

```

# =====
# AI-FlowersVision.ipynb – Fast Version
# =====

# Phase 1 – Setup & Imports
# -----
!pip install -q tensorflow==2.14.1 matplotlib seaborn

import os
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report, confusion_matrix
import numpy as np

# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Dataset path
DATA_DIR = "/content/drive/MyDrive/Image_Classification_Project/flowers"
assert os.path.exists(DATA_DIR), f"Dataset not found: {DATA_DIR}"
print("✅ Dataset loaded from:", DATA_DIR)

IMG_SIZE = (160, 160)
BATCH_SIZE = 32
SEED = 42

# =====
# Phase 2 – Splitting the Dataset
# =====
# Automatically split dataset into training/validation/test
train_ds = tf.keras.utils.image_dataset_from_directory(
    DATA_DIR,
    validation_split=0.3, # 30% reserved for val+test
    subset="training",
    seed=SEED,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE
)

val_test_ds = tf.keras.utils.image_dataset_from_directory(
    DATA_DIR,
    validation_split=0.3,
    subset="validation",
    seed=SEED,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE
)

# Further split validation/test (15% each)
val_batches = int(0.5 * len(val_test_ds))
val_ds = val_test_ds.take(val_batches)
test_ds = val_test_ds.skip(val_batches)

class_names = train_ds.class_names
n_classes = len(class_names)
print("Classes:", class_names)

# Prefetch for performance
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.prefetch(buffer_size=AUTOTUNE)

# Show sample images
plt.figure(figsize=(10, 6))
for images, labels in train_ds.take(1):
    for i in range(9):
        plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")

```

```

plt.suptitle("Sample Training Images", fontsize=14)
plt.tight_layout()
plt.show()

# =====
# Phase 3 – Data Augmentation
# =====
data_augmentation = keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
])

# Visualize augmentation example
for images, _ in train_ds.take(1):
    plt.figure(figsize=(8, 8))
    for i in range(9):
        augmented = data_augmentation(images)
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(augmented[0].numpy().astype("uint8"))
        plt.axis("off")
    plt.suptitle("Augmented Samples", fontsize=14)
    plt.show()
    break

# =====
# Phase 4 – Model Development (Fast Transfer Learning)
# =====
base_model = tf.keras.applications.MobileNetV2(
    input_shape=IMG_SIZE + (3,),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = False # freeze for speed

inputs = keras.Input(shape=IMG_SIZE + (3,))
x = data_augmentation(inputs)
x = tf.keras.applications.mobilenet_v2.preprocess_input(x)
x = base_model(x, training=False)
x = layers.GlobalAveragePooling2D()(x)
x = layers.Dropout(0.3)(x)
outputs = layers.Dense(n_classes, activation="softmax")(x)
model = keras.Model(inputs, outputs)

model.compile(
    optimizer=keras.optimizers.Adam(1e-4),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model.summary()

# =====
# Phase 5 – Training & Testing
# =====
EPOCHS = 8 # small for fast execution
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=EPOCHS,
    verbose=2
)

# Plot accuracy/loss
plt.figure(figsize=(12,4))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.legend(); plt.title('Accuracy')
plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.legend(); plt.title('Loss')
plt.show()

# =====
# Phase 6 – Model Evaluation
# =====

```

```
loss, acc = model.evaluate(test_ds)
print(f"✅ Test Accuracy: {acc:.3f}")

# Collect predictions
y_true = []
y_pred = []
for images, labels in test_ds:
    preds = model.predict(images)
    y_true.extend(labels.numpy())
    y_pred.extend(np.argmax(preds, axis=1))

# Classification report
print("\nClassification Report:")
print(classification_report(y_true, y_pred, target_names=class_names))

# Confusion matrix
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=class_names, yticklabels=class_names, cmap='Blues')
plt.xlabel('Predicted'); plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# =====
# Phase 7 – Save Model (optional)
# =====
SAVE_DIR = "/content/drive/MyDrive/Image_Classification_Project/models_fast"
os.makedirs(SAVE_DIR, exist_ok=True)
model.save(os.path.join(SAVE_DIR, "flowers_mobilenetv2_fast.h5"))
print("📁 Model saved to:", SAVE_DIR)
```

