Redes Neuronales Convolucionales para la detección de Salmonella *spp*. en aves de granja

Melissa Ortega Alzate

Monografía presentada para optar al título de Especialista en Analítica y Ciencia de Datos

Asesor
David Manuel Villanueva Valdés, Magíster (MSc)

Universidad de Antioquia
Facultad de Ingeniería
Especialización en Analítica y Ciencia de Datos
Medellín, Antioquia, Colombia
2024

Cita	(Ortega Alzate, 2024)
Referencia **Estilo APA 7 (2020)**	Ortega Alzate, M. (2024). R*edes Neuronales Convolucionales para la detección de Salmonella spp. en aves de granja* [Trabajo de grado especialización]. Universidad de Antioquia, Medellín, Colombia.

Especialización en Analítica y Ciencia de Datos, Cohorte VI.

Centro de Investigación Ambientales y de Ingeniería (CIA).

UNIVERSIDAD DE ANTIOQUIA
Vicerrectoría de Docencia

Sistema de Bibliotecas

Centro de Documentación Ingeniería (CENDOI)

Repositorio Institucional: http://bibliotecadigital.udea.edu.co

Universidad de Antioquia - www.udea.edu.co

Rector: John Jairo Arboleda Céspedes.
Decano: Julio Cesar Saldarriaga Molina
Jefe departamento: Diego José Luis Botia Valderrama

Dedicatoria

A mi familia, por su apoyo incondicional.

Agradecimientos

Mi gratitud para Iluma Alliance,
fuente de aprendizaje e inspiración.

Tabla de contenido

Lista de tablas

Lista de figuras

Siglas, acrónimos y abreviaturas

APA	American Psychological Association
AUC	Area Under Curve (Área bajo la curva ROC)
CNN	Convolutional Neural Network (Red Neuronal Convolucional)
DL	Deep Learning (Aprendizaje profundo)
Esp.	Especialista
IA	Inteligencia Artificial
ICA	Instituto Colombiano Agropecuario
FC	Fully-conected (totalmente conectadas)
FP	Falsos positivos
FN	Falsos negativos
ML	Machine Learning (Aprendizaje de máquinas)
OMS	Organización Mundial de la Salud
RNA	Red Neuronal Artificial
ROC	Receiver Operating Characteristic (Característica Operativa del Receptor)
UdeA	Universidad de Antioquia

Resumen

La detección temprana de Salmonella *spp.* en aves de granja es crucial para mejorar la productividad de las compañías avícolas del país, pues no sólo se trata de reducir la mortalidad de sus aves y mantener la rentabilidad y reputación de las empresas, sino de garantizar la producción alimentos inocuos para la población. Sin embargo, la capacidad de respuesta y el costo de los métodos tradicionales limita el desempeño microbiológico de estas compañías.

En esta monografía se compararon diferentes arquitecturas de Redes Neuronales Convolucionales (CNN) para la detección de salmonella *spp*. en imágenes de heces de aves de granja. Se utilizó un conjunto de 5029 imágenes tomadas en África entre 2020 y 2021, y se clasificaron entre "Healthy" y "Salmonella". Tras la limpieza y preprocesamiento de datos, se encontró que la implementación del regularizador L2 y una capa de Dropout (0.5) en combinación con el optimizador Adam (lr = 0.0001), son los parámetros de partida óptimos para la técnica de transferencia de aprendizaje en los modelos pre entrenados VGG16 y ResNet50v2.

Los resultados mostraron que ambos modelos superaron al modelo base en términos de precisión y sensibilidad, con VGG16 obteniendo una precisión de 98.42 % y ResNet50v2 de 93.67 %. En conclusión, se demostró la efectividad de las técnicas de visión por computadora y el aprendizaje profundo en la detección de Salmonella *spp*., lo cual proporciona una base sólida para futuras investigaciones con imágenes tomadas en Colombia.

El código del proyecto se encuentra disponible en el repositorio de GitHub asociado (https://bit.ly/49EMvhb).

Palabras clave: Aprendizaje profundo, Avicultura, Clasificación de imágenes, ResNet50v2, Seguridad alimentaria, Salmonella *spp.*, Transferencia de aprendizaje, VGG16.

Abstract

Early detection of Salmonella *spp*. in poultry is crucial for improving the productivity of poultry companies in the country, as it not only reduces the mortality in their farms and maintains the profitability and reputation of the companies, but also guarantees the production of safe food for the population. However, the response capacity and cost of traditional methods limits the microbiological performance of these companies.

In this project, different architectures of Convolutional Neural Networks (CNN) were compared for the detection of salmonella *spp*. in images of poultry feces. A set of 5029 images taken in Africa between 2020 and 2021 was used, and they were classified as "Healthy" or "Salmonella." After data cleaning and preprocessing, it was found that the implementation of the L2 regularizer and a Dropout layer (0.5) in combination with the Adam optimizer (lr = 0.0001) are the optimal starting parameters for the transfer learning technique in the pre-trained VGG16 and ResNet50v2 models.

The results showed that both models outperformed the baseline model in terms of accuracy and recall, with VGG16 achieving a precision of 98.42% and ResNet50v2 of 93.67%. In conclusion, the effectiveness of computer vision and deep learning techniques for the detection of Salmonella *spp*. was demonstrated, which provides a solid foundation for future research with images taken in Colombia.

The code of this project is available in the associated GitHub repository (https://bit.ly/49EMvhb).

Keywords: Deep Learning, Food Safety, Image classification, Poultry, Salmonella *spp.*, ResNet50v2, Transfer-learning, VGG16.

1. Descripción del problema

Según la Organización Mundial de la Salud (OMS), cada año se enferman alrededor de 600 millones de personas en el mundo por consumir alimentos contaminados y 420.000 mueren a raíz de enfermedades transmitidas por estos alimentos [1]. Por esta razón, las compañías involucradas en la producción de proteína de origen animal adquieren la responsabilidad de implementar planes de gestión de calidad para garantizar la inocuidad alimentaria.

Normalmente, estos planes incluyen un programa de muestreo microbiológico para la detección de microorganismos patógenos durante sus procesos productivos. Sin embargo, las metodologías de microbiología tradicional implican que una muestra sea tomada y enviada a un laboratorio para su diagnóstico, lo cual demanda tiempo y recursos a las compañías, limitando la toma de decisiones asertivas y su desempeño microbiológico.

1.1. Problema de negocio

La inocuidad alimentaria es una prioridad para las organizaciones públicas y privadas alrededor del mundo, así como para las compañías involucradas en la cadena de producción de proteína animal. Un contaminante común es la presencia de especies enterohemorrágicas de Salmonella en alimentos, uno de los microorganismos que más afectan la salud de los consumidores de proteína de origen animal [1].

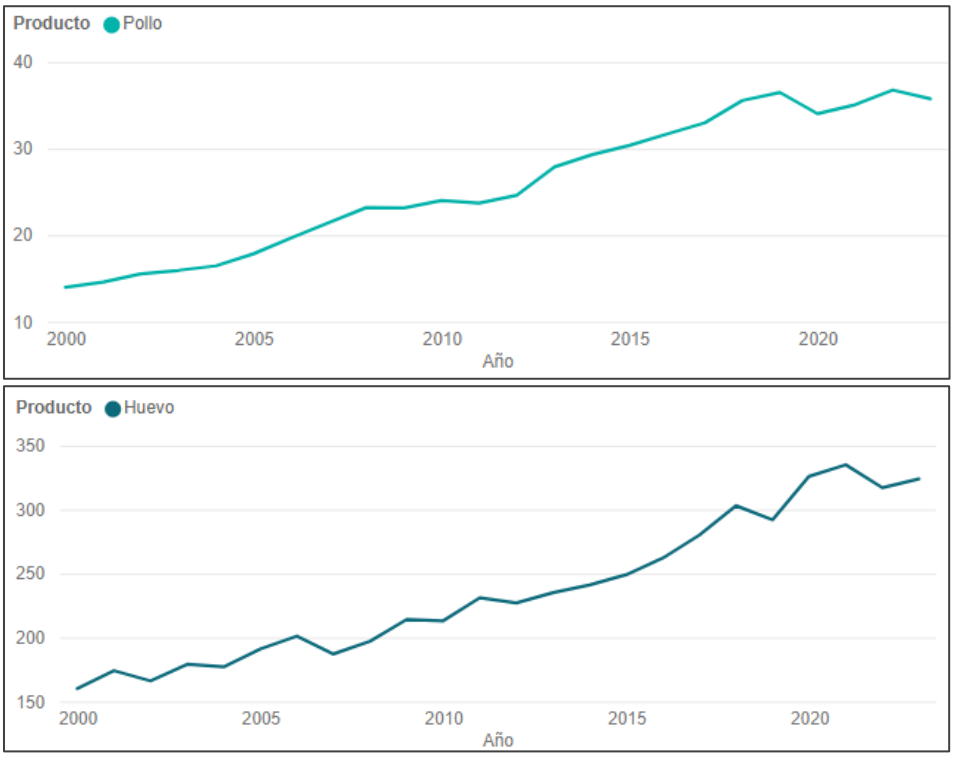
En Colombia, el consumo per cápita de huevo durante el 2023 fue de 324 unidades (16,864.4 millones de huevos producidos), mientras que el consumo per cápita de pollo durante el mismo año fue de 35.8 kg, con una producción total de 1,817,130 toneladas de pollo. En 2022, Colombia se posicionó como el 14º y 4° país con mayor producción per cápita de pollo y huevo en el mundo, respectivamente [2].

El constante aumento en el consumo per cápita de pollo y huevo (**Figura *1***) ha traído oportunidades competitivas para la industria avícola del país [3]. Sin embargo, esta tendencia también ha planteado desafíos significativos en términos de calidad y producción para los productores, quienes deben mantener altos niveles de productividad y rentabilidad mientras

minimizan la incidencia de enfermedades y mortalidad en las aves. Lo anterior, sin comprometer la calidad de los productos avícolas. Para enfrentar estos desafíos, se implementan procedimientos de detección de microorganismos patógenos durante el proceso productivo.

Figura 1

Arriba: Consumo per cápita de proteína de origen animal en Colombia. Arriba: pollo (kilogramos por habitante). Abajo: huevo (unidades por habitante)



Nota. Fuente http://bit.ly/3QT15cC(Federación Nacional de Avicultores de Colombia, 2024).

Específicamente en las granjas, el Instituto Colombiano Agropecuario (ICA) estableció en el año 2019 el "Programa Nacional de Control y Disminución de Prevalencia[1] de las Salmonellas Paratíficas […] en aves de corral dentro del territorio nacional". No obstante, la metodología descrita implica tomar al menos 20 muestras cloacales de aves diferentes cada mes y enviarlas a un laboratorio de microbiología certificado por el ICA para su tratamiento y diagnóstico [4]. Estos métodos tradicionales implican tiempo, recursos humanos y costos para las compañías, ya que se requiere personal capacitado para la toma de muestras y una logística adecuada para el

[1] Prevalencia: proporción de casos de una enfermedad en un periodo de tiempo, respecto a la población existente en la zona de objeto de estudio [40].

almacenamiento y transporte de estas hasta el laboratorio. Además, el procesamiento y diagnóstico de las muestras puede demorar hasta 5 días desde su recepción en el laboratorio [5]. La detección de Salmonella *spp*. en una muestra por métodos de microbiología tradicional cuesta entre 50.000 y 150.000 pesos colombianos, el Instituto Nacional de Salud tenía un costo de 104.000 pesos colombianos para el 2021 [6]. Estos recursos, aumentan con el volumen de muestras tomadas, lo que plantea dudas sobre su **escalabilidad**.

Por otro lado, los errores humanos son comunes, desde omisiones en la toma de muestras, formatos de remisión mal diligenciados, contaminación cruzada, demoras durante el transporte, deficiencia en la refrigeración durante el almacenamiento o errores en la implementación de los métodos en el laboratorio, pueden comprometer la **confiabilidad** de los resultados.

Adicionalmente, tomar muestras cloacales puede causar estrés a las aves, lo que provoca pérdida de peso, y afectar la productividad de las granjas. Esta problemática también se evidencia, cuando los productores de aves reducen su rentabilidad debido a aves muertas por Salmonelosis. Esta situación no solo impacta en la capacidad de las granjas para cumplir con los estándares exigidos por sus clientes, sino que también afecta su reputación y, en última instancia, su rentabilidad. Además, las empresas posteriores en la cadena de producción se ven obligadas a aumentar su inversión en puntos críticos de control, desinfectantes, inhibidores, etc., para controlar la carga microbiológica en sus instalaciones. En última instancia, esta problemática puede tener consecuencias graves, como enfermedades o incluso muertes causadas por la ingesta de alimentos contaminados [1].

En conjunto, el procedimiento actual para la detección de Salmonella *spp*. en granjas es lento, difícil de seguir y podría proporcionar resultados poco representativos del desempeño microbiológico real de las granjas.

Por lo tanto, es necesario desarrollar métodos alternativos para la detección de estos microorganismos en los animales de granja, que sean confiables y rápidos[7]. Por ejemplo, mediante la implementación de algoritmos de *Machine Learning* (ML) para la detección de Salmonella *spp.* a partir del reconocimiento de patrones en imágenes de heces de aves de granja.

Este reconocimiento es un factor determinante para mejorar la productividad de las compañías, reducir la carga microbiológica en la cadena de producción de proteína animal, y garantizar el consumo de alimentos inocuos desde el principio de la cadena de producción.

1.2. Aproximación desde la analítica de datos

En el ámbito del control de calidad como área de la industria de la Ciencia de Datos, se propone programar un algoritmo de *ML* que serviría para detectar la presencia de Salmonella *spp*. en heces de aves de granja. Específicamente, se plantea la implementación de técnicas de Deep Learning (DL, Aprendizaje profundo), las cuáles según [8], han tomado relevancia para resolver problemas de visión por computador, incluso en el área de la agricultura [9], [10].

Por ejemplo, el uso del modelo de **segmentación** de imágenes "Segment Anything" de MetaAI Research para predecir el peso de las aves y encontrar patrones en su comportamiento [11], [12] y la **detección** automatizada de objetos en imágenes para el conteo automatizado de aves mediante cámaras instaladas en los galpones [13], [14]. Así como tecnologías ya implementadas en Colombia como PigVision^{TM}, la cual integra IoT y visión de computadores para estimar el peso de los cerdos durante su crecimiento con una precisión del 95% [15], [16]. En cuanto a la **clasificación** de imágenes, se respalda el uso de modelos de modelos de DL para la detección de enfermedades a partir de imágenes en animales [17], humanos [18], [19] y plantas[20].

Además, se reconoce que las técnicas de ML clásicas se ven restringidas en el procesamiento y extracción de características de imágenes [21], por lo que se opta por utilizar arquitecturas de Redes Neuronales Convolucionales (CNN), que, como se evidenció han sido ampliamente probadas en problemas de visión por computadoras.

Este enfoque alternativo podría reducir significativamente los recursos dedicados a garantizar la inocuidad alimentaria. Se esperaría una disminución tanto en el tiempo necesario para obtener resultados, pasando de 5 a 1 día hábil, como en el personal requerido para llevar a cabo las metodologías actuales. Asimismo, se reducirían las acciones correctivas que podrían generar costos adicionales a las empresas que adquieren las aves.

Por otro lado, las granjas podrían mejorar su desempeño. Esto se traduciría en una optimización de la logística relacionada con la toma, almacenamiento, transporte y diagnóstico de muestras microbiológicas, permitiendo procesar un mayor número de muestras en un período de tiempo dado. Como resultado, se obtendrían conclusiones respaldadas por una mayor cantidad de información.

Además de los beneficios operativos, la implementación de esta tecnología podría tener un impacto positivo en la reputación y competitividad de las empresas. Aquellas granjas que adopten la Inteligencia Artificial en sus procesos, posicionándose como líderes en innovación y compromiso con la seguridad alimentaria, se destacarían como proveedores confiables [22].

En una eventual implementación de los modelos predictivos desarrollados, es importante considerar el riesgo de falsos negativos y falsos positivos. Los falsos negativos podrían subestimar el riesgo microbiológico, afectando la productividad y la reputación de una granja. Por otro lado, los falsos positivos podrían llevar a decisiones incorrectas, como la implementación innecesaria de tratamientos médicos. En estos casos, es crucial establecer un protocolo de seguimiento que incluya la validación de los resultados del modelo con métodos tradicionales de microbiología.

Además, la solución analítica que se propone:

- No proporciona recomendaciones sobre la cantidad y representatividad de los datos necesarios para la detección de microorganismos patógenos en granjas.
- No incluye sugerencias sanitarias para el manejo de granjas que resulten identificadas como contaminadas con microorganismos patógenos.
- No incluye análisis de las posibles de contaminación.
- No contempla el despliegue del algoritmo entrenado y ajustado en ninguna herramienta como aplicaciones web para dispositivos móviles o computadores.

Respecto a riesgos para desarrollar el proyecto de aprendizaje, se encuentra que:

- El conjunto de datos disponible no es suficiente para el entrenamiento de los algoritmos.
- Los recursos computacionales disponibles no son suficientes para ejecutar los algoritmos.

- La complejidad en la selección de modelos, señalada como un desafío significativo por estudios previos [23].

1.3. Origen de los datos

El conjunto de datos fue obtenido de la página Web de Kaggle, se titula "Chicken Disease Image Classification". Contiene un archivo .csv con las etiquetas de cada imagen y 8067 fotografías de heces de aves de granja. Las fotografías fueron capturadas en granjas de aves en Tanzania (África) entre septiembre 2020 y febrero 2021, utilizando la aplicación Open Data Kit[2] en dispositivos móviles [24]. El conjunto de datos está disponible para su descarga gratuita en Kaggle y no hay ninguna restricción o condición de uso establecida. No se modificaría, combinaría o utilizará el conjunto de datos con propósitos diferentes a los enmarcados en este proyecto.

1.4. Métricas de desempeño

La métrica más utilizada en la literatura para evaluar el desempeño de modelos de *ML* es la precisión (**accuracy**), que indica cuántas predicciones acertadas se realizan respecto al total de predicciones [5], se espera al menos una precisión alrededor del +-5 % comparada con la precisión del método realizado en los laboratorios de microbiología tradicional, que oscila entre el 90 % y 95 % (este valor cambia en cada laboratorio y debe ser notificado junto con la incertidumbre [25]). La RNA propuesta también será evaluada utilizando la sensibilidad **(recall),** que indica el número de imágenes clasificadas correctamente en relación con el total de imágenes de esa clase. Los tratamientos médicos se aplican normalmente en el agua de bebida de los galpones, es decir, al galpón en su conjunto. En este sentido, una tolerancia del 15 % a falsos negativos es razonable. En otras palabras, se espera un recall de al menos 85 % para detectar la mayoría de las aves contaminadas con Salmonella *spp*.

Adicionalmente, se emplearán **matrices de confusión** para comprender la tasa de verdaderos positivos y verdaderos negativos alcanzados. Finalmente, se usa la métrica **AUC/ROC** (Receiver Operating Characteristic), para comparar el desempeño de las distintas arquitecturas de CNN desarrolladas entre sí y con relación a un predictor aleatorio. Esta curva muestra la tasa de

[2] https://getodk.org/

verdaderos positivos (1 – especificidad) para diferentes umbrales de clasificación. Por lo tanto, el área bajo la curva más cercana a 1 indicará el modelo con mejor rendimiento respecto a discriminación entre las dos clases [8].

Se espera que el modelo predictivo pueda tener influencia sobre la **mortalidad** de las aves, este es un indicador clave que está relacionado con la productividad de las granjas. La mortalidad, expresada en porcentaje, da cuenta del número de aves fallecidas respecto al total de aves iniciales en el galpón, ya sea por día o acumulada desde el inicio de la cría. Estas pérdidas diarias reducen el tamaño del lote, lo que no solo implica la pérdida de ingresos por aves no vendidas, sino también la realizada en agua, alimento y logística de las aves que no podrán comercializarse. Se espera que el modelo reduzca el tiempo de respuesta a eventos de Salmonelosis al aumentar el número de muestras disponibles, mejorando la precisión en la detección de aves enfermas.

2. Objetivos

2.1. Objetivo general

Desarrollar un método alternativo basado en algoritmos de Inteligencia Artificial (IA) para la detección de salmonella *spp*. en imágenes de heces de aves de granja tomadas en África entre 2020 y 2021.

2.2. Objetivos específicos

- Preparar un conjunto de imágenes de heces de aves de granja para obtener trenes de entrenamiento, validación y prueba para su posterior uso en modelos de *Machine Learning*.

- Comparar la precisión y sensibilidad de 14 combinaciones de parámetros en una Red Neuronal Convolucional simple, que incluyen optimizador, tasa de aprendizaje, implementación de regularizadores y Dropout, para identificar los mejores parámetros de partida.

- Entrenar y comparar dos arquitecturas diferentes de Redes Neuronales Convolucionales diferentes para la clasificación de imágenes: VGG16 y ResNet50v2 y realizar ajuste fino para mejorar la precisión (accuracy), sensibilidad (recall), matrices de confusión y el área bajo la curva ROC (AUC) de cada modelo.

3. Datos

En esta sección, se describe el acceso a los datos originales y sus características, incluyendo formatos, resoluciones, etiquetado, número de registros, tamaño total en MB, etc. Luego se muestra cómo se construyeron los subconjuntos de entrenamiento, validación y prueba. Finalmente, se presenta analítica descriptiva de los mismos.

3.1. Datos originales

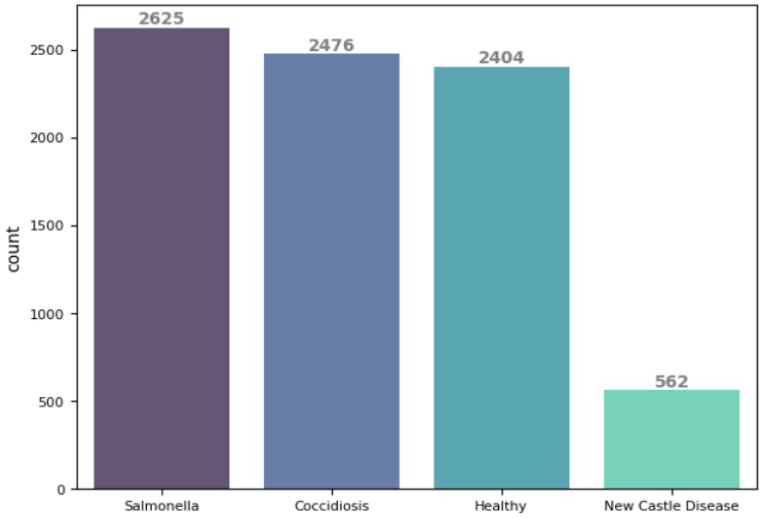
El archivo descargado desde Kaggle “archive.zip” tiene un tamaño de 249,1 MB. Esta carpeta comprimida contiene 8067 imágenes y un archivo *.csv con las etiquetas correspondientes.

El archivo “train_data.csv”, pesa 220 KB y consta de dos columnas: “images” y “label”, en donde se asocia el nombre de cada imagen con su respectiva etiqueta. Según la información mostrada en el archivo “01_data_prep_EDA.ipynb” disponible en el repositorio de GitHub (Ortega A., 2024), el archivo con las etiquetas comprende 8067 instancias y 2 columnas, ambas de tipo “Object”. Las instancias se encuentran clasificadas originalmente en 4 enfermedades diferentes: “Salmonella”, “Coccidiosis”, “Newcastle Disease” y “Healthy”. No hay valores nulos en ninguna columna, y el archivo cargado utilizando Python ocupa 126.2+ KB, siendo fácil de procesar en términos de almacenamiento y carga.

La **Figura 2** muestra la frecuencia de las instancias para cada clase del archivo “train_data.csv”.

Figura 2

Frecuencia de las instancias para la variable de salida



Nota. Fuente: https://bit.ly/3MO6hx3 (Kaggle, 2022). Elaboración propia, código disponible en el repositorio asociado al proyecto (Ortega A., 2023).

Adicionalmente, la carpeta descargada contiene una subcarpeta llamada “Train” en donde se encuentran las imágenes en formato JPG. Todas tienen un tamaño de 224 x 224 pixeles y utilizan una codificación RGB, lo que significa que cada imagen tiene 3 canales de color representados por números enteros de 8 bits sin signo (uint8), representando valores enteros no negativos. Este tipo de codificación permite representar valores enteros no negativos en un rango de 0 a 255 para cada píxel de la imagen a color [26].

Las imágenes están etiquetadas con la clase correspondiente tanto en el nombre del archivo, como en el archivo “train_data.csv”. Las imágenes etiquetadas con presencia de Salmonella *spp.* están marcadas como “salmo” o “pcrsalmo”, mientras que las imágenes sin Salmonella *spp.* están marcadas como “healthy” o “pcrhealthy”. Cada imagen tiene la palabra “healthy.” o “salmo.” Seguida de un consecutivo, por ejemplo: “healthy.2053.jpg” o “pcrsalmo.1.jpg”. La **Figura 3** muestra ejemplos de estas imágenes.

Figura 3

Ejemplo de las imágenes disponibles. Arriba de izquierda a derecha: salmo.11, salmo.41, salmo.27. Abajo, de izquierda a derecha: healthy.67, healthy.761, healthy.2049

Nota. Fuente: https://bit.ly/3MO6hx3 (Kaggle, 2022).

Se asume que las etiquetas están correctamente asignadas y que no hay imágenes duplicadas. La **diversidad** del conjunto de datos es amplia, abarcando imágenes tomadas durante un periodo de 6 meses, lo que se espera permita al algoritmo generalizar a partir de ellas. Es importante señalar que las fotografías fueron capturadas en África y pueden no reflejar el entorno o condiciones en Colombia. Además, no se han actualizado desde febrero 2021 y algunas imágenes se ven borrosas. Estudios previos han concluido que "los conjuntos de datos pueden ser ruidosos o de baja calidad cuando se recopilan en entornos agrícolas interiores y exteriores hostiles" [5].

3.2. Datsets

Limpieza de datos

El proceso de limpieza y obtención de los subconjuntos de datos se encuentra documentado en el archivo "01_data_prep_EDA.ipynb" del repositorio de GitHub asociado (https://bit.ly/49EMvhb). En este proceso, se seleccionaron únicamente las imágenes

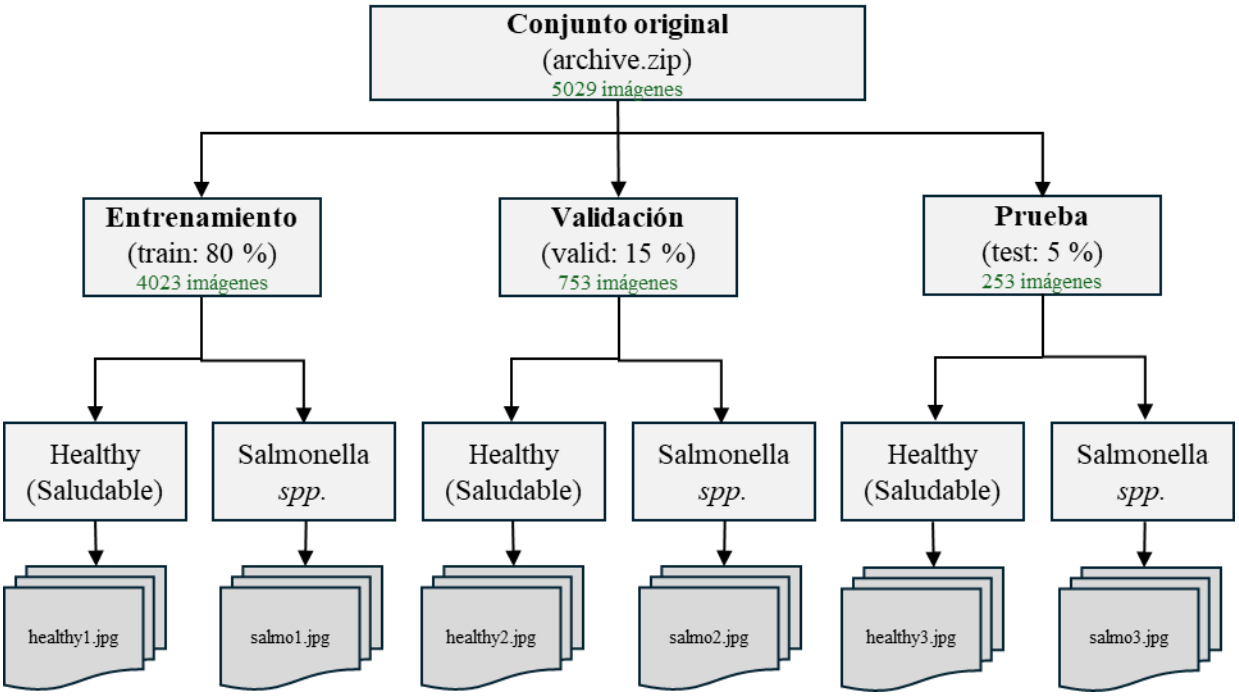
pertenecientes a las clases "Healthy" y "Salmonella", descartando las demás clases presentes en el conjunto de datos original.

División de los datos en trenes de entrenamiento, prueba y validación

A continuación, se dividió el conjunto de datos en subconjuntos de entrenamiento, prueba y validación con una proporción de 80%, 15% y 5%, respectivamente. Esta división se justifica para evaluar el rendimiento del modelo y prevenir el sobreajuste, así como para validar su capacidad de generalización en nuevos datos no vistos durante el entrenamiento. La distribución aleatoria y proporcional de las clases se garantizó utilizando la librería 'os' de Python para mover los archivos a las carpetas de destino correspondientes. Cada carpeta destino contiene dos subcarpetas, una para cada clase. La estructura del directorio creado se muestra en la **Figura 4**.

Figura 4

Estructura del directorio con las imágenes luego de la limpieza y división de los datos en conjuntos de entrenamiento, validación y prueba



Nota. Elaboración propia (Microsoft Power Point).

3.3. Analítica descriptiva

El análisis exploratorio de los datos se detalla en el archivo "01_data_prep_EDA.ipynb" del repositorio de GitHub asociado (https://bit.ly/49EMvhb). En este análisis, se recopiló información de las imágenes creando una tabla llamada "*images_metadata*". Esta tabla (DataFrame) contiene información relevante de cada imagen, como la ruta del archivo, la categoría (etiqueta), alto, ancho, relación de aspecto, tamaño, canales, tipo de dato y rango de píxeles. Para obtener esta información, se iteró a través de los directorios de imágenes y se leyó cada archivo de imagen utilizando la biblioteca de Python ***mpimg***. La **Figura 5** muestra una vista preliminar del DataFrame obtenido.

Figura 5

Vista previa de 10 registros del DataFrame "images_metadata"

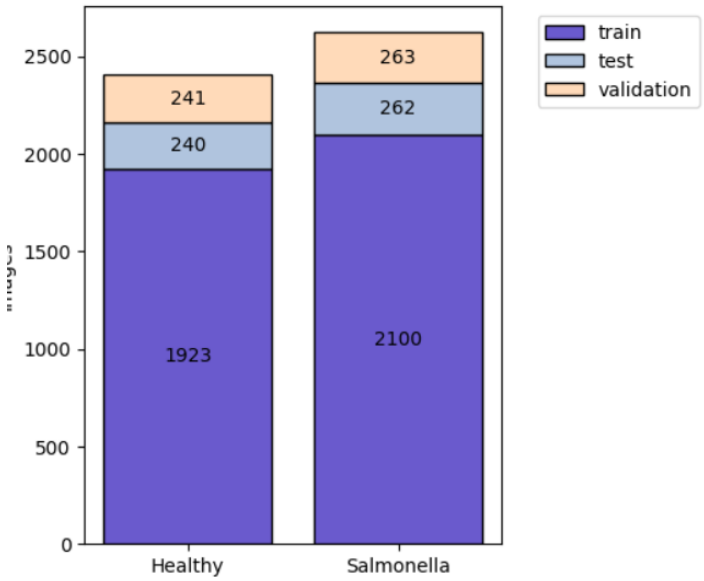
	path	label	height	width	aspect_ratio	size	channels	dtype	pixel_range
0	/tmp/data2/test/healthy/healthy.881.jpg	healthy	224	224	1.0	50176	3	uint8	2 - 255
1	/tmp/data2/test/healthy/healthy.1817.jpg	healthy	224	224	1.0	50176	3	uint8	27 - 254
2	/tmp/data2/test/healthy/healthy.640.jpg	healthy	224	224	1.0	50176	3	uint8	14 - 255
3	/tmp/data2/test/healthy/healthy.799.jpg	healthy	224	224	1.0	50176	3	uint8	17 - 255
4	/tmp/data2/test/healthy/pcrhealthy.320.jpg	healthy	224	224	1.0	50176	3	uint8	22 - 255
5	/tmp/data2/test/healthy/pcrhealthy.236.jpg	healthy	224	224	1.0	50176	3	uint8	0 - 255
6	/tmp/data2/test/healthy/healthy.309.jpg	healthy	224	224	1.0	50176	3	uint8	8 - 232
7	/tmp/data2/test/healthy/healthy.777.jpg	healthy	224	224	1.0	50176	3	uint8	0 - 255
8	/tmp/data2/test/healthy/healthy.1202.jpg	healthy	224	224	1.0	50176	3	uint8	24 - 247
9	/tmp/data2/test/healthy/healthy.1272.jpg	healthy	224	224	1.0	50176	3	uint8	16 - 253

Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto (Ortega A., 2024).

Además, se confirmó la ausencia de imágenes duplicadas (con el mismo nombre de archivo) y se contaron las imágenes de cada clase ('Healthy' y 'Salmonella'). La **Figura 6** muestra un equilibrio notable entre las dos categorías, con el 52,1 % de los registros en la categoría "Salmonella" y el 47,9 % en la categoría "Healthy". Se implementó una función para visualizar imágenes aleatorias y sus respectivas clases, lo que permitió tener una mejor idea de la variabilidad y características de las imágenes en el conjunto de datos.

Figura 6

Frecuencia de las instancias para la variable de salida luego de limpiar los datos



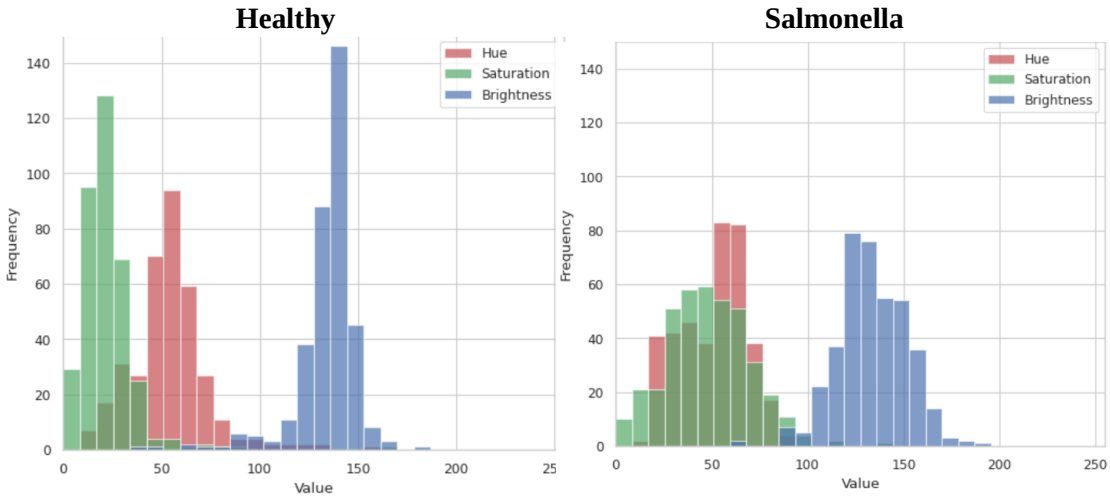
Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto (Ortega A., 2024).

Por su parte, la **Figura *7*** muestra histogramas para conocer la distribución de los pixeles de las imágenes de validación para cada clase. En cuanto a la saturación, la distribución de píxeles de las imágenes con Salmonella *spp.* tienen un valor medio mucho más alto (49.09 frente a 21.47). Esto indica que los colores en las imágenes de esta clase son mucho más intensos y vibrantes que en las imágenes clasificadas como “Healthy”.

Respecto al brillo y al matiz (Hue), la distribución de píxeles de las imágenes de Salmonella *spp.* y Healthy tienen valores promedio similares. Esto sugiere que las imágenes de Salmonella *spp.* puede ser ligeramente más brillantes que las imágenes sanas, pero la diferencia no es tan significativa como en el caso de la saturación. Además, los colores en las imágenes de Salmonella *spp.* pueden ser más variados en términos de matiz.

Figura 7

Distribución de pixeles por clase



Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto (Ortega A., 2024).

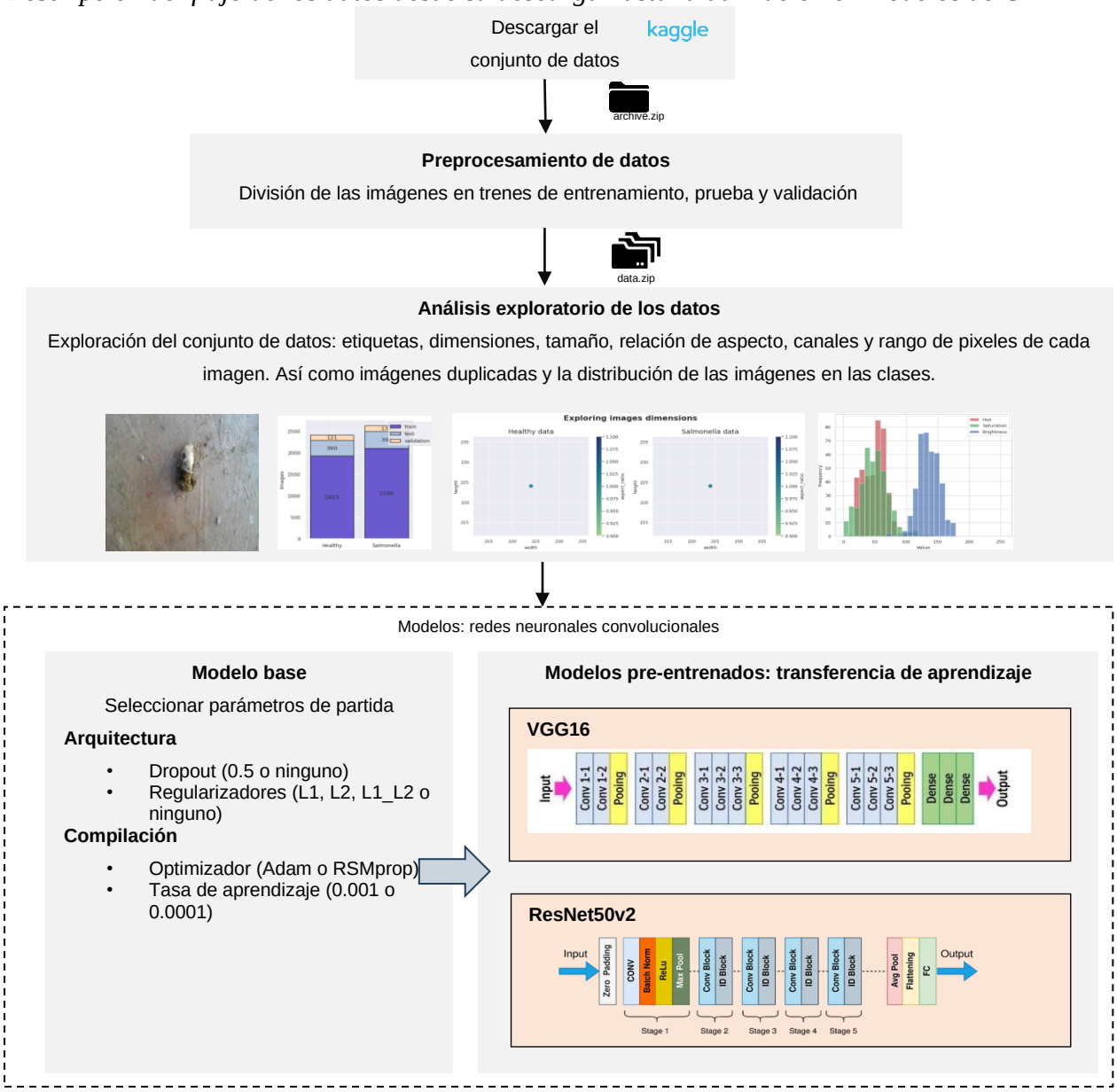
En general, la distribución de píxeles de las imágenes de Salmonella *spp.* se caracteriza por colores más intensos y vibrantes en comparación con las imágenes sanas, lo que podría ser útil para diferenciarlas en el proceso de clasificación.

4. Proceso de analítica

4.1. Pipeline principal

Figura 8

Descripción del flujo de los datos desde su descarga hasta la utilización en modelos de CNN



4.2. Preprocesamiento

Se realizaron diversas operaciones de preprocesamiento para mejorar el rendimiento de los modelos. Primero, se normalizaron las imágenes ajustando sus valores de píxeles entre 0 y 1, lo que favorece el rendimiento y convergencia de los modelos [8], [10], así como la demanda computacional. Luego, se estandarizó el tamaño de las imágenes a 224x224 píxeles para asegurar consistencia dimensional.

Además, se aplicaron técnicas de aumentación de datos en el conjunto de entrenamiento, como rotaciones (hasta 20 grados), traslaciones horizontal y vertical (hasta el 20% del ancho y alto), cizallamiento (hasta el 20%), zoom (hasta el 20%) y volteo horizontal, para generar más muestras de entrenamiento y mejorar la robustez de los modelos [8].

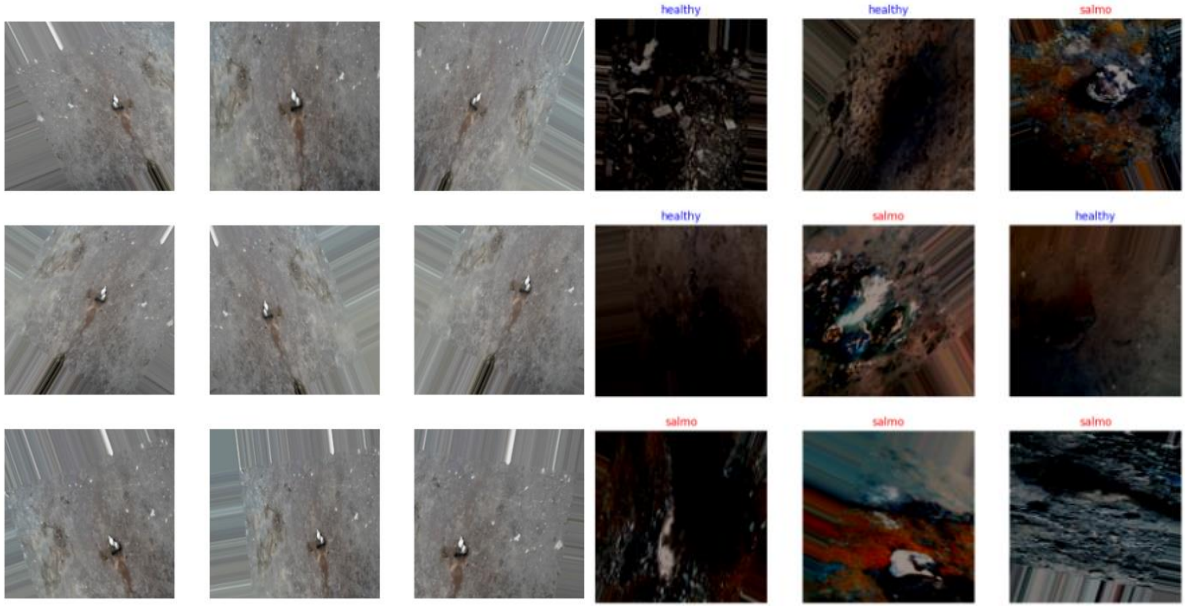
Se utilizaron generadores de la biblioteca **ImageDataGenerator** para cargar las imágenes en lotes durante el entrenamiento, validación y prueba. Estos generadores facilitan el manejo de grandes conjuntos de datos y permiten, no sólo aplicar transformaciones de aumentación de datos de manera eficiente, sino que favorecen el entrenamiento y evaluación de los modelos al generar el flujo de imágenes desde el directorio para cada época de entrenamiento por lotes y no cargando todas las imágenes en memoria RAM [27]. No se encontraron errores o excepciones durante la implementación de ImageDataGenerator, lo que indica que las imágenes no están corruptas.

Adicionalmente, cuando se utiliza el modelo pre entrenado VGG16 se incorpora el preprocesamiento propio del modelo que incluye escalar los valores de los píxeles de 0-255 a 0-1, sustraer el valor medio de RGB de la imagen de ImageNet (que es [0.485, 0.456, 0.406]) y después dividir cada canal por la desviación estándar de los mismos canales en la imagen de ImageNet (que es [0.229, 0.224, 0.225]).

En la **Tabla 1** se muestran otros preprocesamientos específicos para la implementación de cada modelo.

Figura 9

Izquierda: ejemplos de aumento de imágenes mediante rotación, traslación, aumento y volteo. Derecha: incluido el preprocesamiento propio de VGG16



Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto (Ortega A., 2024).

4.3. Modelos

Después de preprocesar las imágenes, se consideró un modelo base y dos modelos pre-entrenados para abordar la clasificación entre las clases "Healthy" y "Salmonella". El modelo base es una arquitectura de red neuronal convolucional simple que sirve como punto de referencia inicial para evaluar el desempeño y guiar el desarrollo de los modelos de transferencia de aprendizaje[28].

Basados en los experimentos de [8] aumentar la profundidad de las CNN y usar transferencia de aprendizaje, pueden mejorar el desempeño las CNN. Por lo tanto, entre los modelos disponles en la API de Keras, se eligieron los modelos VGG16 y ResNet50v2, lo anterior, basado en una combinación de factores: la precisión, el tamaño, la cantidad de parámetros y la velocidad de inferencia del modelo publicados en la documentación oficial de Keras [8], [29], [30].

Según la documentación oficial de Keras, el modelo VGG16 tiene una precisión de aproximadamente el 71.3% en la métrica Top-1 y el 90.1% en la métrica Top-5, con un tamaño de 528 MB y 138.4 millones de parámetros. Por otro lado, el modelo ResNet50v2 ofrece una precisión ligeramente mejor, con aproximadamente el 74.9% en la métrica Top-1 y el 92.1% en la métrica

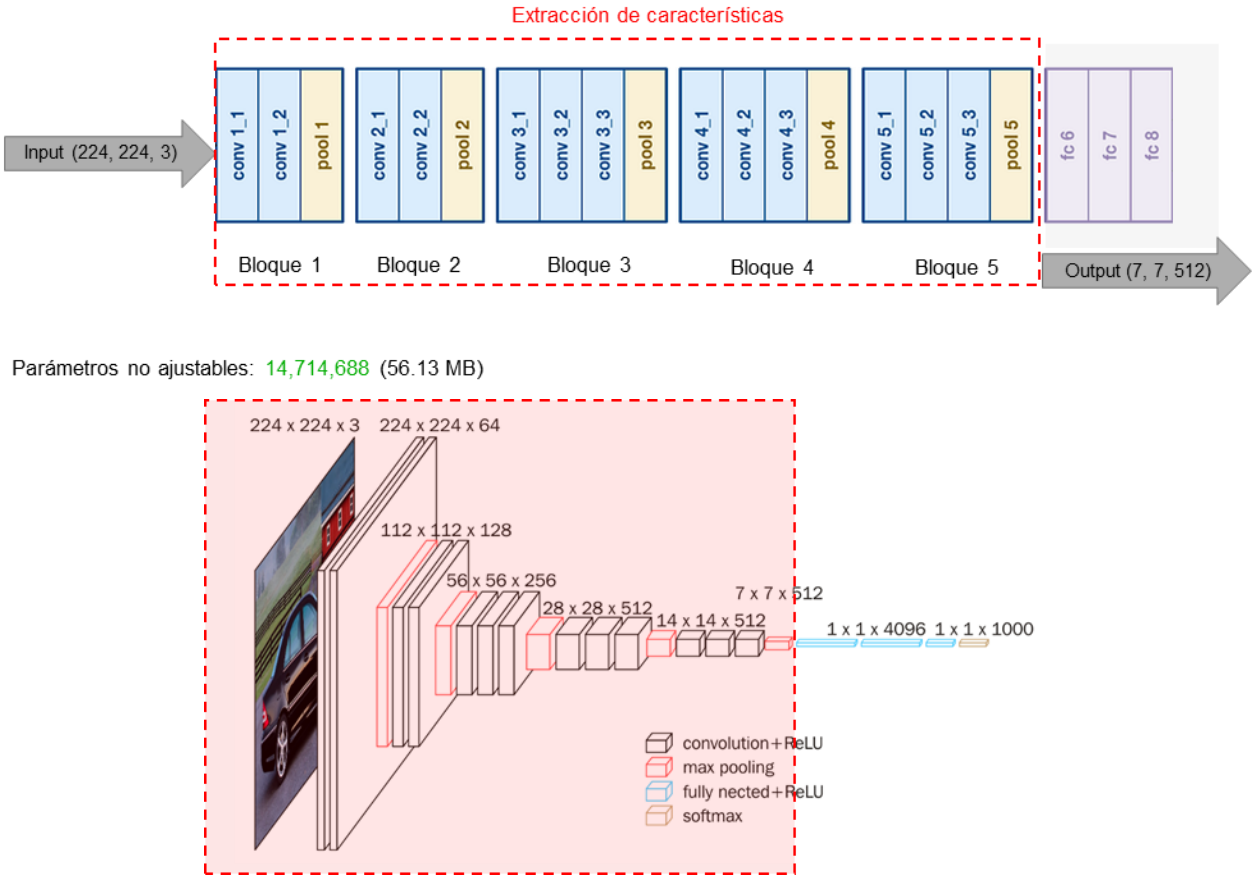
Top-5, con un tamaño más pequeño de 98 MB y 25.6 millones de parámetros [29]. Ambos modelos son relativamente ligeros en comparación con los otros modelos disponibles, lo que facilita su implementación en entornos con recursos computacionales limitados [31], [32].

A continuación, se describen la arquitectura y configuraciones de los modelos pre entrenados:

4.3.1. VGG16: es una arquitectura de CNN propuesta por Simonyan y Zisserman en 2014, que se destacó por su desempeño en tareas de clasificación de imágenes en el desafío ImageNet de ILSVRC ese año [33]. Esta red consta de 16 capas, incluyendo 13 capas convolucionales y 3 capas totalmente conectadas (FC: fully-connected). Sus capas convolucionales utilizan filtros de tamaño 3x3 y las capas FC tienen 4096 unidades cada una. A diferencia del modelo base, incluye capas de Max Pooling después de algunas, no todas, las capas convolucionales para reducir la dimensionalidad de las características extraídas. Este modelo fue entrenado en 4 GPUs por 2-3 semanas [34]. La **Figura 10** muestra esquemáticamente la arquitectura del modelo VGG16.

Figura 10

Arquitectura y resumen de la red VGG16 implementada

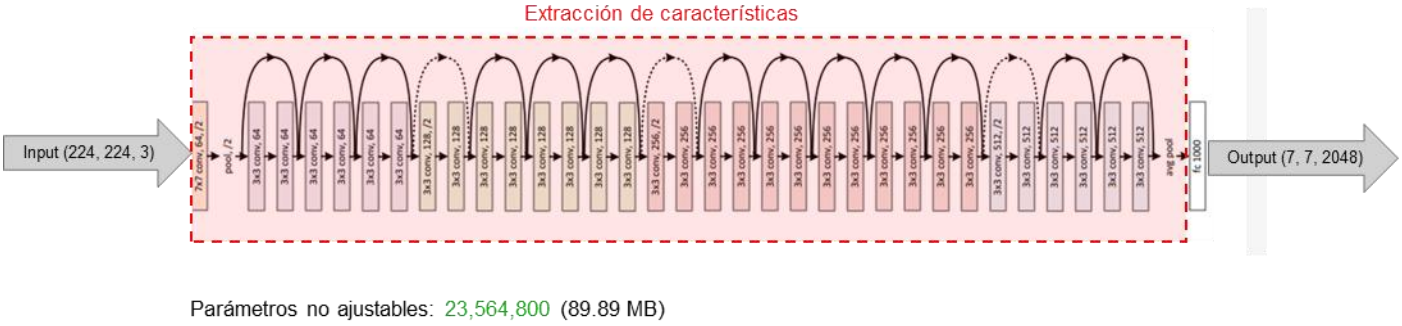


Nota: Tomadas y modificadas de [35].

4.3.2. ResNet50v2: es la segunda versión del modelo de Redes Neuronales Residuales, propuesto por Microsoft Research en 2015. La versión original ganó en 2015 el desafío ImageNet de ILSVRC y está compuesta por 49 capas convolucionales organizadas en 4 bloques de "residual learning" y una capa FC. Lo anterior, facilita la propagación de gradientes a través de las capas y mejora el entrenamiento. En su publicación original, los autores destacan que la versión 1 de esta red es 8 veces más profunda que VGG16, pero aun así tiene menor complejidad [8], [32]. La función *resnet50v2.preprocess_input* de Keras se utiliza para preprocesar las imágenes de entrada antes de pasarlas al modelo ResNet50v2 [29].

Figura 11

Arquitectura y resumen de la red ResNet50v2 implementada



Nota. Tomada y modificada de [36].

Ambos modelos fueron implementados para extraer las características de las imágenes, ya que en su entrenamiento original han aprendido a partir de una gran cantidad de imágenes [8]. En la **Sección 0** (pág. 33) se describe la metodología implementada para usar estas arquitecturas.

4.4. Métricas

En esta sección se describe cómo se calculan las métricas de desempeño de los modelos de aprendizaje profundo [8]. Para la precisión y la sensibilidad se utilizó la API 'compile' de Keras, así:

a. Precisión (accuracy): es la proporción de predicciones verdaderas positivas (VP) y verdaderas negativas (VN) entre el total de predicciones, es decir, da cuenta de la frecuencia con que las predicciones del modelo coinciden con las etiquetas verdaderas. Se calcula así:

$$Precisión\ (accuracy) = \frac{VP + VN}{VP + VN + FP + FN}$$

b. Sensibilidad (recall): es la proporción de predicciones verdaderas positivas (VP) entre el total de muestras positivas reales (VP + FN). Se calcula así:

$$Sensibilidad\ (recall) = \frac{VP}{VP + FN}$$

c. Matriz de confusión: es la visualización del desempeño de los algoritmos, cada columna representa el número de predicciones de cada clase y cada fila las instancias de la clase real.

Verdaderos negativos (VN)	Falsos positivos (FP)
Falsos negativos (FN)	verdaderos positivos (VP)

d. AUC/ROC: El AUC mide el área bajo la curva ROC que representa gráficamente la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) a diferentes umbrales de clasificación. Un AUC de 0.5 indica que el modelo tiene un rendimiento similar al azar. Un AUC de 1.0 indica un rendimiento perfecto del modelo.

$$AUC = \int_0^1 TPR(FPR)d(FPR)$$

5. Metodología

5.1. Baseline

Para establecer un punto de referencia y determinar parámetros óptimos iniciales, se desarrolló un modelo base utilizando una arquitectura simple de CNN. La **Figura 12** muestra la arquitectura propuesta, la cual consta de 4 capas convolucionales con 16, 32, 64 y 128 filtros de 3 x 3 píxeles, seguidas de una capa de Max Pooling que opera sobre una ventana de 2 x 2 píxeles.

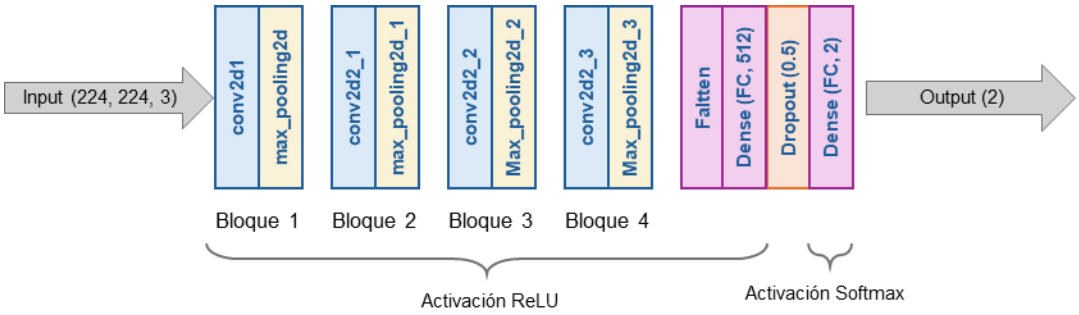
Esta arquitectura clásica es ampliamente usada y permite al modelo aprender características cada vez más abstractas y complejas de las imágenes a medida que avanza a través de las capas. El tamaño de los filtros también es una elección común en las CNN, de hecho, es el mismo tamaño utilizado en el modelo pre-entrenados VGG16 que se compara más adelante [8].

Se utilizaron funciones de activación ReLU en todas las capas de convolución y densas debido a su simplicidad y eficacia en modelos de DL. La función ReLU devuelve 0 para entradas negativas y la entrada sin cambios para entradas positivas, lo que ayuda al modelo a aprender características no lineales de las imágenes y a prevenir el problema de la saturación de gradiente. Esta elección de función de activación también se utiliza en las capas convolucionales de VGG16 y ResNet50v2, proporcionando consistencia en el enfoque de aprendizaje [8], [28].

Además, se exploraron diversos métodos de regularización introduciendo términos de penalización en la función de pérdida, entre ellos L1, L2, su combinación (L1_L2) y ausencia. La regularización L1, introduce un término de penalización proporcional a la suma de los valores absolutos de los pesos de las conexiones entre las neuronas. Esta técnica promueve la esparces en los pesos, forzando muchos de ellos a cero, lo que facilita la selección automática de características y simplifica el modelo. La regularización L2, implica la adición de un término de penalización proporcional a la suma de los cuadrados de los pesos, lo cual tiende a reducir la magnitud de los pesos sin forzarlos a cero. La combinación de L1 y L2, aprovecha los beneficios de ambos enfoques [37].

Figura 12

Arquitectura y resumen de la arquitectura del modelo base



Layer (type)	Output Shape	Param #	
conv2d (Conv2D)	(None, 222, 222, 16)	448	
max_pooling2d (MaxPooling2D)	(None, 111, 111, 16)	0	
conv2d_1 (Conv2D)	(None, 109, 109, 32)	4,640	
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 32)	0	
conv2d_2 (Conv2D)	(None, 52, 52, 64)	18,496	
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0	
conv2d_3 (Conv2D)	(None, 24, 24, 128)	73,856	
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 128)	0	
flatten (Flatten)	(None, 18432)	0	Regularizador opcional (L1, L2 o L1, L2, ninguno)
dense (Dense)	(None, 512)	9,437,696	
dropout (Dropout)	(None, 512)	0	Dropout opcional (0,5)
dense_1 (Dense)	(None, 2)	1,026	

Total params: 9,536,162 (36.38 MB)
Trainable params: 9,536,162 (36.38 MB)
Non-trainable params: 0 (0.00 B)

Nota. Elaboración propia (Microsoft Power Point).

El *Dropout*, por otro lado, es una técnica que consiste en desactivar aleatoriamente un porcentaje de neuronas en una capa durante el entrenamiento del modelo. Esto previene el sobreajuste al forzar al modelo a no depender demasiado de neuronas específicas [37]. Se estableció un valor de 0.5 por las siguientes razones: (1) desactivar el 50% de las neuronas mejoraría su capacidad de generalización sobre datos no vistos. (2) Según el artículo original de *Dropout* por [37] un valor de 0.5 ha demostrado ser efectivo en diversas aplicaciones y arquitecturas de redes neuronales. (3) Un valor más alto podría llevar a la subutilización de la capacidad del modelo, mientras que un valor más bajo podría no ofrecer suficiente regularización.

Finalmente, se utilizó una capa densa de salida con dos unidades y activación *softmax* para favorecer la interpretabilidad del modelo. La activación softmax normaliza las salidas de la capa densa para que sumen 1, permitiendo interpretar la salida como la probabilidad de pertenencia a cada clase [8], [28].

Para la compilación del modelo base, se usó la función de pérdida "categorical_crossentropy" debido a su compatibilidad con la activación softmax. Esta función mide la diferencia entre las distribuciones de probabilidad predichas y reales de las etiquetas de clase [28]. Además, se compararon los optimizadores RMSprop o Adam con sus tasas de aprendizaje predeterminadas (0.001) y una menor de 0.0001. Las diferentes combinaciones de parámetros y el resultado de su evaluación en el conjunto de validación se muestran en la **Tabla 2**.

Los parámetros de entramiento EPOCHS = 50, BATCH_SIZE = 64, STEPS_PER_EPOCH = 20 y VALIDATION_STEPS = 10 se mantuvieron constantes durante esta comparación. Se reconoce que pueden influir en el rendimiento y desempeño de los modelos. En cada época, el modelo procesa 20 baches de 64 imágenes cada uno, lo que resulta en un total de 1280 imágenes por época. Dado que el conjunto de entrenamiento tiene 4023 imágenes, esto representa aproximadamente el 32% del conjunto de entrenamiento que se utiliza en cada iteración de entrenamiento.

5.2. Validación

Para la validación, se utilizaron modelos pre-entrenados para extraer y aprender características de las imágenes, omitiendo sus capas densas intrínsecas. A continuación, se añadieron capas personalizadas en la parte superior de estos modelos. Primero, se tomó la salida de la última capa convolucional y se aplicó una capa de promedio global (GlobalAveragePooling2D), que es comúnmente usada para reducir la dimensionalidad de la salida y hacerla más resistente a la variabilidad en la posición y tamaño de los objetos en las imágenes.

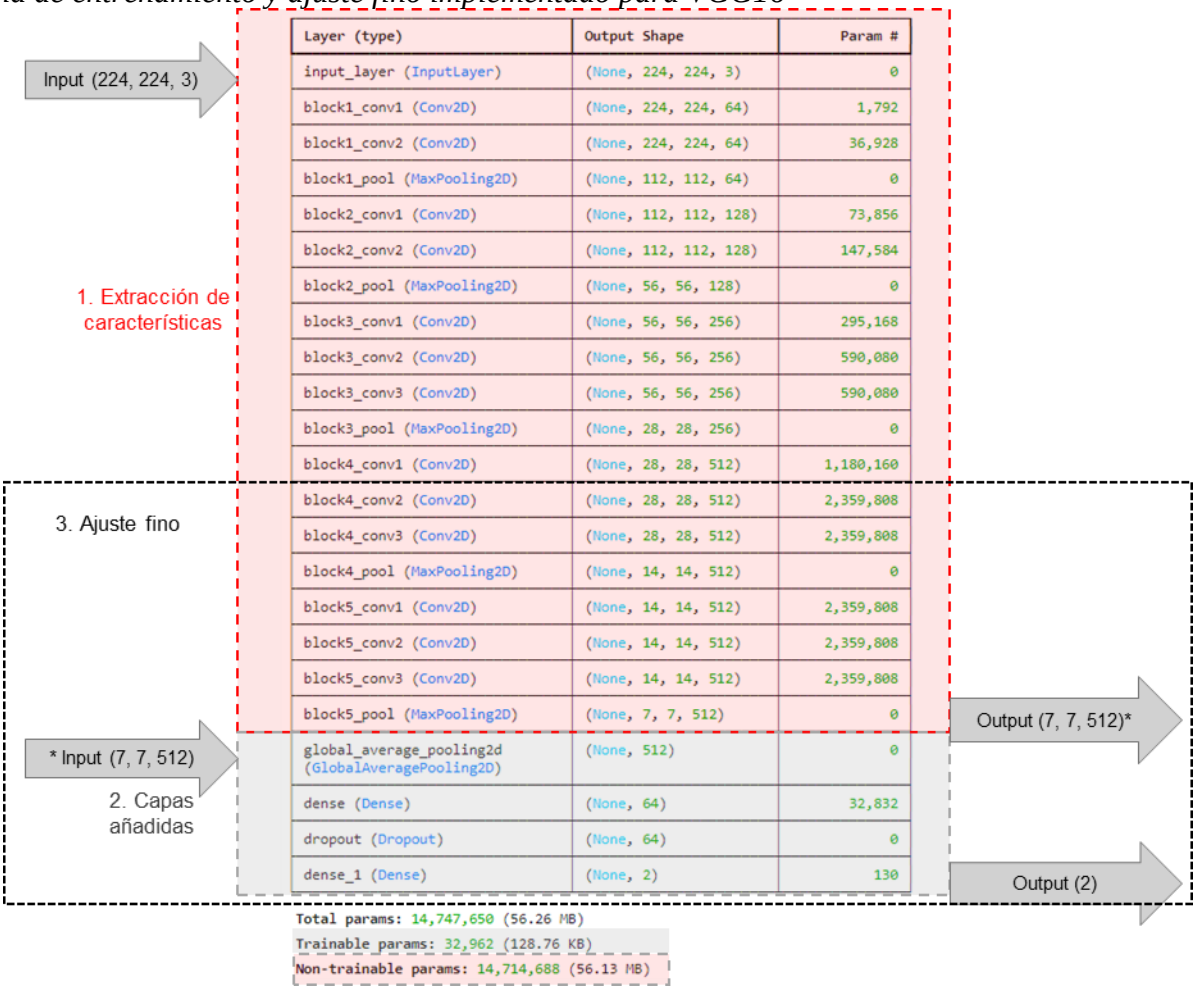
Luego, se añadió una capa densa (`Dense(64, activation='relu'))` de 512 unidades y activación ReLU para introducir no linealidad al modelo. También se incluyó una capa de `Dropout(0.5)` para regularizar el modelo durante el entrenamiento. Finalmente, se agrega una capa

densa de salida (`Dense(2, activation='softmax'))` con 2 unidades y activación softmax para obtener las probabilidades de pertenencia a cada una. Estos parámetros de la arquitectura de las capas personalizadas resultaron del modelo base previamente descrito.

Inicialmente, se congelaron todas las capas de los modelos pre-entrenados para entrenar solo las capas añadidas, y el modelo se compiló con “categorical_crossentropy”, el optimizador Adam con una tasa de aprendizaje de 0.0001 (mejor resultado de las combinaciones del modelo base), y las métricas precisión (accuracy) y sensibilidad (recall). Tras un entrenamiento inicial, se descongelaron las 3 capas añadidas: “GlobalAveragePooling”, densa (512), densa (2) y 7 capas superiores más de la red original para permitir su ajuste fino, recompilando el modelo con una tasa de aprendizaje más baja y realizando un entrenamiento adicional. La **Figura 13** ilustra el procedimiento en VGG16.

Figura 13

Esquema de entrenamiento y ajuste fino implementado para VGG16



Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1,792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36,928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73,856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147,584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295,168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590,080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590,080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 512)	0
dense (Dense)	(None, 64)	32,832
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 2)	130

Total params: 14,747,650 (56.26 MB)
Trainable params: 32,962 (128.76 KB)
Non-trainable params: 14,714,688 (56.13 MB)

Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto: 03_transfer_learning_VGG16.ipynb (Ortega A., 2024). Modificada en Microsoft Power Point.

Los modelos finales fueron evaluados en los conjuntos de entrenamiento, validación y prueba, y se compararon los resultados con los del modelo base. La evaluación del modelo final incluyó el análisis del historial de entrenamiento y la comparación de métricas de rendimiento antes y después del ajuste fino en cada uno. La **Tabla 1** muestra una comparación de la arquitectura de estas dos redes.

Tabla 1

Comparación de la arquitectura y aspectos varios de los modelos implementados

Aspecto	VGG16	ResNet50
Año	2014	2015
Conjunto de imágenes	ImageNet	ImageNet
Capas convolucionales	16	50
Dimensión de entrada	224x224x3	224x224x3
Función de activación	ReLu	ReLu
Preprocesamiento	Convertir las imágenes de RGB a BGR y centrar cada canal de color en cero con respecto al conjunto de datos ImageNet, sin escalar.	Convertir las imágenes de RGB a BGR y normalización de los pixeles entre -1 y 1 centrando los valores en 0.
Regularizadores L1, L2	No	No
Parámetros (M)	138.4	25.5
Filtros	(3x3) 64, 128, 256, 512	La mayoría (3x3) 64, 128, 256, 512
Pooling	(2x2)	
Capa de Dropout	Sí, 0.5	No
Función capa de predicción	Softmax	Softmax
Función de pérdida	Categorical crossentropy	Categorical crossentropy
Tasa de error	7.3	3.57
Capas densas	3	1
FLOPs[3]	19.6	3.6
Entrenamiento	Más lento debido a mayor tamaño	Más rápido debido a menor complejidad

Nota: elaboración propia, diversas fuentes [8], [29], [32].

5.3. Iteraciones y evolución

Las iteraciones subsiguientes, se enfocaron en mejorar las métricas obtenidas en el modelo base con los siguientes dos puntos metodológicos:

[3] FLOPs: Floating Point Operations Per Second, es una medida de rendimiento computacional que indica cuántas operaciones de punto flotante un sistema informático puede realizar en un segundo.

- **Entrenamiento inicial con capas congeladas:** todas las capas se mantuvieron congeladas, entrenando únicamente las nuevas capas añadidas. Este enfoque permitió establecer una base sólida sin sobre ajustar el modelo inicial.
- **Ajuste fino con capas descongeladas:** se descongelaron las capas superiores para realizar un ajuste fino de los modelos. Este ajuste permitió que el modelo refinara las características previamente aprendidas.

5.4. Herramientas

Se emplearon diversas herramientas para facilitar el preprocesamiento de datos, el modelado, la evaluación y la gestión del código:

- **Jupyter Notebook:** utilizado para el desarrollo y ejecución interactiva del código, permitiendo una integración fluida de código, texto y visualizaciones para documentar el flujo de trabajo y los resultados obtenidos.
- **Google Collab y Kaggle:** se utilizaron como interfaces de desarrollo para ejecutar los Notebooks usando las GPU disponibles en cada caso. Las características se muestran en la **Figura 14**.
- **TensorFlow y Keras:** estas bibliotecas fueron utilizadas para construir y entrenar los modelos de CNN, así como definir y ajustar las arquitecturas VGG16 y ResNet50v2.
- **Scikit-learn:** empleada para evaluar el rendimiento de los modelos con las métricas descritas (precisión y recall).
- **Git y GitHub:** se utilizó para el control de versiones del código, y GitHub sirvió como repositorio remoto para almacenar y compartir el proyecto.

Figura 14

Características del hardware y software utilizado en Google Collab

```
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.104.05             Driver Version: 535.104.05   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P0              26W /  70W |    103MiB / 15360MiB |      3%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
+---------------------------------------------------------------------------------------+
```

Nota. Código disponible en el repositorio asociado al proyecto: 03_transfer_learning_VGG16.ipynb (Ortega A., 2024).

6. Resultados y discusión

6.1. Métricas

6.1.1. Baseline

Los resultados iniciales muestran una variabilidad significativa en el desempeño del modelo en las distintas iteraciones evaluadas. La precisión y la sensibilidad oscilan ampliamente. Mientras que las iteraciones 1 y 2 destacan con una precisión y recall superiores al 90 %, otras iteraciones como la 4, 5, 6, 8, 12 y 13 muestran valores de accuracy menores a 50 %. Los resultados de las iteraciones del modelo base se muestran en la **Tabla 2**.

Tabla 2

Resultados de las 14 iteraciones del modelo base evaluado en el conjunto de validación

Optimizador: Adam

Iter.	Dropout	Tasa de aprendizaje	Regularizador	Tiempo (s)	Pérdida (Loss)	Precisión (Accuracy)	Recall
1	0.5	0.001	L2	2.064246	0.346588	0.905138	0.905138
2	0.5	0.0001	L2	0.849627	0.530615	0.916996	0.916996
3	NA	0.0001	NA	0.929514	0.596743	0.743083	0.743083
4	N/A	0.001	L2	0.843437	3.656449	0.407115	0.407115
5	0.5	0.001	L1	0.920575	105.542320	0.509881	0.509881
6	0.5	0.001	L1, L2	1.002730	105.988235	0.521739	0.521739
7	0.5	0.001	N/A	0.817490	4.159244	0.695652	0.695652

Optimizador: RMSprop

Iter.	Dropout	Tasa de aprendizaje	Regularizador	Tiempo (s)	Pérdida (Loss)	Precisión (Accuracy)	Recall
8	0.5	0.001	L2	0.819710	1.871595	0.521739	0.521739
9	0.5	0.0001	L2	0.757079	0.658700	0.766798	0.766798
10	N/A	0.0001	N/A	0.803251	8.862959	0.786561	0.786561
11	N/A	0.001	L2	0.812602	2.694685	0.671937	0.671937
12	0.5	0.001	L1	0.792441	49.105217	0.521739	0.521739
13	0.5	0.001	L1, L2	0.872821	49.070683	0.521739	0.521739
14	0.5	0.001	N/A	0.812607	0.639003	0.841897	0.841897

El uso de Dropout de 0.5 mejoró la precisión en validación, comparando las iteraciones con y sin Dropout, la 1 con la 4 y la 8 con 11, se muestra una mejora significativa cuando se incluye un hallazgo corroborado en múltiples estudios [8], [37], [38]. Respecto al optimizador, no se evidenció una diferencia tan significativa. Sin embargo, Adam mostró un rendimiento ligeramente

superior en algunas de las iteraciones, la literatura respalda el uso de este optimizador como una opción efectiva para CNN (Kingma & Ba, 2014).

La inclusión de regularizadores L1 y L2, o su combinación no tuvo un impacto significativo en el rendimiento del modelo en términos de pérdida, precisión o recall. En varias iteraciones, independientemente de la presencia o ausencia de regularizadores, las métricas similares. Por ejemplo, al comparar la iteración 1 con la 7 que utilizan Adam, el uso del regularizador L2 mejoró el desempeño. Sin embargo, ocurre lo contrario cuando se comparan las iteraciones 8 y 14 que usaron el optimizador RMSprop. Esto sugiere que, para el conjunto de imágenes específico del regularizador L2 pueden contribuir en muy baja proporción al rendimiento del modelo. En la literatura, los resultados son también muy amplios, se han encontrado estudios en donde la implementación de estas técnicas mejora el desempeño de los modelos, pero otros en donde no.

Solo los modelos 1 y 2 lograron avanzar en el entrenamiento, mientras que los demás fueron detenidos por "Early stopping" después de 20 épocas, siendo la primera época la más eficiente para estos modelos. Las iteraciones 4, 5, 6, 8, 12 y 13 lo cual podría deberse a que la tasa de aprendizaje sea alta, lo que impide que el modelo siga aprendiendo después de cierto punto. Por ejemplo, cuando se cambió la tasa de aprendizaje en las iteraciones 8 y 9 de 0.001 a 0.0001, hubo una mejoría notable. Otro motivo puede ser la arquitectura del modelo, que, al no ser muy profunda, alcanzó su capacidad máxima de aprendizaje. Por último, la implementación de los métodos de regularización, sobre todo en sus combinaciones, limitaron la capacidad del modelo para mejorar, como se observa en las iteraciones 6 y 13, estabilizando el aprendizaje después de cierto número de épocas. Las curvas ROC/AUC de las 14 iteraciones se muestran en el **Anexo 1**.

Aunque la iteración 1 y 2 presentaron las mejores métricas en validación, la iteración 2 requiere más épocas para alcanzarla. Por lo tanto, una combinación similar a la iteración 2, con optimizador Adam a una tasa de aprendizaje de 0.0001 y Dropout de 0.5, representó el punto de partida para los modelos de transferencia de aprendizaje utilizados en adelante.

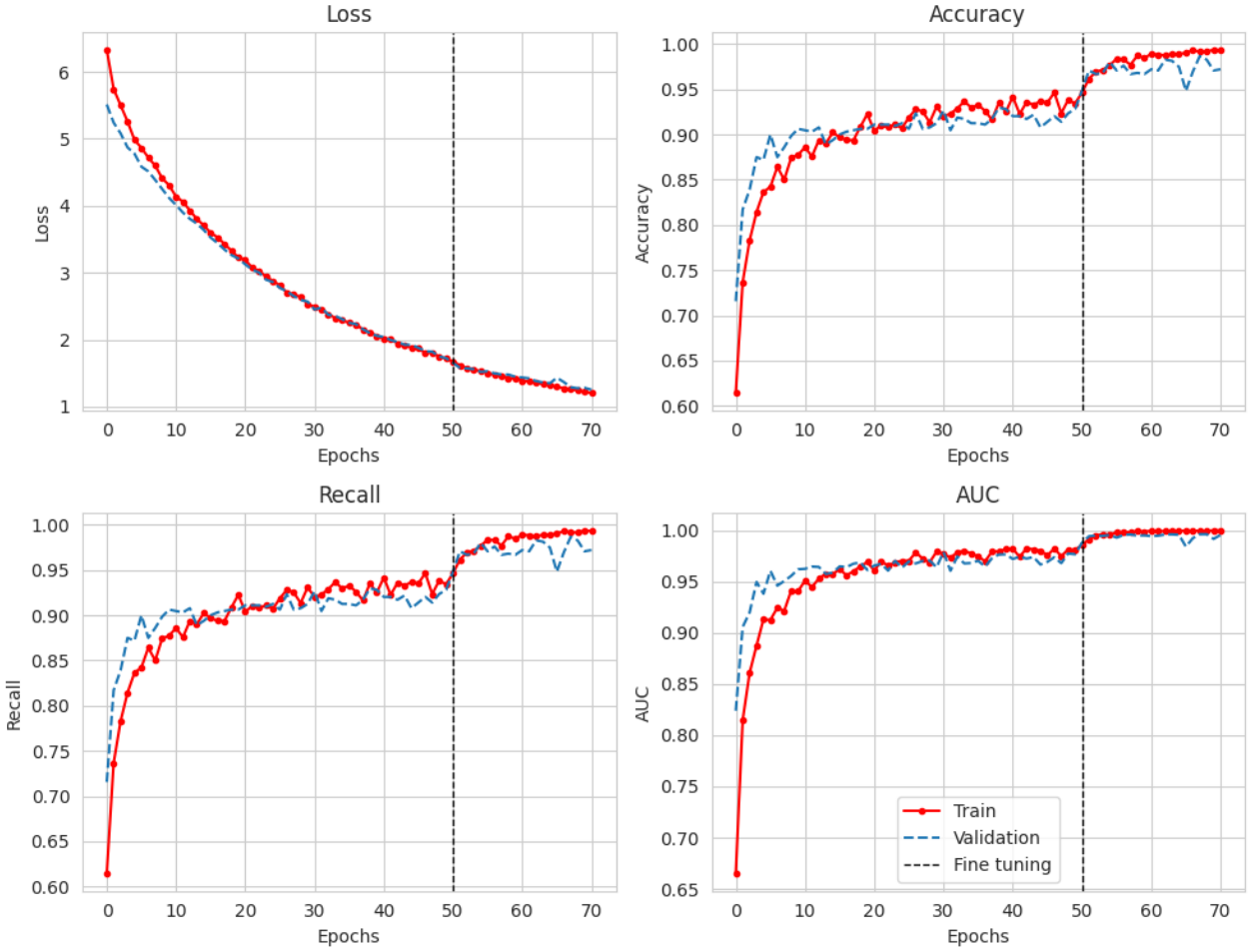
6.1.2. Transferencia de aprendizaje

Siguiendo con los resultados obtenidos del modelo base, se implementaron los modelos de transferencia de aprendizaje: VGG16 y ResNet50v2. A continuación, se presentan y analizan los resultados de estas implementaciones.

El historial de entrenamiento del modelo de transferencia de aprendizaje VGG16 y ResNet50v2 se muestran en la **Figura 15** y **Figura 16**, respectivamente. En general las métricas de precisión (accuracy), sensibilidad (recall) y AUC muestran un aumento gradual a lo largo del entrenamiento, tanto para el conjunto de entrenamiento como de validación. La mejora en estas métricas se mantuvo durante las primeras 50 épocas y no se activó el mecanismo de EarlyStopping para ninguno de los dos modelos, lo que sugiere que podrían seguir siendo entrenado durante más épocas. Para esto, se aplicó el proceso de ajuste fino, descongelando algunas de las capas y entrenándolas durante 20 épocas adicionales con una tasa de aprendizaje más baja.

Figura 15

Historial de entrenamiento del modelo de transferencia de aprendizaje VGG16

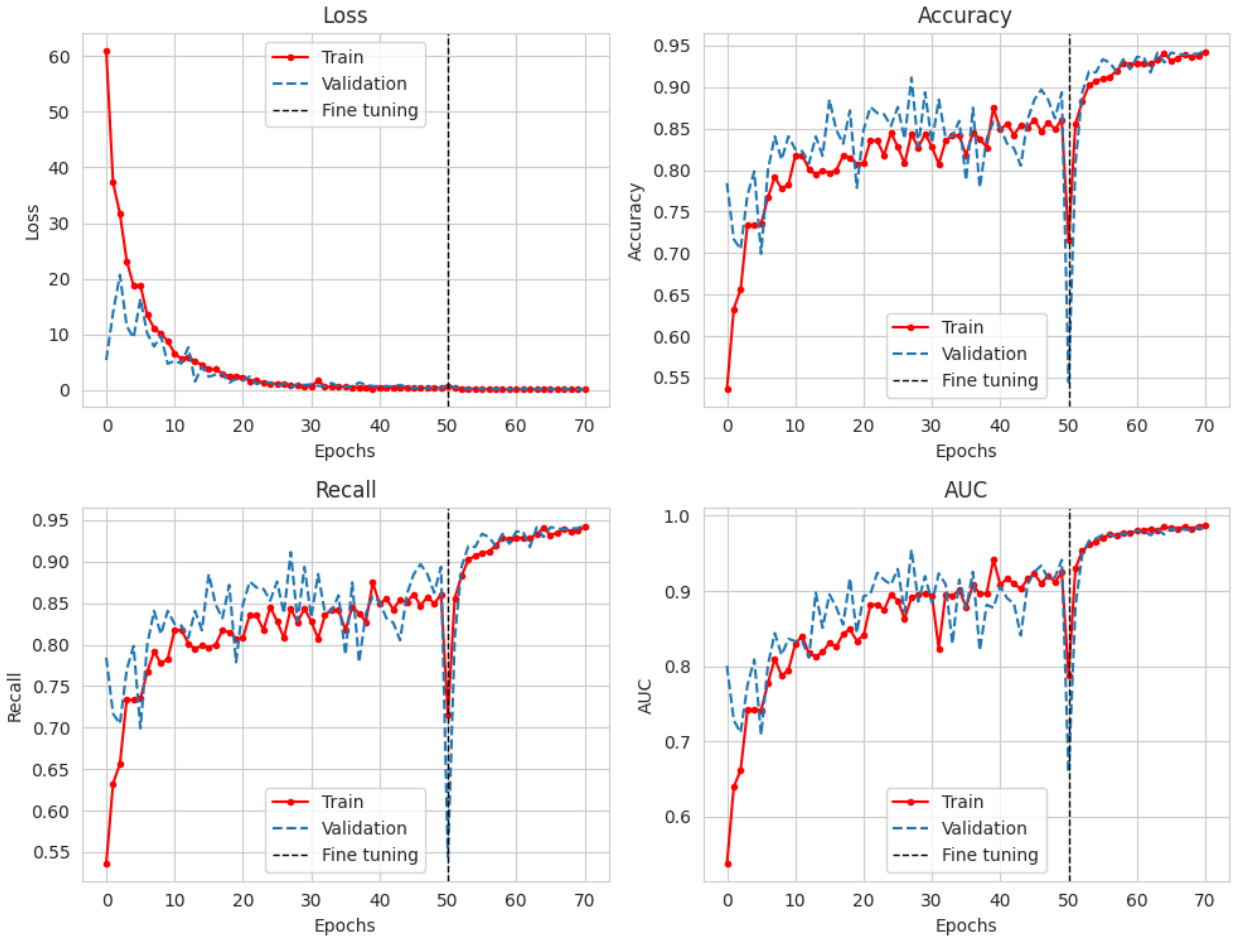


Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto: 03_transfer_learning_VGG16.ipynb (Ortega A., 2024).

Hay dos diferencias notables en el historial de entrenamiento de cada modelo. En las primeras 50 épocas de entrenamiento, VGG16 mostró una convergencia más estrecha entre las métricas de entrenamiento y validación (**Figura *15***) en comparación con ResNet50 para todas las métricas. Sin embargo, después del ajuste fino ResNet50v2 mostró una convergencia mayor entre las métricas de validación y entrenamiento. Esto puede deberse a su arquitectura más profunda (**Figura *16***).

Figura 16

Historial de entrenamiento del modelo de transferencia de aprendizaje ResNet50v2



Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto: 04_transfer_learning_ResNet50v2.ipynb (Ortega A., 2024).

La segunda diferencia, es que la pérdida durante el entrenamiento de cada modelo se comportó muy diferente: en VGG16 comenzó en alrededor de 6 y disminuyó gradualmente a lo largo de las 50 épocas de entrenamiento inicial. Por otro lado, la pérdida de ResNet50 comenzó en un valor de aproximadamente 60 y disminuyó rápidamente durante las primeras épocas, antes de nivelarse y continuar disminuyendo a un ritmo más lento. En la **Tabla 3** se presentan las mejores métricas alcanzadas durante el entrenamiento y la época en que se logró, tanto antes como después de este proceso.

Tabla 3

Resultados del entrenamiento de VGG16 y ResNet50v2 antes y después del ajuste fino. Se muestra la época respectiva entre paréntesis

Modelo	Métrica (validación)	Antes de ajuste fino	Después de ajuste fino
VGG16	Precisión	0.929688 (38)	0.98672 (67)
VGG16	Sensibilidad	0.929688 (38)	0.98672 (67)
VGG16	AUC	0.978848 (30)	0.996181 (67)
ResNet50v2	Precisión	0.911504 (27)	0.944223 (70)
ResNet50v2	Sensibilidad	0.955126 (27)	0.944223 (70)
ResNet50v2	AUC	0.911504 (27)	0.984506 (69)

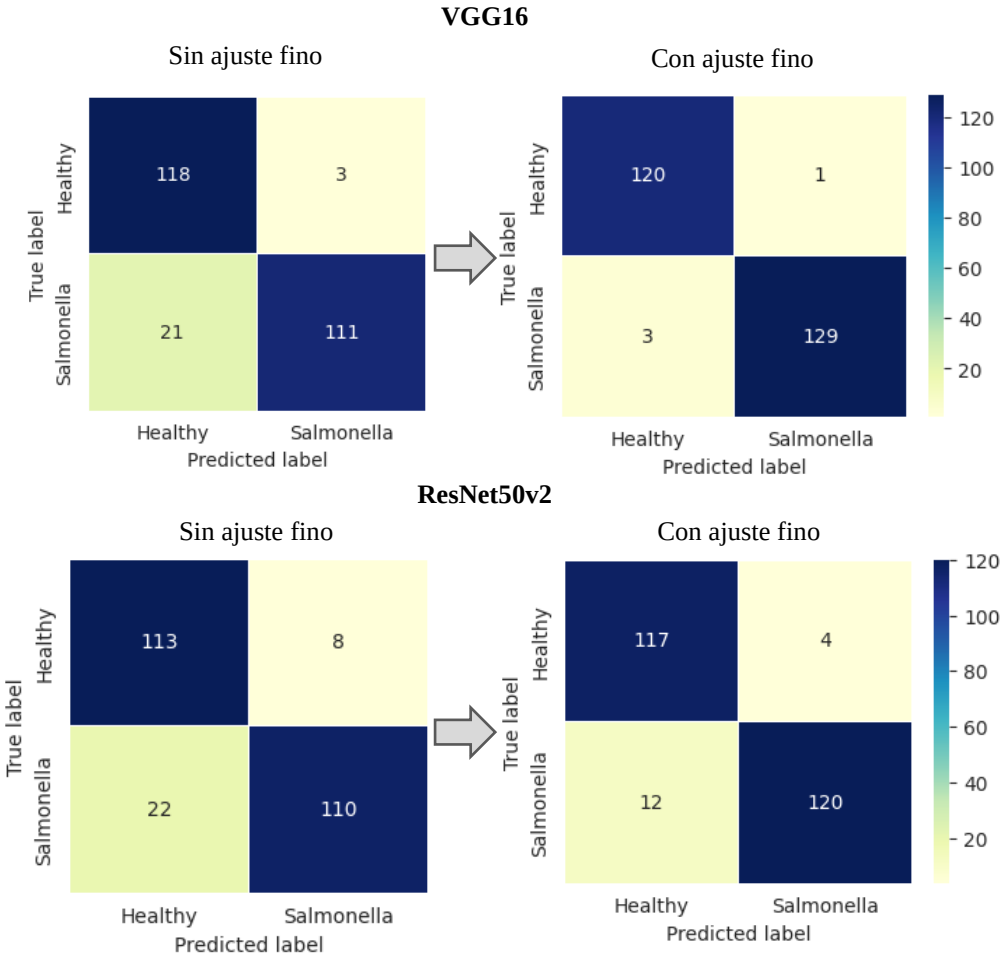
Se observa que ambos modelos mejoraron notablemente todas sus métricas después del ajuste fino, excepto la sensibilidad (recall) en ResNet50v2. De esta manera, se confirma que el ajuste fino es una metodología que mejora el desempeño de los modelos pre entrenados de DL utilizados para el problema de clasificación abordado. Adicionalmente, se observa que VGG16 requiere más épocas en alcanzar mejores métricas antes del ajuste fino que ResNet50v2, pero menos después del ajuste fino.

En general, aunque ambos modelos mostraron una mejora en sus métricas después del proceso de ajuste fino, VGG16 tuvo un mejor desempeño tanto antes como después de este. Esto podría deberse a que VGG16 tiene una arquitectura más sencilla y, por lo tanto, es más fácil de entrenar y ajustar.

Después del entrenamiento, se evaluaron los modelos en el conjunto de prueba, es decir, imágenes no vistas con anterioridad. Los resultados se presentados en la son consistentes con lo visto durante el entrenamiento, en donde ambos modelos mostraron una mejora en su desempeño después del ajuste fino, siendo VGG16 el modelo que muestra una mejora más significativa. También se puede observar que la mayoría de los errores ocurrieron al clasificar imágenes etiquetadas con Salmonella *spp*. como saludables, es decir, falsos negativos. Este resultado, implicaría para el negocio la subestimación de contaminación microbiológica en sus granjas. Los resultados de las matrices de confusión son respaldados por las curvas ROC/AUC del Anexo 3.

Figura 17

Matrices de confusión para CGG16 y ResNet50v2 antes y después del ajuste fino



Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto (Ortega A., 2024).

Es importante tener en cuenta que el desempeño del modelo puede variar según el conjunto de datos utilizado para su prueba. Por lo que se recomienda realizar más pruebas y validación cruzada para obtener una evaluación más precisa, incluso con fotografías más recientes tomadas en Colombia. La **Tabla 4** muestra los resultados al evaluar los modelos en el conjunto de prueba.

Tabla 4
Resumen de resultados evaluados en el conjunto de prueba

Modelo	Precisión	Sensibilidad	AUC	Tiempo (s)	Parámetros
Baseline	0.91699	0.91699		0.849627	9,536,162
VGG16	0.98419	0.98419	0.99114	1.341505	1,497,837
ResNet50v2	0.93675	0.93675	0.98846	0.78036	2,461,491

Los resultados muestran que VGG16 y ResNet50v2 superaron al modelo base en términos de precisión y sensibilidad. Además, VGG16 obtuvo mejores métricas que el modelo base y ResNet50v2. Sin embargo, es importante tener en cuenta el tiempo de entrenamiento y el número de parámetros de cada modelo, ya que el tiempo de entrenamiento de VGG16 fue mayor en comparación con el modelo base, lo que puede afectar el rendimiento y la velocidad en una eventual implementación del modelo.

6.2. Evaluación cualitativa

Como se describió en la definición del problema (Sección 1.1, pág. 11), la clasificación precisa de imágenes de heces de aves puede ser de gran ayuda para la industria avícola del país, ya que permite una detección rápida de enfermedades y facilita la toma de decisiones, especialmente al considerar una mayor cantidad de muestras para decidir, por ejemplo, si medicar o no un galpón.

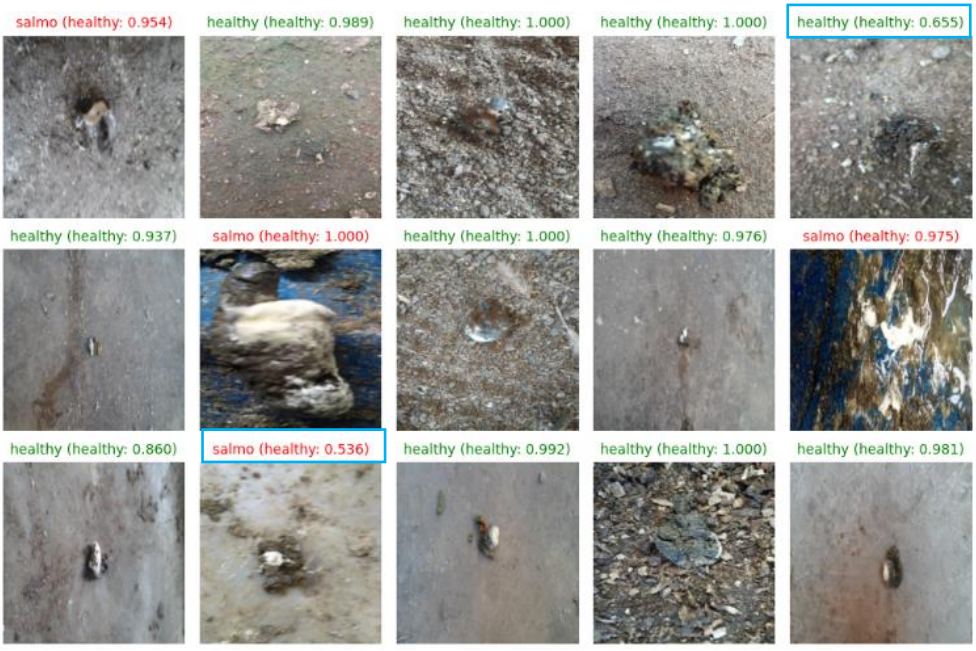
En términos de sobre entrenamiento, se observó que los modelos tienen un buen equilibrio entre la complejidad del modelo y el ajuste a los datos, lo que se refleja en una buena precisión y sensibilidad cuando se probaron sobre el conjunto de prueba (matrices de confusión y curvas ROC). Sin embargo, es importante tener en cuenta que el conjunto de datos utilizado en el entrenamiento corresponde a imágenes tomadas en África entre 2020 y 2021, es importante actualizar la base de datos y evaluar la generalización del modelo a nuevos datos, que no hacen parte del conjunto encontrado en Kaggle. La **Figura 18** muestra las predicciones hechas por cada modelo, con su respectiva probabilidad de pertenencia a cada clase. Esto muestra que el modelo ResNet50v2 presenta mayor incertidumbre en algunas predicciones. Sin embargo, en general, los modelos logran clasificar correctamente la mayoría de las muestras, incluso aquellas con cierta incertidumbre.

Figura 18

Predicciones de los modelos en el tren de prueba. El título de cada imagen representa la etiqueta real (predicción: probabilidad). Arriba: VGG16, Abajo: ResNet50v2.

VGG16

ResNet50v2



Nota. Elaboración propia, código disponible en el repositorio asociado al proyecto (Ortega A., 2024).

6.3. Consideraciones de producción

Es importante recordar que la solución diseñada impactaría al negocio en la medida en que mejora su capacidad para ayudar en la toma decisiones rápidas, por ejemplo, si medicar un galpón o no. La capacidad de enviar una foto y recibir una predicción implica el uso de tecnologías como Kafka, Databricks, Spark Streaming o similares para garantizar que el modelo esté siempre actualizado y pueda tomar decisiones precisas en tiempo real.

La infraestructura de despliegue elegida es Google Cloud Platform, en donde se utilizarán dos servicios principales: Cloud Storage (S3) para almacenar los archivos de configuración, el modelo entrenado, las clases y las imágenes que se reciban; y AI Platform, en donde se creará un Notebook de Tensorflow con el código que permita cargar los modelos guardados en Cloud Storage. Este Notebook tendrá la configuración para recibir las imágenes y otro archivo que realice la inferencia, así como un archivo requirements.txt y un archivo Docker. Posteriormente, se debe tokenizar (binarizar) todo el proyecto y subirlo a AI Platform. La utilización de estos servicios en la nube permitirá desplegar el modelo de forma rápida y escalable, adaptándose a las demandas cambiantes del entorno de producción.

Una vez el modelo esté desplegado, se deben configurar alertas para detectar cuando el error del modelo supere el 30 %. Estos mecanismos permiten tomar acciones correctivas rápidas y garantizar que el modelo siga siendo, al menos igual de preciso. Otras buenas prácticas de programación en Ciencia de Datos, como el versionado y la documentación, son fundamentales para garantizar una posible transición del modelo a producción.

A estas consideraciones técnicas, deben añadirse implicaciones éticas y legales, pues se trata de almacenar y utilizar información confidencial de las compañías productoras de aves de granja. Es importante recordar que el estado de salud de sus aves no sólo impacta su productividad y reputación, sino la seguridad alimentaria del país.

6.4. Trabajos futuros

1. **Modularizar:** debido a que se presentaron 4 notebooks de Jupyter diferentes [39], se repitieron fragmentos de código para generar figuras y tablas de resultados, estos fragmentos pueden ser funciones que se importen en cada Notebook para mejorar la escalabilidad y facilitar futuras actualizaciones del proyecto.
2. **Mejoramiento de la resolución:** experimentar con diferentes técnicas de preprocesamiento de imágenes para aumentar su resolución y determinar si mejora la capacidad del modelo para detectar características relevantes.
3. **Técnicas de balanceo de datos:** implementar y evaluar técnicas de balanceo de datos para mejorar la representación de la clase minoritaria (Healthy) y evaluar el impacto en las métricas de desempeño.
4. **Análisis del tiempo de entrenamiento:** incluir en el análisis la duración del entrenamiento de cada modelo para comprender mejor el costo computacional asociado. Esto puede hacerse personalizando los callbacks, en donde se registre el tiempo en que comienza el entrenamiento y finalización de cada época y general.
5. **Optimización de hiperparámetros:** para los modelos con mejor desempeño, investigar la influencia de otros hiperparámetros como la paciencia en EarlyStopping que se mantuvo constante durante este proyecto, así como el tamaño del lote.
6. **Exploración de otros modelos:** probar arquitecturas como GoogleNet para evaluar si ofrecen mejoras en precisión y eficiencia en comparación con los modelos actuales.
7. **Despliegue en producción**: realizar un despliegue adecuado del modelo utilizando herramientas como MLflow para su puesta en producción y uso en campo. Esto permitiría evaluar la viabilidad del modelo en un entorno real y su potencial impacto práctico.

Referencias

[1] OMS, «Food Safety». Accedido: 31 de mayo de 2024. [En línea]. Disponible en: https://www.who.int/health-topics/food-safety#tab=tab_1

[2] Federación Nacional de Avicultores de Colombia, «Estadísticas». Accedido: 21 de marzo de 2024. [En línea]. Disponible en: https://fenavi.org/estadisticas/consumo-per-capita-mundo-pollo/

[3] FENAVI, «Boletín FENAVIQUIN», *https://fenavi.org/boletin-fenaviquin/fenaviquin-edicion-390-noviembre-15-de-2023/*, 15 de noviembre de 2023.

[4] Instituto Colombiano Agropecuario (ICA), *Resolución No. 00017754 de 2019*. Colombia, 2019.

[5] L. Ma, J. Yi, N. Wisuthiphaet, M. Earles, y N. Nitin, «Accelerating the Detection of Bacteria in Food Using Artificial Intelligence and Optical Imaging», *Appl Environ Microbiol*, vol. 89, n.º 1, ene. 2023, doi: 10.1128/aem.01828-22.

[6] Instituto Nacional de Salud, *Resolución Número 0140 de 2021*. Colombia: https://www.ins.gov.co/TyS/Documents/LISTA%20DE%20PRECIOS%20A%C3%91O%202021%20-%20PUBLICACION%20PAGINA%20WEB%20INS.pdf, 2021.

[7] T. Obe *et al.*, «Controlling Salmonella: strategies for feed, the farm, and the processing plant», *Poult Sci*, vol. 102, n.º 12, p. 103086, dic. 2023, doi: 10.1016/j.psj.2023.103086.

[8] L. Alzubaidi *et al.*, «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions», *J Big Data*, vol. 8, n.º 1, 2021, doi: 10.1186/s40537-021-00444-8.

[9] R. O. Ojo, A. O. Ajayi, H. A. Owolabi, L. O. Oyedele, y L. A. Akanbi, «Internet of Things and Machine Learning techniques in poultry health and welfare management: A systematic literature review», *Comput Electron Agric*, vol. 200, p. 107266, sep. 2022, doi: 10.1016/j.compag.2022.107266.

[10] D. Verma, N. Goel, y V. K. Garg, «A Review of Machine Learning Models for Disease Prediction in Poultry Chickens», pp. 723-737, 2023, doi: 10.1007/978-981-99-4626-6_59.

[11] Xiao Yang, Zihao Wu, Haixing Dai, Ramesh Bahadur Bist, Sachin Subedi, y Jin Sun, «An innovative segment anything model for precision poultry monitoring», *Comput Electron Agric*, vol. 222, jul. 2022.

[12] M. Campbell *et al.*, «A computer vision approach to monitor activity in commercial broiler chickens using trajectory-based clustering analysis», *Comput Electron Agric*, vol. 217, p. 108591, feb. 2024, doi: 10.1016/j.compag.2023.108591.

[13] L. Cao *et al.*, «Automated chicken counting in surveillance camera environments based on the point supervision algorithm: Lc-densefcn», *Agriculture (Switzerland)*, vol. 11, n.º 6, 2021, doi: 10.3390/agriculture11060493.

[14] O. Geffen, Y. Yitzhaky, N. Barchilon, S. Druyan, y I. Halachmi, «A machine vision system to detect and count laying hens in battery cages», *Animal*, vol. 14, n.º 12, pp. 2628-2634, 2020, doi: 10.1017/S1751731120001676.

[15] Asimetrix, «PigVision», https://asimetrix.co/es/smart-farm.

[16] Sento Electronics Scaling Hub, «Pigvision, a Sento Production», https://www.youtube.com/watch?v=jKnB4DrinUQ.

[17] Y. Cui, X. Kong, C. Chen, y Y. Li, «Research on broiler health status recognition method based on improved YOLOv5», *Smart Agricultural Technology*, vol. 6, p. 100324, dic. 2023, doi: 10.1016/j.atech.2023.100324.

[18] S. Zakareya, H. Izadkhah, y J. Karimpour, «A New Deep-Learning-Based Model for Breast Cancer Diagnosis from Medical Images», *Diagnostics*, vol. 13, n.º 11, 2023, doi: 10.3390/diagnostics13111944.

[19] Z. Yue, L. Ma, y R. Zhang, «Comparison and Validation of Deep Learning Models for the Diagnosis of Pneumonia», *Comput Intell Neurosci*, vol. 2020, pp. 1-8, sep. 2020, doi: 10.1155/2020/8876798.

[20] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, y D. P. Hughes, «Deep Learning for Image-Based Cassava Disease Detection», *Front Plant Sci*, vol. 8, oct. 2017, doi: 10.3389/fpls.2017.01852.

[21] K. P. Ferentinos, «Deep learning models for plant disease detection and diagnosis», *Comput Electron Agric*, vol. 145, 2018, doi: 10.1016/j.compag.2018.01.009.

[22] S. A. Shaik Mazhar y D. Akila, «Machine Learning and Sensor Roles for Improving Livestock Farming Using Big Data», 2023, pp. 181-190. doi: 10.1007/978-981-19-2538-2_17.

[23] M. E. Hossain, M. A. Kabir, L. Zheng, D. L. Swain, S. McGrath, y J. Medway, «A systematic review of machine learning techniques for cattle identification: Datasets,

methods and future directions», *Artificial Intelligence in Agriculture*, vol. 6, pp. 138-155, 2022, doi: 10.1016/j.aiia.2022.09.002.

[24] Kaggle, «Chicken Disease Image Classification». Accedido: 31 de mayo de 2024. [En línea]. Disponible en: https://www.kaggle.com/datasets/allandclive/chicken-disease-1

[25] International Organization for Standardization (ISO), *ISO 16140-2:2016 Microbiology of the food chain -- Method validation -- Part 2: Protocol for the validation of alternative (proprietary) methods against a reference method*.

[26] «Scalars — NumPy v1.26 Manual», https://numpy.org/doc/stable/reference/arrays.scalars.html#numpy.uint.

[27] Keras, «Keras Data Generators», https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.

[28] M. Vakalopoulou, S. Christodoulidis, N. Burgos, O. Colliot, y V. Lepetit, «Deep Learning: Basics and Convolutional Neural Networks (CNNs)», *Neuromethods*, vol. 197, pp. 77-115, jul. 2023, doi: 10.1007/978-1-0716-3195-9_3.

[29] «Keras Applications», https://keras.io/api/applications/.

[30] A. Krizhevsky, I. Sutskever, y G. E. Hinton, «ImageNet classification with deep convolutional neural networks», *Commun ACM*, vol. 60, n.º 6, 2017, doi: 10.1145/3065386.

[31] G. Huang, Z. Liu, L. Van Der Maaten, y K. Q. Weinberger, «Densely connected convolutional networks», en *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.243.

[32] K. He, X. Zhang, S. Ren, y J. Sun, «Deep residual learning for image recognition», en *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.90.

[33] S. Das, «CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more…». Accedido: 26 de mayo de 2024. [En línea]. Disponible en: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

[34] K. Simonyan y A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition», sep. 2014.

[35] T. Thaker, «VGG 16 Easiest Explanation», https://medium.com/nerd-for-tech/vgg-16-easiest-explanation-12453b599526.

[36] JananiBabu, «Residual Networks (ResNet50)», https://jananisbabu.github.io/ResNet50_From_Scratch_Tensorflow/.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, y R. Salakhutdinov, «Dropout: A simple way to prevent neural networks from overfitting», *Journal of Machine Learning Research*, vol. 15, 2014.

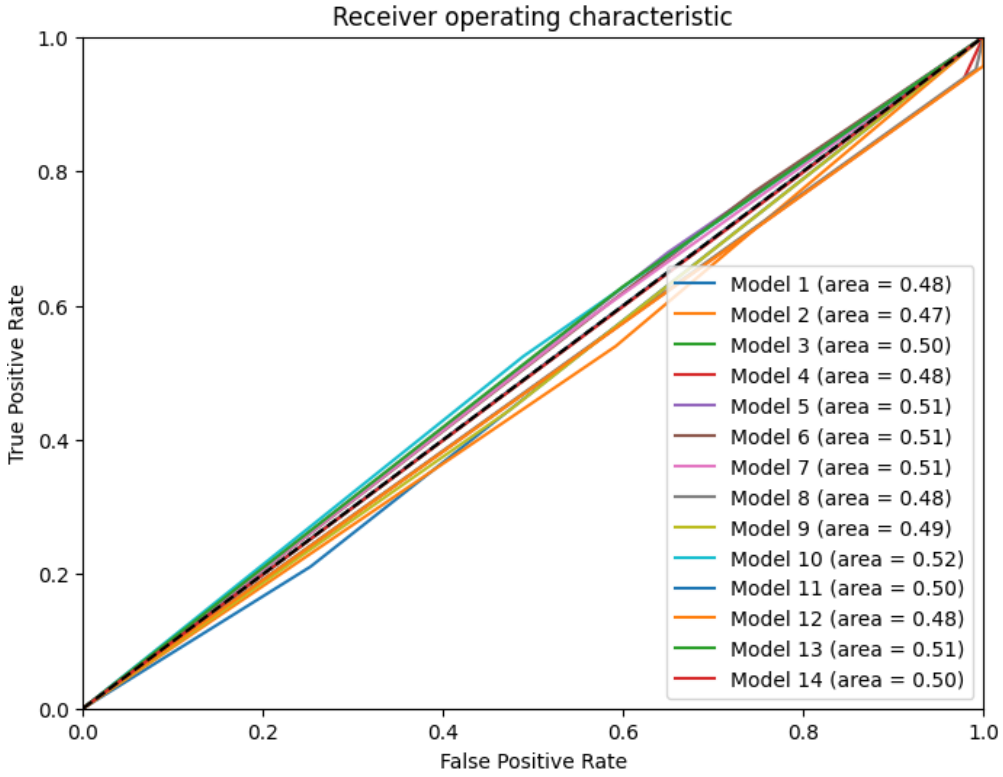
[38] M. Dimiccoli, A. Cartas, y P. Radeva, «Activity recognition from visual lifelogs: State of the art and future challenges», *Multimodal Behavior Analysis in the Wild: Advances and Challenges.*, pp. 121-134, ene. 2019, doi: 10.1016/B978-0-12-814601-9.00017-1.

[39] M. Ortega Alzate, «Redes Neuronales Convolucionales para la detección de Salmonella spp. en aves de granja», 2024. Accedido: 21 de marzo de 2024. [En línea]. Disponible en: https://github.com/melissaortegaa/monografia

[40] Real Academia Española (RAE), «Prevalencia», https://dle.rae.es/prevalencia. Accedido: 31 de mayo de 2024. [En línea]. Disponible en: https://dle.rae.es/prevalencia

Anexos

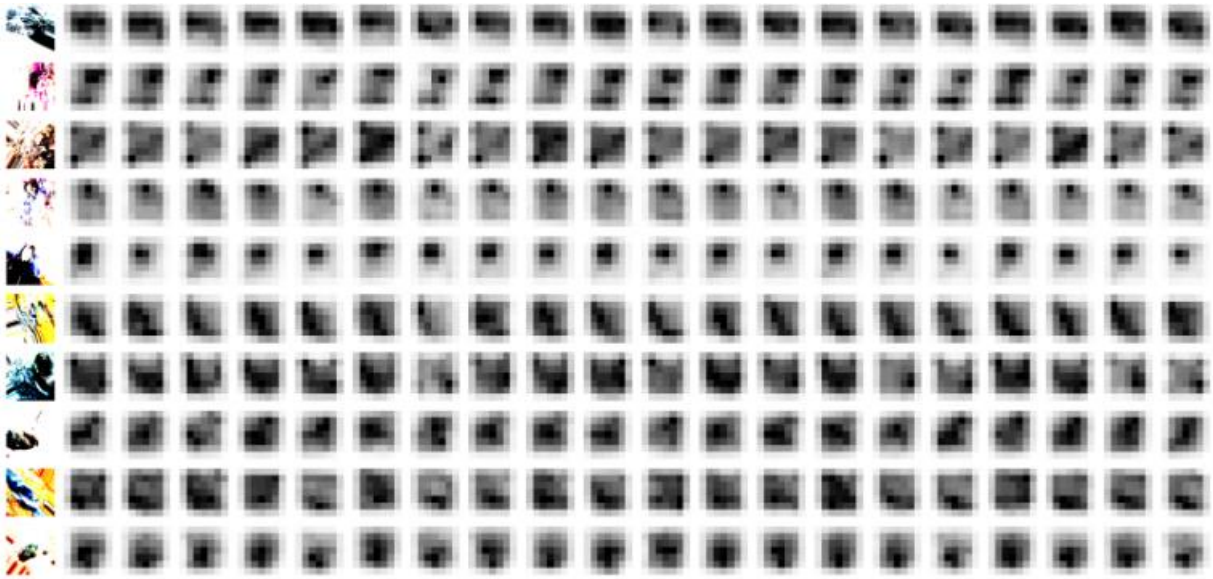
Anexo 1. Curvas ROC/AUC para las 14 iteraciones del modelo base



Anexo 2. Activaciones de la última capa convolucional de cada modelo

Se muestran 20 mapas de características para 10 imágenes aleatorias del subconjunto de entrenamiento. Cada mapa (cada columna) destaca diferentes patrones de los daos de entrada. Las activaciones son de mucha menor resolución que las imágenes originales, pero retienen la información importante.

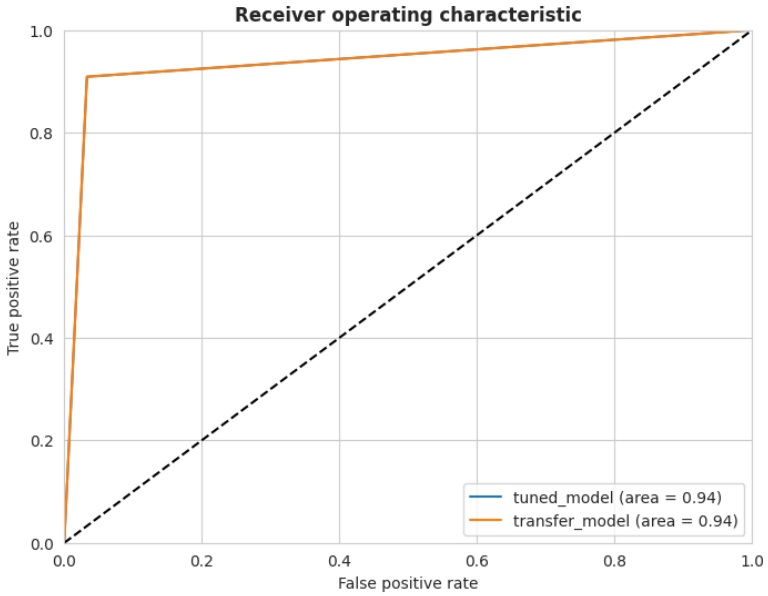
VGG16: block5_pool



ResNet50v2: conv5_block3_out

Anexo 3. Curvas ROC/AUC para cada modelo

VGG16



ResNet50v2