

# Práctica 2:

# Clasificación de

# ímágenes médicas



**Aplicaciones Avanzadas de la Inteligencia Artificial**

Marzo 2025

Máster en Ingeniería Informática

Pablo Hidalgo Delgado. NIA: 100451225

Sergio Caño Amor. NIA: 100429678

Eduardo Enrique Angulo Luna: NIA: 100545176

## **ÍNDICE**

<b>1. Introducción</b>	<b>3</b>
<b>2. Análisis exploratorio de datos (EDA)</b>	<b>3</b>
<b>3. Preproceso</b>	<b>4</b>
3.1 Dimensión única y modo RGB	4
3.2 Conversión a clases binarias (problema binario)	4
3.3 División en datos de train, validation y test	5
3.4 Aleatorización	5
3.5 Normalización	5
3.6 Data augmentation (problema multiclase)	5
<b>4. Clasificador binario</b>	<b>6</b>
4.1 Construcción del modelo	6
4.2 Experimentación	7
4.3 Resultados obtenidos	7
4.4 Análisis de resultados	8
4.5 Interpretabilidad del modelo	9
<b>5. Clasificador multiclase</b>	<b>9</b>
5.1 Construcción del modelo	10
5.2 Experimentación	10
5.3 Resultados obtenidos	10
5.4 Análisis de resultados	12
5.5 Interpretabilidad del modelo	13
<b>6. Utilidad clínica</b>	<b>14</b>
<b>7. Conclusiones</b>	<b>14</b>

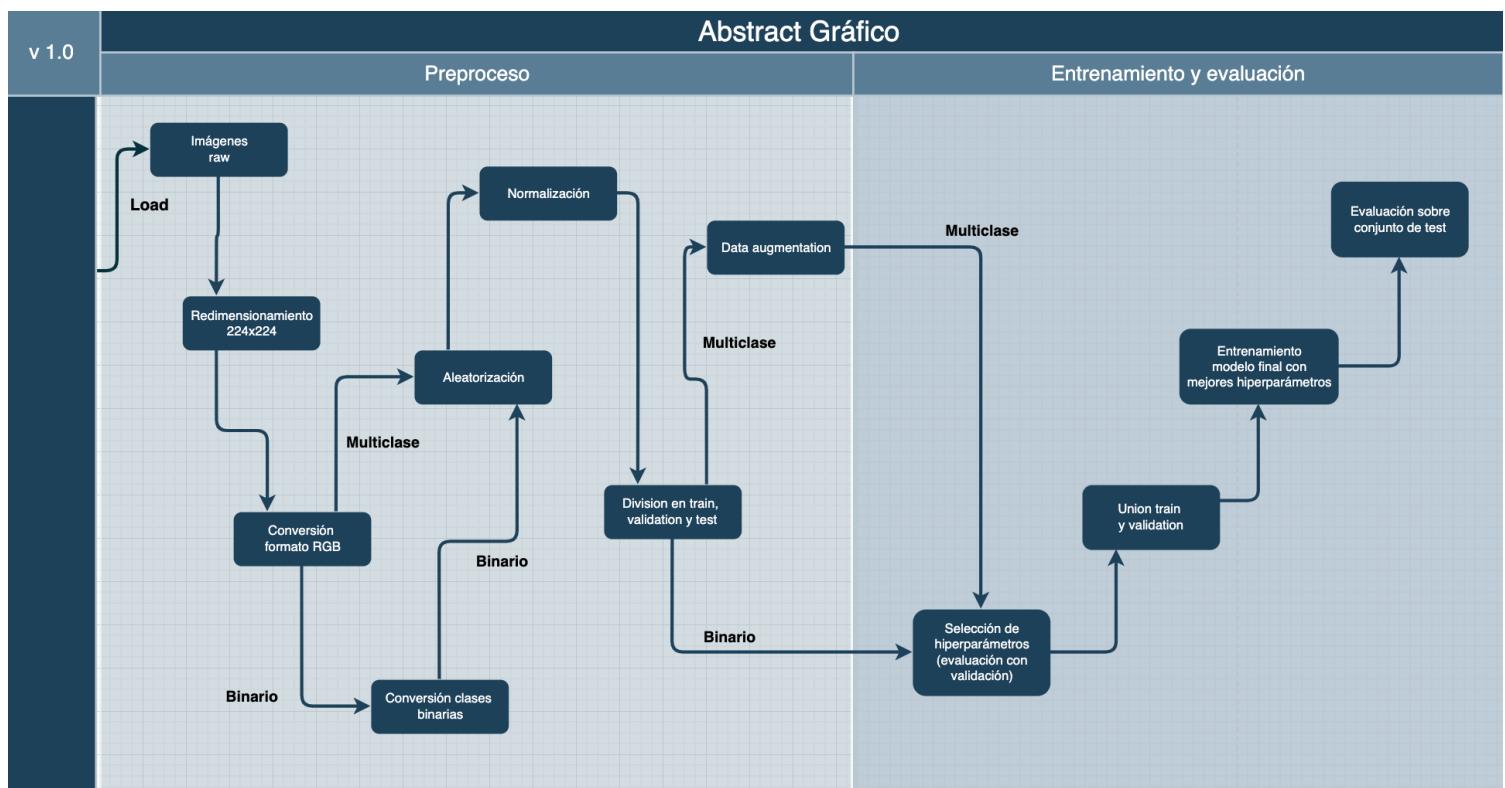
## RESUMEN EJECUTIVO

Este trabajo se centra en el desarrollo de dos modelos de clasificación de imágenes médicas, específicamente para la detección de enfermedades respiratorias como COVID-19 y neumonía bacteriana. El objetivo es, a partir de imágenes radiológicas, construir dos redes neuronales convolucionales (CNN) que puedan distinguir entre:

- Pacientes enfermos (COVID-19 y neumonía) y personas sanas (clasificador binario)
- Pacientes con COVID-19, pacientes con neumonía bacteriana y pacientes sanos (clasificador multiclas)

Para abordar este problema, se ha decidido emplear un modelo de transfer learning basado en la arquitectura ResNet50, utilizando los pesos del modelo pre-entrenado DeepCOVID-XR para el clasificador binario e ImageNet para el clasificador multiclas.

El proceso que se ha seguido para llevar a cabo este trabajo se puede ver reflejado en el siguiente diagrama:



Los resultados obtenidos sobre el conjunto de test para ambos modelos son los siguientes:

	Accuracy	Sensibilidad	Especificidad	F1-Score	AUC
<b>Binario</b>	0.98	0.97	0.99	0.98	0.996
<b>Multiclas</b>	0.96	[0.44, 0.97, 1.0]	[0.99, 0.95, 1.0]	[0.53, 0.95, 1.0]	0.974

## 1. Introducción

El objetivo de esta práctica es construir dos modelos capaces de clasificar imágenes médicas, específicamente radiografías de tórax:

- Un clasificador binario, cuyo objetivo es determinar si una radiografía de tórax presenta indicios de alguna enfermedad (COVID-19 o neumonía bacteriana) o si corresponde a un paciente sano.
- Un clasificador multiclasa, que será capaz de diferenciar entre tres categorías: COVID-19, neumonía bacteriana o sano.

Siguiendo la recomendación de la profesora, optamos por utilizar un modelo pre-entrenado, ya que esto nos permitirá obtener mejores resultados al aprovechar características previamente aprendidas en redes profundas aplicadas a imágenes médicas.

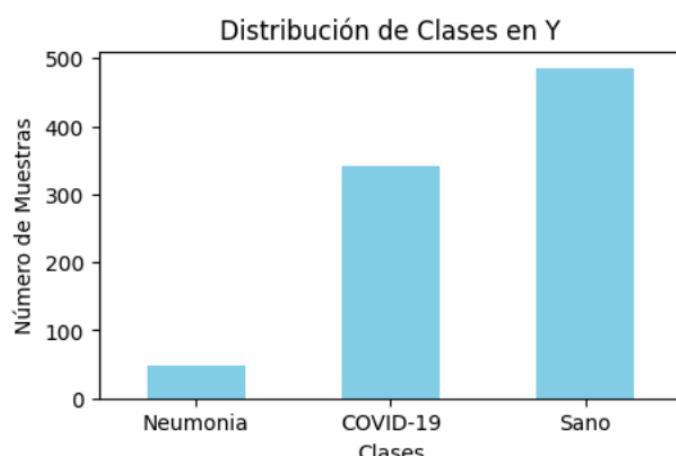
## 2. Análisis exploratorio de datos (EDA)

Antes de empezar con la división en datos de entrenamiento y test, realizamos un Análisis Exploratorio de Datos (EDA). De esta manera, investigamos y resumimos las características más importantes del conjunto de datos que disponemos. Esto nos sirve para comprender mejor las imágenes y optimizar la obtención de nuestro modelo.

Por medio de este análisis, observamos que el conjunto se compone de un total de 876 imágenes con dimensiones variables y distintos formatos de color (RGB, RGBA, P y L). Disponemos de 3 clases diferentes, y cada clase cuenta con un número distinto de imágenes, concretamente:

- Clase COVID-19: 342 imágenes
- Clase neumonía bacteriana: 48 imágenes
- Clase sano: 486 imágenes

Se puede observar claramente la distribución de clases en la siguiente gráfica:



A partir del análisis de la gráfica presentada, se puede observar que nos enfrentamos a un problema desbalanceado. En particular, la clase correspondiente a *Neumonía* presenta un número considerablemente menor de instancias en comparación con las clases *COVID-19* y *Normal*.

Este desbalance es especialmente relevante en cuanto al clasificador multiclas se refiere, ya que la escasez de ejemplos de la clase *Neumonía Bacteriana* podría afectar negativamente el desempeño del modelo, introduciendo un sesgo hacia las clases con mayor representación. Esto evidencia la necesidad de implementar técnicas de data augmentation enfocadas en la clase *Neumonía* con el objetivo de mitigar el desbalance y mejorar el rendimiento general del modelo.

En el caso del clasificador binario, el impacto de este desbalance es menos crítico, dado que la clase *Enfermo* agrupa tanto las instancias de *COVID-19* como las de *neumonía bacteriana*, alcanzando aproximadamente 400 imágenes. Este valor está relativamente equilibrado en comparación con las instancias de la clase *Sano*.

### 3. Preproceso

Previamente a entrenar los modelos, debemos realizar una transformación de los datos para que puedan ser interpretados por los algoritmos de CNN de manera eficiente y correcta. Las transformaciones de datos que realizamos son las siguientes:

#### 3.1 Dimensión única y modo RGB

Como primer paso en el preprocesamiento de las imágenes, realizamos la conversión de todas las imágenes al formato RGB. Esta transformación se llevó a cabo debido a que las imágenes originales presentaban diferencias en su formato (RGB, RGBA, P, L). Dado que decidimos utilizar un modelo pre-entrenado para la construcción de los clasificadores, es fundamental estandarizar el número de canales de todas las imágenes a 3 (RGB), ya que estos modelos han sido entrenados con imágenes en dicho formato. Esta homogeneización asegura la compatibilidad con la arquitectura seleccionada y evita errores durante la fase de entrenamiento.

Posteriormente, todas las imágenes fueron redimensionadas a un tamaño uniforme de 224x224 píxeles. Esta dimensión fue elegida debido a que es la requerida por varios modelos pre-entrenados utilizados comúnmente en tareas de clasificación de imágenes, incluidos los que utilizamos nosotros.

#### 3.2 Conversión a clases binarias (problema binario)

Para afrontar nuestro problema binario, transformamos el problema en una tarea de clasificación binaria, donde agrupamos las clases en “sano” (0) y “enfermo” (1). Para ello, utilizamos el método `where` de la librería numpy, asignando el valor 0 a las imágenes correspondientes a personas sanas (clase 2) y el valor 1 a aquellas con neumonía bacteriana o COVID-19 (clases 0 y 1).

### **3.3 División en datos de train, validation y test**

Dividimos el conjunto de datos en train (80%), y test (20%). Posteriormente, dividimos los datos de entrenamiento en train train (80%) y train validation (20%). Este proceso lo realizamos mediante la función ‘train\_test\_split()’ de la librería de scikit-learn. Además, utilizamos el parámetro stratify para realizar una partición estratificada, es decir, mantener la misma proporción de clases en ambos conjuntos. Esta decisión se fundamenta en la limitada cantidad de imágenes disponibles para cada clase. Sin la estratificación, existe la posibilidad de obtener conjuntos de datos con un número escaso de muestras para algunas clases.

### **3.4 Aleatorización**

En nuestro caso, es importante aleatorizar los datos ya que las imágenes vienen ordenadas por clase, por lo que el modelo podría captar patrones relacionados con este orden.

La aleatorización la realizamos a la vez que la división de datos especificando el parámetro shuffle = True en la función train\_test\_split().

### **3.5 Normalización**

Dado que en esta práctica decidimos utilizar un modelo pre-entrenado, en particular ResNet50 con pesos del modelo DeepCOVID-XR (clasificador binario) e ImageNet (clasificador multiclase), es necesario asegurar que las imágenes estén normalizadas de la misma forma que las imágenes con las que estos modelos fueron entrenados originalmente. En este sentido, es importante considerar que DeepCOVID-XR probablemente fue inicializado a partir de un modelo base pre-entrenado en ImageNet, por lo que mantener la misma estrategia de normalización es clave para asegurar la coherencia entre los datos de entrenamiento original y nuestros datos actuales.

Por esta razón, aplicamos la función de preprocessamiento específica de ResNet50 (preprocess\_input), la cual ajusta los valores de los píxeles para que tengan la misma escala y distribución que las imágenes de ImageNet. Este paso es fundamental para asegurar que las capas convolucionales preentrenadas puedan interpretar correctamente las imágenes nuevas y aprovechar al máximo los pesos previamente aprendidos.

En cuanto a la inclusión de capas de normalización adicionales, como BatchNormalization, no fue necesario agregarlas manualmente. ResNet50 ya incorpora capas de BatchNormalization en su estructura interna, lo que garantiza que las activaciones se mantengan estables durante el entrenamiento.

### **3.6 Data augmentation (problema multiclase)**

Para nuestro clasificador multiclase, implementamos la técnica de data augmentation con el objetivo de aumentar la cantidad de ejemplos de la clase “Neumonía Bacteriana”, que presenta un número significativamente inferior de instancias en comparación con las otras dos clases, como se ha mostrado en el EDA. Esta técnica tiene como objetivo equilibrar el conjunto de datos, lo que es crucial para mejorar el rendimiento del modelo, puesto que el desbalance de clases puede llevar a un sesgo hacia las clases mayoritarias.

El data augmentation solo se aplicará sobre los datos de entrenamiento (train), ya que nuestro objetivo es mejorar la capacidad del modelo para generalizar y evitar que aprenda patrones específicos de los datos de validación o test. La aplicación de estas transformaciones en los datos de validación y test podría afectar la evaluación real del modelo en datos no vistos.

El proceso de data augmentation ha incluido varias transformaciones sobre las imágenes de la clase "Neumonía Bacteriana", entre las cuales se encuentran:

- Rotación: Rotación de las imágenes entre  $-15^{\circ}$  y  $15^{\circ}$ .
- Desplazamiento horizontal y vertical: Traslaciones horizontales y verticales de hasta un 5%.
- Zoom: Aplicación de un zoom de  $\pm 10\%$  sobre las imágenes.
- Ajuste de brillo: Modificación del brillo de las imágenes en un rango de 0.8 a 1.2.
- Volteo horizontal: Volteo de las imágenes en dirección horizontal, cuando es aplicable.
- Relleno de píxeles: Relleno de los píxeles fuera de los bordes utilizando el método de "nearest" (más cercano).

Estas transformaciones han sido configuradas para generar un número fijo de imágenes aumentadas por cada imagen original, permitiendo crear una mayor diversidad en los datos de entrenamiento. En este caso, hemos generado 6 imágenes aumentadas por cada imagen original de la clase "Neumonía Bacteriana".

## 4. Clasificador binario

En esta sección se explicará la construcción, entrenamiento, experimentación, evaluación e interpretabilidad del clasificador binario.

### 4.1 Construcción del modelo

Para la construcción de nuestro clasificador binario, utilizamos la arquitectura ResNet50 como base, aprovechando los pesos pre-entrenados del modelo DeepCOVID-XR, que ha sido entrenado previamente con imágenes de radiografías torácicas para clasificar si corresponden a COVID-19 o no. Este enfoque es particularmente adecuado para nuestro problema, ya que el objetivo de nuestro clasificador binario es el mismo.

Una vez cargado el modelo y los pesos correspondientes, configuramos el parámetro “include\_top=False” para excluir la capa superior del modelo y poder incorporar una capa densa personalizada para la clasificación. Las capas convolucionales preentrenadas se mantienen congeladas (“base\_model.trainable = False”), lo que nos permite aprovechar los conocimientos adquiridos por el modelo sin necesidad de entrenar nuevamente esas capas. Para procesar la salida de las capas convolucionales, aplanamos la matriz resultante y agregamos una capa densa con un número variable de neuronas (parámetro) y la función de activación ReLU. Además, incluimos una capa de dropout para prevenir el sobreajuste durante el entrenamiento. Finalmente, la capa de salida consta de 1 neurona con una función de activación sigmoide, adecuada para un problema de clasificación binaria.

## 4.2 Experimentación

Realizamos varios experimentos con distintos modelos para la clasificación de las imágenes modificando algunos de los hiperparámetros como la tasa de aprendizaje, número de neuronas y tasa de dropout. De esta manera, entrenamos y evaluamos un total de 8 modelos con distintos valores de los hiperparámetros mencionados.

Hemos de mencionar que utilizamos el optimizador Adam y optimizamos la función de pérdida binary\_crossentropy. Esta función calcula la entropía cruzada entre las etiquetas reales y las predicciones del modelo, y es especialmente adecuada para problemas de clasificación binaria, donde la salida deseada es un valor binario (0 o 1) que indica la clase a la que pertenece cada muestra.

Es importante señalar que en todos estos experimentos empleamos un mismo número de épocas, 50. No obstante, utilizamos la técnica de "Early Stopping" con un umbral de paciencia establecido en 5 épocas. Además, configuramos el parámetro restore\_best\_weights = True, permitiendo que el modelo devuelto sea aquel que ha exhibido el menor error de validación durante el proceso de entrenamiento.

A partir de estos modelos construidos, seleccionaremos aquel que maximice el accuracy y el área bajo la curva ROC, y lo entrenaremos con los datos de train y validación conjuntamente para optimizar el rendimiento del modelo y garantizar una generalización adecuada. Al entrenar con ambos conjuntos de datos, el modelo tiene acceso a una variedad más amplia de patrones y características, lo que puede ayudar a mejorar su capacidad para reconocer y adaptarse a diferentes situaciones.

## 4.3 Resultados obtenidos

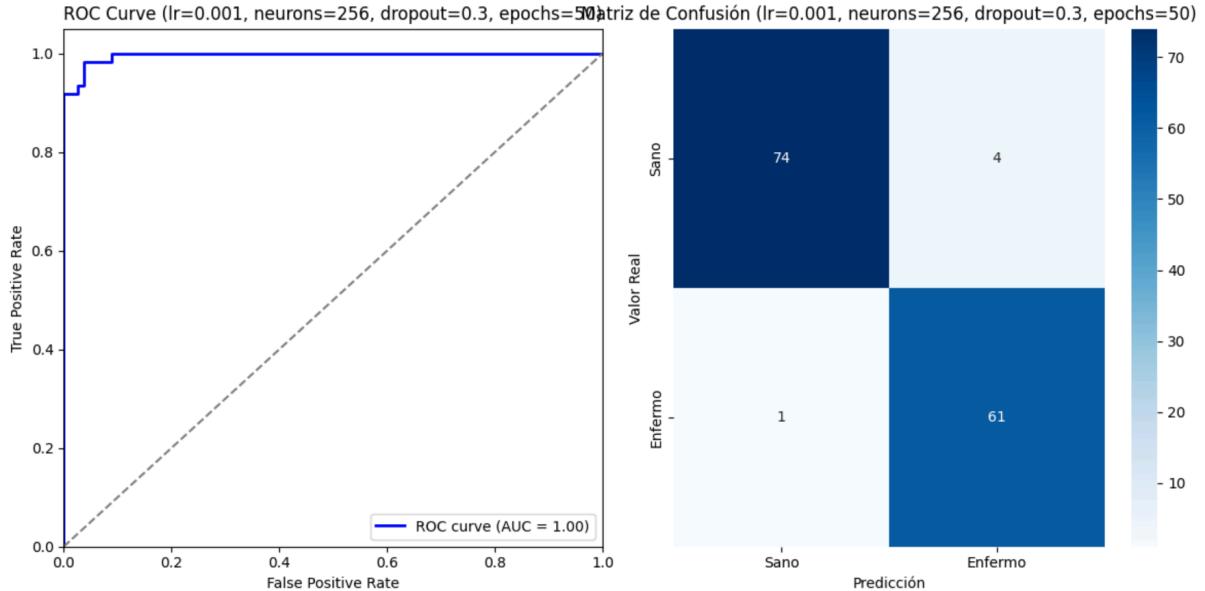
Los resultados de los distintos modelos construidos son los siguientes:

lr	neurons	dropout	epochs	val_acc	val_auc	sensitivity	specificity	f1_score
0.0001	128	0.3	50	0.9642857142857143	0.9958643507030605	0.967741935483871	0.9615384615384616	0.96
0.0001	128	0.5	50	0.9785714285714285	0.9956575682382134	0.9516129032258065	1.0	0.9752066115702479
0.0001	256	0.3	50	0.9785714285714285	0.9975186104218362	0.967741935483871	0.9871794871794872	0.975609756097561
0.0001	256	0.5	50	0.9714285714285714	0.9966914805624484	0.967741935483871	0.9743589743589743	0.967741935483871
0.001	128	0.3	50	0.9642857142857143	0.9729114971050454	0.9838709677419355	0.9487179487179487	0.9606299212598425
0.001	128	0.5	50	0.9642857142857143	0.9767369727047147	0.9193548387096774	1.0	0.957983193277311
0.001	256	0.3	50	0.9642857142857143	0.9962779156327544	0.9838709677419355	0.9487179487179487	0.9606299212598425
0.001	256	0.5	50	0.9857142857142858	0.9965880893300249	0.967741935483871	1.0	0.9836065573770492

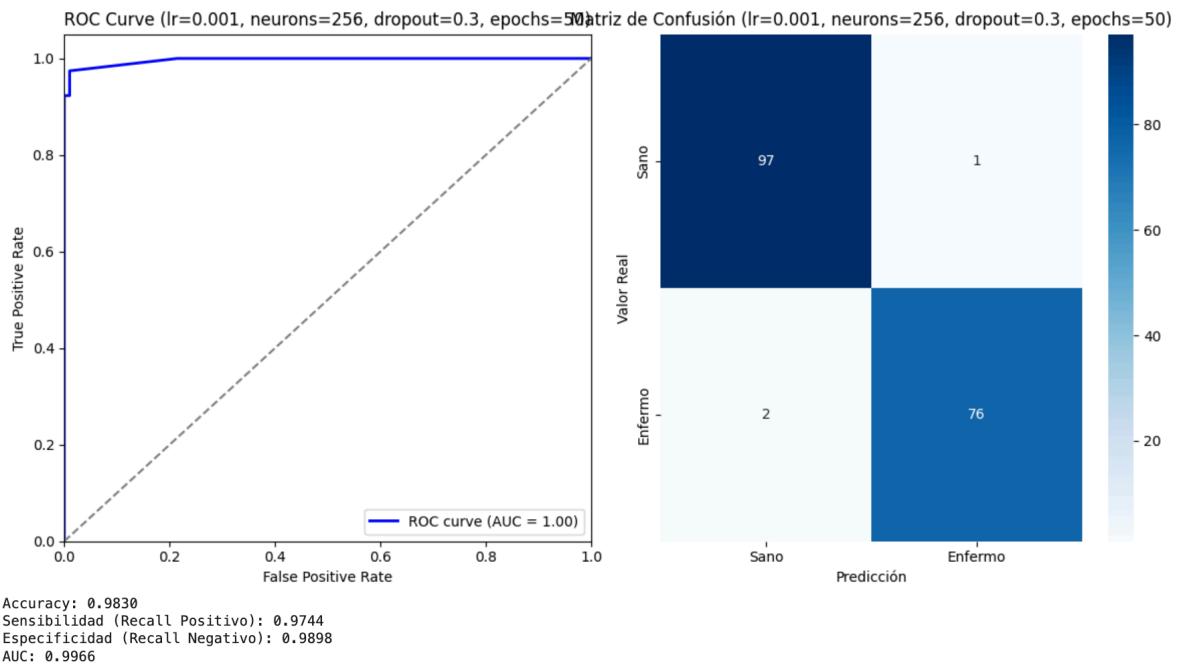
Escogemos el modelo que maximiza el accuracy de validación y el área bajo la curva ROC. No obstante, dado que varios modelos presentan un desempeño muy similar en estas métricas, priorizamos aquel con la mayor sensibilidad. Esta decisión se fundamenta en la naturaleza del problema, para el cual consideramos que es más importante detectar correctamente los casos positivos de COVID-19, ya que un falso negativo podría implicar el no aislamiento oportuno de un paciente infectado, aumentando así el riesgo de propagación y retrasando el inicio de un tratamiento adecuado. Por lo tanto, consideramos que en este contexto es preferible maximizar la sensibilidad, aún si esto implica sacrificar en cierta medida la especificidad. De esta manera, los hiperparámetros escogidos son los siguientes:

- $lr = 0.001$ , neuronas = 256, dropout = 0.3

La curva ROC y la matriz de confusión para este modelo son (conjunto de validación):



Entrenamos un nuevo modelo con estos hiperparámetros sobre los datos de train y validación conjuntamente. Evaluamos el modelo con los datos de test y los resultados obtenidos son los siguientes:



#### 4.4 Análisis de resultados

Observando los experimentos realizados y evaluados con el conjunto de validación, podemos afirmar que, generalmente, se obtienen muy buenos resultados para cualquier combinación de hiperparámetros, con un accuracy y AUC por encima de 0.95 en todos los experimentos. Esto hace evidente el éxito del transfer learning y del modelo

DeepCOVID-XR que ya nos proporciona unos pesos bastante adecuados para nuestro problema.

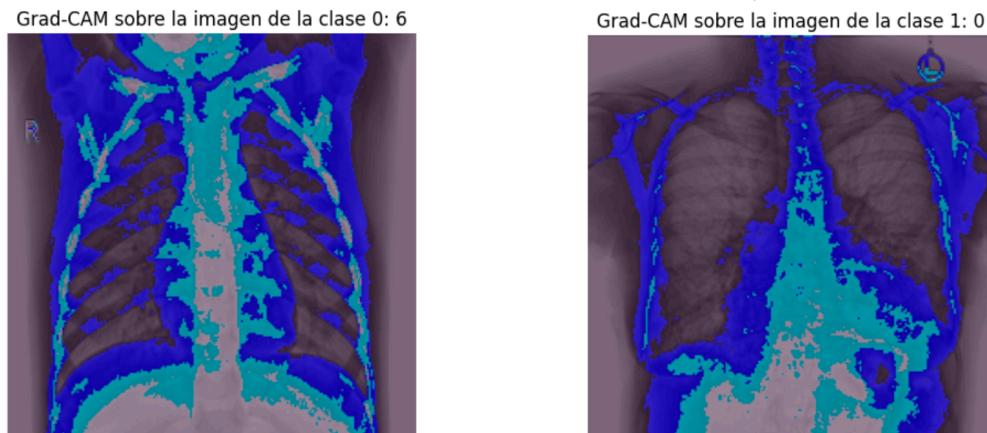
En relación con el modelo escogido ( $\text{lr} = 0.001$ , neuronas = 256, dropout = 0.3), los resultados obtenidos muestran un desempeño sobresaliente. La curva ROC alcanza un AUC de 0.9966, indicando una excelente capacidad de discriminación entre pacientes sanos y enfermos. El accuracy es del 98.30%, con una sensibilidad de 97.44% y una especificidad de 98.98%, lo que confirma su eficacia para detectar casos positivos sin comprometer significativamente la detección de casos negativos.

El análisis de la matriz de confusión evidencia una mínima cantidad de errores, con solo 1 falso positivo y 2 falsos negativos. Esto sugiere que el modelo logra un equilibrio adecuado entre la detección de casos positivos y negativos, alineándose con el objetivo de reducir el riesgo de falsos negativos.

#### 4.5 Interpretabilidad del modelo

Para analizar la interpretabilidad de nuestro modelo final, aplicamos la técnica Grad-CAM (Gradient-weighted Class Activation Mapping), que permite visualizar las regiones de la imagen en las que el modelo se enfocó para realizar su predicción.

Así, el mapa de activación obtenido para una imagen de cada clase es el siguiente:



Los mapas de activación obtenidos muestran que, al clasificar a un paciente como sano (clase 0), el modelo centra su atención en la región del tórax, las costillas y la columna vertebral. En cambio, al identificar a un paciente enfermo (clase 1, con neumonía bacteriana o COVID-19), el modelo se enfoca en ciertas áreas de los pulmones y la parte central e inferior del tórax.

Sería recomendable contrastar estos hallazgos con un profesional de la salud especializado en radiografías torácicas para validar si el modelo está basando sus decisiones en información médica relevante y adecuada.

### 5. Clasificador multiclas

En esta sección se explicará la construcción, experimentación, entrenamiento, evaluación e interpretabilidad del clasificador multiclas.

## **5.1 Construcción del modelo**

Después de realizar varias pruebas iniciales utilizando los pesos pre-entrenados del modelo DeepCOVID-XR, nos damos cuenta de que obtenemos resultados insatisfactorios para el clasificador multiclase. En particular, el modelo tiene dificultades para predecir la clase 0 (neumonía bacteriana), y rara vez la clasifica correctamente, tendiendo a predecir entre las clases 1 (COVID) y 2 (sano). Al analizar esta situación, hemos llegado a la conclusión de que esto puede deberse a que el modelo DeepCOVID-XR fue entrenado específicamente para distinguir entre COVID y sano, y no fue diseñado para reconocer neumonía bacteriana.

En consecuencia, hemos decidido continuar utilizando la arquitectura ResNet, pero con pesos pre-entrenados de "ImageNet", los cuales han sido entrenados sobre un conjunto de imágenes generales. Esta aproximación debería permitir que el modelo sea más flexible y capaz de generalizar mejor a las imágenes de radiografías, mejorando así su capacidad para clasificar correctamente las tres clases correspondientes a nuestro problema.

De esta manera, al igual que para el clasificador binario, configuramos el parámetro "include\_top=False" para excluir la capa superior del modelo y mantenemos congeladas las capas convolucionales mediante "base\_model.trainable = False". Posteriormente, aplanamos la matriz resultante y agregamos una capa densa con un número variable de neuronas (ajustable mediante un parámetro) y la función de activación ReLU. Además, incorporamos una capa de dropout para prevenir el sobreajuste durante el entrenamiento. No obstante, a diferencia del clasificador binario, la capa de salida consta de 3 neuronas, cada una con una función de activación softmax, que nos permitirá obtener la probabilidad de pertenecer a cada una de las tres clases posibles.

## **5.2 Experimentación**

De la misma manera que para el clasificador binario, realizamos varios experimentos con diferentes modelos para la clasificación de imágenes, ajustando hiperparámetros como la tasa de aprendizaje, el número de neuronas y la tasa de dropout. También utilizamos el optimizador Adam.

Sin embargo, en contraste con el problema binario, en este caso usamos la función de pérdida sparse\_categorical\_crossentropy, que es adecuada para problemas multiclase sin necesidad de codificación One-Hot Encoding (OHE) de la variable de salida.

En todos los experimentos, mantenemos constante el número de épocas en 50 y aplicamos la técnica de "Early Stopping" con un umbral de paciencia de 5 épocas. Además, configuramos el parámetro restore\_best\_weights = True, tal y como hemos hecho con el clasificador binario.

A partir de estos modelos construidos, seleccionamos aquel que maximice el accuracy, pero también nos fijamos en las métricas adicionales de sensibilidad, especificidad y F1-score.

### 5.3 Resultados obtenidos

Los resultados de los distintos modelos construidos son los siguientes:

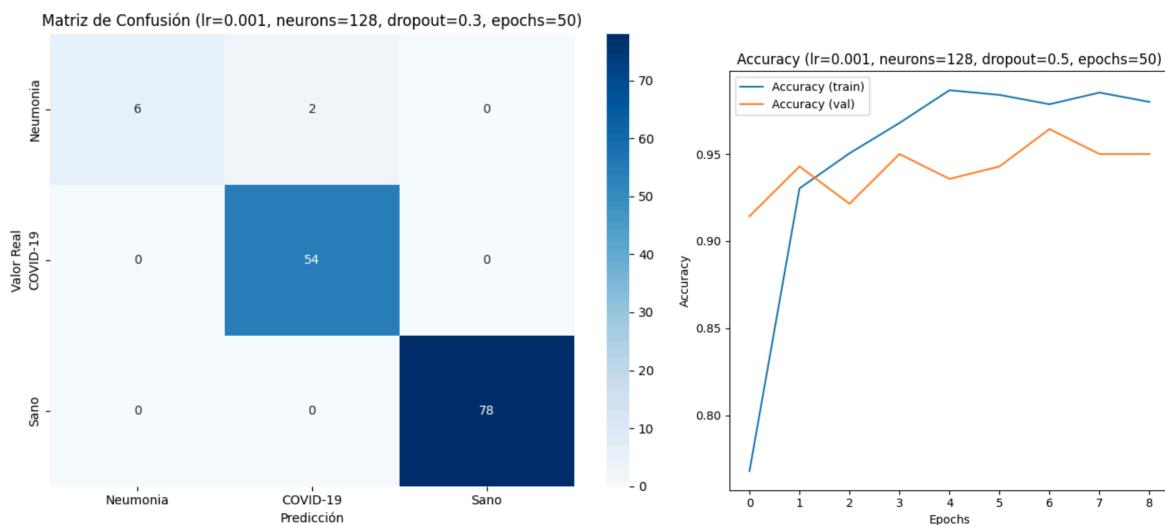
experimentos\_multiclasa\_resultados

lr	neurons	dropout	epochs	val_acc	sensitivity	specificity	f1_scores
0,0001	128	0.3	50	0,957	[0.5, 0.981, 0.987]	[0.992, 0.942, 1.0]	[0.615, 0.946, 0.994]
0,0001	128	0.5	50	0,964	[0.625, 0.963, 1.0]	[0.985, 0.965, 1.0]	[0.667, 0.954, 1.0]
0,0001	256	0.3	50	0,964	[0.625, 0.981, 0.987]	[0.992, 0.953, 1.0]	[0.714, 0.955, 0.994]
0,0001	256	0.5	50	0,971	[0.625, 0.981, 1.0]	[0.992, 0.965, 1.0]	[0.714, 0.964, 1.0]
0,001	128	0.3	50	0,986	[0.75, 1.0, 1.0]	[1.0, 0.977, 1.0]	[0.857, 0.982, 1.0]
0,001	128	0.5	50	0,964	[0.75, 0.963, 0.987]	[0.985, 0.965, 1.0]	[0.75, 0.954, 0.994]
0,001	256	0.3	50	0,964	[0.5, 0.981, 1.0]	[0.992, 0.953, 1.0]	[0.615, 0.955, 1.0]
0,001	256	0.5	50	0,957	[0.5, 0.963, 1.0]	[1.0, 0.953, 0.968]	[0.667, 0.945, 0.987]

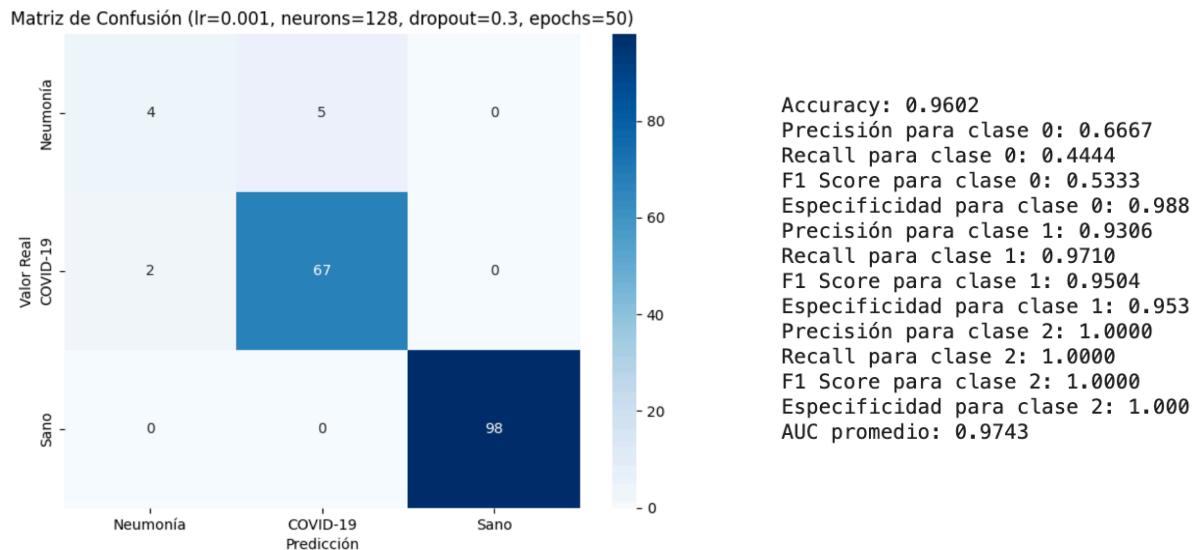
El modelo que seleccionamos es aquel que maximiza el accuracy de validación, así como las métricas adicionales de sensibilidad, especificidad y F1-score. Al observar la tabla de resultados obtenidos, podemos ver que un modelo sobresale por su rendimiento superior en todas estas métricas en comparación con los demás. Este modelo se corresponde con la siguiente configuración de hiperparámetros:

- lr = 0.001, neuronas = 128, dropout = 0.3

La matriz de confusión y la evolución del accuracy de validación para este modelo son:



Entrenamos un nuevo modelo con estos hiperparámetros sobre los datos de train y validación conjuntamente. Evaluamos el modelo con los datos de test y los resultados obtenidos son los siguientes:



#### 5.4 Análisis de resultados

Tras realizar múltiples experimentos con nuestro modelo de clasificación multiclase de imágenes torácicas, hemos observado un patrón consistente en los resultados: la sensibilidad para la clase 0 (Neumonía bacteriana) es considerablemente baja. Esto puede deberse a la escasez de imágenes disponibles para esta clase. Aunque aplicamos data augmentation para intentar mitigar este problema, el modelo no ha sido capaz de generalizar correctamente las transformaciones aplicadas, ya que las radiografías nunca se toman rotadas o con modificaciones geométricas significativas en la práctica clínica.

En cuanto al mejor modelo obtenido, presenta un accuracy global del 96%, lo que indica que en términos generales realiza predicciones correctas en la mayoría de los casos. Sin embargo, al analizar métricas más detalladas por clase, encontramos discrepancias significativas en el rendimiento, especialmente para la clase “Neumonía bacteriana”, como nos ha ocurrido durante la evaluación del modelo con el conjunto de validación:

- Para la clase 0 (Neumonía bacteriana):

- Sensibilidad: 0.44 (baja capacidad de detectar correctamente casos de neumonía)
- Especificidad: 0.99 (el modelo rara vez confunde otras clases con neumonía)
- F1-score: 0.53 (bajo equilibrio entre precisión y sensibilidad)

El modelo tiende a cometer muchos falsos negativos en esta clase, lo que implica que varios casos de neumonía bacteriana no son detectados correctamente. Esto sugiere, como hemos mencionado anteriormente, que el data augmentation aplicado no ha funcionado bien para mejorar la generalización del modelo en esta categoría.

- Para la clase 1 (COVID-19):

- Sensibilidad: 0.97 (alta capacidad de detección de casos)
- Especificidad: 0.95 (baja tasa de falsos positivos)
- F1-score: 0.95

La detección de COVID-19 es altamente efectiva, con una combinación equilibrada de precisión y sensibilidad.

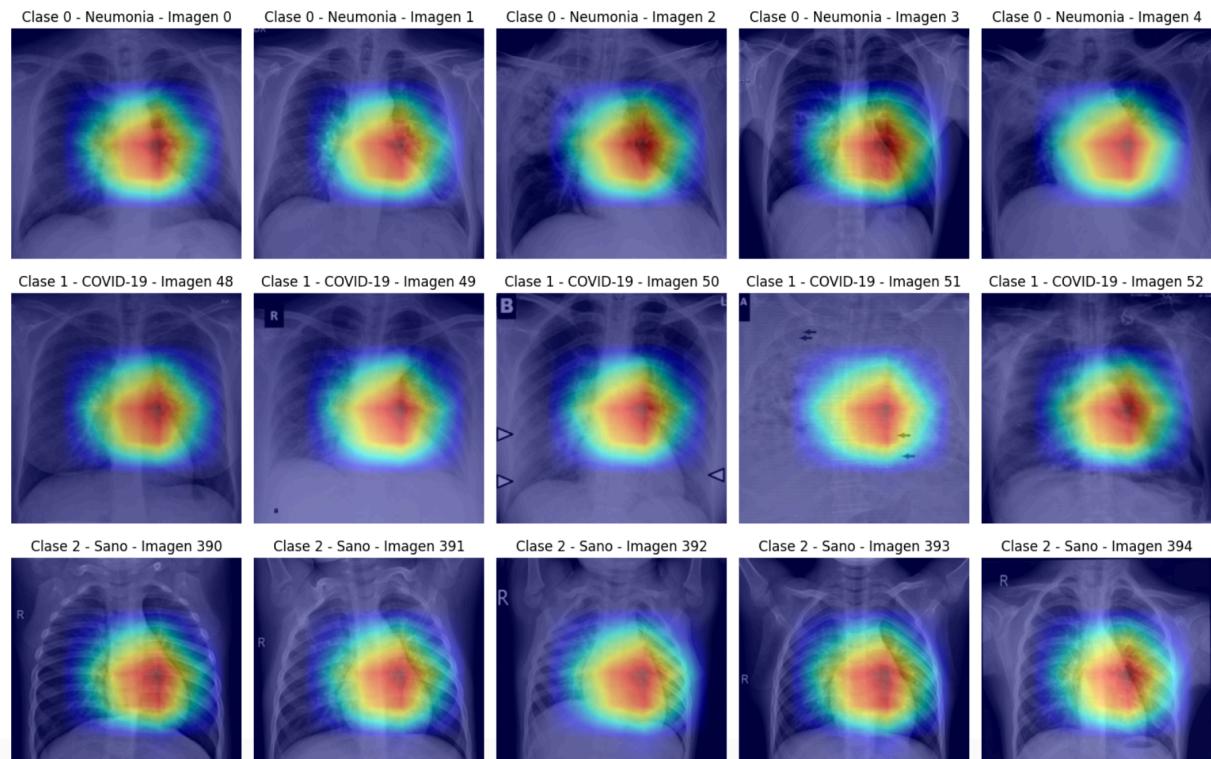
- Para la clase 2 (Sano):

- Sensibilidad: 1.0 (todos los casos sanos son detectados correctamente)
- Especificidad: 1.0 (ningún caso es erróneamente clasificado como sano)
- F1-score: 1.0

El modelo clasifica correctamente a los pacientes sanos sin margen de error, lo que indica un alto desempeño en esta clase.

## 5.5 Interpretabilidad del modelo

Para evaluar la interpretabilidad del modelo de clasificación de imágenes de tórax, hemos utilizado la técnica Class Activation Mapping (CAM) con el objetivo de visualizar las regiones donde la CNN enfoca su atención al realizar una predicción. Este enfoque permite verificar la coherencia de las decisiones del modelo con el conocimiento médico y detectar posibles sesgos o errores en la clasificación. Los mapas de activación generados para algunas imágenes de cada clase son los siguientes:



Analizando las activaciones, se puede afirmar que el modelo se centra principalmente en la región pulmonar. Sin embargo, a simple vista, las activaciones para las tres categorías muestran patrones muy similares, lo que dificulta extraer conclusiones definitivas sobre qué características específicas está utilizando el modelo para diferenciar entre neumonía bacteriana, COVID-19 y pulmones sanos. Dado que las diferencias entre las activaciones no son claramente distinguibles, sería recomendable consultar a un experto en radiología para evaluar si los patrones resaltados por el modelo realmente corresponden con hallazgos clínicos significativos.

## 6. Utilidad clínica

En cuanto al clasificador binario, con un accuracy de 0.98 y una sensibilidad de 0.97, podemos afirmar que podría ser de gran utilidad clínica. Esto, por supuesto, asumiendo que las imágenes utilizadas en el entrenamiento y evaluación son representativas de las que se obtienen en la práctica médica diaria y que están correctamente etiquetadas. Si las condiciones en las que se aplicara el modelo en un entorno real son similares a las del conjunto de datos utilizado, su utilidad sería significativa.

Si bien la decisión final debe recaer siempre en un profesional de la salud, este clasificador puede servir como una herramienta de apoyo en el diagnóstico, ayudando a los médicos a tomar decisiones más informadas o incluso permitiendo que los pacientes que se realicen una radiografía tengan una evaluación preliminar que les ayude a tomar medidas oportunas.

Tanto los médicos como los pacientes deben ser conscientes de la fiabilidad del modelo en cada caso específico. Por ejemplo, si el sistema clasifica a un paciente como enfermo (ya sea con COVID-19 o neumonía bacteriana), es fundamental conocer la probabilidad de que esa clasificación sea correcta, garantizando así una interpretación adecuada de los resultados.

No obstante, como se ha mencionado anteriormente, para que este modelo pueda ser de utilidad clínica, deberíamos contrastar estos resultados, especialmente la interpretación del modelo y mapas de activación, con un especialista en radiografías torácicas.

Por otro lado, respecto al clasificador multiclase, no podríamos darle una utilidad clínica debido a su baja sensibilidad (0.44) a la hora de detectar neumonía bacteriana.

## 7. Conclusiones

En conclusión, se han desarrollado y entrenado dos modelos de clasificación utilizando imágenes médicas de radiografías torácicas. El primer modelo es un clasificador binario, cuyo rendimiento ha sido destacado con un accuracy de 98.30%, una sensibilidad de 97.44% y un F1-score de 98.0%. Este modelo, basado en la arquitectura ResNet50 y los pesos preentrenados de DeepCOVID-XR, ha demostrado ser extremadamente eficaz en la detección de COVID-19 y neumonía bacteriana, lo que resalta su potencial como herramienta complementaria en el diagnóstico clínico. El uso de transfer learning ha sido clave en la optimización del modelo, alcanzando métricas sólidas con un esfuerzo de entrenamiento reducido. Además, la correcta optimización de los hiperparámetros ha

permitido priorizar la detección de casos positivos, minimizando así el riesgo de falsos negativos.

A pesar del buen desempeño, hemos identificado ciertas limitaciones, especialmente en cuanto a la interpretabilidad del modelo. Los mapas de activación sugieren que el modelo se enfoca en las regiones pulmonares. No obstante, se deben validar estos resultados con especialistas en radiología, así como evaluar el desempeño en entornos hospitalarios con datos externos, para garantizar la fiabilidad del modelo antes de su implementación en la práctica clínica.

Por otro lado, el clasificador multiclase, basado también en ResNet50, ha logrado un accuracy global de 96%. Sin embargo, el modelo muestra limitaciones en la detección de la clase neumonía bacteriana, con una baja sensibilidad de 0.44, lo que resulta en una alta tasa de falsos negativos. Este desafío podría estar relacionado con la escasez de datos para esta clase y la dificultad del modelo para generalizar a partir del data augmentation, cuyo enfoque en transformaciones geométricas no refleja completamente las variaciones de las radiografías reales. En contraste, el modelo ha demostrado un excelente rendimiento en la detección de COVID-19 y en la identificación de sujetos sanos, con una sensibilidad de 0.97 y 1.0, respectivamente.

El análisis de los mapas de activación del modelo multiclase también muestra que la red se enfoca correctamente en la región pulmonar, lo que es clínicamente relevante. Sin embargo, la similitud en las activaciones entre las clases dificulta la interpretación de los criterios utilizados por el modelo para diferenciar entre ellas. Este aspecto resalta la necesidad de validar los hallazgos con especialistas en radiología.

En resumen, aunque el clasificador multiclase presenta un buen desempeño general, su aplicación clínica aún enfrenta retos, especialmente en la detección de neumonía bacteriana. Para mejorar su rendimiento, se sugiere la recopilación de más datos específicos para esta clase, la implementación de otras técnicas de balanceo de clases y la exploración de enfoques avanzados de data augmentation.

Por último, cabe mencionar que hemos tenido especiales dificultades a la hora de ejecutar el código de esta práctica ya que la GPU no siempre estaba disponible en Google Collab. Además, en algunas ocasiones, la RAM llegaba a su límite y se generaba un error "ResourceExhaustedError: Graph execution error". Estos inconvenientes afectaron la ejecución de nuestros experimentos. También queríamos comentar que hemos dedicado un gran esfuerzo a la práctica, dado que 2 de los 3 integrantes del grupo no tenían apenas conocimientos sobre Inteligencia Artificial, y menos sobre CNNs.