```
import os, zipfile, shutil, random
from glob import glob
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2, ResNet50, DenseNet121, EfficientNetB0
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report, confusion_matrix

# Paths
ZIP_LOCAL = "/content/archive (1).zip"
EXTRACT_DIR = "/content/PlantVillage"
```

```
if not os.path.exists(EXTRACT_DIR):
    os.makedirs(EXTRACT_DIR, exist_ok=True)

with zipfile.ZipFile(ZIP_LOCAL, "r") as z:
    z.extractall(EXTRACT_DIR)

print("Extraction done✅.")
print("Sample contents:", os.listdir(EXTRACT_DIR)[:10])
```

Show hidden output

```
import pathlib
from sklearn.model_selection import train_test_split

# find the root folder containing classes
base_dir = None
for root, dirs, files in os.walk(EXTRACT_DIR):
    if len(dirs) > 5:   # PlantVillage has many class folders (diseases)
        base_dir = root
        break

print("Detected dataset root:", base_dir)
classes = os.listdir(base_dir)
print("Total classes:", len(classes))

# make new structured folders
structured_dir = "/content/PlantVillage_split"
if os.path.exists(structured_dir):
    shutil.rmtree(structured_dir)
os.makedirs(structured_dir)

for split in ["train", "val", "test"]:
    for c in classes:
        os.makedirs(os.path.join(structured_dir, split, c))

# split data
for c in classes:
    files = glob(os.path.join(base_dir, c, "*"))
    train_files, test_files = train_test_split(files, test_size=0.2, random_state=42)
    train_files, val_files = train_test_split(train_files, test_size=0.2, random_state=42)

    for f in train_files:
        shutil.copy(f, os.path.join(structured_dir, "train", c))
    for f in val_files:
        shutil.copy(f, os.path.join(structured_dir, "val", c))
    for f in test_files:
        shutil.copy(f, os.path.join(structured_dir, "test", c))

print("Splitting done✅.")
```

```
Detected dataset root: /content/PlantVillage/PlantVillage
Total classes: 15
Splitting done✅.
```

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 32

train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=20,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
```

```
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

val_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory(
    os.path.join(structured_dir, "train"),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="categorical"
)

val_gen = val_datagen.flow_from_directory(
    os.path.join(structured_dir, "val"),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="categorical"
)

test_gen = val_datagen.flow_from_directory(
    os.path.join(structured_dir, "test"),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    shuffle=False
)

NUM_CLASSES = train_gen.num_classes
print("Data generators ready✅. Classes:", train_gen.class_indices)
```

```
Found 13196 images belonging to 15 classes.
Found 3308 images belonging to 15 classes.
Found 4134 images belonging to 15 classes.
Data generators ready✅. Classes: {'Pepper__bell___Bacterial_spot': 0, 'Pepper__bell___healthy': 1, 'Potato___Early_blight': 2
```

```
def build_model(base_model_fn, input_shape=(224,224,3), num_classes=NUM_CLASSES):
    base_model = base_model_fn(weights="imagenet", include_top=False, input_shape=input_shape)
    base_model.trainable = False  # freeze base

    x = GlobalAveragePooling2D()(base_model.output)
    x = Dropout(0.3)(x)
    output = Dense(num_classes, activation="softmax")(x)
    model = Model(inputs=base_model.input, outputs=output)

    model.compile(optimizer=Adam(1e-4),
                  loss="categorical_crossentropy",
                  metrics=["accuracy"])
    return model
```

```
EPOCHS = 50

models = {
    "MobileNetV2": build_model(MobileNetV2),
    "ResNet50": build_model(ResNet50),
    "DenseNet121": build_model(DenseNet121),
    "EfficientNetB0": build_model(EfficientNetB0),
}

histories = {}

for name, model in models.items():
    print(f"\n🔵 Training {name}...")
    h = model.fit(train_gen,
                  validation_data=val_gen,
                  epochs=EPOCHS,
                  verbose=1)
    histories[name] = h
    model.save(f"/content/{name}_crop.h5")
    print(f" Saved✅ {name}")
```

Show hidden output

```
preds = []
for name, model in models.items():
    print(f"Predicting with {name}...")
    p = model.predict(test_gen, verbose=1)
    preds.append(p)
```

```
ensemble_pred = np.mean(preds, axis=0)
y_true = test_gen.classes
y_pred = np.argmax(ensemble_pred, axis=1)

print("\n✅ Ensemble Evaluation")
print(classification_report(y_true, y_pred, target_names=list(test_gen.class_indices.keys())))
```

```
import seaborn as sns

cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(12,10))
sns.heatmap(cm, annot=False, cmap="Blues")
plt.title("Ensemble Confusion Matrix")
plt.show()
```

```
from tensorflow.keras.preprocessing import image

def predict_image(img_path):
    img = image.load_img(img_path, target_size=IMG_SIZE)
    x = image.img_to_array(img) / 255.
    x = np.expand_dims(x, axis=0)

    # each model
    for name, model in models.items():
        pred = model.predict(x)
        print(f"{name} → {list(test_gen.class_indices.keys())[np.argmax(pred)]}")

    # ensemble
    pred_ens = np.mean([m.predict(x) for m in models.values()], axis=0)
    print(f"\n🌿 Ensemble → {list(test_gen.class_indices.keys())[np.argmax(pred_ens)]}")

# Example:
# predict_image("/content/PlantVillage_split/test/Tomato___Late_blight/1234.jpg")
```