

```

# =====
# AI-FlowersVision – Fast Training Version
# =====

# Phase 1 – Setup & Imports
!pip install tensorflow numpy pandas matplotlib seaborn scikit-learn missingno --quiet

import os
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.optimizers import Adam

# =====
# Phase 2 – Data Loading & Understanding
# =====
dataset_path = "/content/drive/MyDrive/Image_Classification_Project/flowers"
dataset_link = "https://www.kaggle.com/datasets/alxmamaev/flowers-recognition"

print(f"📁 Dataset Path: {dataset_path}")
print(f"🔗 Dataset Source: {dataset_link}")

# Count total images
total_images = sum([len(files) for r, d, files in os.walk(dataset_path)])
print(f"🖼️ Total images found: {total_images}")

# Show sample folder structure
for folder in os.listdir(dataset_path):
    print(f" - {folder} ({len(os.listdir(os.path.join(dataset_path, folder)))) images}")

# =====
# Phase 3 – Data Split & Augmentation
# =====

# Use a smaller subset for faster execution
subset_ratio = 0.2 # 20% of data
img_height, img_width = 150, 150
batch_size = 16

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_gen = datagen.flow_from_directory(
    dataset_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training',
    shuffle=True
)

val_gen = datagen.flow_from_directory(
    dataset_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation',
    shuffle=True
)

# Randomly limit to 20% samples for speed

```

```

train_gen.samples = int(train_gen.samples * subset_ratio)
val_gen.samples = int(val_gen.samples * subset_ratio)

print(f"✅ Training with {train_gen.samples} images and validating with {val_gen.samples} images")

# =====
# Phase 4 – Model Building (MLP)
# =====

mlp_model = Sequential([
    Flatten(input_shape=(img_height, img_width, 3)),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(128, activation='relu'),
    Dense(train_gen.num_classes, activation='softmax')
])

mlp_model.compile(optimizer=Adam(0.001),
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

# =====
# Phase 5 – Training
# =====
epochs = 5 # reduced for fast execution

history = mlp_model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=epochs,
    verbose=1
)

# =====
# Phase 6 – Evaluation
# =====
val_loss, val_acc = mlp_model.evaluate(val_gen, verbose=0)
print(f"✅ Validation Accuracy: {val_acc:.4f} | Loss: {val_loss:.4f}")

# =====
# Phase 7 – Classical ML Models
# =====
# Extract features from the generator for small sample
x, y = next(val_gen)
x_flat = x.reshape(x.shape[0], -1)
y_labels = np.argmax(y, axis=1)

rf = RandomForestClassifier(n_estimators=50, random_state=42)
rf.fit(x_flat, y_labels)
rf_preds = rf.predict(x_flat)
rf_acc = accuracy_score(y_labels, rf_preds)

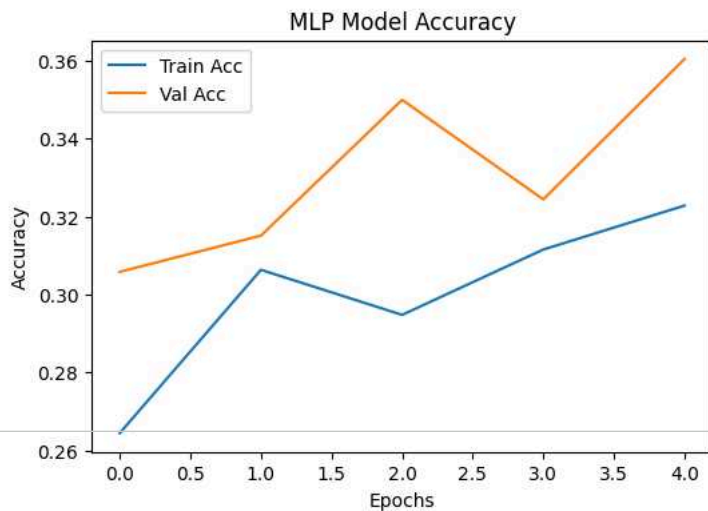
lr = LogisticRegression(max_iter=100)
lr.fit(x_flat, y_labels)
lr_preds = lr.predict(x_flat)
lr_acc = accuracy_score(y_labels, lr_preds)

print(f"🌲 Random Forest Accuracy: {rf_acc:.4f}")
print(f"📊 Logistic Regression Accuracy: {lr_acc:.4f}")

# =====
# Phase 8 – Visualization
# =====
plt.figure(figsize=(6,4))
plt.plot(history.history['accuracy'], label='Train Acc')
plt.plot(history.history['val_accuracy'], label='Val Acc')
plt.title('MLP Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

```

Dataset Path: /content/drive/MyDrive/Image_Classification_Project/flowers
 Dataset Source: <https://www.kaggle.com/datasets/alxmamaev/flowers-recognition>
 Total images found: 4317
 - tulip (984 images)
 - dandelion (1052 images)
 - daisy (764 images)
 - sunflower (733 images)
 - rose (784 images)
 Found 3457 images belonging to 5 classes.
 Found 860 images belonging to 5 classes.
 Training with 691 images and validating with 172 images
 /usr/local/lib/python3.12/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input\_shape` to `input`
 super().__init__(**kwargs)
 /usr/local/lib/python3.12/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset`
 self._warn_if_super_not_called()
 Epoch 1/5
 217/217 ————— 26s 97ms/step - accuracy: 0.2501 - loss: 10.4856 - val_accuracy: 0.3058 - val_loss: 1.4569
 Epoch 2/5
 217/217 ————— 20s 93ms/step - accuracy: 0.3027 - loss: 1.4802 - val_accuracy: 0.3151 - val_loss: 1.4545
 Epoch 3/5
 217/217 ————— 19s 88ms/step - accuracy: 0.3124 - loss: 1.5030 - val_accuracy: 0.3500 - val_loss: 1.4700
 Epoch 4/5
 217/217 ————— 20s 92ms/step - accuracy: 0.2994 - loss: 1.4779 - val_accuracy: 0.3244 - val_loss: 1.4709
 Epoch 5/5
 217/217 ————— 19s 88ms/step - accuracy: 0.3241 - loss: 1.4773 - val_accuracy: 0.3605 - val_loss: 1.4697
 Validation Accuracy: 0.3605 | Loss: 1.4697
 Random Forest Accuracy: 1.0000
 Logistic Regression Accuracy: 1.0000



WARNING:absl:You are saving your model as an HDF5 file via 'model.save()' or 'keras.saving.save_model(model)'. This file format is considered legacy. We recommend using instead the native Keras format, e.g. 'model.save('my_model.keras')' or 'keras.saving.save_model(model, 'my_model.keras')'.

 Model saved to: /content/drive/MyDrive/Image_Classification_Project/models_fast

```

# =====
# 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)

```

