveló que los modelos basados en redes neuronales presentaron mejor sensibilidad y especificidad en la clasificación de toses relacionadas con COVID-19, reduciendo los falsos negativos en comparación con modelos tradicionales. Sin embargo, se observó una variabilidad significativa en los

resultados de los diferentes folds, lo que sugiere que el modelo aún es sensible a la distribución de los datos de entrenamiento;

- Se identificó que ciertos problemas en los datos, como la presencia de ruido y etiquetas incorrectas en las muestras de entrenamiento, afectaron el rendimiento de los modelos evaluados. La validación cruzada ayudó a mitigar estos efectos, pero futuras mejoras en la curación de datos podrían aumentar la robustez del clasificador;

- El modelo de redes neuronales convolucionales desarrollado logró identificar patrones diferenciadores entre toses de pacientes positivos y negativos para COVID-19, lo que sugiere su potencial como herramienta complementaria en el diagnóstico temprano de la enfermedad;

- El sistema diseñado ofrece una solución no invasiva y accesible para el diagnóstico de COVID-19, permitiendo obtener resultados a partir de grabaciones de toses. Esto abre la posibilidad de su integración en aplicaciones móviles o plataformas de telemedicina para la detección rápida y económica en contextos de atención primaria o autoevaluación.

4.2 Recomendaciones

A partir de las conclusiones obtenidas, se plantean las siguientes recomendaciones para futuros trabajos y mejoras en el modelo:

- Optimizar la calidad del conjunto de datos mediante técnicas de preprocesamiento avanzadas, como filtrado de ruido, normalización de amplitudes y segmentación más precisa de los eventos de tos;

- Explorar arquitecturas más sofisticadas, como modelos basados en transformadores o redes neuronales recurrentes (*RNNs*), que podrían capturar mejor las características temporales de las señales de audio;

- Implementar técnicas de aumento de datos (*data augmentation*) para mejorar la generalización del modelo, generando variaciones sintéticas de las toses a partir de transformaciones en la frecuencia y el tiempo;

- Ampliar la evaluación del modelo con bases de datos más representativas y en condiciones más cercanas al uso real, asegurando que el clasificador mantenga un rendimiento estable en entornos clínicos o en aplicaciones móviles;
- Comparar el desempeño del modelo con estudios previos en clasificación de toses para identificar oportunidades de mejora y validar su utilidad en el diagnóstico temprano de enfermedades respiratorias;
- Se sugiere la incorporación de metadatos adicionales, como la presencia de síntomas, historial médico y otros factores de salud, para mejorar la capacidad de predicción del modelo y contextualizar mejor la clasificación de los pacientes.
- Dado que algunos folds presentaron sobreajuste, sería conveniente investigar estrategias adicionales de regularización, como el uso de dropout más agresivo, penalización en los pesos (*L2 regularization*) o incluso la implementación de técnicas de aprendizaje semi-supervisado.
- Para aplicaciones en el mundo real, se recomienda desarrollar una interfaz de usuario que permita la recopilación de grabaciones de tos en tiempo real, asegurando que los datos sean capturados en condiciones controladas para minimizar el impacto del ruido ambiental.
- En trabajos futuros, se podrían explorar arquitecturas híbridas más avanzadas que combinen modelos basados en aprendizaje profundo con enfoques de *transfer learning* a partir de modelos preentrenados en reconocimiento de audio.
- Finalmente, se sugiere realizar pruebas piloto en entornos clínicos para evaluar la viabilidad del modelo en un contexto real de diagnóstico y ajustar su desempeño en función de las necesidades del personal médico y los pacientes.

BIBLIOGRAFÍA

Arora, S. (2013). Gradient descent: Offline, online, and randomly. Lecture notes, Princeton University. Available online at https://www.cs.princeton.edu/courses/archive/fall13/cos521/lectures/lec19.pdf.

Ashish Vaswani, Noam Shazeer, N. P. J. U. L. J. A. N. G. . K. and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010. Curran Associates Inc.

Brown, C. and Johnson, E. (2021). Coughvid: Using ai to detect covid-19 through audio analysis. *Journal of Medical AI*, 8(2):123–130.

ESPOL (2023). La inteligencia artificial fue el tema central de la ii edición del congreso de transformación digital inhouse tech espol. In *Congreso de Transformación Digital: INHOUSE TECH ESPOL*. Accedido el 2 de febrero de 2025.

Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., and Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118.

et al., T. B. B. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:1877–1901.

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory*. John Wiley Sons.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558.

Instituto Nacional de Salud Pública e Investigación (2022). Proyectos de investigación relacionados con covid-19 en ecuador. Recuperado de https://www.inspi.gob.ec.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. PMLR.

Islam, T., Sheakh, M. A., Tahosin, M. S., Hena, M. H., Akash, S., Bin Jardan, Y., FentahunWondmie, G., Nafidi, H.-A., and Bourhia, M. (2024). Predictive modeling for breast cancer classification in the context of bangladeshi patients by use of machine learning approach with explainable ai. *Scientific Reports*, 14.

Jocher, G. et al. (2023). Yolov8: Ultralytics official repository. Recuperado de https://github.com/ultralytics/ultralytics.

Johnson, M. and Patel, A. (2019). Convolutional neural networks for audio analysis: State of the art. *Audio Processing Today*, 8(4):123–140.

Kaiming He, Xiangyu Zhang, S. R. and Sun, J. (2016). Deep residual learning for image recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

Laguarta, J., Hueto, I., and Subirana, B. (2020). Covid-19 artificial intelligence diagnosis using only cough recordings. *IEEE Open Journal of Engineering in Medicine and Biology*, 1:275–281.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

Nakamura, O. (1990). Unsupervised learning for image pattern recognition. *Journal of Computer Vision*, 2(4):339–348.

Nitish Srivastava, Geoffrey Hinton, A. K. I. S. and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

OpenAI (2023). Gpt-4 technical report. Recuperado de https://openai.com/research/gpt-4.

Pahar, M., Klopper, M., Warren, R., and Niesler, T. (2021). Covid-19 cough classification using machine learning and global smartphone recordings. *Computers in Biology and Medicine*, 135:104572.

Segovia Jácome, S. W. (2023). Implementación de un sistema de diagnóstico asistido en zonas rurales utilizando dispositivos portátiles y técnicas de aprendizaje automático.

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. *School of Computer Science, Carnegie Mellon University*. Available online at https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Smith, J. and Doe, J. (2020). Acoustic analysis for covid-19 diagnosis: Challenges and advances. *Journal of Medical Acoustics*, 12(3):45–60.

Soloway, D. and Haley, P. J. (1997). Neural generalized predictive control: a newton-raphson implementation. Technical report.

Spotify Research Team (2021). Spotify recommendations: Audio analysis for personalization. Recuperado de https://research.spotify.com.

Sánchez, L. and Rodríguez, P. (2020). Implementación de algoritmos de aprendizaje profundo para la clasificación de datos biomédicos.

Topol, E. J. (2019). High-performance medicine: the convergence of human and artificial intelligence. *Nature Medicine*, 25(1):44–56.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

Wang, W., Shang, Q., and Lu, H. (2023). Automatic covid-19 detection from cough sounds using multi-headed convolutional neural networks. *Applied Sciences*, 13:6976.

APÉNDICES

APÉNDICE A

A.1 Derivadas Matriciales y Gradientes

A continuación, se presentan las demostraciones para justificar los gradientes de la función objetivo que se busca minimizar en la representación del modelo.

Definición 1

Sea A de dimensiones $m \times n$ y B de dimensiones $n \times p$. El producto AB se define como:

$$C = AB, \tag{A.1}$$

donde C es una matriz $m \times p$, cuyo elemento (i, j) está dado por:

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \tag{A.2}$$

para todo $i = 1, 2, \ldots, m$ y $j = 1, 2, \ldots, p$.

Proposición 1: Sea A de dimensiones $m \times n$ y $\mathbf{x}$ un vector columna de dimensiones $n \times 1$. Entonces el producto:

$$\mathbf{z} = A\mathbf{x}, \tag{A.3}$$

está dado por:

$$z_i = \sum_{k=1}^{n} a_{ik} x_k, \tag{A.4}$$

para todo $i = 1, 2, \ldots, m$.

De manera similar, sea $\mathbf{y}$ un vector columna de dimensiones $m \times 1$, entonces el producto:

$$\mathbf{z}^T = \mathbf{y}^T A, \tag{A.5}$$

está dado por:

$$z_i = \sum_{k=1}^{n} a_{ki} y_k, \tag{A.6}$$

para todo $i = 1, 2, \ldots, n$.

Finalmente, el escalar resultante del producto:

$$\alpha = \mathbf{y}^T A\mathbf{x}, \tag{A.7}$$

está dado por:

$$\alpha = \sum_{j=1}^{m}\sum_{k=1}^{n} a_{jk} y_j x_k. \tag{A.8}$$

Demostración: Estas son aplicaciones directas de la Definición 1.

Definición 2: Sea $\mathbf{y} = \psi(\mathbf{x})$, donde $\mathbf{y} \in \mathbb{R}^m$ y $\mathbf{x} \in \mathbb{R}^n$. La matriz Jacobiana de primer orden se define como:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \frac{\partial y_m}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}. \tag{A.9}$$

Proposición 2: Si $\mathbf{y} = A\mathbf{x}$, donde $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$, y $A \in \mathbb{R}^{m\times n}$, entonces:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = A. \tag{A.10}$$

Demostración: El i-ésimo elemento de $\mathbf{y}$ está dado por:

$$y_i = \sum_{k=1}^{n} a_{ik} x_k. \tag{A.11}$$

Derivando y_i respecto a x_j, se obtiene:

$$\frac{\partial y_i}{\partial x_j} = a_{ij}, \tag{A.12}$$

para $i = 1, 2, \ldots, m$ y $j = 1, 2, \ldots, n$. Por lo tanto, la derivada total es:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = A. \tag{A.13}$$

Proposición 3: *Si* $\mathbf{y} = A\mathbf{x}$*, donde* A *no depende de* $\mathbf{x}$ *y* $\mathbf{x}$ *es función de* $\mathbf{z}$*, entonces:*

$$\frac{\partial \mathbf{y}}{\partial \mathbf{z}} = A\frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.14}$$

Demostración: El i-ésimo elemento de $\mathbf{y}$ está dado por:

$$y_i = \sum_{k=1}^{n} a_{ik}x_k. \tag{A.15}$$

Dado que x_k depende de z_j, derivamos y_i respecto a z_j:

$$\frac{\partial y_i}{\partial z_j} = \sum_{k=1}^{n} a_{ik}\frac{\partial x_k}{\partial z_j}. \tag{A.16}$$

En notación matricial, lo anterior equivale a:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{z}} = A\frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.17}$$

Proposición 4 *Sea el escalar* α *definido por:*

$$\alpha = \mathbf{y}^T A\mathbf{x}, \tag{A.18}$$

donde $\mathbf{y}$ *es* $m \times 1$*,* $\mathbf{x}$ *es* $n \times 1$*,* A *es* $m \times n$*, y* A *es independiente de* $\mathbf{x}$ *y* $\mathbf{y}$*. Entonces:*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{y}^T A \tag{A.19}$$

y

$$\frac{\partial \alpha}{\partial \mathbf{y}} = \mathbf{x}^T A^T. \tag{A.20}$$

Demostración: Definiendo:

$$W = \mathbf{y}^T A, \tag{A.21}$$

se tiene que:

$$\alpha = W\mathbf{x}. \tag{A.22}$$

Por la Proposición 2, se deduce que:

$$\frac{\partial \alpha}{\partial \mathbf{x}} = W = \mathbf{y}^T A. \tag{A.23}$$

Dado que α es un escalar, también se puede escribir como:

$$\alpha = \alpha^T = \mathbf{x}^T A^T \mathbf{y}. \tag{A.24}$$

Aplicando nuevamente la Proposición 2, se obtiene:

$$\frac{\partial \alpha}{\partial \mathbf{y}} = \mathbf{x}^T A^T. \tag{A.25}$$

Proposición 5 *Para el caso especial en el que el escalar α está dado por la forma cuadrática:*

$$\alpha = \mathbf{x}^T A \mathbf{x}, \tag{A.26}$$

donde $\mathbf{x}$ es $n \times 1$, A es $n \times n$, y A no depende de $\mathbf{x}$, entonces:

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T (A + A^T). \tag{A.27}$$

Demostración: Por definición:

$$\alpha = \sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} x_i x_j. \tag{A.28}$$

Diferenciando respecto al k-ésimo elemento de $\mathbf{x}$, se tiene:

$$\frac{\partial \alpha}{\partial x_k} = \sum_{j=1}^{n} a_{kj} x_j + \sum_{i=1}^{n} a_{ik} x_i. \tag{A.29}$$

Para todo $k = 1, 2, \ldots, n$, esto se puede escribir como:

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T A^T + \mathbf{x}^T A = \mathbf{x}^T (A + A^T). \tag{A.30}$$

Proposición 6 *Para el caso especial donde A es una matriz simétrica, se tiene:*

$$\alpha = \mathbf{x}^T A \mathbf{x}, \tag{A.31}$$

y la derivada de α respecto a $\mathbf{x}$ es:

$$\frac{\partial \alpha}{\partial \mathbf{x}} = 2\mathbf{x}^T A. \tag{A.32}$$

Demostración: Por la Proposición 8, sabemos que la derivada de una forma cuadrática general es:

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T (A + A^T). \tag{A.33}$$

Dado que A es simétrica, se cumple que $A = A^T$. Sustituyendo en la expresión anterior, obtenemos:

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T(A + A) = 2\mathbf{x}^T A. \tag{A.34}$$

Proposición 7 *Sea el escalar α definido por:*

$$\alpha = \mathbf{y}^T\mathbf{x}, \tag{A.35}$$

donde $\mathbf{y}$ y $\mathbf{x}$ son funciones del vector $\mathbf{z}$. Entonces:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T\frac{\partial \mathbf{y}}{\partial \mathbf{z}} + \mathbf{y}^T\frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.36}$$

Demostración: Por definición:

$$\alpha = \sum_{j=1}^{n} x_j y_j. \tag{A.37}$$

Diferenciando respecto al k-ésimo elemento de $\mathbf{z}$, se obtiene:

$$\frac{\partial \alpha}{\partial z_k} = \sum_{j=1}^{n}\left(x_j\frac{\partial y_j}{\partial z_k} + y_j\frac{\partial x_j}{\partial z_k}\right). \tag{A.38}$$

Para todo $k = 1, 2, \ldots, n$, esto se puede escribir en forma matricial como:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \frac{\partial \alpha}{\partial \mathbf{y}}\frac{\partial \mathbf{y}}{\partial \mathbf{z}} + \frac{\partial \alpha}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{z}} = \mathbf{x}^T\frac{\partial \mathbf{y}}{\partial \mathbf{z}} + \mathbf{y}^T\frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.39}$$

Proposición 8 *Sea el escalar α definido por:*

$$\alpha = \mathbf{x}^T\mathbf{x}, \tag{A.40}$$

donde $\mathbf{x}$ es una función del vector $\mathbf{z}$. Entonces:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = 2\mathbf{x}^T\frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.41}$$

Demostración: Aplicamos la Proposición 7, donde se establece que la derivada de un producto escalar $\mathbf{y}^T\mathbf{x}$ respecto a $\mathbf{z}$ es:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T\frac{\partial \mathbf{y}}{\partial \mathbf{z}} + \mathbf{y}^T\frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.42}$$

En este caso, $\mathbf{y} = \mathbf{x}$, por lo que:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T \frac{\partial \mathbf{x}}{\partial \mathbf{z}} + \mathbf{x}^T \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.43}$$

Simplificando los términos, se obtiene:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = 2\mathbf{x}^T \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.44}$$

Proposición 9 *Sea el escalar α definido por:*

$$\alpha = \mathbf{y}^T A \mathbf{x}, \tag{A.45}$$

donde $\mathbf{y}$ es $m \times 1$, $\mathbf{x}$ es $n \times 1$, A es $m \times n$, y tanto $\mathbf{y}$ como $\mathbf{x}$ son funciones del vector $\mathbf{z}$, mientras que A no depende de $\mathbf{z}$. Entonces:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T A^T \frac{\partial \mathbf{y}}{\partial \mathbf{z}} + \mathbf{y}^T A \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.46}$$

Demostración: Definiendo:

$$\mathbf{w}^T = \mathbf{y}^T A, \tag{A.47}$$

se tiene:

$$\alpha = \mathbf{w}^T \mathbf{x}. \tag{A.48}$$

Aplicando la Proposición 7, se obtiene:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T \frac{\partial \mathbf{w}}{\partial \mathbf{z}} + \mathbf{w}^T \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.49}$$

Sustituyendo nuevamente para $\mathbf{w}$, llegamos a:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T A^T \frac{\partial \mathbf{y}}{\partial \mathbf{z}} + \mathbf{y}^T A \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.50}$$

Proposición 10 *Sea el escalar α definido por la forma cuadrática:*

$$\alpha = \mathbf{x}^T A \mathbf{x}, \tag{A.51}$$

donde $\mathbf{x}$ es $n \times 1$, A es $n \times n$, y $\mathbf{x}$ es una función del vector $\mathbf{z}$, mientras que A no depende de $\mathbf{z}$. Entonces:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T (A + A^T) \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.52}$$

Demostración: Por la Proposición 9, sabemos que si:

$$\alpha = \mathbf{y}^T A \mathbf{x}, \tag{A.53}$$

entonces la derivada parcial respecto a $\mathbf{z}$ es:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T A^T \frac{\partial \mathbf{y}}{\partial \mathbf{z}} + \mathbf{y}^T A \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.54}$$

En este caso, $\mathbf{y} = \mathbf{x}$, por lo que la expresión se simplifica a:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T A^T \frac{\partial \mathbf{x}}{\partial \mathbf{z}} + \mathbf{x}^T A \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.55}$$

Agrupando términos, se obtiene:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T (A + A^T) \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.56}$$

Proposición 11 *Para el caso especial donde A es una matriz simétrica y:*

$$\alpha = \mathbf{x}^T A \mathbf{x}, \tag{A.57}$$

donde $\mathbf{x}$ es una función del vector $\mathbf{z}$ y A no depende de $\mathbf{z}$. Entonces:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = 2\mathbf{x}^T A \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.58}$$

Demostración: Por la Proposición 10, se tiene que:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T (A + A^T) \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.59}$$

Dado que A es simétrica, se cumple que $A = A^T$. Sustituyendo, obtenemos:

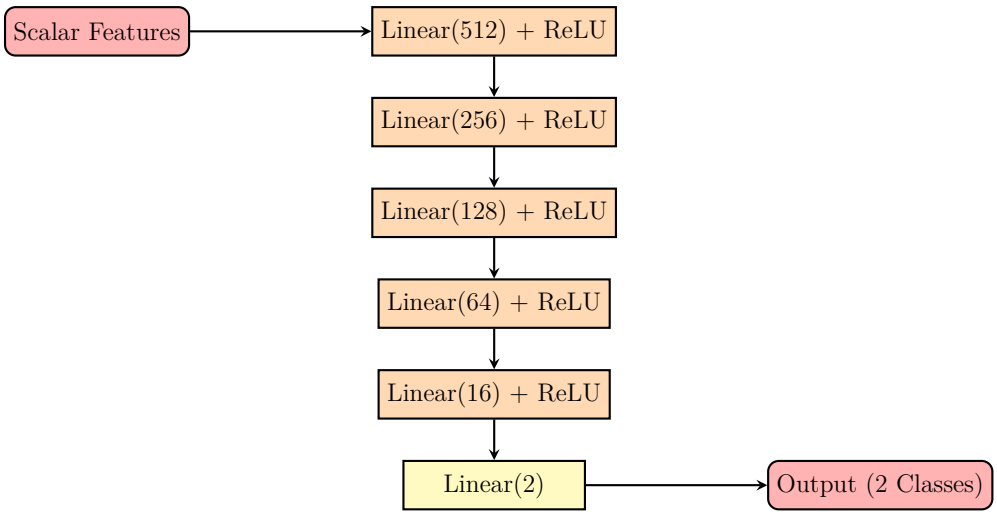
$$\frac{\partial \alpha}{\partial \mathbf{z}} = \mathbf{x}^T (A + A) \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.60}$$

Simplificando:

$$\frac{\partial \alpha}{\partial \mathbf{z}} = 2\mathbf{x}^T A \frac{\partial \mathbf{x}}{\partial \mathbf{z}}. \tag{A.61}$$

APÉNDICE B

B.1 Arquitectura de Red: CoughNet



B.2 Arquitectura de Red: CoughNetwithCNN