```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!mkdir -p ~/.kaggle
!cp /content/kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

# Create folder in drive
!mkdir -p /content/drive/MyDrive/dataset/flowers

# Install Kaggle CLI
!pip install --quiet kaggle

# Download
!kaggle datasets download -d alxmamaev/flowers-recognition \
    -p /content/drive/MyDrive/dataset/flowers
!unzip -q /content/drive/MyDrive/dataset/flowers/flowers-
recognition.zip \
    -d /content/drive/MyDrive/dataset/flowers
```

```
Dataset URL: https://www.kaggle.com/datasets/alxmamaev/flowers-
recognition
License(s): unknown
```

```
# Check dataset structure
!find /content/drive/MyDrive/dataset/flowers -maxdepth 2 -type d
```

```
/content/drive/MyDrive/dataset/flowers
/content/drive/MyDrive/dataset/flowers/flowers
/content/drive/MyDrive/dataset/flowers/flowers/daisy
/content/drive/MyDrive/dataset/flowers/flowers/dandelion
/content/drive/MyDrive/dataset/flowers/flowers/rose
/content/drive/MyDrive/dataset/flowers/flowers/sunflower
/content/drive/MyDrive/dataset/flowers/flowers/tulip
```

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing import
image_dataset_from_directory, image

DATASET_PATH = '/content/drive/MyDrive/dataset/flowers/flowers'
IMG_SIZE = (224, 224)
BATCH_SIZE = 32

train_ds = image_dataset_from_directory(
    DATASET_PATH,
    validation_split=0.2,
```

```
    subset='training',
    seed=123,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE
)
val_ds = image_dataset_from_directory(
    DATASET_PATH,
    validation_split=0.2,
    subset='validation',
    seed=123,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE
)

class_names = train_ds.class_names
print("Clase:", class_names)
```

```
Found 4317 files belonging to 5 classes.
Using 3454 files for training.
Found 4317 files belonging to 5 classes.
Using 863 files for validation.
Clase: ['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
```

```
# Freeze transfer-learning base
base_model = tf.keras.applications.MobileNetV2(
    input_shape=IMG_SIZE + (3,),
    include_top=False,
    weights='imagenet'
)
base_model.trainable = False

# Add layers
from tensorflow.keras import layers, models
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(len(train_ds.class_names), activation='softmax')
])

# Compile
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

# Train
EPOCHS = 10
history = model.fit(
    train_ds,
```

```
    validation_data=val_ds,
    epochs=EPOCHS
)

# Evaluate
loss, acc = model.evaluate(val_ds)
print(f"Validation Accuracy: {acc*100:.2f}%")
```

```
Epoch 1/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 34s 220ms/step - accuracy: 0.3217 - loss:
1.6117 - val_accuracy: 0.5261 - val_loss: 1.2365
Epoch 2/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 19s 174ms/step - accuracy: 0.5392 - loss:
1.1952 - val_accuracy: 0.5655 - val_loss: 1.1657
Epoch 3/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 20s 168ms/step - accuracy: 0.5940 - loss:
1.0913 - val_accuracy: 0.6188 - val_loss: 1.0781
Epoch 4/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 19s 158ms/step - accuracy: 0.6250 - loss:
1.0140 - val_accuracy: 0.6246 - val_loss: 1.0499
Epoch 5/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 22s 174ms/step - accuracy: 0.6378 - loss:
0.9647 - val_accuracy: 0.6396 - val_loss: 1.0333
Epoch 6/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 23s 197ms/step - accuracy: 0.6612 - loss:
0.9301 - val_accuracy: 0.6211 - val_loss: 1.0203
Epoch 7/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 37s 157ms/step - accuracy: 0.6766 - loss:
0.8890 - val_accuracy: 0.6327 - val_loss: 1.0140
Epoch 8/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 22s 169ms/step - accuracy: 0.6742 - loss:
0.8840 - val_accuracy: 0.6501 - val_loss: 0.9766
Epoch 9/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 22s 183ms/step - accuracy: 0.6992 - loss:
0.8443 - val_accuracy: 0.6466 - val_loss: 0.9962
Epoch 10/10
108/108 ━━━━━━━━━━━━━━━━━━━━ 18s 159ms/step - accuracy: 0.7012 - loss:
0.8204 - val_accuracy: 0.6466 - val_loss: 1.0051
27/27 ━━━━━━━━━━━━━━━━━━━━ 3s 100ms/step - accuracy: 0.6243 - loss:
1.0290
Validation Accuracy: 64.66%
```

```
# Load image
img_path =
'/content/drive/MyDrive/dataset/flowers/flowers/rose/11944957684_2cc80
6276e.jpg'
img = image.load_img(img_path, target_size=IMG_SIZE)
plt.imshow(img)
plt.axis('off')
```

```
# Preprocess and prediction
gray = image.img_to_array(img) / 255.0
input_arr = np.expand_dims(gray, axis=0)
preds = model.predict(input_arr)
predicted_class = class_names[np.argmax(preds[0])]
print(f"Clasa prezisă pentru imagine: {predicted_class}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
Clasa prezisă pentru imagine: rose
```