01_dclt_image_classification

April 28, 2025

`M9 · Deep Learning aplicada - Visión artificial (Intel Scenes)`

`Clasificación multiclase 150×150 · 6 etiquetas`

`Proyecto desarrollado por Diego Cesar Lerma Torres para IMF Smart Education`

Caso práctico del módulo **M9**
Deep learning aplicada: NLP y visión artificial
Del Master en Inteligencia Artificial

Este proyecto aplica técnicas avanzadas de **deep learning** para clasificar imágenes de escenas, combinando diversas áreas del conocimiento en inteligencia artificial.

1 Dataset

Intel Image Classification Dataset

Este conjunto de datos contiene imágenes de escenas naturales alrededor del mundo, capturadas en distintas condiciones y escenarios. El objetivo es clasificar cada imagen en una de seis categorías diferentes.

- **Número de imágenes**: Aproximadamente 25,000.
- **Dimensiones**: 150x150 píxeles por imagen.
- **Categorías**:
  - 0: Buildings (edificios)
  - 1: Forest (bosque)
  - 2: Glacier (glaciar)
  - 3: Mountain (montaña)
  - 4: Sea (mar)
  - 5: Street (calle)
- **Estructura del dataset**:
  - **Entrenamiento (Train)**: ~14,000 imágenes.
  - **Pruebas (Test)**: ~3,000 imágenes.
  - **Predicción (Prediction)**: ~7,000 imágenes.

Cada partición está disponible en archivos comprimidos independientes.

Fuente: El dataset fue inicialmente publicado en Analytics Vidhya por Intel como parte de un desafío de clasificación de imágenes.

Objetivo del proyecto:
El propósito principal de este conjunto de datos es servir de base para el entrenamiento de redes neuronales profundas (CNNs) capaces de clasificar escenas naturales con alta precisión, fortaleciendo habilidades en visión artificial y aprendizaje profundo.

> Agradecimientos a Intel y Analytics Vidhya por proporcionar este valioso recurso para la comunidad.

2 Configuración inicial

[50]:
```
# !pip install -r requirements.txt

# En local
%pip install -r ../requirements.txt
```

```
Requirement already satisfied: pandas in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from -r ../requirements.txt
(line 1)) (2.2.3)
Requirement already satisfied: numpy in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from -r ../requirements.txt
(line 2)) (2.1.3)
Requirement already satisfied: matplotlib in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from -r ../requirements.txt
(line 3)) (3.10.1)
Requirement already satisfied: scikit-learn in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from -r ../requirements.txt (line 4)) (1.6.1)
Requirement already satisfied: keras-tuner in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from -r ../requirements.txt (line 5)) (1.4.7)
Requirement already satisfied: jinja2 in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from -r ../requirements.txt
(line 7)) (3.1.6)
Requirement already satisfied: tensorflow[and-cuda] in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from -r ../requirements.txt (line 6)) (2.19.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from pandas->-r ../requirements.txt (line 1)) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from pandas->-r ../requirements.txt (line 1)) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from pandas->-r ../requirements.txt (line 1)) (2025.2)
Requirement already satisfied: contourpy>=1.0.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from matplotlib->-r ../requirements.txt (line 3)) (1.3.2)
```

```
Requirement already satisfied: cycler>=0.10 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from matplotlib->-r ../requirements.txt (line 3)) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from matplotlib->-r ../requirements.txt (line 3)) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from matplotlib->-r ../requirements.txt (line 3)) (1.4.8)
Requirement already satisfied: packaging>=20.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from matplotlib->-r ../requirements.txt (line 3)) (25.0)
Requirement already satisfied: pillow>=8 in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from matplotlib->-r
../requirements.txt (line 3)) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from matplotlib->-r ../requirements.txt (line 3)) (3.2.3)
Requirement already satisfied: scipy>=1.6.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from scikit-learn->-r ../requirements.txt (line 4)) (1.15.2)
Requirement already satisfied: joblib>=1.2.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from scikit-learn->-r ../requirements.txt (line 4)) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from scikit-learn->-r ../requirements.txt (line 4)) (3.6.0)
Requirement already satisfied: keras in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from keras-tuner->-r
../requirements.txt (line 5)) (3.9.2)
Requirement already satisfied: requests in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from keras-tuner->-r
../requirements.txt (line 5)) (2.32.3)
Requirement already satisfied: kt-legacy in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from keras-tuner->-r
../requirements.txt (line 5)) (1.0.5)
Requirement already satisfied: absl-py>=1.0.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (2.2.2)
Requirement already satisfied: astunparse>=1.6.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (0.6.0)
```

```
Requirement already satisfied: google-pasta>=0.1.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (3.4.0)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3
in /home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (5.29.4)
Requirement already satisfied: setuptools in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from tensorflow[and-cuda]->-r
../requirements.txt (line 6)) (80.0.0)
Requirement already satisfied: six>=1.12.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (3.0.1)
Requirement already satisfied: typing-extensions>=3.6.6 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (4.13.2)
Requirement already satisfied: wrapt>=1.11.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (1.71.0)
Requirement already satisfied: tensorboard~=2.19.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (2.19.0)
Requirement already satisfied: h5py>=3.11.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (3.13.0)
Requirement already satisfied: ml-dtypes<1.0.0,>=0.5.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (0.5.1)
Requirement already satisfied: nvidia-cublas-cu12==12.5.3.2 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (12.5.3.2)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.5.82 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (12.5.82)
Requirement already satisfied: nvidia-cuda-nvcc-cu12==12.5.82 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
```

```
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (12.5.82)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.5.82 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (12.5.82)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.5.82 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (12.5.82)
Requirement already satisfied: nvidia-cudnn-cu12==9.3.0.75 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (9.3.0.75)
Requirement already satisfied: nvidia-cufft-cu12==11.2.3.61 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6))
(11.2.3.61)
Requirement already satisfied: nvidia-curand-cu12==10.3.6.82 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6))
(10.3.6.82)
Requirement already satisfied: nvidia-cusolver-cu12==11.6.3.83 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6))
(11.6.3.83)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.1.3 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (12.5.1.3)
Requirement already satisfied: nvidia-nccl-cu12==2.23.4 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (2.23.4)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.5.82 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorflow[and-cuda]->-r ../requirements.txt (line 6)) (12.5.82)
Requirement already satisfied: charset-normalizer<4,>=2 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from requests->keras-tuner->-r ../requirements.txt (line 5)) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from requests->keras-tuner->-r ../requirements.txt (line 5)) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from requests->keras-tuner->-r ../requirements.txt (line 5)) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from requests->keras-tuner->-r ../requirements.txt (line 5))
(2025.4.26)
Requirement already satisfied: markdown>=2.6.8 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorboard~=2.19.0->tensorflow[and-cuda]->-r ../requirements.txt
(line 6)) (3.8)
```

```
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorboard~=2.19.0->tensorflow[and-cuda]->-r ../requirements.txt
(line 6)) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from tensorboard~=2.19.0->tensorflow[and-cuda]->-r ../requirements.txt
(line 6)) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from jinja2->-r ../requirements.txt (line 7)) (3.0.2)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from astunparse>=1.6.0->tensorflow[and-cuda]->-r ../requirements.txt
(line 6)) (0.45.1)
Requirement already satisfied: rich in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from keras->keras-tuner->-r
../requirements.txt (line 5)) (14.0.0)
Requirement already satisfied: namex in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from keras->keras-tuner->-r
../requirements.txt (line 5)) (0.0.9)
Requirement already satisfied: optree in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from keras->keras-tuner->-r
../requirements.txt (line 5)) (0.15.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from rich->keras->keras-tuner->-r ../requirements.txt (line 5))
(3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/home/diego/code/learning/ai/image-classification/.venv/lib/python3.12/site-
packages (from rich->keras->keras-tuner->-r ../requirements.txt (line 5))
(2.19.1)
Requirement already satisfied: mdurl~=0.1 in /home/diego/code/learning/ai/image-
classification/.venv/lib/python3.12/site-packages (from markdown-it-
py>=2.2.0->rich->keras->keras-tuner->-r ../requirements.txt (line 5)) (0.1.2)
Note: you may need to restart the kernel to use updated packages.
```

[ ]:
```
# Subir el kaggle.json, en caso de querer descargar directamente la base de␣
↪datos de su origen

#from google.colab import files
#files.upload()
```

[3]:
```
# Mover el archivo descargado a su lugar

#!mkdir -p ~/.kaggle
#!mv kaggle.json ~/.kaggle/
```

```
#!chmod 600 ~/.kaggle/kaggle.json
```

[51]:
```
import os
import json
import zipfile
import random
from pathlib import Path

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras import layers, models, regularizers, mixed_precision
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,␣
 ↪ReduceLROnPlateau
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import keras_tuner as kt

mixed_precision.set_global_policy('mixed_float16')

SEED               = 42
IMG_SIZE           = (150, 150)
BATCH_SIZE         = 64
EPOCHS             = 50
VAL_SPLIT          = 0.20
TEST_SPLIT         = 0.25
AUTOTUNE           = tf.data.AUTOTUNE
PROJECT_ROOT       = Path('../')
DATA_DIR           = PROJECT_ROOT/'data'
FULL_DATA  = DATA_DIR / 'data_all'
MODELS_DIR         = PROJECT_ROOT/'models'
HIST_DIR           = PROJECT_ROOT/'history'
for d in (DATA_DIR, FULL_DATA, MODELS_DIR, HIST_DIR):
    d.mkdir(parents=True, exist_ok=True)

random.seed(SEED);
np.random.seed(SEED);
tf.random.set_seed(SEED)
print(tf.__version__)
```

```
2.19.0
```

Ejecutar solo la primera vez. Descargar y descomprimir desde Kaggle

Las siguientes veces, esta sección debe comentarse u omitirse su ejecución

[ ]:
```
!kaggle datasets download -d puneet6060/intel-image-classification -p␣
↪{PROJECT_ROOT}
```

[6]:
```
with zipfile.ZipFile(PROJECT_ROOT/'intel-image-classification.zip') as z: z.
↪extractall(DATA_DIR)
```

3 1. Descarga y unificación de carpetas seg_train + seg_test

[14]:
```
"""# 1. Unificación de carpetas seg_train + seg_test"""

import shutil

def merge_folders_corrected(src_root: Path, dst_root: Path):
    """
    Fusiona las imágenes de las subcarpetas de clase encontradas dentro de
    src_root/seg_train/seg_train/<clase> y src_root/seg_test/seg_test/<clase>
    en dst_root/<clase>.
    Elimina las carpetas originales seg_train/ y seg_test/ de src_root después.
    """
    print(f"Iniciando fusión de carpetas desde {src_root} hacia {dst_root}")
    dst_root.mkdir(parents=True, exist_ok=True)

    split_names = ['seg_train', 'seg_test']

    for split in split_names:
        split_base_dir = src_root / split
        print(f"Procesando división: {split}")

        nested_split_dir = split_base_dir / split

        if nested_split_dir.exists() and nested_split_dir.is_dir():
            print(f"  Encontrada estructura anidada en: {nested_split_dir}")
            source_class_dir_parent = nested_split_dir
        elif split_base_dir.exists() and split_base_dir.is_dir():
             contains_class_dirs = any(item.is_dir() for item in split_base_dir.
↪iterdir())
             if contains_class_dirs:
                 print(f"  Estructura anidada no encontrada, usando directorio␣
↪base: {split_base_dir}")
                 source_class_dir_parent = split_base_dir
             else:
                 print(f"  Directorio {split_base_dir} no contiene␣
↪subdirectorios de clase esperados. Omitiendo.")
                 continue
        else:
```

```
            print(f"  Directorio base {split_base_dir} no encontrado o no es un␣
↪directorio. Omitiendo.")
            continue

        class_count = 0
        image_count = 0
        for class_dir in source_class_dir_parent.iterdir():
            if class_dir.is_dir():
                class_name = class_dir.name
                target_class_dir = dst_root / class_name
                target_class_dir.mkdir(parents=True, exist_ok=True)
                class_count += 1

                files_in_class = list(class_dir.iterdir())
                print(f"    Procesando clase '{class_name}'␣
↪({len(files_in_class)} archivos)...")
                for img_path in files_in_class:
                    if img_path.is_file():
                        dst_img_path = target_class_dir / img_path.name
                        try:
                            shutil.move(str(img_path), str(dst_img_path))
                            image_count += 1
                        except Exception as e:
                            print(f"      Error moviendo {img_path.name} a␣
↪{target_class_dir}: {e}")

        print(f"  Procesadas {class_count} clases y {image_count} imágenes para␣
↪la división '{split}'.")

    print("\nLimpiando carpetas originales...")
    for split in split_names:
        original_split_dir = src_root / split
        if original_split_dir.exists():
            try:
                shutil.rmtree(original_split_dir)
                print(f"  Eliminada carpeta: {original_split_dir}")
            except OSError as e:
                print(f"  Error eliminando {original_split_dir}: {e}")
        else:
            print(f"  La carpeta {original_split_dir} no existe, no se necesita␣
↪eliminar.")

    print(f"\nFusión completada. Datos unificados en: {dst_root}")
```

[ ]:
```
if DATA_DIR.exists():
    merge_folders_corrected(DATA_DIR, FULL_DATA)
else:
```

```
    print(f"ERROR: El directorio de origen {DATA_DIR} no existe. No se puede␣
 ↪ejecutar la fusión.")

if FULL_DATA.exists():
    print("\nContenido del directorio unificado (data_all):")
    class_names_merged = sorted([d.name for d in FULL_DATA.iterdir() if d.
 ↪is_dir()])
    print(f"Clases encontradas: {class_names_merged}")
    total_images = 0
    for class_dir in FULL_DATA.iterdir():
        if class_dir.is_dir():
            count = len(list(class_dir.glob('*.*')))
            print(f"- {class_dir.name}: {count} imágenes")
            total_images += count
    print(f"Total imágenes unificadas: {total_images}")
else:
    print(f"ERROR: El directorio destino {FULL_DATA} no se creó correctamente.")
```

4 2. Generadores de imágenes

[28]:
```
def build_dataset(directory, img_size=IMG_SIZE, batch_size=BATCH_SIZE,
                  test_split=TEST_SPLIT, seed=SEED):

    ds_full = tf.keras.utils.image_dataset_from_directory(
        directory,
        image_size=img_size,
        batch_size=batch_size,
        shuffle=True,
        seed=seed,
        label_mode='int')

    class_names = ds_full.class_names


    total      = ds_full.cardinality().numpy()
    test_count = int(total * test_split)

    test_ds    = ds_full.take(test_count)
    train_val  = ds_full.skip(test_count)

    val_count  = int(train_val.cardinality().numpy() * VAL_SPLIT)
    val_ds     = train_val.take(val_count)
    train_ds   = train_val.skip(val_count)


    train_ds = train_ds.cache().prefetch(AUTOTUNE)
```

```
    val_ds   = val_ds.cache().prefetch(AUTOTUNE)
    test_ds  = test_ds.cache().prefetch(AUTOTUNE)

    return train_ds, val_ds, test_ds, class_names
```

[29]:
```
train_ds, val_ds, test_ds, class_names = build_dataset(FULL_DATA)
NUM_CLASSES = len(class_names)
print(class_names)
```

```
Found 17034 files belonging to 6 classes.
['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
```

[30]:
```
for cls in sorted((FULL_DATA).iterdir()):
    print(f"{cls.name:<10}  {len(list(cls.glob('*'))):5d} imágenes")

# Debe mostrarse que existen varias imagenes repartidas en 6 carpetas. De lo␣
 ↪contrario, revisar antes de comenzar el entrenamiento.
```

```
buildings    2628 imágenes
forest       2745 imágenes
glacier      2957 imágenes
mountain     3037 imágenes
sea          2784 imágenes
street       2883 imágenes
```

5 3. Callbacks comunes

[52]:
```
def common_callbacks(name):
    return [
        EarlyStopping(monitor='val_loss',
                      patience=5,
                      restore_best_weights=True),

        ModelCheckpoint(MODELS_DIR/f'{name}.keras',
                        monitor='val_loss',
                        save_best_only=True),

        ReduceLROnPlateau(monitor='val_loss',
                          patience=2,
                          factor=0.3,
                          verbose=1)
    ]
```

[53]:
```
def get_optimizer(lr=1e-3):
    """Crea un optimizador Adam envuelto en LossScaleOptimizer."""
    base_opt = tf.keras.optimizers.Adam(learning_rate=lr)
    return mixed_precision.LossScaleOptimizer(base_opt)
```

6 4. Modelo 1 — CNN Base

[33]:
```python
def cnn_base(input_shape=IMG_SIZE+(3,), num_classes=NUM_CLASSES):
    model = tf.keras.Sequential([
        tf.keras.Input(shape=input_shape),
        layers.Rescaling(1./255),
        layers.Conv2D(32, 3, padding='same', use_bias=False),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.MaxPooling2D(),
        layers.GlobalAveragePooling2D(),
        layers.Dense(64, activation='relu'),
        layers.Dense(num_classes, activation='softmax')
    ], name='cnn_base')

    model.compile(optimizer=get_optimizer(1e-3),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

[34]:
```python
print("\n--- Entrenando Modelo Base ---")
base_model = cnn_base()
base_model.summary()

history_base = base_model.fit(
    train_ds,
    epochs=EPOCHS,
    validation_data=val_ds,
    callbacks=common_callbacks('base_model')
)

# Guardar historial
print("Guardando historial de Modelo Base v2...")
json.dump(history_base.history, open(HIST_DIR/'base_model.json','w'))
```

```
--- Entrenando Modelo Base ---
```

Model: "cnn_base"

Layer (type)	Output Shape	Param #
rescaling_3 (Rescaling)	(None, 150, 150, 3)	0
conv2d_13 (Conv2D)	(None, 150, 150, 32)	864
batch_normalization_9	(None, 150, 150, 32)	128

(BatchNormalization)		
activation_6 (Activation)	(None, 150, 150, 32)	0
max_pooling2d_7 (MaxPooling2D)	(None, 75, 75, 32)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 32)	0
dense_6 (Dense)	(None, 64)	2,112
dense_7 (Dense)	(None, 6)	390

Total params: 3,494 (13.65 KB)

Trainable params: 3,430 (13.40 KB)

Non-trainable params: 64 (256.00 B)

```
Epoch 1/50
161/161            6s 19ms/step -
accuracy: 0.3829 - loss: 1.5492 - val_accuracy: 0.2289 - val_loss: 1.6952 -
learning_rate: 0.0010
Epoch 2/50
161/161            1s 8ms/step -
accuracy: 0.5488 - loss: 1.1839 - val_accuracy: 0.3965 - val_loss: 1.4517 -
learning_rate: 0.0010
Epoch 3/50
161/161            1s 8ms/step -
accuracy: 0.5994 - loss: 1.0603 - val_accuracy: 0.5152 - val_loss: 1.1258 -
learning_rate: 0.0010
Epoch 4/50
161/161            1s 8ms/step -
accuracy: 0.6252 - loss: 0.9870 - val_accuracy: 0.5527 - val_loss: 1.0711 -
learning_rate: 0.0010
Epoch 5/50
161/161            1s 8ms/step -
accuracy: 0.6451 - loss: 0.9397 - val_accuracy: 0.5301 - val_loss: 1.1670 -
learning_rate: 0.0010
Epoch 6/50
156/161            0s 7ms/step -
accuracy: 0.6575 - loss: 0.9051
Epoch 6: ReduceLROnPlateau reducing learning rate to 0.0003000000142492354.
161/161            1s 8ms/step -
accuracy: 0.6579 - loss: 0.9046 - val_accuracy: 0.5258 - val_loss: 1.2007 -
```

```
learning_rate: 0.0010
Epoch 7/50
161/161             1s 8ms/step -
accuracy: 0.6768 - loss: 0.8675 - val_accuracy: 0.6734 - val_loss: 0.8536 -
learning_rate: 3.0000e-04
Epoch 8/50
161/161             1s 8ms/step -
accuracy: 0.6858 - loss: 0.8540 - val_accuracy: 0.6754 - val_loss: 0.8492 -
learning_rate: 3.0000e-04
Epoch 9/50
161/161             1s 8ms/step -
accuracy: 0.6893 - loss: 0.8470 - val_accuracy: 0.6770 - val_loss: 0.8468 -
learning_rate: 3.0000e-04
Epoch 10/50
161/161             1s 9ms/step -
accuracy: 0.6916 - loss: 0.8408 - val_accuracy: 0.6734 - val_loss: 0.8459 -
learning_rate: 3.0000e-04
Epoch 11/50
161/161             1s 8ms/step -
accuracy: 0.6936 - loss: 0.8354 - val_accuracy: 0.6770 - val_loss: 0.8413 -
learning_rate: 3.0000e-04
Epoch 12/50
161/161             1s 8ms/step -
accuracy: 0.6938 - loss: 0.8304 - val_accuracy: 0.6797 - val_loss: 0.8361 -
learning_rate: 3.0000e-04
Epoch 13/50
161/161             1s 8ms/step -
accuracy: 0.6960 - loss: 0.8259 - val_accuracy: 0.6820 - val_loss: 0.8330 -
learning_rate: 3.0000e-04
Epoch 14/50
161/161             1s 8ms/step -
accuracy: 0.6989 - loss: 0.8217 - val_accuracy: 0.6832 - val_loss: 0.8308 -
learning_rate: 3.0000e-04
Epoch 15/50
161/161             1s 8ms/step -
accuracy: 0.6994 - loss: 0.8178 - val_accuracy: 0.6828 - val_loss: 0.8293 -
learning_rate: 3.0000e-04
Epoch 16/50
161/161             3s 20ms/step -
accuracy: 0.7009 - loss: 0.8142 - val_accuracy: 0.6848 - val_loss: 0.8301 -
learning_rate: 3.0000e-04
Epoch 17/50
161/161             1s 8ms/step -
accuracy: 0.7022 - loss: 0.8108 - val_accuracy: 0.6812 - val_loss: 0.8256 -
learning_rate: 3.0000e-04
Epoch 18/50
161/161             1s 8ms/step -
accuracy: 0.7039 - loss: 0.8077 - val_accuracy: 0.6859 - val_loss: 0.8228 -
```

```
learning_rate: 3.0000e-04
Epoch 19/50
161/161             1s 8ms/step -
accuracy: 0.7054 - loss: 0.8048 - val_accuracy: 0.6824 - val_loss: 0.8231 -
learning_rate: 3.0000e-04
Epoch 20/50
161/161             1s 8ms/step -
accuracy: 0.7056 - loss: 0.8020 - val_accuracy: 0.6863 - val_loss: 0.8211 -
learning_rate: 3.0000e-04
Epoch 21/50
161/161             1s 8ms/step -
accuracy: 0.7071 - loss: 0.7994 - val_accuracy: 0.6859 - val_loss: 0.8188 -
learning_rate: 3.0000e-04
Epoch 22/50
161/161             1s 8ms/step -
accuracy: 0.7070 - loss: 0.7970 - val_accuracy: 0.6918 - val_loss: 0.8126 -
learning_rate: 3.0000e-04
Epoch 23/50
161/161             1s 8ms/step -
accuracy: 0.7081 - loss: 0.7945 - val_accuracy: 0.6926 - val_loss: 0.8116 -
learning_rate: 3.0000e-04
Epoch 24/50
161/161             1s 8ms/step -
accuracy: 0.7075 - loss: 0.7923 - val_accuracy: 0.6914 - val_loss: 0.8104 -
learning_rate: 3.0000e-04
Epoch 25/50
161/161             1s 8ms/step -
accuracy: 0.7090 - loss: 0.7904 - val_accuracy: 0.6898 - val_loss: 0.8147 -
learning_rate: 3.0000e-04
Epoch 26/50
156/161             0s 7ms/step -
accuracy: 0.7092 - loss: 0.7885
Epoch 26: ReduceLROnPlateau reducing learning rate to 9.000000427477062e-05.
161/161             1s 8ms/step -
accuracy: 0.7094 - loss: 0.7884 - val_accuracy: 0.6887 - val_loss: 0.8151 -
learning_rate: 3.0000e-04
Epoch 27/50
161/161             1s 8ms/step -
accuracy: 0.7137 - loss: 0.7822 - val_accuracy: 0.7152 - val_loss: 0.7710 -
learning_rate: 9.0000e-05
Epoch 28/50
161/161             1s 8ms/step -
accuracy: 0.7141 - loss: 0.7784 - val_accuracy: 0.7160 - val_loss: 0.7701 -
learning_rate: 9.0000e-05
Epoch 29/50
161/161             1s 8ms/step -
accuracy: 0.7134 - loss: 0.7777 - val_accuracy: 0.7168 - val_loss: 0.7705 -
learning_rate: 9.0000e-05
```

```
Epoch 30/50
161/161             1s 8ms/step -
accuracy: 0.7134 - loss: 0.7770 - val_accuracy: 0.7172 - val_loss: 0.7699 -
learning_rate: 9.0000e-05
Epoch 31/50
161/161             1s 8ms/step -
accuracy: 0.7142 - loss: 0.7764 - val_accuracy: 0.7168 - val_loss: 0.7699 -
learning_rate: 9.0000e-05
Epoch 32/50
161/161             1s 8ms/step -
accuracy: 0.7143 - loss: 0.7756 - val_accuracy: 0.7180 - val_loss: 0.7695 -
learning_rate: 9.0000e-05
Epoch 33/50
161/161             1s 8ms/step -
accuracy: 0.7142 - loss: 0.7751 - val_accuracy: 0.7184 - val_loss: 0.7690 -
learning_rate: 9.0000e-05
Epoch 34/50
161/161             1s 8ms/step -
accuracy: 0.7149 - loss: 0.7745 - val_accuracy: 0.7188 - val_loss: 0.7686 -
learning_rate: 9.0000e-05
Epoch 35/50
161/161             1s 9ms/step -
accuracy: 0.7149 - loss: 0.7739 - val_accuracy: 0.7180 - val_loss: 0.7679 -
learning_rate: 9.0000e-05
Epoch 36/50
161/161             2s 10ms/step -
accuracy: 0.7153 - loss: 0.7733 - val_accuracy: 0.7188 - val_loss: 0.7677 -
learning_rate: 9.0000e-05
Epoch 37/50
161/161             2s 10ms/step -
accuracy: 0.7159 - loss: 0.7727 - val_accuracy: 0.7188 - val_loss: 0.7670 -
learning_rate: 9.0000e-05
Epoch 38/50
161/161             1s 8ms/step -
accuracy: 0.7161 - loss: 0.7722 - val_accuracy: 0.7188 - val_loss: 0.7669 -
learning_rate: 9.0000e-05
Epoch 39/50
161/161             3s 21ms/step -
accuracy: 0.7162 - loss: 0.7716 - val_accuracy: 0.7180 - val_loss: 0.7663 -
learning_rate: 9.0000e-05
Epoch 40/50
161/161             1s 8ms/step -
accuracy: 0.7163 - loss: 0.7711 - val_accuracy: 0.7180 - val_loss: 0.7663 -
learning_rate: 9.0000e-05
Epoch 41/50
161/161             1s 8ms/step -
accuracy: 0.7167 - loss: 0.7705 - val_accuracy: 0.7188 - val_loss: 0.7651 -
learning_rate: 9.0000e-05
```

```
Epoch 42/50
161/161             1s 9ms/step -
accuracy: 0.7166 - loss: 0.7700 - val_accuracy: 0.7191 - val_loss: 0.7650 -
learning_rate: 9.0000e-05
Epoch 43/50
161/161             2s 10ms/step -
accuracy: 0.7169 - loss: 0.7695 - val_accuracy: 0.7188 - val_loss: 0.7644 -
learning_rate: 9.0000e-05
Epoch 44/50
161/161             2s 10ms/step -
accuracy: 0.7170 - loss: 0.7689 - val_accuracy: 0.7184 - val_loss: 0.7641 -
learning_rate: 9.0000e-05
Epoch 45/50
161/161             2s 11ms/step -
accuracy: 0.7168 - loss: 0.7684 - val_accuracy: 0.7180 - val_loss: 0.7639 -
learning_rate: 9.0000e-05
Epoch 46/50
161/161             2s 11ms/step -
accuracy: 0.7176 - loss: 0.7679 - val_accuracy: 0.7203 - val_loss: 0.7633 -
learning_rate: 9.0000e-05
Epoch 47/50
161/161             2s 12ms/step -
accuracy: 0.7176 - loss: 0.7673 - val_accuracy: 0.7195 - val_loss: 0.7633 -
learning_rate: 9.0000e-05
Epoch 48/50
161/161             2s 12ms/step -
accuracy: 0.7176 - loss: 0.7668 - val_accuracy: 0.7195 - val_loss: 0.7623 -
learning_rate: 9.0000e-05
Epoch 49/50
161/161             2s 11ms/step -
accuracy: 0.7184 - loss: 0.7663 - val_accuracy: 0.7195 - val_loss: 0.7617 -
learning_rate: 9.0000e-05
Epoch 50/50
161/161             2s 11ms/step -
accuracy: 0.7187 - loss: 0.7658 - val_accuracy: 0.7207 - val_loss: 0.7616 -
learning_rate: 9.0000e-05
Guardando historial de Modelo Base v2…
```

7 5. Modelo 2 — CNN Avanzada (más profundidad + Dropout + BatchNorm)

[38]:
```python
def cnn_advanced(input_shape=IMG_SIZE+(3,), num_classes=NUM_CLASSES):
    """
    Versión 3: Filtros [32, 64, 64], Orden Conv->BN->Activation
    """
    inputs = layers.Input(shape=input_shape)
```

```
    x = layers.Rescaling(1./255)(inputs)

    for filters in [32, 64, 64]:
        x = layers.Conv2D(filters, 3, padding='same', use_bias=False)(x)
        x = layers.BatchNormalization()(x)
        x = layers.Activation('relu')(x)

        x = layers.Conv2D(filters, 3, padding='same', use_bias=False)(x)
        x = layers.BatchNormalization()(x)
        x = layers.Activation('relu')(x)

        x = layers.MaxPooling2D()(x)
        x = layers.Dropout(0.20)(x)

    x = layers.Flatten()(x)
    x = layers.Dense(256, activation='relu')(x)
    x = layers.Dropout(0.40)(x)
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = tf.keras.Model(inputs, outputs, name='cnn_advanced')

    model.compile(optimizer=get_optimizer(5e-4),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

[40]:
```
print("\n--- Creando y Verificando CNN Advanced ---")
adv_model_instance = cnn_advanced()

print("--- Model Summary ---")
adv_model_instance.summary(line_length=100)

callbacks = [tf.keras.callbacks.TerminateOnNaN()] +␣
 ↪common_callbacks('advance_model')


print("\n--- Iniciando entrenamiento CNN Advanced ---")
history_adv = adv_model_instance.fit(
    train_ds,
    epochs=EPOCHS,
    validation_data=val_ds,
    callbacks=callbacks
)

print("Guardando historial de CNN Avanzado...")
json.dump(history_adv.history, open(HIST_DIR/'advance_model.json','w'))
```

```
--- Creando y Verificando CNN Advanced ---
--- Model Summary ---
Model: "cnn_advanced"
```

Layer (type)	Output Shape	Param #
input_layer_5 (InputLayer)	(None, 150, 150, 3)	0
cast_42 (Cast)	(None, 150, 150, 3)	0
rescaling_5 (Rescaling)	(None, 150, 150, 3)	0
conv2d_20 (Conv2D)	(None, 150, 150, 32)	864
batch_normalization_16 (BatchNormalization)	(None, 150, 150, 32)	128
activation_13 (Activation)	(None, 150, 150, 32)	0
conv2d_21 (Conv2D)	(None, 150, 150, 32)	9,216
batch_normalization_17 (BatchNormalization)	(None, 150, 150, 32)	128
activation_14 (Activation)	(None, 150, 150, 32)	0
max_pooling2d_11 (MaxPooling2D)	(None, 75, 75, 32)	0
dropout_12 (Dropout)	(None, 75, 75, 32)	0

```
conv2d_22 (Conv2D)                    (None, 75, 75, 64)                ␣
↪         18,432

batch_normalization_18                (None, 75, 75, 64)                ␣
↪            256
(BatchNormalization)                                                    ␣
↪

activation_15 (Activation)            (None, 75, 75, 64)                ␣
↪            0

conv2d_23 (Conv2D)                    (None, 75, 75, 64)                ␣
↪         36,864

batch_normalization_19                (None, 75, 75, 64)                ␣
↪            256
(BatchNormalization)                                                    ␣
↪

activation_16 (Activation)            (None, 75, 75, 64)                ␣
↪            0

max_pooling2d_12 (MaxPooling2D)       (None, 37, 37, 64)                ␣
↪            0

dropout_13 (Dropout)                  (None, 37, 37, 64)                ␣
↪            0

conv2d_24 (Conv2D)                    (None, 37, 37, 64)                ␣
↪         36,864

batch_normalization_20                (None, 37, 37, 64)                ␣
↪            256
(BatchNormalization)                                                    ␣
↪

activation_17 (Activation)            (None, 37, 37, 64)                ␣
↪            0

conv2d_25 (Conv2D)                    (None, 37, 37, 64)                ␣
↪         36,864

batch_normalization_21                (None, 37, 37, 64)                ␣
↪            256
(BatchNormalization)                                                    ␣
↪
```

```
 activation_18 (Activation)                  (None, 37, 37, 64)                ␣
↪              0

 max_pooling2d_13 (MaxPooling2D)             (None, 18, 18, 64)                ␣
↪              0

 dropout_14 (Dropout)                        (None, 18, 18, 64)                ␣
↪              0

 flatten_4 (Flatten)                         (None, 20736)                     ␣
↪              0

 dense_10 (Dense)                            (None, 256)                       ␣
↪      5,308,672

 dropout_15 (Dropout)                        (None, 256)                       ␣
↪              0

 dense_11 (Dense)                            (None, 6)                         ␣
↪          1,542


 Total params: 5,450,598 (20.79 MB)

 Trainable params: 5,449,958 (20.79 MB)

 Non-trainable params: 640 (2.50 KB)


--- Iniciando entrenamiento CNN Advanced ---
Epoch 1/50
161/161             0s 47ms/step -
accuracy: 0.2433 - loss: 8.4697
2025-04-28 17:30:33.876186: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_183', 248 bytes spill stores, 248 bytes spill loads

161/161             14s 57ms/step -
accuracy: 0.2439 - loss: 8.4543 - val_accuracy: 0.2680 - val_loss: 2.0326 -
learning_rate: 5.0000e-04
Epoch 2/50
Epoch 2/50
```

```
161/161            3s 21ms/step -
accuracy: 0.5747 - loss: 1.2600 - val_accuracy: 0.2816 - val_loss: 1.9575 -
learning_rate: 5.0000e-04
Epoch 3/50
161/161            3s 20ms/step -
accuracy: 0.6521 - loss: 0.9817 - val_accuracy: 0.3648 - val_loss: 1.8716 -
learning_rate: 5.0000e-04
Epoch 4/50
161/161            3s 21ms/step -
accuracy: 0.7072 - loss: 0.8006 - val_accuracy: 0.6164 - val_loss: 1.0116 -
learning_rate: 5.0000e-04
Epoch 5/50
161/161            3s 21ms/step -
accuracy: 0.7538 - loss: 0.7030 - val_accuracy: 0.6395 - val_loss: 0.9872 -
learning_rate: 5.0000e-04
Epoch 6/50
161/161            3s 21ms/step -
accuracy: 0.7763 - loss: 0.6337 - val_accuracy: 0.7648 - val_loss: 0.7041 -
learning_rate: 5.0000e-04
Epoch 7/50
161/161            3s 20ms/step -
accuracy: 0.8043 - loss: 0.5713 - val_accuracy: 0.7492 - val_loss: 0.7491 -
learning_rate: 5.0000e-04
Epoch 8/50
161/161            3s 21ms/step -
accuracy: 0.8106 - loss: 0.5429 - val_accuracy: 0.7953 - val_loss: 0.6192 -
learning_rate: 5.0000e-04
Epoch 9/50
161/161            5s 32ms/step -
accuracy: 0.8292 - loss: 0.4781 - val_accuracy: 0.8109 - val_loss: 0.5614 -
learning_rate: 5.0000e-04
Epoch 10/50
161/161            3s 19ms/step -
accuracy: 0.8341 - loss: 0.4549 - val_accuracy: 0.7867 - val_loss: 0.6452 -
learning_rate: 5.0000e-04
Epoch 11/50
158/161            0s 17ms/step -
accuracy: 0.8502 - loss: 0.4196
Epoch 11: ReduceLROnPlateau reducing learning rate to 0.0001501000071246177.
161/161            3s 18ms/step -
accuracy: 0.8501 - loss: 0.4199 - val_accuracy: 0.7977 - val_loss: 0.5961 -
learning_rate: 5.0000e-04
Epoch 12/50
161/161            3s 18ms/step -
accuracy: 0.8696 - loss: 0.3652 - val_accuracy: 0.8516 - val_loss: 0.4441 -
learning_rate: 1.5000e-04
Epoch 13/50
161/161            3s 18ms/step -
```

```
accuracy: 0.8824 - loss: 0.3226 - val_accuracy: 0.8527 - val_loss: 0.4334 -
learning_rate: 1.5000e-04
Epoch 14/50
161/161             3s 17ms/step -
accuracy: 0.8900 - loss: 0.2999 - val_accuracy: 0.8445 - val_loss: 0.4754 -
learning_rate: 1.5000e-04
Epoch 15/50
161/161             0s 16ms/step -
accuracy: 0.8984 - loss: 0.2842
Epoch 15: ReduceLROnPlateau reducing learning rate to 4.500000213738531e-05.
161/161             3s 17ms/step -
accuracy: 0.8984 - loss: 0.2842 - val_accuracy: 0.8488 - val_loss: 0.4456 -
learning_rate: 1.5000e-04
Epoch 16/50
161/161             3s 17ms/step -
accuracy: 0.9004 - loss: 0.2635 - val_accuracy: 0.8441 - val_loss: 0.4743 -
learning_rate: 4.5000e-05
Epoch 17/50
159/161             0s 16ms/step -
accuracy: 0.9106 - loss: 0.2491
Epoch 17: ReduceLROnPlateau reducing learning rate to 1.3500000204658135e-05.
161/161             3s 17ms/step -
accuracy: 0.9106 - loss: 0.2491 - val_accuracy: 0.8406 - val_loss: 0.4830 -
learning_rate: 4.5000e-05
Epoch 18/50
161/161             3s 17ms/step -
accuracy: 0.9157 - loss: 0.2342 - val_accuracy: 0.8496 - val_loss: 0.4638 -
learning_rate: 1.3500e-05
Guardando historial de CNN Avanzado…
```

8 6. Modelo 3 — Hyperparameter Tuning (Keras Tuner)

[58]:
```python
def model_builder(hp):
    """Define el modelo basado en la arquitectura funcional."""

    hp_filters = hp.Choice('filters', values=[32, 48, 64])
    hp_activation = hp.Choice('activation', values=['relu', 'swish'])
    hp_pooling = hp.Choice('pooling', ['flatten', 'global_avg'])
    hp_dense_units = hp.Int('dense_units', min_value=64, max_value=256, step=64)
    hp_dropout_conv = hp.Float('dropout_conv', min_value=0.1, max_value=0.3,␣
 ↪step=0.05)
    hp_dropout_dense = hp.Float('dropout_dense', min_value=0.2, max_value=0.5,␣
 ↪step=0.1)
    hp_l2 = hp.Float('l2', min_value=1e-6, max_value=1e-4, sampling='log')
    hp_lr = hp.Choice('learning_rate', values=[1e-3, 5e-4, 1e-4])
```

```
    inputs = layers.Input(shape=IMG_SIZE + (3,))
    x = layers.Rescaling(1./255)(inputs)

    filter_progression = [hp_filters, hp_filters * 2, hp_filters * 4]

    for filters in filter_progression:
        x = layers.Conv2D(filters, 3, padding='same', use_bias=False,
                          kernel_regularizer=tf.keras.regularizers.l2(hp_l2))(x)
        x = layers.BatchNormalization()(x)
        x = layers.Activation(hp_activation)(x)

        x = layers.Conv2D(filters, 3, padding='same', use_bias=False,
                          kernel_regularizer=tf.keras.regularizers.l2(hp_l2))(x)
        x = layers.BatchNormalization()(x)
        x = layers.Activation(hp_activation)(x)

        x = layers.MaxPooling2D()(x)
        x = layers.Dropout(hp_dropout_conv)(x)

    if hp_pooling == 'flatten':
        x = layers.Flatten()(x)
    else:
        x = layers.GlobalAveragePooling2D()(x)

    x = layers.Dense(hp_dense_units, activation=hp_activation,
                     kernel_regularizer=regularizers.l2(hp_l2))(x)
    x = layers.Dropout(hp_dropout_dense)(x)
    outputs = layers.Dense(NUM_CLASSES, activation='softmax')(x)

    model = tf.keras.Model(inputs, outputs)

    optimizer = tf.keras.optimizers.Adam(learning_rate=hp_lr) # Mantener Adam␣
↪por ahora

    optimizer = mixed_precision.LossScaleOptimizer(optimizer)

    model.compile(optimizer=optimizer,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

[60]:
```
tuner = kt.Hyperband(
    model_builder,
    objective='val_accuracy',
    max_epochs=20,
    factor=3,
    hyperband_iterations=1,
```

```
    directory='kt_logs_hyperband',
    project_name='intel_hp_hyperband_v2',
    overwrite=True)

search_callbacks = [
    EarlyStopping(monitor='val_accuracy', patience=3)
]

print(f"Iniciando búsqueda de hiperparámetros con Hyperband (max_epochs=20)...")
tuner.search(train_ds,
             validation_data=val_ds,
             callbacks=search_callbacks)

print("\nBúsqueda completada. Obteniendo y re-entrenando el mejor modelo...")

best_hp = tuner.get_best_hyperparameters(1)[0]
print("Mejores Hiperparámetros encontrados:")
for param, value in best_hp.values.items():
    print(f"- {param}: {value}")

best_hp_model = tuner.hypermodel.build(best_hp)
best_hp_model.summary()

best_hp_model.save(MODELS_DIR / 'hp_best_structure_untrained.keras',␣
 ↪include_optimizer=False)

print("\n--- Iniciando entrenamiento FINAL del mejor modelo HP ---")

final_history_hp = best_hp_model.fit(
    train_ds,
    epochs=EPOCHS,
    validation_data=val_ds,

    callbacks=common_callbacks('hp')
)

print("Guardando historial y modelo final HP...")
json.dump(final_history_hp.history, open(HIST_DIR / 'hp.json', 'w'))
best_hp_model.save(MODELS_DIR / 'hp.keras')
```

```
Trial 30 Complete [00h 01m 16s]
val_accuracy: 0.806640625

Best val_accuracy So Far: 0.841796875
Total elapsed time: 00h 27m 34s

Búsqueda completada. Obteniendo y re-entrenando el mejor modelo…
Mejores Hiperparámetros encontrados:
```

- filters: 48
- activation: swish
- pooling: global_avg
- dense_units: 64
- dropout_conv: 0.15000000000000002
- dropout_dense: 0.2
- l2: 3.5557471228176156e-06
- learning_rate: 0.0005
- tuner/epochs: 20
- tuner/initial_epoch: 7
- tuner/bracket: 1
- tuner/round: 1
- tuner/trial_id: 0020

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 150, 150, 3)	0
rescaling_1 (Rescaling)	(None, 150, 150, 3)	0
conv2d_6 (Conv2D)	(None, 150, 150, 48)	1,296
batch_normalization_6 (BatchNormalization)	(None, 150, 150, 48)	192
activation_6 (Activation)	(None, 150, 150, 48)	0
conv2d_7 (Conv2D)	(None, 150, 150, 48)	20,736
batch_normalization_7 (BatchNormalization)	(None, 150, 150, 48)	192
activation_7 (Activation)	(None, 150, 150, 48)	0
max_pooling2d_3 (MaxPooling2D)	(None, 75, 75, 48)	0
dropout_4 (Dropout)	(None, 75, 75, 48)	0
conv2d_8 (Conv2D)	(None, 75, 75, 96)	41,472
batch_normalization_8 (BatchNormalization)	(None, 75, 75, 96)	384
activation_8 (Activation)	(None, 75, 75, 96)	0

conv2d_9 (Conv2D)	(None, 75, 75, 96)	82,944
batch_normalization_9 (BatchNormalization)	(None, 75, 75, 96)	384
activation_9 (Activation)	(None, 75, 75, 96)	0
max_pooling2d_4 (MaxPooling2D)	(None, 37, 37, 96)	0
dropout_5 (Dropout)	(None, 37, 37, 96)	0
conv2d_10 (Conv2D)	(None, 37, 37, 192)	165,888
batch_normalization_10 (BatchNormalization)	(None, 37, 37, 192)	768
activation_10 (Activation)	(None, 37, 37, 192)	0
conv2d_11 (Conv2D)	(None, 37, 37, 192)	331,776
batch_normalization_11 (BatchNormalization)	(None, 37, 37, 192)	768
activation_11 (Activation)	(None, 37, 37, 192)	0
max_pooling2d_5 (MaxPooling2D)	(None, 18, 18, 192)	0
dropout_6 (Dropout)	(None, 18, 18, 192)	0
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 192)	0
dense_2 (Dense)	(None, 64)	12,352
dropout_7 (Dropout)	(None, 64)	0
dense_3 (Dense)	(None, 6)	390

Total params: 659,542 (2.52 MB)

Trainable params: 658,198 (2.51 MB)

Non-trainable params: 1,344 (5.25 KB)

```
--- Iniciando entrenamiento FINAL del mejor modelo HP ---
Epoch 1/50
161/161             17s 73ms/step -
accuracy: 0.5666 - loss: 1.0997 - val_accuracy: 0.1984 - val_loss: 3.2733 -
learning_rate: 5.0000e-04
Epoch 2/50
161/161             8s 50ms/step -
accuracy: 0.7330 - loss: 0.6973 - val_accuracy: 0.4625 - val_loss: 1.6967 -
learning_rate: 5.0000e-04
Epoch 3/50
161/161             8s 51ms/step -
accuracy: 0.7834 - loss: 0.5854 - val_accuracy: 0.7270 - val_loss: 0.7353 -
learning_rate: 5.0000e-04
Epoch 4/50
161/161             8s 49ms/step -
accuracy: 0.8162 - loss: 0.5172 - val_accuracy: 0.7949 - val_loss: 0.5705 -
learning_rate: 5.0000e-04
Epoch 5/50
161/161             8s 47ms/step -
accuracy: 0.8301 - loss: 0.4690 - val_accuracy: 0.8207 - val_loss: 0.4828 -
learning_rate: 5.0000e-04
Epoch 6/50
161/161             8s 47ms/step -
accuracy: 0.8410 - loss: 0.4354 - val_accuracy: 0.8258 - val_loss: 0.4755 -
learning_rate: 5.0000e-04
Epoch 7/50
161/161             8s 52ms/step -
accuracy: 0.8508 - loss: 0.4088 - val_accuracy: 0.8227 - val_loss: 0.4737 -
learning_rate: 5.0000e-04
Epoch 8/50
161/161             8s 50ms/step -
accuracy: 0.8603 - loss: 0.3875 - val_accuracy: 0.8328 - val_loss: 0.4718 -
learning_rate: 5.0000e-04
Epoch 9/50
161/161             8s 47ms/step -
accuracy: 0.8649 - loss: 0.3786 - val_accuracy: 0.7969 - val_loss: 0.5546 -
learning_rate: 5.0000e-04
Epoch 10/50
159/161             0s 45ms/step -
accuracy: 0.8729 - loss: 0.3612
Epoch 10: ReduceLROnPlateau reducing learning rate to 0.0001500000071246177.
161/161             8s 48ms/step -
accuracy: 0.8729 - loss: 0.3612 - val_accuracy: 0.7969 - val_loss: 0.6024 -
learning_rate: 5.0000e-04
Epoch 11/50
161/161             8s 47ms/step -
accuracy: 0.8811 - loss: 0.3308 - val_accuracy: 0.8449 - val_loss: 0.4251 -
learning_rate: 1.5000e-04
```

```
Epoch 12/50
161/161             8s 49ms/step -
accuracy: 0.8955 - loss: 0.2943 - val_accuracy: 0.8445 - val_loss: 0.4240 -
learning_rate: 1.5000e-04
Epoch 13/50
161/161             8s 50ms/step -
accuracy: 0.8967 - loss: 0.2868 - val_accuracy: 0.8566 - val_loss: 0.4030 -
learning_rate: 1.5000e-04
Epoch 14/50
161/161             8s 49ms/step -
accuracy: 0.9024 - loss: 0.2756 - val_accuracy: 0.8641 - val_loss: 0.3871 -
learning_rate: 1.5000e-04
Epoch 15/50
161/161             8s 48ms/step -
accuracy: 0.9020 - loss: 0.2707 - val_accuracy: 0.8559 - val_loss: 0.4134 -
learning_rate: 1.5000e-04
Epoch 16/50
160/161             0s 44ms/step -
accuracy: 0.9066 - loss: 0.2663
Epoch 16: ReduceLROnPlateau reducing learning rate to 4.500000213738531e-05.
161/161             8s 47ms/step -
accuracy: 0.9066 - loss: 0.2663 - val_accuracy: 0.8520 - val_loss: 0.4214 -
learning_rate: 1.5000e-04
Epoch 17/50
161/161             8s 50ms/step -
accuracy: 0.9106 - loss: 0.2495 - val_accuracy: 0.8859 - val_loss: 0.3251 -
learning_rate: 4.5000e-05
Epoch 18/50
161/161             8s 52ms/step -
accuracy: 0.9187 - loss: 0.2373 - val_accuracy: 0.8879 - val_loss: 0.3191 -
learning_rate: 4.5000e-05
Epoch 19/50
161/161             8s 48ms/step -
accuracy: 0.9162 - loss: 0.2298 - val_accuracy: 0.8902 - val_loss: 0.3192 -
learning_rate: 4.5000e-05
Epoch 20/50
160/161             0s 49ms/step -
accuracy: 0.9162 - loss: 0.2331
Epoch 20: ReduceLROnPlateau reducing learning rate to 1.3500000204658135e-05.
161/161             8s 51ms/step -
accuracy: 0.9162 - loss: 0.2330 - val_accuracy: 0.8871 - val_loss: 0.3258 -
learning_rate: 4.5000e-05
Epoch 21/50
161/161             8s 48ms/step -
accuracy: 0.9214 - loss: 0.2245 - val_accuracy: 0.8973 - val_loss: 0.2974 -
learning_rate: 1.3500e-05
Epoch 22/50
161/161             8s 48ms/step -
```

```
accuracy: 0.9223 - loss: 0.2174 - val_accuracy: 0.8957 - val_loss: 0.2979 -
learning_rate: 1.3500e-05
Epoch 23/50
160/161             0s 45ms/step -
accuracy: 0.9233 - loss: 0.2169
Epoch 23: ReduceLROnPlateau reducing learning rate to 4.050000006827758e-06.
161/161             8s 47ms/step -
accuracy: 0.9233 - loss: 0.2168 - val_accuracy: 0.8941 - val_loss: 0.2981 -
learning_rate: 1.3500e-05
Epoch 24/50
161/161             8s 48ms/step -
accuracy: 0.9188 - loss: 0.2193 - val_accuracy: 0.8977 - val_loss: 0.2959 -
learning_rate: 4.0500e-06
Epoch 25/50
161/161             8s 47ms/step -
accuracy: 0.9246 - loss: 0.2147 - val_accuracy: 0.8977 - val_loss: 0.2946 -
learning_rate: 4.0500e-06
Epoch 26/50
161/161             8s 52ms/step -
accuracy: 0.9219 - loss: 0.2151 - val_accuracy: 0.8984 - val_loss: 0.2949 -
learning_rate: 4.0500e-06
Epoch 27/50
160/161             0s 46ms/step -
accuracy: 0.9250 - loss: 0.2122
Epoch 27: ReduceLROnPlateau reducing learning rate to 1.2149999747634864e-06.
161/161             8s 48ms/step -
accuracy: 0.9251 - loss: 0.2122 - val_accuracy: 0.9000 - val_loss: 0.2952 -
learning_rate: 4.0500e-06
Epoch 28/50
161/161             7s 46ms/step -
accuracy: 0.9261 - loss: 0.2123 - val_accuracy: 0.8996 - val_loss: 0.2952 -
learning_rate: 1.2150e-06
Epoch 29/50
160/161             0s 45ms/step -
accuracy: 0.9244 - loss: 0.2124
Epoch 29: ReduceLROnPlateau reducing learning rate to 3.644999992502562e-07.
161/161             8s 47ms/step -
accuracy: 0.9245 - loss: 0.2124 - val_accuracy: 0.8996 - val_loss: 0.2946 -
learning_rate: 1.2150e-06
Epoch 30/50
161/161             8s 48ms/step -
accuracy: 0.9239 - loss: 0.2105 - val_accuracy: 0.9000 - val_loss: 0.2949 -
learning_rate: 3.6450e-07
Guardando historial y modelo final HP…
```

9 7. Modelo 4 — Transfer Learning + Fine Tuning

[61]:
```
def transfer_finetune(base='MobileNetV2',
                      img_size=IMG_SIZE,
                      num_classes=NUM_CLASSES,
                      unfreeze_from=100,
                      hub_size=224):

    """Feature-extraction + fine-tuning con red pre-entrenada."""
    base_model = getattr(tf.keras.applications, base)(
        include_top=False,
        weights='imagenet',
        input_shape=(hub_size, hub_size, 3)
    )
    base_model.trainable = False

    inputs  = layers.Input(shape=img_size + (3,))
    x = layers.Resizing(hub_size, hub_size)(inputs)
    x = layers.Rescaling(1./255)(x)
    x = base_model(x, training=False)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.Dense(128, activation='relu')(x)
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = tf.keras.Model(inputs, outputs, name=f'{base}_finetune')

    opt_fe = get_optimizer(1e-3)
    model.compile(optimizer=opt_fe,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    hist_fe = model.fit(train_ds,
                        epochs=5,
                        validation_data=val_ds)

    opt_ft = get_optimizer(1e-5)

    base_model.trainable = True
    print(f"Fine-tuning: Descongelando desde la capa {unfreeze_from}")
    for i, layer in enumerate(base_model.layers):
        if i < unfreeze_from:
            layer.trainable = False
        else:
            if i % 10 == 0 or i >= len(base_model.layers) - 5:
              print(f"  - Capa {i} ({layer.name}): Trainable = {layer.
↪trainable}")

    print("\nRe-compilando modelo para fine-tuning con LR bajo...")
```

```
    model.compile(optimizer=opt_ft,
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])


    print("Iniciando fase de fine-tuning...")
    initial_epoch_ft = hist_fe.epoch[-1] + 1

    hist_ft = model.fit(train_ds,
                        epochs=EPOCHS,
                        validation_data=val_ds,
                        callbacks=common_callbacks('fine_tuning'))

    history = {k: hist_fe.history.get(k, []) + hist_ft.history[k]
               for k in hist_ft.history.keys()
               }
    return model, history
```

[62]:
```
tl_model, history_ft = transfer_finetune()
json.dump(history_ft, open(HIST_DIR / 'fine_tuning.json', 'w'))
tl_model.save(MODELS_DIR / 'fine_tuning.keras', include_optimizer=False)
```

```
Epoch 1/5

2025-04-28 18:56:17.675602: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_6489', 268 bytes spill stores, 268 bytes spill loads

2025-04-28 18:56:17.726344: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_6489', 276 bytes spill stores, 276 bytes spill loads

2025-04-28 18:56:17.781057: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_6489', 408 bytes spill stores, 408 bytes spill loads

2025-04-28 18:56:17.793891: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_6489', 16 bytes spill stores, 16 bytes spill loads


160/161             0s 12ms/step -
accuracy: 0.8043 - loss: 0.5201

2025-04-28 18:56:22.197956: I
```

```
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_1449_0', 176 bytes spill stores, 524 bytes spill loads

2025-04-28 18:56:22.370662: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_1449', 104 bytes spill stores, 104 bytes spill loads

2025-04-28 18:56:22.389392: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_1449', 5144 bytes spill stores, 5204 bytes spill loads

2025-04-28 18:56:22.418414: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_1449', 4984 bytes spill stores, 4984 bytes spill loads

2025-04-28 18:56:22.439210: I
external/local_xla/xla/stream_executor/cuda/subprocess_compilation.cc:346] ptxas
warning : Registers are spilled to local memory in function
'gemm_fusion_dot_1449', 12 bytes spill stores, 12 bytes spill loads


161/161            13s 55ms/step -
accuracy: 0.8051 - loss: 0.5179 - val_accuracy: 0.9062 - val_loss: 0.2469
Epoch 2/5
161/161            2s 15ms/step -
accuracy: 0.9168 - loss: 0.2257 - val_accuracy: 0.9078 - val_loss: 0.2547
Epoch 3/5
161/161            2s 14ms/step -
accuracy: 0.9315 - loss: 0.1852 - val_accuracy: 0.8922 - val_loss: 0.2963
Epoch 4/5
161/161            2s 14ms/step -
accuracy: 0.9408 - loss: 0.1615 - val_accuracy: 0.8789 - val_loss: 0.3236
Epoch 5/5
161/161            2s 15ms/step -
accuracy: 0.9472 - loss: 0.1444 - val_accuracy: 0.8906 - val_loss: 0.2965
Fine-tuning: Descongelando desde la capa 100
  - Capa 100 (block_11_expand_relu): Trainable = True
  - Capa 110 (block_12_depthwise): Trainable = True
  - Capa 120 (block_13_depthwise): Trainable = True
  - Capa 130 (block_14_depthwise_relu): Trainable = True
  - Capa 140 (block_15_project): Trainable = True
  - Capa 149 (block_16_project): Trainable = True
  - Capa 150 (block_16_project_BN): Trainable = True
  - Capa 151 (Conv_1): Trainable = True
```

```
  - Capa 152 (Conv_1_bn): Trainable = True
  - Capa 153 (out_relu): Trainable = True

Re-compilando modelo para fine-tuning con LR bajo…
Iniciando fase de fine-tuning…
Epoch 1/50

2025-04-28 18:56:45.020210: E
external/local_xla/xla/stream_executor/cuda/cuda_timer.cc:86] Delay kernel timed
out: measured time has sub-optimal accuracy. There may be a missing warmup
execution, please investigate in Nsight Systems.
2025-04-28 18:56:45.217044: E
external/local_xla/xla/stream_executor/cuda/cuda_timer.cc:86] Delay kernel timed
out: measured time has sub-optimal accuracy. There may be a missing warmup
execution, please investigate in Nsight Systems.

161/161            21s 64ms/step -
accuracy: 0.7919 - loss: 0.6268 - val_accuracy: 0.9129 - val_loss: 0.2672 -
learning_rate: 1.0000e-05
Epoch 2/50
161/161            3s 20ms/step -
accuracy: 0.9198 - loss: 0.2220 - val_accuracy: 0.9176 - val_loss: 0.2569 -
learning_rate: 1.0000e-05
Epoch 3/50
161/161            3s 21ms/step -
accuracy: 0.9496 - loss: 0.1535 - val_accuracy: 0.9203 - val_loss: 0.2484 -
learning_rate: 1.0000e-05
Epoch 4/50
161/161            3s 20ms/step -
accuracy: 0.9687 - loss: 0.1107 - val_accuracy: 0.9211 - val_loss: 0.2477 -
learning_rate: 1.0000e-05
Epoch 5/50
161/161            3s 18ms/step -
accuracy: 0.9792 - loss: 0.0812 - val_accuracy: 0.9187 - val_loss: 0.2497 -
learning_rate: 1.0000e-05
Epoch 6/50
158/161            0s 16ms/step -
accuracy: 0.9888 - loss: 0.0602
Epoch 6: ReduceLROnPlateau reducing learning rate to 2.9999999242136253e-06.
161/161            3s 18ms/step -
accuracy: 0.9889 - loss: 0.0601 - val_accuracy: 0.9176 - val_loss: 0.2513 -
learning_rate: 1.0000e-05
Epoch 7/50
161/161            3s 18ms/step -
accuracy: 0.9944 - loss: 0.0450 - val_accuracy: 0.9172 - val_loss: 0.2509 -
learning_rate: 3.0000e-06
Epoch 8/50
160/161            0s 16ms/step -
accuracy: 0.9955 - loss: 0.0411
```

```
Epoch 8: ReduceLROnPlateau reducing learning rate to 8.999999636216671e-07.
161/161             3s 19ms/step -
accuracy: 0.9955 - loss: 0.0411 - val_accuracy: 0.9180 - val_loss: 0.2497 -
learning_rate: 3.0000e-06
Epoch 9/50
161/161             3s 19ms/step -
accuracy: 0.9963 - loss: 0.0376 - val_accuracy: 0.9145 - val_loss: 0.2482 -
learning_rate: 9.0000e-07
```

10 8. Modelo 5 — Data Augmentation

[63]:
```
data_augmentation_layers = tf.keras.Sequential(
    [
        layers.RandomFlip("horizontal", seed=SEED),
        layers.RandomRotation(0.2, seed=SEED),
        layers.RandomZoom(0.2, seed=SEED),
        layers.RandomTranslation(height_factor=0.2, width_factor=0.2,␣
 ↪seed=SEED),
    ],
    name="data_augmentation",
)

try:
    base_model_for_aug = tl_model
except NameError:
    print("Cargando modelo base para aumentación desde archivo...")
    base_model_for_aug = tf.keras.models.load_model(MODELS_DIR / 'fine_tuning.
 ↪keras')

inputs = tf.keras.Input(shape=IMG_SIZE + (3,))
x = data_augmentation_layers(inputs, training=True)
outputs = base_model_for_aug(x)
aug_model = tf.keras.Model(inputs, outputs, name='model_with_augmentation')

aug_model.compile(optimizer=get_optimizer(1e-5),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

history_aug = aug_model.fit(
    train_ds,
    epochs=EPOCHS,
    validation_data=val_ds,
    callbacks=common_callbacks('data_aug')
)

json.dump(history_aug.history, open(HIST_DIR / 'data_aug.json', 'w'))
```

```
aug_model.save(MODELS_DIR / 'data_aug.keras')
```

```
Epoch 1/50
161/161             16s 59ms/step -
accuracy: 0.7371 - loss: 0.7842 - val_accuracy: 0.9137 - val_loss: 0.2632 -
learning_rate: 1.0000e-05
Epoch 2/50
161/161             8s 53ms/step -
accuracy: 0.7970 - loss: 0.5615 - val_accuracy: 0.9195 - val_loss: 0.2465 -
learning_rate: 1.0000e-05
Epoch 3/50
161/161             9s 55ms/step -
accuracy: 0.8160 - loss: 0.4965 - val_accuracy: 0.9180 - val_loss: 0.2455 -
learning_rate: 1.0000e-05
Epoch 4/50
161/161             8s 51ms/step -
accuracy: 0.8279 - loss: 0.4708 - val_accuracy: 0.9180 - val_loss: 0.2484 -
learning_rate: 1.0000e-05
Epoch 5/50
161/161             9s 56ms/step -
accuracy: 0.8398 - loss: 0.4367 - val_accuracy: 0.9184 - val_loss: 0.2359 -
learning_rate: 1.0000e-05
Epoch 6/50
161/161             9s 53ms/step -
accuracy: 0.8468 - loss: 0.4078 - val_accuracy: 0.9184 - val_loss: 0.2371 -
learning_rate: 1.0000e-05
Epoch 7/50
161/161             8s 52ms/step -
accuracy: 0.8595 - loss: 0.3788 - val_accuracy: 0.9172 - val_loss: 0.2358 -
learning_rate: 1.0000e-05
Epoch 8/50
161/161             8s 51ms/step -
accuracy: 0.8659 - loss: 0.3714 - val_accuracy: 0.9141 - val_loss: 0.2409 -
learning_rate: 1.0000e-05
Epoch 9/50
161/161             9s 57ms/step -
accuracy: 0.8687 - loss: 0.3576 - val_accuracy: 0.9145 - val_loss: 0.2354 -
learning_rate: 1.0000e-05
Epoch 10/50
161/161             9s 53ms/step -
accuracy: 0.8806 - loss: 0.3382 - val_accuracy: 0.9156 - val_loss: 0.2424 -
learning_rate: 1.0000e-05
Epoch 11/50
160/161             0s 48ms/step -
accuracy: 0.8755 - loss: 0.3335
Epoch 11: ReduceLROnPlateau reducing learning rate to 2.9999999242136253e-06.
161/161             8s 53ms/step -
accuracy: 0.8755 - loss: 0.3334 - val_accuracy: 0.9145 - val_loss: 0.2376 -
```

```
learning_rate: 1.0000e-05
Epoch 12/50
161/161             8s 52ms/step -
accuracy: 0.8838 - loss: 0.3174 - val_accuracy: 0.9148 - val_loss: 0.2387 -
learning_rate: 3.0000e-06
Epoch 13/50
160/161             0s 46ms/step -
accuracy: 0.8888 - loss: 0.3071
Epoch 13: ReduceLROnPlateau reducing learning rate to 8.999999636216671e-07.
161/161             8s 52ms/step -
accuracy: 0.8888 - loss: 0.3070 - val_accuracy: 0.9145 - val_loss: 0.2388 -
learning_rate: 3.0000e-06
Epoch 14/50
161/161             8s 51ms/step -
accuracy: 0.8855 - loss: 0.3206 - val_accuracy: 0.9145 - val_loss: 0.2399 -
learning_rate: 9.0000e-07
```

11 9. Evaluación en test y comparativa final

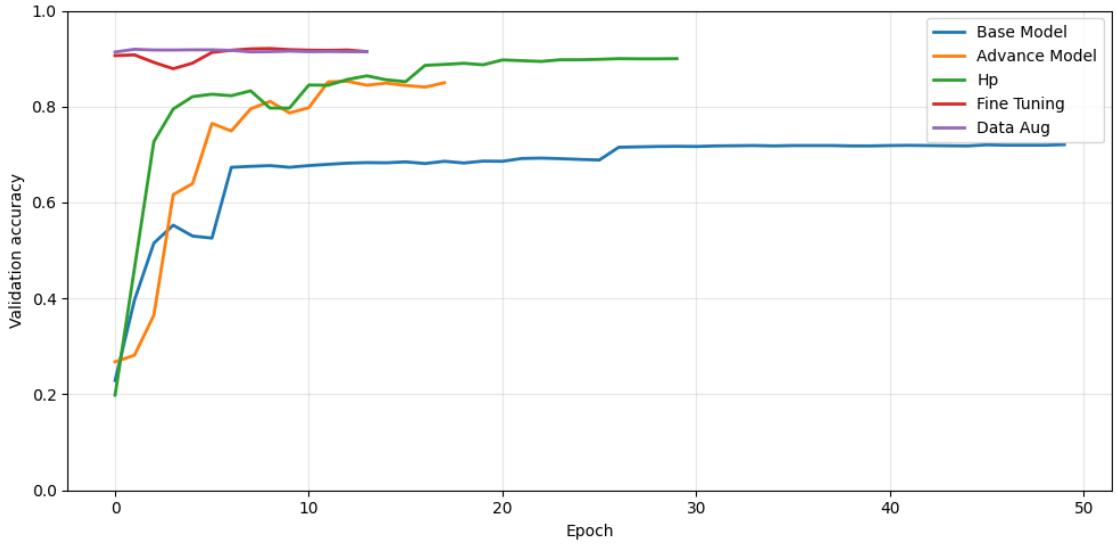
[64]:
```
def evaluate_and_log(model_path, name):
    model = tf.keras.models.load_model(model_path)
    loss, acc = model.evaluate(test_ds, verbose=0)
    return {'Model_Name': name, 'Test_Accuracy': acc}
```

[65]:
```
results = pd.DataFrame([
    evaluate_and_log(MODELS_DIR/'base_model.keras',   'base_model'),
    evaluate_and_log(MODELS_DIR/'advance_model.keras','advance_model'),
    evaluate_and_log(MODELS_DIR/'hp.keras',           'hp'),
    evaluate_and_log(MODELS_DIR/'fine_tuning.keras',  'fine_tuning'),
    evaluate_and_log(MODELS_DIR/'data_aug.keras',     'data_aug')
])

display(results.style.background_gradient(cmap='Reds', subset=['Test_Accuracy'])
                   .format({'Test_Accuracy':'{:.3f}'}))

plt.figure(figsize=(10,5))
for m, color in␣
 ↪zip(['base_model','advance_model','hp','fine_tuning','data_aug'],
                    ['tab:blue','tab:orange','tab:green','tab:red','tab:
 ↪purple']):
    hist = json.load(open(HIST_DIR/f'{m}.json'))
    plt.plot(hist['val_accuracy'], label=m.replace('_',' ').title(),␣
 ↪linewidth=2)
plt.legend(); plt.ylabel('Validation accuracy'); plt.xlabel('Epoch'); plt.
 ↪ylim(0,1); plt.grid(alpha=.3)
plt.tight_layout(); plt.show()
```

```
<pandas.io.formats.style.Styler at 0x7f91801a8140>
```



12 10. Visualización rápida de predicciones con matplotlib

[66]:
```
model  = tf.keras.models.load_model(MODELS_DIR / 'data_aug.keras')
model.summary(line_length=80)

NUM_IMAGES = 12
test_iter  = test_ds.unbatch().take(NUM_IMAGES)

plt.figure(figsize=(12, 9))
for idx, (img, true_lab) in enumerate(test_iter):
    pred_prob   = model.predict(img[tf.newaxis, ...], verbose=0)
    pred_label  = tf.argmax(pred_prob, axis=1).numpy()[0]

    ax = plt.subplot(3, 4, idx + 1)
    plt.imshow(img.numpy().astype("uint8"))
    ax.axis("off")

    correct = (pred_label == true_lab.numpy())
    color   = "green" if correct else "red"
    ax.set_title(
        f"GT: {class_names[true_lab]} \nPred: {class_names[pred_label]}",
        fontsize=9, color=color, pad=4
    )

plt.tight_layout()
plt.show()
```

```
Model: "model_with_augmentation"
```

Layer (type)	Output Shape	Param #
input_layer_4 (InputLayer)	(None, 150, 150, 3)	0
data_augmentation (Sequential)	(None, 150, 150, 3)	0
MobileNetV2_finetune (Functional)	(None, 6)	2,422,726

Total params: 6,475,096 (24.70 MB)

Trainable params: 2,026,182 (7.73 MB)

Non-trainable params: 396,544 (1.51 MB)

Optimizer params: 4,052,370 (15.46 MB)