

CLASIFICACIÓN DE IMÁGENES DE COVID-19

FINAL PROJECT

Marta Cuevas Rodríguez

Universidad de Málaga | 2025

Herramientas y algoritmos en Bioinformática

OVERVIEW

- INTRODUCTION
- OBJETIVOS
- CONFIGURACIÓN INICIAL
- MODELO SIMPLE
- DROPOUT Y EARLYSTOPPING
- ADAPTACIÓN DE MODELO
- MODELO PREENTRENADO
- MODELO DE DOS CLASES
- CONCLUSION

INTRODUCTION

La pandemia de COVID-19 ha afectado gravemente los sistemas de salud a nivel mundial. En este contexto, las radiografías de tórax se han convertido en una herramienta clave para la detección rápida de la enfermedad, ya que permiten identificar signos de infecciones respiratorias, como opacidades pulmonares. Este trabajo se centra en desarrollar un modelo de red neuronal convolucional (CNN) para clasificar imágenes de radiografías en cuatro categorías: COVID-19, NORMAL, PNEUMONIA y Lung Opacity, con el objetivo de analizar el rendimiento del modelo aplicando técnicas de preprocesamiento, aumento de datos y optimización de hiperparámetros.

OBJETIVO

Desarrollar y evaluar un modelo basado en redes neuronales convolucionales (CNN) para clasificar radiografías de tórax en las categorías COVID-19, NORMAL, PNEUMONIA y Lung

OBJETIVOS ESPECÍFICOS

- Preprocesar el conjunto de datos COVID-19 Radiography Dataset.
- Diseñar y entrenar una CNN para la clasificación de imágenes.
- Implementar técnicas para mejorar la generalización.
- Ajustar hiperparámetros y aplicar estrategias como Early Stopping para evitar el sobreajuste.

CONFIGURACIÓN INICIAL

- **Lenguaje:** Python con TensorFlow/Keras.
- **Hardware:** Aceleración con GPU (CUDA) (con errores de inicialización).
- **Dataset:** COVID-19 Radiography Dataset (4 clases: COVID, NORMAL, PNEUMONIA, Lung_Opacity).
- **Preprocesamiento:** Redimensionamiento a 150x150 y normalización de imágenes.
- **División:** 80% entrenamiento, 20% validación.
- **Modelo:** Red neuronal convolucional simple.

MODELO CNN SIMPLE CON CUATRO CLASES



MODELO

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(len(CLASSES), activation='softmax') # Una salida por clase
])
```

- 3 bloques de capas Conv2D + MaxPooling2D (ReLU).
- Capa densa con 128 neuronas (ReLU) y Dropout 50%.
- Capa de salida con softmax (4 clases).
- Imagen redimensionada a 150x150 y normalizada.
- Entrenado con tamaño de lote 32, función de pérdida categorical crossentropy, optimizador Adam, durante 10 épocas.

MODELO CNN SIMPLE CON CUATRO CLASES

RESULTADOS

Clase	Precision	Recall	F1-Score	Support
<i>COVID</i>	0.17	0.17	0.17	1446
<i>NORMAL</i>	0.29	0.29	0.29	2404
<i>PNEUMONIA</i>	0.49	0.49	0.49	4076
<i>Lung Opacity</i>	0.06	0.07	0.07	538
Accuracy			0.35	8464
Macro Avg	0.26	0.26	0.26	8464
Weighted Avg	0.36	0.35	0.35	8464

MODELO CNN CON DROPOUT Y EARLY STOPPING



MODELO

```
model = models.Sequential([
    # Bloque 1
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE[0], IMG_SIZE[1], 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.2),

    # Bloque 2
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.2),

    # Clasificador
    layers.Flatten(),
    layers.Dense(64, activation='relu'), # Reducimos el tamaño de la capa densa
    layers.Dropout(0.3),
    layers.Dense(len(CLASSES), activation='softmax') # Capa de salida
])
```

- **Dropout**: Añadido un 20% de Dropout después de cada bloque convolucional y un 30% antes de la capa de salida.
- **Early Stopping**: Configurado un callback para detener el entrenamiento si la pérdida de validación no mejoraba durante 3 épocas consecutivas.
- Se redujo el número de neuronas en la capa densa final a 64.

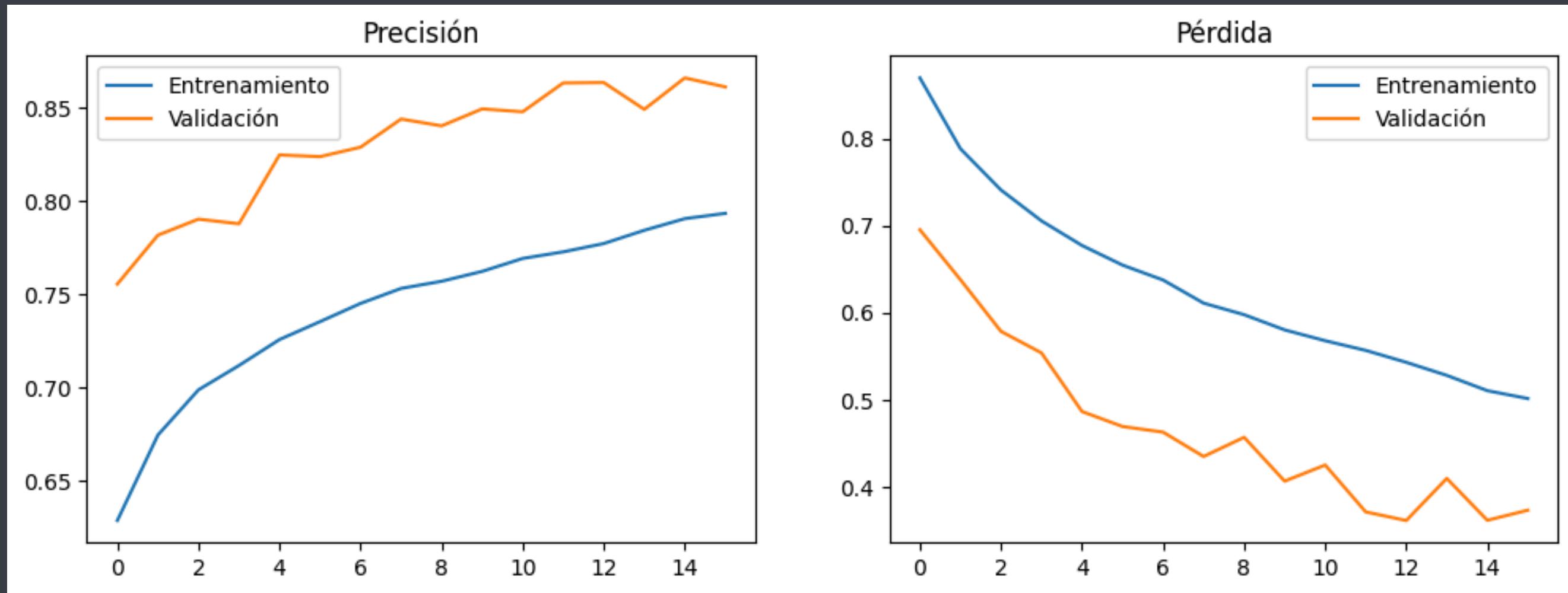
MODELO CNN CON DROPOUT Y EARLY STOPPING

RESULTADOS

Clase	Precisión	Recall	F1-Score	Soporte
<i>COVID</i>	0.19	0.19	0.19	1446
<i>NORMAL</i>	0.28	0.25	0.26	2404
<i>PNEUMONIA</i>	0.48	0.52	0.50	4076
<i>Lung Opacity</i>	0.06	0.05	0.05	538
Exactitud			0.36	8464
Promedio Macro	0.25	0.25	0.25	8464
Promedio Ponderado	0.35	0.36	0.35	8464

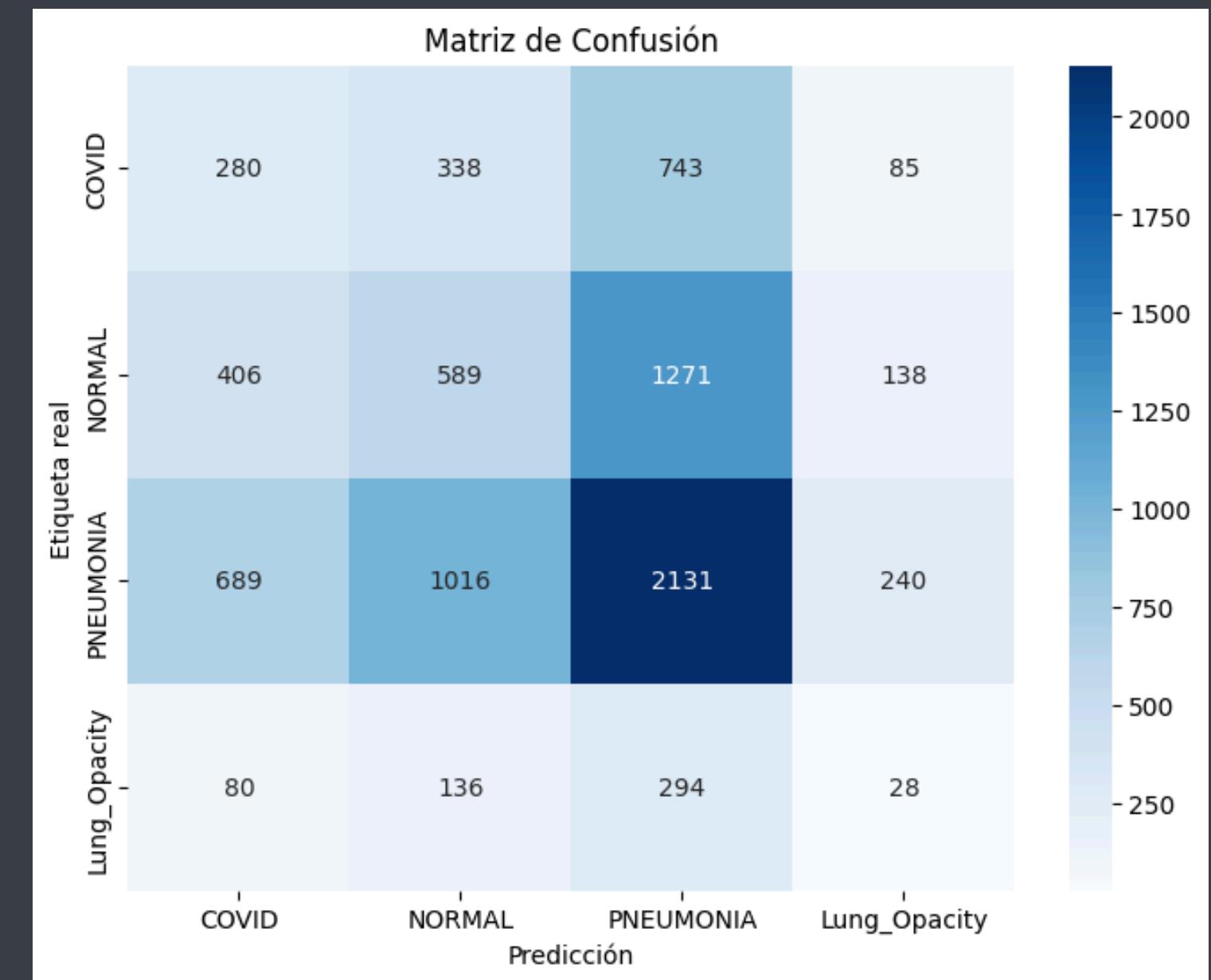
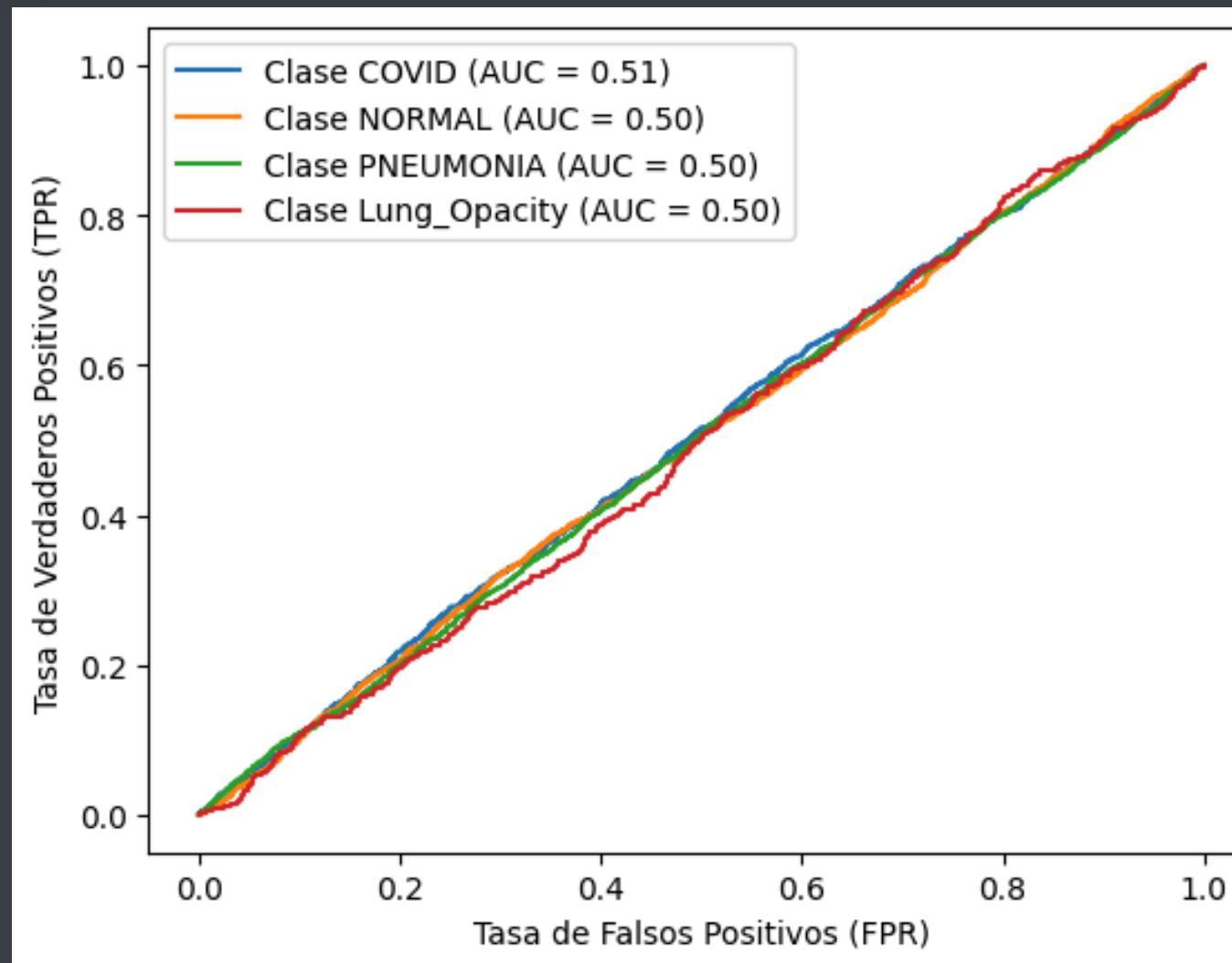
MODELO CNN CON DROPOUT Y EARLY STOPPING

RESULTADOS



MODELO CNN CON DROPOUT Y EARLY STOPPING

RESULTADOS



ADAPTACIÓN DEL MODELO

○ MODELO

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(image_size[0], image_size[1], 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(categories), activation='softmax')
])
```

- **Tres capas convolucionales** con 32, 64 y 128 filtros, seguidas de Max Pooling.
- **Dropout** del 50% después de la capa densa para reducir sobreajuste.
- **Funciones de activación:** ReLU en capas internas y Softmax en salida.
- Optimización con Adam y función de pérdida Categorical Crossentropy.

ADAPTACIÓN DEL MODELO



RESULTADOS

```
Found 33866 images belonging to 4 classes.  
Found 8464 images belonging to 4 classes.  
Epoch 1/20  
530/530 ————— 599s 1s/step - accuracy: 0.5172 - loss: 1.1066 - val_accuracy: 0.6590 - val_loss: 0.8270  
Epoch 2/20  
21/530 ————— 8:25 994ms/step - accuracy: 0.6312 - loss: 0.9021
```

MODELO CON MOBILENETV2 PREENTRENADO



MODELO

```
# Cargar el modelo MobileNetV2 preentrenado sin las capas superiores
base_model = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=False, input_shape=(image_size[0], image_size[1], 3))

# Congelar las capas de la base
base_model.trainable = True
for layer in base_model.layers[:-50]: # Congela la mayoría, pero descongela las últimas 50 capas
    layer.trainable = False

# Construir el modelo
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(categories), activation='softmax')
])
```

- **MobileNetV2** preentrenado con pesos de ImageNet como base para extracción de características.
- **Congelación parcial:** Últimas 50 capas desbloqueadas para ajustar características al dataset específico.
- Añadida **GlobalAveragePooling2D**, capa densa con ReLU y Dropout (50%), finalizando con Softmax para clasificación multiclase.

MODELO CON MOBILENETV2 PREENTRENADO

RESULTADOS

```
Found 33866 images belonging to 4 classes.  
Found 8464 images belonging to 4 classes.  
Epoch 1/20  
1059/1059 ————— 573s 531ms/step - accuracy: 0.6817 - loss: 0.8502 - val_accuracy: 0.7413 - val_loss: 1.3015  
Epoch 2/20  
368/1059 ————— 5:35 486ms/step - accuracy: 0.7654 - loss: 0.6234
```

MODELO REDUCIDO A DOS CLASES

● MODELO

```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(image_size[0], image_size[1], 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(len(categories), activation='softmax')
])
```

- **Reducción de clases:** Solo COVID y NORMAL para simplificar a clasificación binaria.
- **Red neuronal estándar:** Tres capas convolucionales, max pooling y capa densa con softmax.
- **Early Stopping:** Monitoreo de precisión en validación para evitar sobreajuste.
- **Sin Dropout:** Se eliminó Dropout para permitir un aprendizaje sin restricciones adicionales.

MODELO REDUCIDO A DOS CLASES

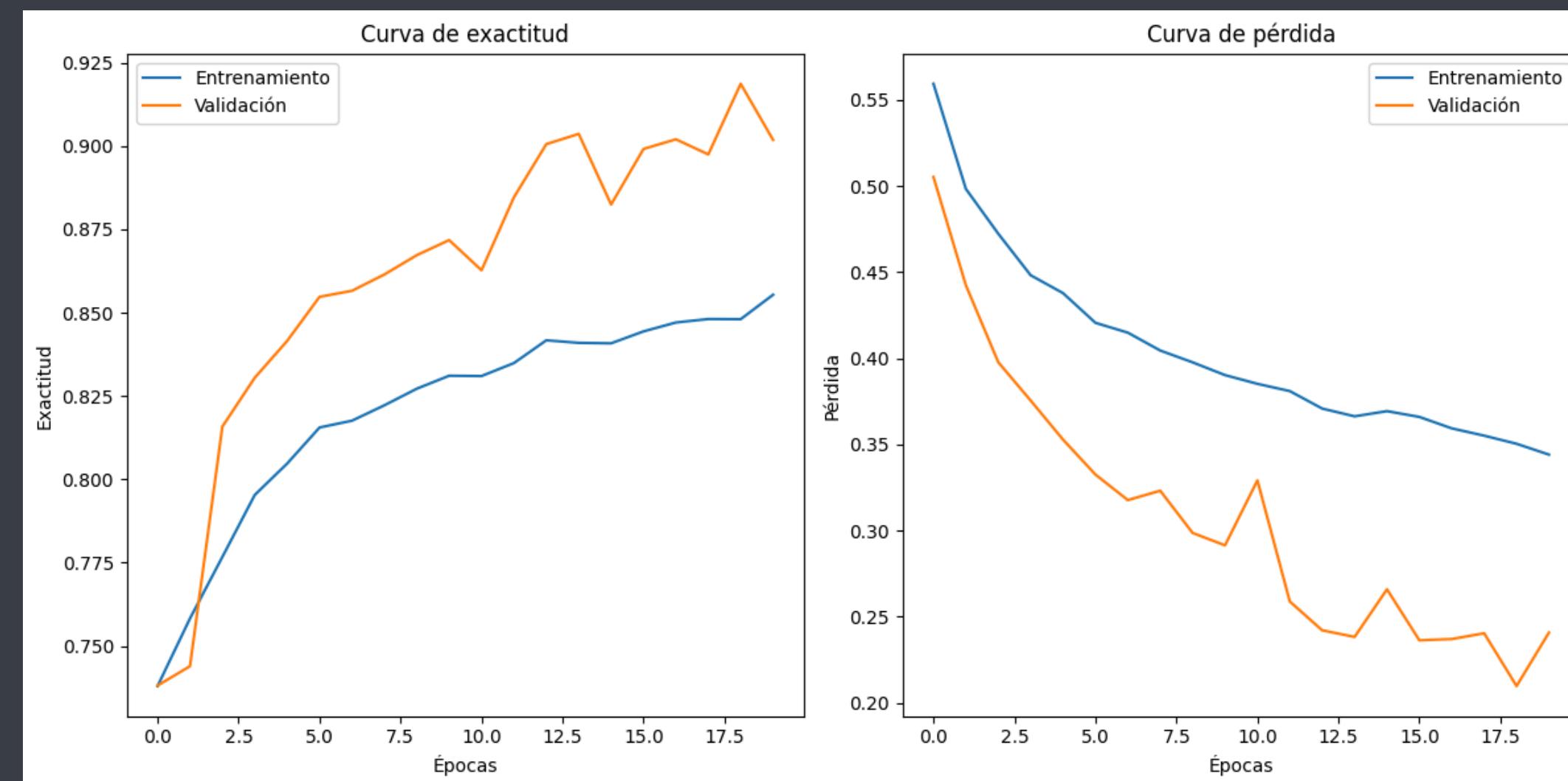


RESULTADOS

Clase	Precisión	Recall	F1-score	Support
COVID	0.27	0.23	0.25	1446
NORMAL	0.74	0.77	0.76	4076
Exactitud				
Macro avg	0.50	0.50	0.50	5522
Weighted avg	0.62	0.63	0.62	5522

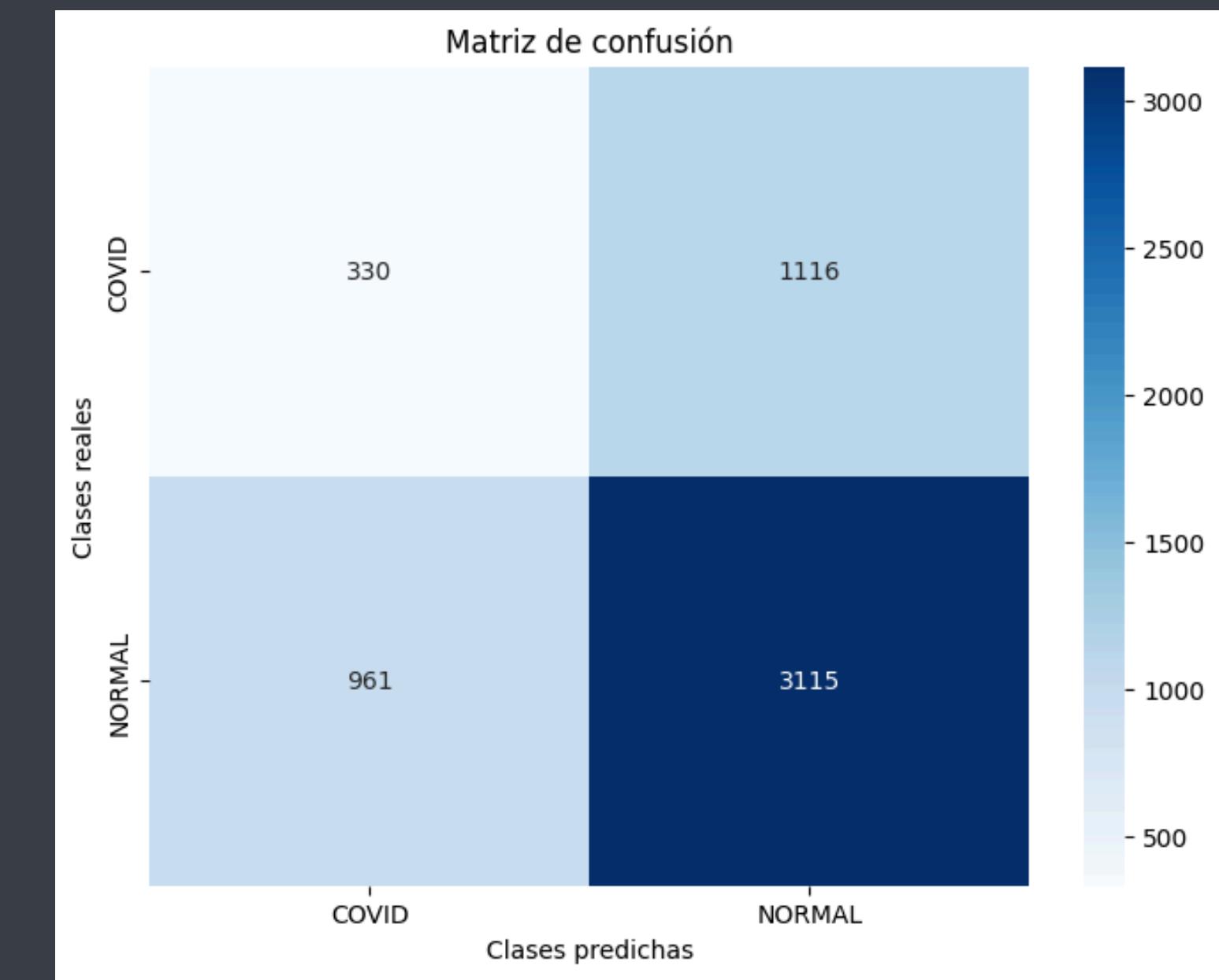
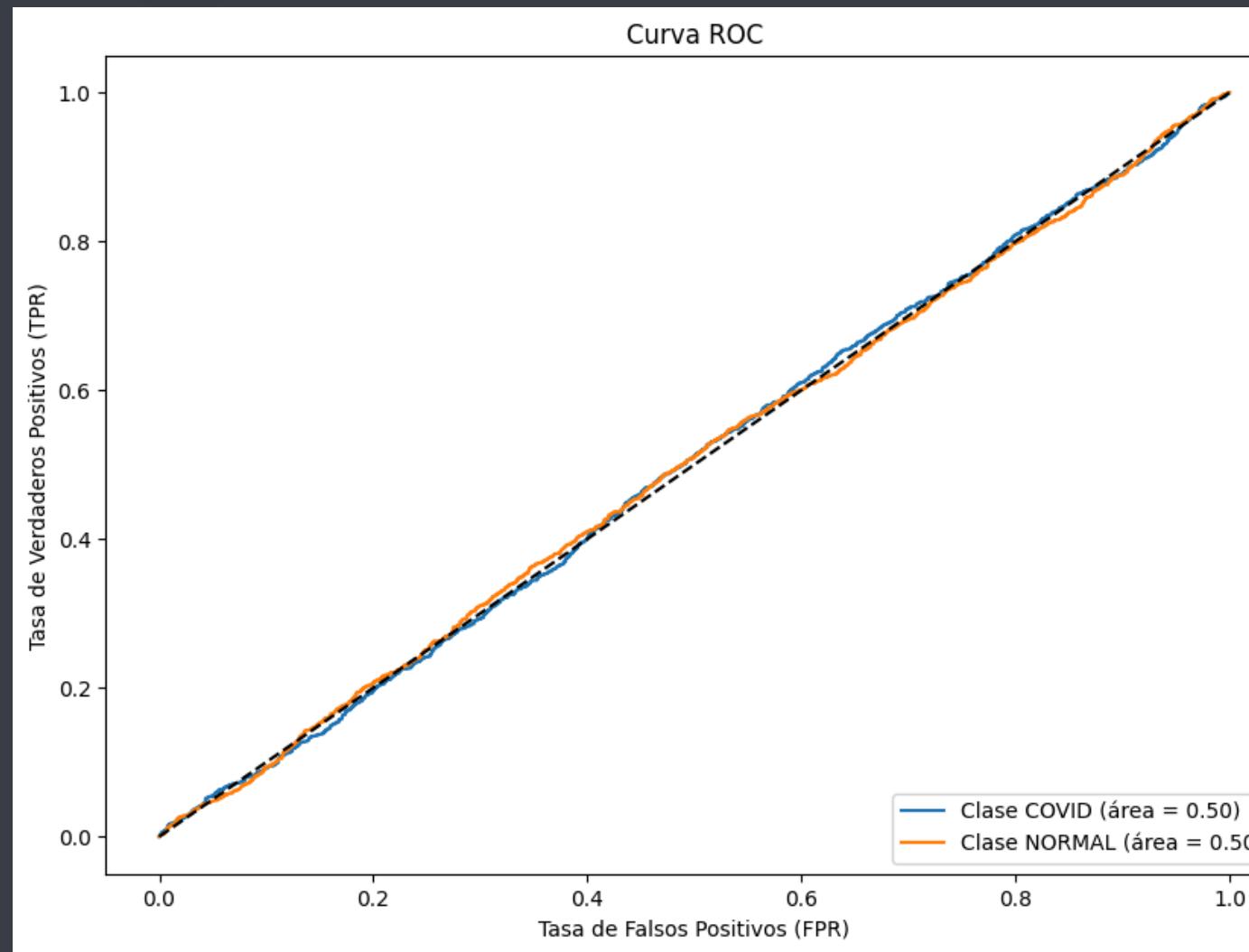
MODELO REDUCIDO A DOS CLASES

RESULTADOS



MODELO REDUCIDO A DOS CLASES

RESULTADOS



CONCLUSIÓN

A pesar de las mejoras implementadas, como la reducción del problema a dos clases y el uso de técnicas de regularización como el Dropout, el rendimiento del modelo sigue siendo subóptimo, especialmente en la detección de la clase COVID. El desbalance entre las clases COVID y NORMAL sigue siendo un desafío importante, afectando la capacidad del modelo para aprender de manera efectiva. Aunque el uso del modelo preentrenado MobileNetV2 no mostró mejoras sustanciales, el conjunto de datos complejo y el desbalance de clases probablemente contribuyen a este resultado. Futuras mejoras podrían incluir el uso de técnicas avanzadas de balanceo de clases, métricas como AUC-ROC para evaluar mejor la clase minoritaria y la implementación de redes más profundas para capturar características complejas.

THANK YOU

FINAL PROYECT

Marta Cuevas Rodríguez

Universidad de Málaga | 2025

Herramientas y algoritmos en Bioinformática