

Caso de Estudio 2:

Curso: Deep Learning

Desarrollado por:

Juan Camilo Sanmiguel Jonathan Alejandro Rubiano

1. Descripción del Problema

En este nuevo caso de estudio, se exploraría el uso de Transfer Learning, aprovechando modelos preentrenados para mejorar la clasificación en las mismas categorías:

El objetivo principal es:

El objetivo es comparar el desempeño de un modelo entrenado desde cero con el de un modelo preentrenado ajustado a este conjunto de datos, optimizando el tiempo de entrenamiento y la precisión de la clasificación.

Trabajaremos con el recurso COVID-19 Radiography Database, reconocido con el *COVID-19 Dataset Award* por la comunidad de Kaggle. Este conjunto de datos fue recopilado por investigadores de la Universidad de Qatar (Qatar), la Universidad de Dhaka (Bangladesh) y colaboradores de Pakistán y Malasia, junto con médicos. Contiene miles de radiografías clasificadas como:

- COVID-19, casos confirmados.
- Normal, pacientes sin hallazgos patológicos.
- Viral Pneumonia, neumonía viral distinta de COVID-19.
- Lung Opacity, opacidades pulmonares no relacionadas con COVID-19.

Preprocesamiento de los datos

Se descargaron los datos desde el repositorio en Kaggle

Se aplico CLAHE como una mejora de contraste activo, eligiendo parámetros CLIP = 2 y GRID = 8, se realizó el ajuste de tamaño para poder usarse dentro de un modelo ya definido a un resize (224×224), padding para evitar pérdidas de información en el cambio de tamaño, normalización y conversión a RGB.

```
In [ ]: import kagglehub

# Download Latest version
path = kagglehub.dataset_download("tawsifurrahman/covid19-radiography-database")

print("Path to dataset files:", path)

Downloading from https://www.kaggle.com/api/v1/datasets/download/tawsifurrahman/covid19-radiography-database?dataset_version_number=5...
100%|██████████| 778M/778M [00:13<00:00, 59.4MB/s]
Extracting files...
Path to dataset files: C:\Users\Camilo\.cache\kagglehub\datasets\tawsifurrahman\covid19-radiography-database\versions\5
```

```
In [ ]: # =====
# 💡 ENTRENAMIENTO DE CLASIFICADOR MULTICLASE CON EFFICIENTNETB0
# =====

import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.applications.efficientnet import preprocess_input
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.metrics import classification_report, confusion_matrix
```

```
# =====
# 📁 CONFIGURACIÓN DE RUTAS
# =====
# Ruta principal que contiene las 4 carpetas de categorías
base_dir = r"C:\Users\Camilo\.cache\kagglehub\datasets\tawsifurrahman\covid19-radiography-database\versions\5\COVID-19_Radiogr
IMG_SIZE = 224
TEST_SIZE = 0.2
BATCH_SIZE = 16
EPOCHS = 50
LEARNING_RATE = 1e-4

# CLAHE params
CLAHE_CLIP = 2.0
CLAHE_GRID = (8, 8)

# =====
# FUNCIONES AUXILIARES
# =====
def apply_clahe_and_resize_gray(path, img_size=IMG_SIZE, clip=CLAHE_CLIP, grid=CLAHE_GRID):
    """
    Lee imagen en grayscale, aplica CLAHE, redimensiona manteniendo aspecto con padding,
    devuelve imagen uint8 en escala 0-255 (single-channel).
    """
    img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        raise FileNotFoundError(f"No se pudo leer: {path}")

    # Aplicar CLAHE
    clahe = cv2.createCLAHE(clipLimit=clip, tileGridSize=grid)
    enhanced = clahe.apply(img)

    # Resize manteniendo aspecto y padding centrado
    h, w = enhanced.shape
    scale = img_size / max(h, w)
    new_w, new_h = int(w * scale), int(h * scale)
    resized = cv2.resize(enhanced, (new_w, new_h), interpolation=cv2.INTER_AREA)

    padded = np.zeros((img_size, img_size), dtype=np.uint8)
    x_offset = (img_size - new_w) // 2
```

```
y_offset = (img_size - new_h) // 2
padded[y_offset:y_offset + new_h, x_offset:x_offset + new_w] = resized

return padded # uint8 (0-255), single channel

def gray_to_rgb_stack(gray_img):
    """
    Convierte imagen (H,W) uint8 a (H,W,3) replicando el canal.
    """
    if gray_img.ndim == 2:
        return np.stack((gray_img,)*3, axis=-1)
    elif gray_img.ndim == 3 and gray_img.shape[2] == 1:
        return np.concatenate([gray_img]*3, axis=-1)
    else:
        return gray_img # ya RGB
```

Se realizo la unión de los datos con sus respectivas categorías para poder ser usados dentro del modelo de Deep learning.

```
In [ ]: # =====#
# CONSTRUIR DATAFRAME con rutas e etiquetas (categorías)
# =====#
data = []
for category in sorted(os.listdir(base_dir)):
    category_path = os.path.join(base_dir, category)
    images_dir = os.path.join(category_path, "images")
    if not os.path.isdir(images_dir):
        print(f"⚠️ No se encontró 'imagenes' en {category_path}, se omite")
        continue
    for fn in sorted(os.listdir(images_dir)):
        if fn.lower().endswith((".jpg", ".jpeg", ".png", ".bmp", ".tif", ".tiff")):
            data.append({"ruta_imagen": os.path.join(images_dir, fn), "categoria": category})

df = pd.DataFrame(data)
if df.empty:
    raise SystemExit("No se encontraron imágenes. Revisa base_dir y estructura de carpetas.")
print(f"✅ Dataset creado: {len(df)} imágenes, {df['categoria'].nunique()} clases")
print(df['categoria'].value_counts())
```

⚠️ No se encontró 'imagenes' en C:\Users\Camilo\.cache\kagglehub\datasets\tawsifurrahman\covid19-radiography-database\versions\5\COVID-19_Radiography_Dataset\COVID.metadata.xlsx, se omite
⚠️ No se encontró 'imagenes' en C:\Users\Camilo\.cache\kagglehub\datasets\tawsifurrahman\covid19-radiography-database\versions\5\COVID-19_Radiography_Dataset\Lung_Opacity.metadata.xlsx, se omite
⚠️ No se encontró 'imagenes' en C:\Users\Camilo\.cache\kagglehub\datasets\tawsifurrahman\covid19-radiography-database\versions\5\COVID-19_Radiography_Dataset\Normal.metadata.xlsx, se omite
⚠️ No se encontró 'imagenes' en C:\Users\Camilo\.cache\kagglehub\datasets\tawsifurrahman\covid19-radiography-database\versions\5\COVID-19_Radiography_Dataset\README.md.txt, se omite
⚠️ No se encontró 'imagenes' en C:\Users\Camilo\.cache\kagglehub\datasets\tawsifurrahman\covid19-radiography-database\versions\5\COVID-19_Radiography_Dataset\Viral_Pneumonia.metadata.xlsx, se omite
✅ Dataset creado: 21165 imágenes, 4 clases
categoria

Normal	10192
Lung_Opacity	6012
COVID	3616
Viral Pneumonia	1345

Name: count, dtype: int64

```
In [ ]: # =====
# MAPEO DE CLASES Y CARGA + PREPROCESAMIENTO (CLAHE + STACK + preprocess_input)
# =====
class_names = sorted(df["categoria"].unique())
class_to_idx = {name: idx for idx, name in enumerate(class_names)}
print("Clases:", class_to_idx)

X_list, y_list = [], []
for idx, row in df.iterrows():
    p = row["ruta_imagen"]
    label = class_to_idx[row["categoria"]]
    try:
        gray = apply_clahe_and_resize_gray(p, img_size=IMG_SIZE)
        rgb = gray_to_rgb_stack(gray) # -> H,W,3 uint8
        X_list.append(rgb)
        y_list.append(label)
    except Exception as e:
        print(f"⚠️ Error cargando {p}: {e}")

X = np.array(X_list, dtype=np.uint8) # shape (N, H, W, 3) uint8
y = np.array(y_list, dtype=np.int32)

print(f"✅ Imágenes cargadas en memoria: {X.shape}, etiquetas: {y.shape}")
```

```
Clases: {'COVID': 0, 'Lung_Opacity': 1, 'Normal': 2, 'Viral Pneumonia': 3}
✅ Imágenes cargadas en memoria: (21165, 224, 224, 3), etiquetas: (21165,)
```

Normalización: Las imágenes se convirtieron a tipo float32 y se normalizaron utilizando la función preprocess_input() propia de EfficientNet, la cual ajusta los valores de los píxeles al rango [-1,1]. Esta normalización es coherente con el proceso de entrenamiento original de EfficientNet y permite mantener una escala numérica estable durante la propagación hacia adelante y hacia atrás, evitando saturación en las activaciones y mejorando la convergencia del optimizador.

Codificación de etiquetas: Las clases fueron transformadas mediante one-hot encoding usando to_categorical(), de modo que el modelo pudiera tratar el problema como una clasificación multiclas.

```
In [ ]: # =====
# NORMALIZACIÓN: usar preprocess_input (EfficientNet espera [-1,1])
# =====
# Convertir a float32 primero
```

```
X = X.astype(np.float32)
X = preprocess_input(X) # esto ajusta según EfficientNet (usa escala -1..1)
y_cat = to_categorical(y, num_classes=len(class_names))
```

División del conjunto de datos: El dataset se dividió en subconjuntos de entrenamiento y prueba utilizando una proporción del 80–20 %, garantizando una distribución balanceada de clases mediante la opción stratify. Esto asegura que todas las categorías estén representadas tanto en el entrenamiento como en la evaluación.

Adicional se realizó un aumento de datos por medio ImageDataGenerator donde se utilizó:

- Rotación de 15°
- Movimiento en ejes horizontal y vertical de 10%
- Una variación de aumento de 10%
- Una inversión de la mitad de las imágenes creadas en el eje horizontal
- Y relleno de las imágenes donde queden espacios negros utilizando los pixeles mas cercanos

Estas operaciones simulan variaciones reales en la posición y orientación de los objetos dentro de las imágenes, permitiendo que la red aprenda representaciones más invariantes a la rotación, traslación y escala.

```
In [ ]: # =====#
# SPLIT TRAIN/TEST
# ======
X_train, X_test, y_train, y_test = train_test_split(
    X, y_cat, test_size=TEST_SIZE, random_state=42, stratify=y_cat
)
print(f"Split: train={X_train.shape[0]}, test={X_test.shape[0]}")

# =====#
# DATA AUGMENTATION
# ======
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
```

```
)  
datagen.fit(X_train)
```

```
Split: train=16932, test=4233
```

Estructura de la CNN y Explicación de las capas empleadas

Se empleó la arquitectura EfficientNetB0 como base del modelo, debido a su equilibrio entre precisión y eficiencia computacional. EfficientNet utiliza un método de escalado compuesto que ajusta simultáneamente la profundidad, el ancho y la resolución de entrada de la red de manera balanceada. Esto le permite alcanzar un rendimiento competitivo con un número significativamente menor de parámetros frente a otras arquitecturas más pesadas como ResNet o VGG.

Para este trabajo se seleccionó la versión B0, que es la más ligera de la familia, priorizando un tiempo de entrenamiento reducido y un menor consumo de recursos, sin sacrificar demasiado la capacidad de generalización.

El modelo base fue cargado con los pesos preentrenados en ImageNet, los cuales capturan características visuales genéricas útiles (bordes, texturas, formas), y posteriormente se eliminó la parte superior (include_top=False) para adaptar la red a la nueva tarea de clasificación.

A continuación, se añadieron capas personalizadas para el ajuste fino:

- Flatten(): transforma los mapas de características bidimensionales obtenidos por la EfficientNet en un vector unidimensional que puede ser procesado por capas densas.
- Dense(256, activation='relu'): capa totalmente conectada con 256 neuronas que aprende combinaciones no lineales de las características extraídas por la base convolucional.
- Dropout(0.3): desactiva aleatoriamente el 30 % de las neuronas durante el entrenamiento, con el fin de reducir el sobreajuste y mejorar la capacidad de generalización del modelo.
- Dense(num_classes, activation='softmax'): capa de salida que calcula la probabilidad de pertenencia de cada imagen a una de las clases del conjunto de datos.

Durante la primera fase de entrenamiento, las capas convolucionales del modelo base se mantuvieron congeladas para conservar las representaciones visuales generales aprendidas en ImageNet. Esta estrategia reduce el riesgo de sobreajuste, especialmente considerando que, aunque el conjunto de datos COVID-19 Radiography Database contiene alrededor de 21 000 imágenes, su variabilidad es limitada debido

a que todas corresponden a radiografías de tórax con estructuras anatómicas similares.

```
In [ ]: # =====
# LIMPIAR SESSION PREVIA (evita cargar pesos antiguos inconsistentes)
# =====
tf.keras.backend.clear_session()

# =====
# MODELO: EfficientNetB0 (INPUT 3 CANALES) - clave: input_shape=(224,224,3)
# =====

# Instanciar sin pesos primero
base_model = EfficientNetB0(weights=None, include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))

# Descargar y cargar pesos manualmente
weights_url = "https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5"
weights_path = tf.keras.utils.get_file("efficientnetb0_notop.h5", origin=weights_url, cache_subdir="models")
base_model.load_weights(weights_path)

# Congelar base (igual que antes)
for layer in base_model.layers:
    layer.trainable = False

# Agregar top personalizado (igual que antes)
x = Flatten()(base_model.output)
x = Dense(256, activation="relu")(x)
x = Dropout(0.3)(x)
output = Dense(len(class_names), activation="softmax")(x)

model = Model(inputs=base_model.input, outputs=output)

# Compilar (igual que antes)
model.compile(optimizer=Adam(learning_rate=LEARNING_RATE),
              loss="categorical_crossentropy",
              metrics=["accuracy"])

print("\nResumen del modelo (asegúrate INPUT 3 canales):")
model.summary()
```

```
WARNING:tensorflow:From c:\Users\Camilo\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\backend\common\global_state.py:82: The name tf.reset_default_graph is deprecated. Please use tf.compat.v1.reset_default_graph instead.
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb0_notop.h5  
16705208/16705208 1s 0us/step
```

Resumen del modelo (asegúrate INPUT 3 canales):

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
rescaling (Rescaling)	(None, 224, 224, 3)	0	input_layer[0][0]
normalization (Normalization)	(None, 224, 224, 3)	7	rescaling[0][0]
stem_conv_pad (ZeroPadding2D)	(None, 225, 225, 3)	0	normalization[0]...
stem_conv (Conv2D)	(None, 112, 112, 32)	864	stem_conv_pad[0]...
stem_bn (BatchNormalizatio...)	(None, 112, 112, 32)	128	stem_conv[0][0]
stem_activation (Activation)	(None, 112, 112, 32)	0	stem_bn[0][0]
block1a_dwconv (DepthwiseConv2D)	(None, 112, 112, 32)	288	stem_activation[...]
block1a_bn (BatchNormalizatio...)	(None, 112, 112, 32)	128	block1a_dwconv[0...]
block1a_activation (Activation)	(None, 112, 112, 32)	0	block1a_bn[0][0]
block1a_se_squeeze (GlobalAveragePool...)	(None, 32)	0	block1a_activati...
block1a_se_reshape (Reshape)	(None, 1, 1, 32)	0	block1a_se_squee...
block1a_se_reduce (Conv2D)	(None, 1, 1, 8)	264	block1a_se_resha...
block1a_se_expand	(None, 1, 1, 32)	288	block1a_se_reduc...

(Conv2D)			
block1a_se_excite (Multiply)	(None, 112, 112, 32)	0	block1a_activati... block1a_se_expan...
block1a_project_co... (Conv2D)	(None, 112, 112, 16)	512	block1a_se_excit...
block1a_project_bn (BatchNormalizatio...)	(None, 112, 112, 16)	64	block1a_project_...
block2a_expand_conv (Conv2D)	(None, 112, 112, 96)	1,536	block1a_project_...
block2a_expand_bn (BatchNormalizatio...)	(None, 112, 112, 96)	384	block2a_expand_c...
block2a_expand_act... (Activation)	(None, 112, 112, 96)	0	block2a_expand_b...
block2a_dwconv_pad (ZeroPadding2D)	(None, 113, 113, 96)	0	block2a_expand_a...
block2a_dwconv (DepthwiseConv2D)	(None, 56, 56, 96)	864	block2a_dwconv_p...
block2a_bn (BatchNormalizatio...)	(None, 56, 56, 96)	384	block2a_dwconv[0]...
block2a_activation (Activation)	(None, 56, 56, 96)	0	block2a_bn[0][0]
block2a_se_squeeze (GlobalAveragePool...)	(None, 96)	0	block2a_activati...
block2a_se_reshape (Reshape)	(None, 1, 1, 96)	0	block2a_se_squee...
block2a_se_reduce (Conv2D)	(None, 1, 1, 4)	388	block2a_se_resha...
block2a_se_expand (Conv2D)	(None, 1, 1, 96)	480	block2a_se_reduc...

block2a_se_excite (Multiply)	(None, 56, 56, 96)	0	block2a_activati... block2a_se_expan...
block2a_project_co... (Conv2D)	(None, 56, 56, 24)	2,304	block2a_se_excit...
block2a_project_bn (BatchNormalizatio...)	(None, 56, 56, 24)	96	block2a_project_...
block2b_expand_conv (Conv2D)	(None, 56, 56, 144)	3,456	block2a_project_...
block2b_expand_bn (BatchNormalizatio...)	(None, 56, 56, 144)	576	block2b_expand_c...
block2b_expand_act... (Activation)	(None, 56, 56, 144)	0	block2b_expand_b...
block2b_dwconv (DepthwiseConv2D)	(None, 56, 56, 144)	1,296	block2b_expand_a...
block2b_bn (BatchNormalizatio...)	(None, 56, 56, 144)	576	block2b_dwconv[0]...
block2b_activation (Activation)	(None, 56, 56, 144)	0	block2b_bn[0][0]
block2b_se_squeeze (GlobalAveragePool...)	(None, 144)	0	block2b_activati...
block2b_se_reshape (Reshape)	(None, 1, 1, 144)	0	block2b_se_squee...
block2b_se_reduce (Conv2D)	(None, 1, 1, 6)	870	block2b_se_resha...
block2b_se_expand (Conv2D)	(None, 1, 1, 144)	1,008	block2b_se_reduc...
block2b_se_excite (Multiply)	(None, 56, 56, 144)	0	block2b_activati... block2b_se_expan...

block2b_project_co... (Conv2D)	(None, 56, 56, 24)	3,456	block2b_se_excit...
block2b_project_bn (BatchNormalizatio...)	(None, 56, 56, 24)	96	block2b_project_...
block2b_drop (Dropout)	(None, 56, 56, 24)	0	block2b_project_...
block2b_add (Add)	(None, 56, 56, 24)	0	block2b_drop[0][...] block2a_project_...
block3a_expand_conv (Conv2D)	(None, 56, 56, 144)	3,456	block2b_add[0][0]
block3a_expand_bn (BatchNormalizatio...)	(None, 56, 56, 144)	576	block3a_expand_c...
block3a_expand_act... (Activation)	(None, 56, 56, 144)	0	block3a_expand_b...
block3a_dwconv_pad (ZeroPadding2D)	(None, 59, 59, 144)	0	block3a_expand_a...
block3a_dwconv (DepthwiseConv2D)	(None, 28, 28, 144)	3,600	block3a_dwconv_p...
block3a_bn (BatchNormalizatio...)	(None, 28, 28, 144)	576	block3a_dwconv[0...]
block3a_activation (Activation)	(None, 28, 28, 144)	0	block3a_bn[0][0]
block3a_se_squeeze (GlobalAveragePool...)	(None, 144)	0	block3a_activati...
block3a_se_reshape (Reshape)	(None, 1, 1, 144)	0	block3a_se_squee...
block3a_se_reduce (Conv2D)	(None, 1, 1, 6)	870	block3a_se_resha...
block3a_se_expand	(None, 1, 1, 144)	1,008	block3a_se_reduc...

(Conv2D)			
block3a_se_excite (Multiply)	(None, 28, 28, 144)	0	block3a_activati... block3a_se_expan...
block3a_project_co... (Conv2D)	(None, 28, 28, 40)	5,760	block3a_se_excit...
block3a_project_bn (BatchNormalizatio...)	(None, 28, 28, 40)	160	block3a_project_...
block3b_expand_conv (Conv2D)	(None, 28, 28, 240)	9,600	block3a_project_...
block3b_expand_bn (BatchNormalizatio...)	(None, 28, 28, 240)	960	block3b_expand_c...
block3b_expand_act... (Activation)	(None, 28, 28, 240)	0	block3b_expand_b...
block3b_dwconv (DepthwiseConv2D)	(None, 28, 28, 240)	6,000	block3b_expand_a...
block3b_bn (BatchNormalizatio...)	(None, 28, 28, 240)	960	block3b_dwconv[0]...
block3b_activation (Activation)	(None, 28, 28, 240)	0	block3b_bn[0][0]
block3b_se_squeeze (GlobalAveragePool...)	(None, 240)	0	block3b_activati...
block3b_se_reshape (Reshape)	(None, 1, 1, 240)	0	block3b_se_squee...
block3b_se_reduce (Conv2D)	(None, 1, 1, 10)	2,410	block3b_se_resha...
block3b_se_expand (Conv2D)	(None, 1, 1, 240)	2,640	block3b_se_reduc...
block3b_se_excite (Multiply)	(None, 28, 28, 240)	0	block3b_activati... block3b_se_expan...

block3b_project_co... (Conv2D)	(None, 28, 28, 40)	9,600	block3b_se_excit...
block3b_project_bn (BatchNormalizatio...)	(None, 28, 28, 40)	160	block3b_project_...
block3b_drop (Dropout)	(None, 28, 28, 40)	0	block3b_project_...
block3b_add (Add)	(None, 28, 28, 40)	0	block3b_drop[0][...] block3a_project_...
block4a_expand_conv (Conv2D)	(None, 28, 28, 240)	9,600	block3b_add[0][0]
block4a_expand_bn (BatchNormalizatio...)	(None, 28, 28, 240)	960	block4a_expand_c...
block4a_expand_act... (Activation)	(None, 28, 28, 240)	0	block4a_expand_b...
block4a_dwconv_pad (ZeroPadding2D)	(None, 29, 29, 240)	0	block4a_expand_a...
block4a_dwconv (DepthwiseConv2D)	(None, 14, 14, 240)	2,160	block4a_dwconv_p...
block4a_bn (BatchNormalizatio...)	(None, 14, 14, 240)	960	block4a_dwconv[0]...
block4a_activation (Activation)	(None, 14, 14, 240)	0	block4a_bn[0][0]
block4a_se_squeeze (GlobalAveragePool...)	(None, 240)	0	block4a_activati...
block4a_se_reshape (Reshape)	(None, 1, 1, 240)	0	block4a_se_squee...
block4a_se_reduce (Conv2D)	(None, 1, 1, 10)	2,410	block4a_se_resha...

block4a_se_expand (Conv2D)	(None, 1, 1, 240)	2,640	block4a_se_reduc...
block4a_se_excite (Multiply)	(None, 14, 14, 240)	0	block4a_activati... block4a_se_expan...
block4a_project_co... (Conv2D)	(None, 14, 14, 80)	19,200	block4a_se_excit...
block4a_project_bn (BatchNormalizatio...)	(None, 14, 14, 80)	320	block4a_project_...
block4b_expand_conv (Conv2D)	(None, 14, 14, 480)	38,400	block4a_project_...
block4b_expand_bn (BatchNormalizatio...)	(None, 14, 14, 480)	1,920	block4b_expand_c...
block4b_expand_act... (Activation)	(None, 14, 14, 480)	0	block4b_expand_b...
block4b_dwconv (DepthwiseConv2D)	(None, 14, 14, 480)	4,320	block4b_expand_a...
block4b_bn (BatchNormalizatio...)	(None, 14, 14, 480)	1,920	block4b_dwconv[0]...
block4b_activation (Activation)	(None, 14, 14, 480)	0	block4b_bn[0][0]
block4b_se_squeeze (GlobalAveragePool...)	(None, 480)	0	block4b_activati...
block4b_se_reshape (Reshape)	(None, 1, 1, 480)	0	block4b_se_squeee...
block4b_se_reduce (Conv2D)	(None, 1, 1, 20)	9,620	block4b_se_resha...
block4b_se_expand (Conv2D)	(None, 1, 1, 480)	10,080	block4b_se_reduc...
block4b_se_excite	(None, 14, 14,	0	block4b_activati...

(Multiply)	480)		block4b_se_expan...
block4b_project_co... (Conv2D)	(None, 14, 14, 80)	38,400	block4b_se_excit...
block4b_project_bn (BatchNormalizatio...)	(None, 14, 14, 80)	320	block4b_project_...
block4b_drop (Dropout)	(None, 14, 14, 80)	0	block4b_project_...
block4b_add (Add)	(None, 14, 14, 80)	0	block4b_drop[0][...] block4a_project_...
block4c_expand_conv (Conv2D)	(None, 14, 14, 480)	38,400	block4b_add[0][0]
block4c_expand_bn (BatchNormalizatio...)	(None, 14, 14, 480)	1,920	block4c_expand_c...
block4c_expand_act... (Activation)	(None, 14, 14, 480)	0	block4c_expand_b...
block4c_dwconv (DepthwiseConv2D)	(None, 14, 14, 480)	4,320	block4c_expand_a...
block4c_bn (BatchNormalizatio...)	(None, 14, 14, 480)	1,920	block4c_dwconv[0...]
block4c_activation (Activation)	(None, 14, 14, 480)	0	block4c_bn[0][0]
block4c_se_squeeze (GlobalAveragePool...)	(None, 480)	0	block4c_activati...
block4c_se_reshape (Reshape)	(None, 1, 1, 480)	0	block4c_se_squee...
block4c_se_reduce (Conv2D)	(None, 1, 1, 20)	9,620	block4c_se_resha...
block4c_se_expand (Conv2D)	(None, 1, 1, 480)	10,080	block4c_se_reduc...

block4c_se_excite (Multiply)	(None, 14, 14, 480)	0	block4c_activati... block4c_se_expan...
block4c_project_co... (Conv2D)	(None, 14, 14, 80)	38,400	block4c_se_excit...
block4c_project_bn (BatchNormalizatio...)	(None, 14, 14, 80)	320	block4c_project_...
block4c_drop (Dropout)	(None, 14, 14, 80)	0	block4c_project_...
block4c_add (Add)	(None, 14, 14, 80)	0	block4c_drop[0][...] block4b_add[0][0]
block5a_expand_conv (Conv2D)	(None, 14, 14, 480)	38,400	block4c_add[0][0]
block5a_expand_bn (BatchNormalizatio...)	(None, 14, 14, 480)	1,920	block5a_expand_c...
block5a_expand_act... (Activation)	(None, 14, 14, 480)	0	block5a_expand_b...
block5a_dwconv (DepthwiseConv2D)	(None, 14, 14, 480)	12,000	block5a_expand_a...
block5a_bn (BatchNormalizatio...)	(None, 14, 14, 480)	1,920	block5a_dwconv[0...]
block5a_activation (Activation)	(None, 14, 14, 480)	0	block5a_bn[0][0]
block5a_se_squeeze (GlobalAveragePool...)	(None, 480)	0	block5a_activati...
block5a_se_reshape (Reshape)	(None, 1, 1, 480)	0	block5a_se_squee...
block5a_se_reduce (Conv2D)	(None, 1, 1, 20)	9,620	block5a_se_resha...

block5a_se_expand (Conv2D)	(None, 1, 1, 480)	10,080	block5a_se_reduc...
block5a_se_excite (Multiply)	(None, 14, 14, 480)	0	block5a_activati... block5a_se_expan...
block5a_project_co... (Conv2D)	(None, 14, 14, 112)	53,760	block5a_se_excit...
block5a_project_bn (BatchNormalizatio...)	(None, 14, 14, 112)	448	block5a_project_...
block5b_expand_conv (Conv2D)	(None, 14, 14, 672)	75,264	block5a_project_...
block5b_expand_bn (BatchNormalizatio...)	(None, 14, 14, 672)	2,688	block5b_expand_c...
block5b_expand_act... (Activation)	(None, 14, 14, 672)	0	block5b_expand_b...
block5b_dwconv (DepthwiseConv2D)	(None, 14, 14, 672)	16,800	block5b_expand_a...
block5b_bn (BatchNormalizatio...)	(None, 14, 14, 672)	2,688	block5b_dwconv[0]...
block5b_activation (Activation)	(None, 14, 14, 672)	0	block5b_bn[0][0]
block5b_se_squeeze (GlobalAveragePool...)	(None, 672)	0	block5b_activati...
block5b_se_reshape (Reshape)	(None, 1, 1, 672)	0	block5b_se_squeee...
block5b_se_reduce (Conv2D)	(None, 1, 1, 28)	18,844	block5b_se_resha...
block5b_se_expand (Conv2D)	(None, 1, 1, 672)	19,488	block5b_se_reduc...
block5b_se_excite	(None, 14, 14,	0	block5b_activati...

(Multiply)	672)		block5b_se_expan...
block5b_project_co... (Conv2D)	(None, 14, 14, 112)	75,264	block5b_se_excit...
block5b_project_bn (BatchNormalizatio...)	(None, 14, 14, 112)	448	block5b_project_...
block5b_drop (Dropout)	(None, 14, 14, 112)	0	block5b_project_...
block5b_add (Add)	(None, 14, 14, 112)	0	block5b_drop[0][...] block5a_project_...
block5c_expand_conv (Conv2D)	(None, 14, 14, 672)	75,264	block5b_add[0][0]
block5c_expand_bn (BatchNormalizatio...)	(None, 14, 14, 672)	2,688	block5c_expand_c...
block5c_expand_act... (Activation)	(None, 14, 14, 672)	0	block5c_expand_b...
block5c_dwconv (DepthwiseConv2D)	(None, 14, 14, 672)	16,800	block5c_expand_a...
block5c_bn (BatchNormalizatio...)	(None, 14, 14, 672)	2,688	block5c_dwconv[0...]
block5c_activation (Activation)	(None, 14, 14, 672)	0	block5c_bn[0][0]
block5c_se_squeeze (GlobalAveragePool...)	(None, 672)	0	block5c_activati...
block5c_se_reshape (Reshape)	(None, 1, 1, 672)	0	block5c_se_squee...
block5c_se_reduce (Conv2D)	(None, 1, 1, 28)	18,844	block5c_se_resha...
block5c_se_expand (Conv2D)	(None, 1, 1, 672)	19,488	block5c_se_reduc...

block5c_se_excite (Multiply)	(None, 14, 14, 672)	0	block5c_activati... block5c_se_expan...
block5c_project_co... (Conv2D)	(None, 14, 14, 112)	75,264	block5c_se_excit...
block5c_project_bn (BatchNormalizatio...)	(None, 14, 14, 112)	448	block5c_project_...
block5c_drop (Dropout)	(None, 14, 14, 112)	0	block5c_project_...
block5c_add (Add)	(None, 14, 14, 112)	0	block5c_drop[0][...] block5b_add[0][0]
block6a_expand_conv (Conv2D)	(None, 14, 14, 672)	75,264	block5c_add[0][0]
block6a_expand_bn (BatchNormalizatio...)	(None, 14, 14, 672)	2,688	block6a_expand_c...
block6a_expand_act... (Activation)	(None, 14, 14, 672)	0	block6a_expand_b...
block6a_dwconv_pad (ZeroPadding2D)	(None, 17, 17, 672)	0	block6a_expand_a...
block6a_dwconv (DepthwiseConv2D)	(None, 7, 7, 672)	16,800	block6a_dwconv_p...
block6a_bn (BatchNormalizatio...)	(None, 7, 7, 672)	2,688	block6a_dwconv[0]...
block6a_activation (Activation)	(None, 7, 7, 672)	0	block6a_bn[0][0]
block6a_se_squeeze (GlobalAveragePool...)	(None, 672)	0	block6a_activati...
block6a_se_reshape (Reshape)	(None, 1, 1, 672)	0	block6a_se_squeee...

block6a_se_reduce (Conv2D)	(None, 1, 1, 28)	18,844	block6a_se_resha...
block6a_se_expand (Conv2D)	(None, 1, 1, 672)	19,488	block6a_se_reduc...
block6a_se_excite (Multiply)	(None, 7, 7, 672)	0	block6a_activati... block6a_se_expan...
block6a_project_co... (Conv2D)	(None, 7, 7, 192)	129,024	block6a_se_excit...
block6a_project_bn (BatchNormalizatio...)	(None, 7, 7, 192)	768	block6a_project_...
block6b_expand_conv (Conv2D)	(None, 7, 7, 1152)	221,184	block6a_project_...
block6b_expand_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block6b_expand_c...
block6b_expand_act... (Activation)	(None, 7, 7, 1152)	0	block6b_expand_b...
block6b_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	28,800	block6b_expand_a...
block6b_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block6b_dwconv[0]...
block6b_activation (Activation)	(None, 7, 7, 1152)	0	block6b_bn[0][0]
block6b_se_squeeze (GlobalAveragePool...)	(None, 1152)	0	block6b_activati...
block6b_se_reshape (Reshape)	(None, 1, 1, 1152)	0	block6b_se_squee...
block6b_se_reduce (Conv2D)	(None, 1, 1, 48)	55,344	block6b_se_resha...
block6b_se_expand	(None, 1, 1,	56,448	block6b_se_reduc...

(Conv2D)	1152)		
block6b_se_excite (Multiply)	(None, 7, 7, 1152)	0	block6b_activati... block6b_se_expan...
block6b_project_co... (Conv2D)	(None, 7, 7, 192)	221,184	block6b_se_excit...
block6b_project_bn (BatchNormalizatio...)	(None, 7, 7, 192)	768	block6b_project_...
block6b_drop (Dropout)	(None, 7, 7, 192)	0	block6b_project_...
block6b_add (Add)	(None, 7, 7, 192)	0	block6b_drop[0][... block6a_project_...
block6c_expand_conv (Conv2D)	(None, 7, 7, 1152)	221,184	block6b_add[0][0]
block6c_expand_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block6c_expand_c...
block6c_expand_act... (Activation)	(None, 7, 7, 1152)	0	block6c_expand_b...
block6c_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	28,800	block6c_expand_a...
block6c_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block6c_dwconv[0... block6c_bn[0][0]
block6c_activation (Activation)	(None, 7, 7, 1152)	0	block6c_bn[0][0]
block6c_se_squeeze (GlobalAveragePool...)	(None, 1152)	0	block6c_activati...
block6c_se_reshape (Reshape)	(None, 1, 1, 1152)	0	block6c_se_squeee...
block6c_se_reduce (Conv2D)	(None, 1, 1, 48)	55,344	block6c_se_resha...

block6c_se_expand (Conv2D)	(None, 1, 1, 1152)	56,448	block6c_se_reduc...
block6c_se_excite (Multiply)	(None, 7, 7, 1152)	0	block6c_activati... block6c_se_expan...
block6c_project_co... (Conv2D)	(None, 7, 7, 192)	221,184	block6c_se_excit...
block6c_project_bn (BatchNormalizatio...)	(None, 7, 7, 192)	768	block6c_project_...
block6c_drop (Dropout)	(None, 7, 7, 192)	0	block6c_project_...
block6c_add (Add)	(None, 7, 7, 192)	0	block6c_drop[0][...] block6b_add[0][0]
block6d_expand_conv (Conv2D)	(None, 7, 7, 1152)	221,184	block6c_add[0][0]
block6d_expand_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block6d_expand_c...
block6d_expand_act... (Activation)	(None, 7, 7, 1152)	0	block6d_expand_b...
block6d_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	28,800	block6d_expand_a...
block6d_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block6d_dwconv[0...]
block6d_activation (Activation)	(None, 7, 7, 1152)	0	block6d_bn[0][0]
block6d_se_squeeze (GlobalAveragePool...)	(None, 1152)	0	block6d_activati...
block6d_se_reshape (Reshape)	(None, 1, 1, 1152)	0	block6d_se_squee...

block6d_se_reduce (Conv2D)	(None, 1, 1, 48)	55,344	block6d_se_resha...
block6d_se_expand (Conv2D)	(None, 1, 1, 1152)	56,448	block6d_se_reduc...
block6d_se_excite (Multiply)	(None, 7, 7, 1152)	0	block6d_activati... block6d_se_expan...
block6d_project_co... (Conv2D)	(None, 7, 7, 192)	221,184	block6d_se_excit...
block6d_project_bn (BatchNormalizatio...)	(None, 7, 7, 192)	768	block6d_project_...
block6d_drop (Dropout)	(None, 7, 7, 192)	0	block6d_project_...
block6d_add (Add)	(None, 7, 7, 192)	0	block6d_drop[0][...] block6c_add[0][0]
block7a_expand_conv (Conv2D)	(None, 7, 7, 1152)	221,184	block6d_add[0][0]
block7a_expand_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block7a_expand_c...
block7a_expand_act... (Activation)	(None, 7, 7, 1152)	0	block7a_expand_b...
block7a_dwconv (DepthwiseConv2D)	(None, 7, 7, 1152)	10,368	block7a_expand_a...
block7a_bn (BatchNormalizatio...)	(None, 7, 7, 1152)	4,608	block7a_dwconv[0...]
block7a_activation (Activation)	(None, 7, 7, 1152)	0	block7a_bn[0][0]
block7a_se_squeeze (GlobalAveragePool...)	(None, 1152)	0	block7a_activati...
block7a_se_reshape	(None, 1, 1,	0	block7a_se_squee...

(Reshape)	1152)		
block7a_se_reduce (Conv2D)	(None, 1, 1, 48)	55,344	block7a_se_resha...
block7a_se_expand (Conv2D)	(None, 1, 1, 1152)	56,448	block7a_se_reduc...
block7a_se_excite (Multiply)	(None, 7, 7, 1152)	0	block7a_activati... block7a_se_expan...
block7a_project_co... (Conv2D)	(None, 7, 7, 320)	368,640	block7a_se_excit...
block7a_project_bn (BatchNormalizatio...)	(None, 7, 7, 320)	1,280	block7a_project_...
top_conv (Conv2D)	(None, 7, 7, 1280)	409,600	block7a_project_...
top_bn (BatchNormalizatio...)	(None, 7, 7, 1280)	5,120	top_conv[0][0]
top_activation (Activation)	(None, 7, 7, 1280)	0	top_bn[0][0]
flatten (Flatten)	(None, 62720)	0	top_activation[0...
dense (Dense)	(None, 256)	16,056,576	flatten[0][0]
dropout (Dropout)	(None, 256)	0	dense[0][0]
dense_1 (Dense)	(None, 4)	1,028	dropout[0][0]

Total params: 20,107,175 (76.70 MB)

Trainable params: 16,057,604 (61.25 MB)

Non-trainable params: 4,049,571 (15.45 MB)

Hiperparámetros de entrenamiento y estrategias para manejar el sobreajuste

El modelo se compiló utilizando el optimizador Adam con una tasa de aprendizaje inicial de 1×10^{-4} . La elección de Adam se fundamenta en su capacidad para combinar las ventajas de RMSProp (adaptación del paso de aprendizaje por parámetro) y Momentum (suavizado de las actualizaciones), permitiendo una convergencia más estable y rápida en problemas donde los gradientes pueden presentar alta variabilidad, como ocurre con los datos aumentados. Aunque RMSProp es recomendado por los autores de EfficientNet, en este trabajo se prefirió Adam debido a su robustez ante tasas de aprendizaje mal ajustadas y su mejor comportamiento empírico en tareas de clasificación médica con conjuntos moderados de datos.

Los hiperparámetros principales se definieron como sigue:

- Tamaño de imagen (IMG_SIZE): 224×224 píxeles, manteniendo compatibilidad con EfficientNetB0.
- Tamaño de lote (BATCH_SIZE): 16, seleccionado como equilibrio entre estabilidad del gradiente y uso eficiente de GPU.
- Número máximo de épocas (EPOCHS): 50, lo que permitió suficiente margen de optimización sin prolongar innecesariamente el entrenamiento.
- Tasa de aprendizaje (LEARNING_RATE): 1×10^{-4} , valor típico para transfer learning que evita saltos abruptos en los pesos.
- Proporción de validación (TEST_SIZE): 0.2, equivalente a una división 80/20 del conjunto de datos.

Para prevenir el sobreajuste y optimizar los recursos, se emplearon dos callbacks de Keras:

- EarlyStopping: monitoreó la métrica de validación (val_accuracy) y detuvo el entrenamiento si no se observaba mejora durante 8 épocas consecutivas. Se activó la restauración automática de los mejores pesos (restore_best_weights=True), garantizando que el modelo final correspondiera al punto de máximo rendimiento.
- ReduceLROnPlateau: redujo automáticamente la tasa de aprendizaje a la mitad (factor=0.5) cuando la precisión de validación se estancaba durante 5 épocas, con un límite inferior de 1×10^{-6} . Esta estrategia permitió refinar el aprendizaje en etapas avanzadas, mejorando la estabilidad y evitando oscilaciones en la pérdida.

Ambas técnicas contribuyeron a un proceso de entrenamiento más eficiente y estable, asegurando que el modelo convergiera hacia un mínimo más robusto sin incurrir en sobreajuste.

```
In [ ]: # =====  
# CALLBACKS
```

```
# =====
early_stopping = EarlyStopping(monitor='val_accuracy', patience=8, mode='max', restore_best_weights=True, verbose=1)
lr_plateau = ReduceLROnPlateau(monitor='val_accuracy', mode='max', factor=0.5, patience=5, min_lr=1e-6, verbose=1)

# =====
# ENTRENAMIENTO
# =====

history = model.fit(
    datagen.flow(X_train, y_train, batch_size=BATCH_SIZE),
    validation_data=(X_test, y_test),
    epochs=EPOCHS,
    callbacks=[early_stopping, lr_plateau],
    verbose=1
)
```

```
c:\Users\Camilo\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

Epoch 1/50
1059/1059 427s 397ms/step - accuracy: 0.7897 - loss: 0.5670 - val_accuracy: 0.8344 - val_loss: 0.4268 - learning_rate: 1.0000e-04

Epoch 2/50
1059/1059 408s 385ms/step - accuracy: 0.8397 - loss: 0.4279 - val_accuracy: 0.8535 - val_loss: 0.3892 - learning_rate: 1.0000e-04

Epoch 3/50
1059/1059 411s 388ms/step - accuracy: 0.8578 - loss: 0.3793 - val_accuracy: 0.8715 - val_loss: 0.3554 - learning_rate: 1.0000e-04

Epoch 4/50
1059/1059 406s 383ms/step - accuracy: 0.8643 - loss: 0.3679 - val_accuracy: 0.8819 - val_loss: 0.3198 - learning_rate: 1.0000e-04

Epoch 5/50
1059/1059 404s 382ms/step - accuracy: 0.8737 - loss: 0.3457 - val_accuracy: 0.8715 - val_loss: 0.3645 - learning_rate: 1.0000e-04

Epoch 6/50
1059/1059 407s 384ms/step - accuracy: 0.8799 - loss: 0.3333 - val_accuracy: 0.8724 - val_loss: 0.3498 - learning_rate: 1.0000e-04

Epoch 7/50
1059/1059 396s 374ms/step - accuracy: 0.8808 - loss: 0.3221 - val_accuracy: 0.8601 - val_loss: 0.3725 - learning_rate: 1.0000e-04

Epoch 8/50
1059/1059 392s 370ms/step - accuracy: 0.8841 - loss: 0.3076 - val_accuracy: 0.8824 - val_loss: 0.3156 - learning_rate: 1.0000e-04

Epoch 9/50
1059/1059 393s 371ms/step - accuracy: 0.8867 - loss: 0.3063 - val_accuracy: 0.8443 - val_loss: 0.4148 - learning_rate: 1.0000e-04

Epoch 10/50
1059/1059 402s 380ms/step - accuracy: 0.8928 - loss: 0.2945 - val_accuracy: 0.8809 - val_loss: 0.3323 - learning_rate: 1.0000e-04

Epoch 11/50
1059/1059 395s 373ms/step - accuracy: 0.8942 - loss: 0.2890 - val_accuracy: 0.8410 - val_loss: 0.4120 - learning_rate: 1.0000e-04

Epoch 12/50
1059/1059 396s 374ms/step - accuracy: 0.8936 - loss: 0.2872 - val_accuracy: 0.9022 - val_loss: 0.2783 - learning_rate: 1.0000e-04

Epoch 13/50
1059/1059 397s 375ms/step - accuracy: 0.8949 - loss: 0.2799 - val_accuracy: 0.8705 - val_loss: 0.3394 - learning_rate: 1.0000e-04

Epoch 14/50
1059/1059 419s 395ms/step - accuracy: 0.8994 - loss: 0.2704 - val_accuracy: 0.8665 - val_loss: 0.3594 - learning_rate: 1.0000e-04

```
arning_rate: 1.0000e-04
Epoch 15/50
1059/1059 421s 398ms/step - accuracy: 0.8997 - loss: 0.2700 - val_accuracy: 0.8786 - val_loss: 0.3332 - le
arning_rate: 1.0000e-04
Epoch 16/50
1059/1059 423s 399ms/step - accuracy: 0.9053 - loss: 0.2574 - val_accuracy: 0.8824 - val_loss: 0.3367 - le
arning_rate: 1.0000e-04
Epoch 17/50
1059/1059 0s 335ms/step - accuracy: 0.9040 - loss: 0.2570
Epoch 17: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
1059/1059 413s 390ms/step - accuracy: 0.9037 - loss: 0.2577 - val_accuracy: 0.8975 - val_loss: 0.2813 - le
arning_rate: 1.0000e-04
Epoch 18/50
1059/1059 440s 416ms/step - accuracy: 0.9101 - loss: 0.2392 - val_accuracy: 0.8956 - val_loss: 0.2953 - le
arning_rate: 5.0000e-05
Epoch 19/50
1059/1059 447s 422ms/step - accuracy: 0.9157 - loss: 0.2336 - val_accuracy: 0.8994 - val_loss: 0.2869 - le
arning_rate: 5.0000e-05
Epoch 20/50
1059/1059 448s 423ms/step - accuracy: 0.9151 - loss: 0.2298 - val_accuracy: 0.8573 - val_loss: 0.4037 - le
arning_rate: 5.0000e-05
Epoch 20: early stopping
Restoring model weights from the end of the best epoch: 12.
```

Resultados obtenidos y métricas de evaluación

```
In [ ]: # =====
# EVALUACIÓN Y GRAFICAS
# =====
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print(f"\n🎯 Precisión en test: {acc*100:.2f}%")

# Curvas
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Entrenamiento')
plt.plot(history.history['val_accuracy'], label='Validación')
plt.title('Evolución de la Precisión')
plt.xlabel('Épocas')
```

```
plt.ylabel('Accuracy')
plt.legend()

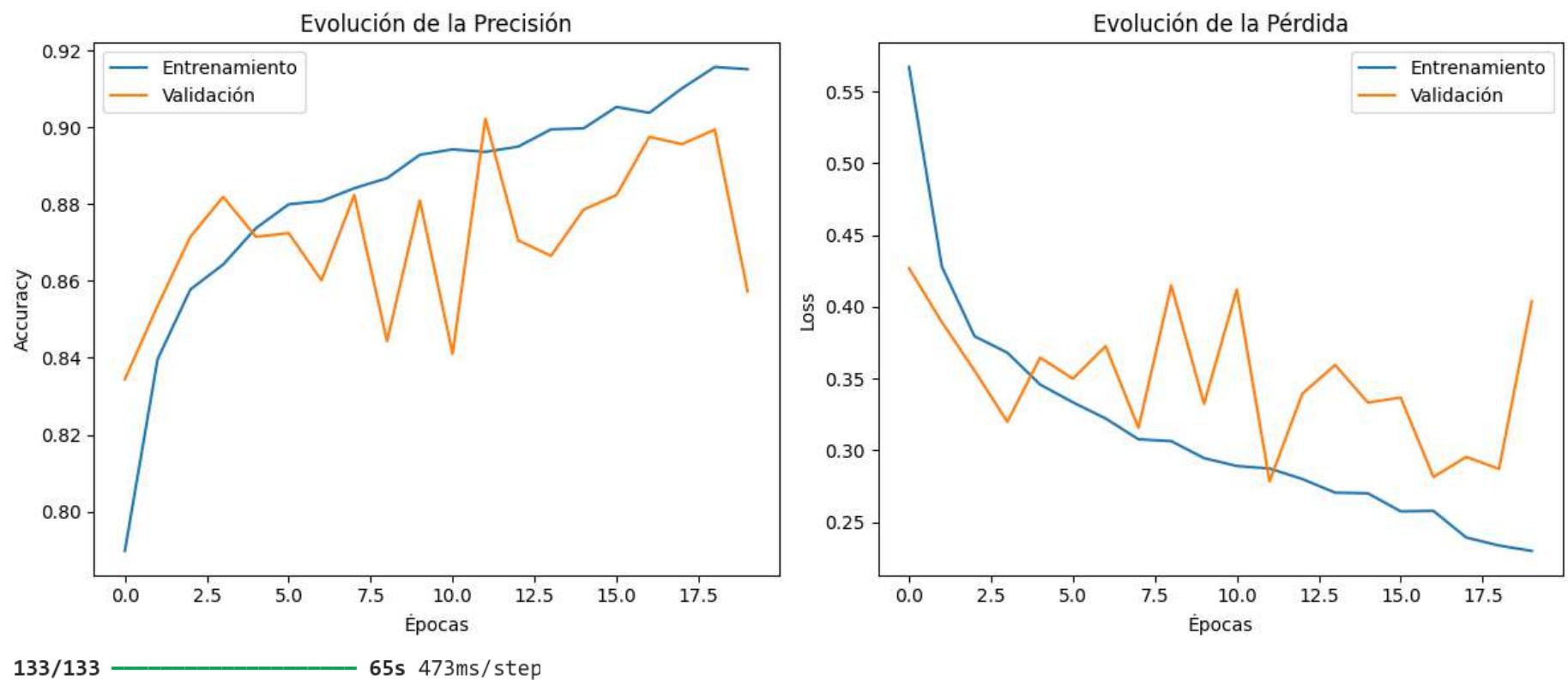
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Entrenamiento')
plt.plot(history.history['val_loss'], label='Validación')
plt.title('Evolución de la Pérdida')
plt.xlabel('Épocas')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()

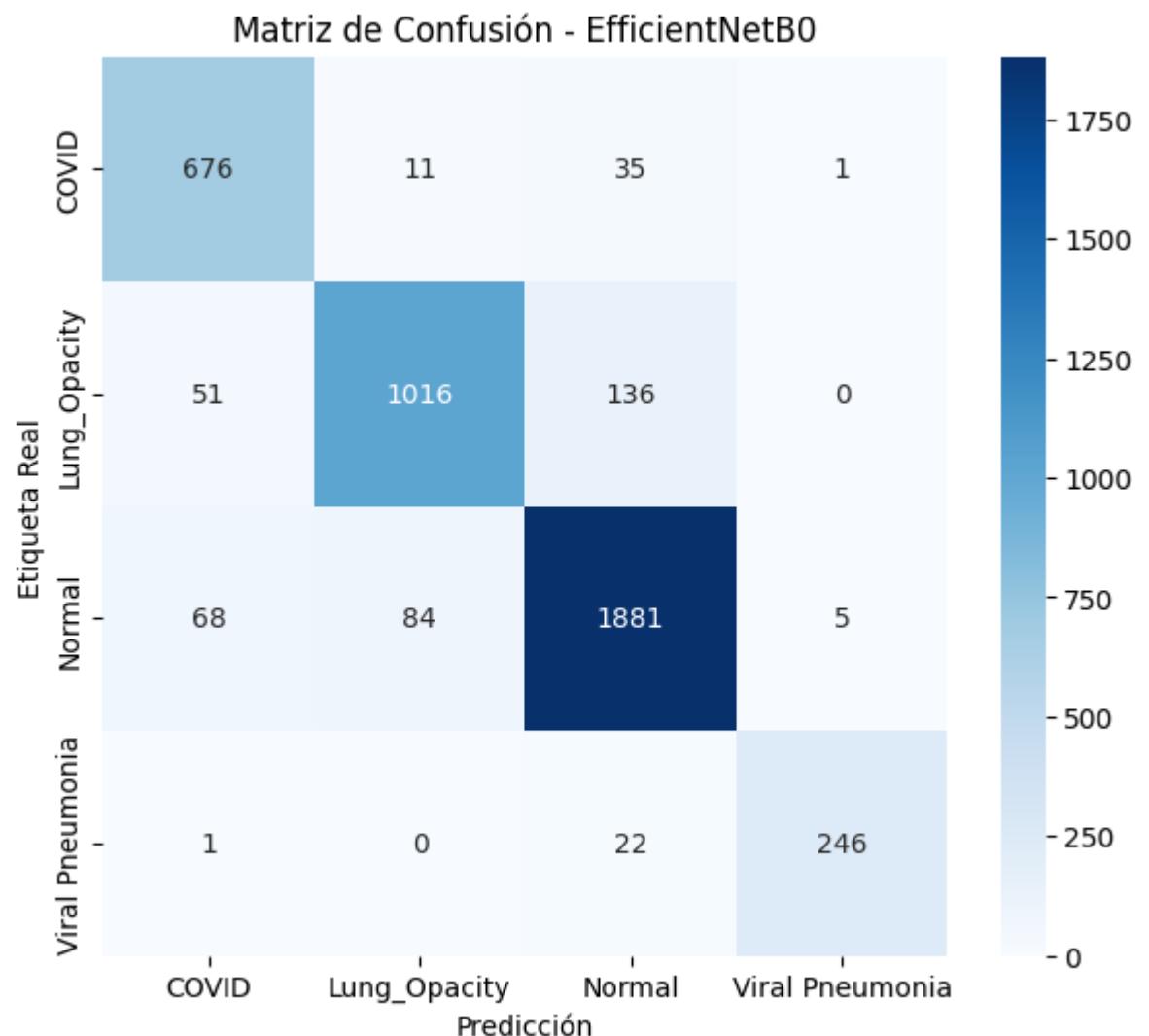
# Matriz de confusión
y_pred = model.predict(X_test)
y_true_labels = np.argmax(y_test, axis=1)
y_pred_labels = np.argmax(y_pred, axis=1)

cm = confusion_matrix(y_true_labels, y_pred_labels)

plt.figure(figsize=(7,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_names, yticklabels=class_names)
plt.title("Matriz de Confusión - EfficientNetB0")
plt.xlabel("Predicción")
plt.ylabel("Etiqueta Real")
plt.show()
```

🎯 Precisión en test: 90.22%





El modelo EfficientNetB0 alcanzó una precisión global del 90.22 % sobre el conjunto de prueba, evidenciando un desempeño competitivo para la tarea de clasificación de radiografías de tórax en cuatro categorías: COVID-19, Lung Opacity, Normal y Viral Pneumonia.

Curvas de entrenamiento

En la evolución de la precisión y la pérdida (Figura 1), se observa un incremento sostenido de la accuracy tanto en entrenamiento como en

validación, acompañado de una disminución progresiva de la loss. Aunque la curva de validación presenta cierta variabilidad, el comportamiento general sugiere una buena capacidad de generalización sin síntomas severos de sobreajuste. Las oscilaciones en la métrica de validación son esperables en modelos con data augmentation, ya que cada lote introduce imágenes transformadas que modifican la distribución de entrada. El uso de EarlyStopping y ReduceLROnPlateau permitió detener el entrenamiento en torno a la época 18, momento en el cual las métricas se estabilizaron, evitando un entrenamiento redundante.

Matriz de confusión

La matriz de confusión (Figura 2) muestra que las clases Normal y Lung Opacity concentran la mayor cantidad de muestras, y el modelo mantiene un desempeño alto en ambas. Las principales confusiones se presentan entre COVID-19 y Lung Opacity, lo cual es coherente desde el punto de vista médico, ya que ambas condiciones generan patrones radiográficos similares de opacidad pulmonar. A pesar de esto, la red logra una buena separación global entre las clases.

```
In [ ]: # Predicciones en test
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Reporte de métricas por clase
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred_classes, target_names=class_names))
```

	133/133 ━━━━━━━━ 63s 471ms/step			
	precision	recall	f1-score	support
COVID	0.85	0.93	0.89	723
Lung_Opacity	0.91	0.84	0.88	1203
Normal	0.91	0.92	0.91	2038
Viral Pneumonia	0.98	0.91	0.94	269
accuracy			0.90	4233
macro avg	0.91	0.90	0.91	4233
weighted avg	0.90	0.90	0.90	4233

Métricas por clase

El reporte de clasificación arroja los siguientes resultados:

- COVID-19: precisión = 0.85, recall = 0.93, F1 = 0.89
- Lung Opacity: precisión = 0.91, recall = 0.84, F1 = 0.88
- Normal: precisión = 0.92, recall = 0.91, F1 = 0.92
- Viral Pneumonia: precisión = 0.98, recall = 0.91, F1 = 0.94

El promedio ponderado de las métricas (weighted avg) es precisión = 0.90, recall = 0.90, F1 = 0.90, lo que confirma la estabilidad del modelo y su consistencia entre clases.

En conjunto, los resultados indican que el modelo logra un equilibrio adecuado entre sensibilidad y precisión, siendo capaz de distinguir correctamente la mayoría de los casos incluso entre clases con patrones radiográficos cercanos.

Descongelar capas

Tras entrenar la versión inicial del modelo con la base convolucional congelada, se realizó un ajuste fino (fine-tuning) con el propósito de refinar los filtros de las capas más profundas de EfficientNetB0 y adaptarlos mejor al dominio médico del conjunto de datos.

Para ello, se descongelaron las últimas 20 capas del modelo y se recompiló la red utilizando una tasa de aprendizaje reducida (5×10^{-6}), con el fin de evitar modificaciones drásticas sobre los pesos preentrenados. El entrenamiento se ejecutó durante 10 épocas adicionales, manteniendo las estrategias de EarlyStopping y ReduceLROnPlateau.

In [23]:

```
# =====
# FINE-TUNING: Descongelar las últimas 20 capas de TODO el modelo
# =====

from tensorflow.keras.optimizers import Adam

# Ver cuántas capas totales tiene el modelo (ya incluye EfficientNet)
print(f"Número total de capas en el modelo completo: {len(model.layers)}")

# Descongelar las últimas 20 capas
for layer in model.layers[-20:]:
```

```
layer.trainable = True

# Comprobación (opcional)
trainable_count = sum([layer.trainable for layer in model.layers])
print(f"💡 Capas entrenables después del cambio: {trainable_count}")

# Recompilar con learning rate bajo
model.compile(
    optimizer=Adam(learning_rate=1e-5),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

print("\n📝 Iniciando fine-tuning (últimas 20 capas del modelo)...")

# Entrenamiento adicional (pocas épocas)
history_finetune = model.fit(
    datagen.flow(X_train, y_train, batch_size=BATCH_SIZE),
    validation_data=(X_test, y_test),
    epochs=10,
    callbacks=[early_stopping, lr_plateau],
    verbose=1
)

# Evaluar nuevamente
loss_ft, acc_ft = model.evaluate(X_test, y_test, verbose=0)
print(f"\n◆ Precisión tras fine-tuning: {acc_ft*100:.2f}%")
```

Número total de capas en el modelo completo: 241

💡 Capas entrenables después del cambio: 21

⌚ Iniciando fine-tuning (últimas 20 capas del modelo)...

Epoch 1/10

1059/1059 479s 445ms/step - accuracy: 0.8207 - loss: 0.7993 - val_accuracy: 0.8838 - val_loss: 0.3688 - learning_rate: 1.0000e-05

Epoch 2/10

1059/1059 464s 438ms/step - accuracy: 0.8663 - loss: 0.4067 - val_accuracy: 0.8783 - val_loss: 0.3497 - learning_rate: 1.0000e-05

Epoch 3/10

1059/1059 462s 437ms/step - accuracy: 0.8816 - loss: 0.3313 - val_accuracy: 0.8826 - val_loss: 0.3273 - learning_rate: 1.0000e-05

Epoch 4/10

1059/1059 464s 438ms/step - accuracy: 0.8901 - loss: 0.3058 - val_accuracy: 0.8826 - val_loss: 0.3263 - learning_rate: 1.0000e-05

Epoch 5/10

1059/1059 0s 387ms/step - accuracy: 0.8909 - loss: 0.2843

Epoch 5: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-06.

1059/1059 470s 444ms/step - accuracy: 0.8945 - loss: 0.2837 - val_accuracy: 0.8720 - val_loss: 0.3546 - learning_rate: 1.0000e-05

Epoch 6/10

1059/1059 467s 441ms/step - accuracy: 0.8976 - loss: 0.2809 - val_accuracy: 0.8906 - val_loss: 0.3092 - learning_rate: 5.0000e-06

Epoch 7/10

1059/1059 467s 441ms/step - accuracy: 0.9031 - loss: 0.2656 - val_accuracy: 0.8887 - val_loss: 0.3178 - learning_rate: 5.0000e-06

Epoch 8/10

1059/1059 470s 444ms/step - accuracy: 0.9011 - loss: 0.2709 - val_accuracy: 0.8859 - val_loss: 0.3202 - learning_rate: 5.0000e-06

Epoch 8: early stopping

Restoring model weights from the end of the best epoch: 1.

- ◆ Precisión tras fine-tuning: 88.38%

In [25]:

```
# =====
# EVALUACIÓN Y GRAFICAS DESPUES DE DESCONGELAR
# =====
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print(f"\n🎯 Precisión en test: {acc*100:.2f}%")
```

```
# Curvas
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Entrenamiento')
plt.plot(history.history['val_accuracy'], label='Validación')
plt.title('Evolución de la Precisión')
plt.xlabel('Épocas')
plt.ylabel('Accuracy')
plt.legend()

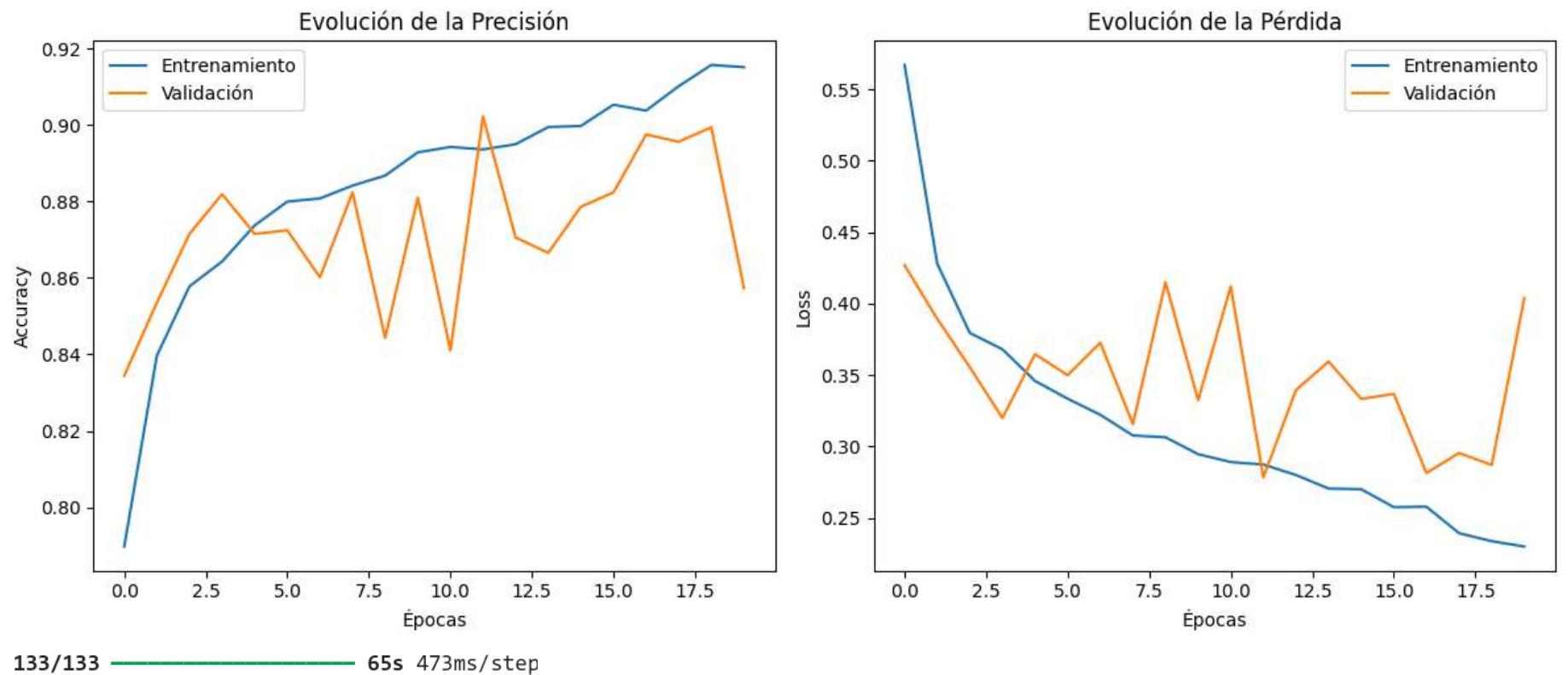
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Entrenamiento')
plt.plot(history.history['val_loss'], label='Validación')
plt.title('Evolución de la Pérdida')
plt.xlabel('Épocas')
plt.ylabel('Loss')
plt.legend()
plt.tight_layout()
plt.show()

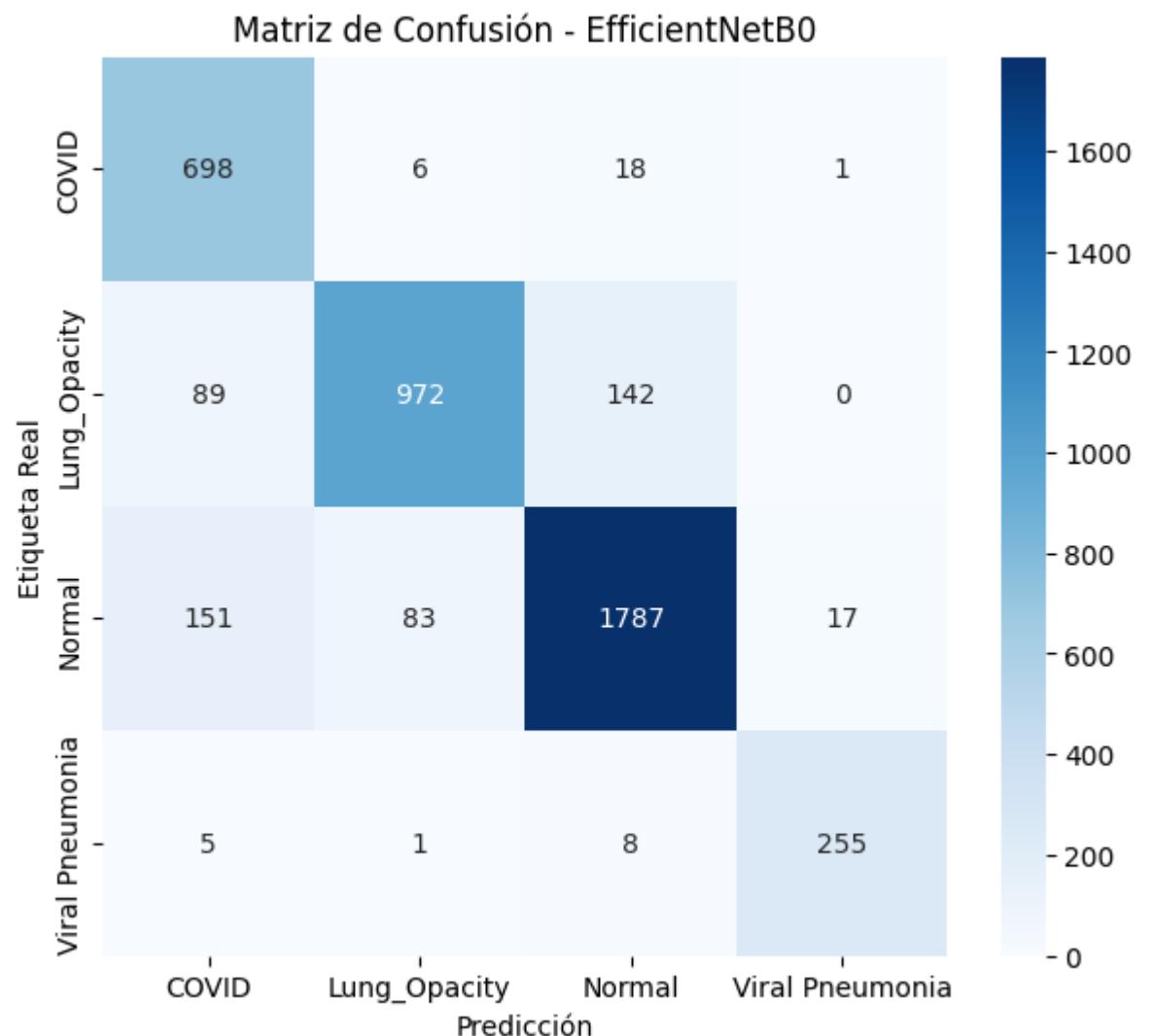
# Matriz de confusión
y_pred = model.predict(X_test)
y_true_labels = np.argmax(y_test, axis=1)
y_pred_labels = np.argmax(y_pred, axis=1)

cm = confusion_matrix(y_true_labels, y_pred_labels)

plt.figure(figsize=(7,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_names, yticklabels=class_names)
plt.title("Matriz de Confusión - EfficientNetB0")
plt.xlabel("Predicción")
plt.ylabel("Etiqueta Real")
plt.show()
```

🎯 Precisión en test: 87.69%





El modelo ajustado alcanzó una precisión del 87.69 % en el conjunto de prueba, ligeramente inferior al resultado obtenido antes del ajuste (90.22 %). Este comportamiento puede atribuirse a que, al actualizar los pesos de las capas profundas, parte de la información general aprendida en ImageNet se ajustó de forma excesiva al conjunto de entrenamiento actual, reduciendo la capacidad de generalización. La matriz de confusión posterior al fine-tuning muestra que las principales confusiones se mantienen entre las clases Normal y Lung Opacity, mientras que COVID-19 y Viral Pneumonia presentan un leve descenso en recall.

```
In [26]: ### Metricas despues de descongelar capas
# Predicciones en test
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

# Reporte de métricas por clase
from sklearn.metrics import classification_report
print(classification_report(y_true, y_pred_classes, target_names=class_names))
```

133/133 ————— 61s 458ms/step

	precision	recall	f1-score	support
COVID	0.74	0.97	0.84	723
Lung_Opacity	0.92	0.81	0.86	1203
Normal	0.91	0.88	0.90	2038
Viral Pneumonia	0.93	0.95	0.94	269
accuracy			0.88	4233
macro avg	0.88	0.90	0.88	4233
weighted avg	0.89	0.88	0.88	4233

Las métricas por clase reflejan este patrón:

- COVID-19: precisión = 0.74, recall = 0.97, F1 = 0.84
- Lung Opacity: precisión = 0.92, recall = 0.81, F1 = 0.86
- Normal: precisión = 0.91, recall = 0.88, F1 = 0.89
- Viral Pneumonia: precisión = 0.93, recall = 0.95, F1 = 0.94

El promedio ponderado (weighted avg) fue precisión = 0.89, recall = 0.88, F1 = 0.88.

En conjunto, el experimento de fine-tuning evidenció que descongelar un número moderado de capas (por ejemplo, 10–15) podría ser más adecuado para este problema, ya que un ajuste demasiado profundo tiende a alterar representaciones previamente optimizadas. Aun así, este paso permitió validar empíricamente la sensibilidad del modelo a la cantidad de capas entrenables y la importancia de regular la tasa de aprendizaje durante el ajuste fino.

Modelo sin transfer learning

In [17]:

```
# =====
# COMPARACIÓN: Modelo entrenado desde cero vs Transfer Learning
# =====

from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import accuracy_score

# Modelo desde cero (sin pesos preentrenados)
base_scratch = EfficientNetB0(weights=None, include_top=False, input_shape=(IMG_SIZE, IMG_SIZE, 3))

x = Flatten()(base_scratch.output)
x = Dense(256, activation="relu")(x)
x = Dropout(0.3)(x)
output = Dense(len(class_names), activation="softmax")(x)
```

```
model_scratch = Model(inputs=base_scratch.input, outputs=output)

model_scratch.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

print("\nEntrenando modelo desde cero (sin transfer learning)...")

# Entrenamos solo unas pocas épocas (para comparación rápida)
history_scratch = model_scratch.fit(
    datagen.flow(X_train, y_train, batch_size=BATCH_SIZE),
    validation_data=(X_test, y_test),
    epochs=8,
    verbose=1
)

# Evaluar desempeño
loss_scratch, acc_scratch = model_scratch.evaluate(X_test, y_test, verbose=0)
print(f"\n◆ Precisión modelo desde cero: {acc_scratch*100:.2f}%")

# =====
# COMPARACIÓN FINAL
# =====

# Supongamos que el modelo con Transfer Learning ya se llama "model"
loss_tl, acc_tl = model.evaluate(X_test, y_test, verbose=0)

print("\n===== COMPARACIÓN FINAL =====")
print(f"Modelo desde cero : {acc_scratch*100:.2f}%")
print(f"Modelo Transfer Learning: {acc_tl*100:.2f}%")
print("=====")
```

Entrenando modelo desde cero (sin transfer learning)...

c:\Users\Camilo\AppData\Local\Programs\Python313\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
self._warn_if_super_not_called()

```
Epoch 1/8  
1059/1059 1778s 2s/step - accuracy: 0.5679 - loss: 1.0840 - val_accuracy: 0.6825 - val_loss: 0.7972  
Epoch 2/8  
1059/1059 1651s 2s/step - accuracy: 0.6738 - loss: 0.8356 - val_accuracy: 0.7361 - val_loss: 0.6830  
Epoch 3/8  
1059/1059 1629s 2s/step - accuracy: 0.7107 - loss: 0.7533 - val_accuracy: 0.7569 - val_loss: 0.6390  
Epoch 4/8  
1059/1059 1759s 2s/step - accuracy: 0.7330 - loss: 0.6898 - val_accuracy: 0.7616 - val_loss: 0.6342  
Epoch 5/8  
1059/1059 1681s 2s/step - accuracy: 0.7599 - loss: 0.6344 - val_accuracy: 0.7763 - val_loss: 0.5953  
Epoch 6/8  
1059/1059 1708s 2s/step - accuracy: 0.7725 - loss: 0.5983 - val_accuracy: 0.7935 - val_loss: 0.5366  
Epoch 7/8  
1059/1059 1702s 2s/step - accuracy: 0.7897 - loss: 0.5688 - val_accuracy: 0.7947 - val_loss: 0.6172  
Epoch 8/8  
1059/1059 1706s 2s/step - accuracy: 0.8045 - loss: 0.5285 - val_accuracy: 0.8193 - val_loss: 0.5026
```

- ◆ Precisión modelo desde cero: 81.93%

===== COMPARACIÓN FINAL =====

Modelo desde cero : 81.93%

Modelo Transfer Learning: 90.22%

Con el objetivo de analizar el impacto del transfer learning, se entrenó una versión del modelo EfficientNetB0 sin pesos preentrenados, es decir, con inicialización aleatoria de todos los parámetros. Este experimento permitió contrastar el comportamiento de la red al aprender completamente desde cero frente al uso de representaciones previamente entrenadas en ImageNet.

A diferencia del modelo con transfer learning, en esta versión no se emplearon callbacks (EarlyStopping ni ReduceLROnPlateau), por lo que el entrenamiento se realizó de forma lineal durante 8 épocas. El modelo alcanzó una precisión del 81.93 % sobre el conjunto de prueba, frente al 90.22 % logrado por la versión con pesos preentrenados.

El desempeño más bajo se explica porque, al no disponer de pesos iniciales provenientes de ImageNet, la red debió aprender tanto características básicas (bordes, contrastes, texturas) como representaciones abstractas desde cero, lo que ralentiza la convergencia y requiere un número mucho mayor de épocas para alcanzar un rendimiento comparable. Si bien una red desde cero podría eventualmente superar el desempeño de una red preentrenada, hacerlo demanda tiempos de entrenamiento mucho más prolongados y recursos computacionales significativamente mayores, lo cual resulta poco práctico en entornos académicos o con hardware limitado.

Por tanto, el uso de transfer learning se consolida como una alternativa eficiente y rentable, ya que permite obtener resultados de alta precisión con un menor número de épocas, aprovechando conocimiento previamente adquirido sin necesidad de grandes volúmenes de datos ni tiempo de entrenamiento excesivo.

Discusión crítica: fortalezas, limitaciones y mejoras

Fortalezas

- **Eficiencia y rendimiento con transferencia:**

EfficientNet demostró un excelente rendimiento, alcanzando alta precisión con un número limitado de épocas de entrenamiento. El uso de transfer learning fue clave, ya que permitió aprovechar el conocimiento previo de redes preentrenadas, mejorando notablemente las métricas de desempeño sin necesidad de entrenar desde cero.

- **Importancia del preprocessamiento:**

Herramientas como CLAHE, la normalización y el data augmentation desempeñaron un papel fundamental. Su combinación permitió obtener un modelo robusto, eficaz y eficiente, capaz de generalizar mejor frente a variaciones en las imágenes.

- **Entrenamiento estable:**

El uso combinado de EarlyStopping y ReduceLROnPlateau evitó entrenamientos redundantes y ayudó a estabilizar la convergencia.

- **Estrategia de aumento de datos:**

La técnica de data augmentation es muy útil ante la limitación de datos reales, ya que genera imágenes sintéticas que enriquecen el conjunto de entrenamiento. Sin embargo, es importante ajustar cuidadosamente los parámetros en este caso, las radiografías estaban centradas, lo que permitió realizar pequeñas rotaciones, zoom y desplazamientos sin perder información relevante. En otros escenarios, una configuración inadecuada podría introducir ruido o distorsionar características críticas, afectando negativamente el aprendizaje del modelo.

Limitaciones

- Comparación desde cero no completamente equivalente

El modelo entrenado desde cero (81.93 %) no utilizó callbacks ni una programación adaptativa del learning rate. Por eso, la comparación con el modelo con transferencia de aprendizaje favorece ligeramente a este último en configuración.

- Sensibilidad al fine-tuning profundo

El descongelamiento de las últimas 20 capas redujo la precisión a 87.69 %. Esto indica que, para este conjunto de datos, un descongelamiento más superficial (entre 10 y 15 capas) o progresivo podría ser más apropiado.

- Prioridad clínica de métricas

Aunque la precisión global es alta, en aplicaciones médicas es importante incluir métricas adicionales como sensibilidad, especificidad o curvas ROC por clase, especialmente para COVID-19 y Lung Opacity, donde se observaron confusiones esperables en la matriz.

- Validación externa

El modelo se evaluó únicamente con una división interna 80/20. A futuro sería necesario probarlo con imágenes de otros hospitales o equipos para verificar su capacidad de generalización fuera del dominio original.

Mejoras inmediatas que se pueden realizar

- Comparación justa

Reentrenar el modelo desde cero aplicando `EarlyStopping` y `ReduceLROnPlateau`, con más épocas y, de ser necesario, usando `class weights` para balancear las clases. Esto permitiría una comparación más equitativa entre configuraciones.

- Descongelamiento gradual

Realizar el descongelamiento por etapas (por ejemplo, de 5 a 10 y luego a 15 capas), usando tasas de aprendizaje muy bajas —del orden de 5×10^{-6} — y esquemas de ajuste dinámico como *cosine decay* o *one-cycle*. Detener el proceso cuando la pérdida de validación deje de mejorar.

- Optimización bayesiana de hiperparámetros

Emplear optimización bayesiana (por ejemplo con KerasTuner u Optuna) para buscar automáticamente la mejor combinación de parámetros como el *learning rate*, el tamaño de lote, la tasa de *dropout*, el número de capas a descongelar y los valores de *patience* o *factor* del ReduceLROnPlateau . Esta técnica reemplaza búsquedas exhaustivas por una exploración más eficiente y guiada.

- Aumentos de datos más avanzados

Explorar técnicas como *MixUp*, *CutMix* o *RandomBrightness/Contrast* de baja intensidad. En radiografías estos aumentos deben aplicarse con cuidado para no distorsionar rasgos anatómicos importantes.

- Calibración de probabilidades

Aplicar métodos como *Temperature Scaling* o *Platt Scaling* para mejorar la confiabilidad de las probabilidades de salida, lo que puede ser relevante en contextos clínicos donde se establecen umbrales de decisión.

- Reporte clínico ampliado

Incluir métricas como AUC, sensibilidad y especificidad por clase, junto con un análisis del impacto de los errores (*falsos negativos* y *falsos positivos*) según su importancia clínica.

Consideraciones Adicionales

- Tiempo y costo del entrenamiento desde cero

Entrenar una EfficientNet completamente desde cero puede llegar a igualar o incluso superar el rendimiento de un modelo con transferencia, pero requiere muchas más épocas, una planificación cuidadosa de la tasa de aprendizaje y, preferiblemente, un conjunto de datos más grande. En este trabajo, el entrenamiento de ocho épocas tomó alrededor de 220 minutos y se stabilizó en una precisión del 81.93 %. Continuar el entrenamiento habría sido posible, pero el costo adicional en tiempo y recursos para cerrar la brecha con el modelo con transferencia sería demasiado alto para los objetivos del curso. Por ello, el *transfer learning* resultó ser la alternativa más práctica y costo-efectiva: permitió alcanzar un mejor rendimiento con menor riesgo de sobreajuste y un tiempo de desarrollo más razonable.

- Sobre el descongelamiento en una fase separada

El descongelamiento se realizó en una segunda fase para evaluar de forma controlada el impacto del *fine-tuning* y documentar sus resultados por separado. En un entorno de producción, podría integrarse todo el proceso en un único entrenamiento con una agenda de *learning rate*

por etapas —primero con la base congelada y luego con descongelamiento progresivo—. En este trabajo se priorizó la claridad y trazabilidad de los resultados por encima de la automatización.

Líneas futuras

- Validar el modelo con datos externos (otros hospitales o escáneres) y aplicar *Test-Time Augmentation* para mayor robustez.
- Implementar *Focal Loss* o *class weights* si se busca mejorar el rendimiento en clases minoritarias o con mayor costo clínico de error.
- Probar versiones superiores del modelo (EfficientNetB1 o B2), que podrían mejorar algunos puntos porcentuales de precisión manteniendo una buena eficiencia.