Pretrained_VS_CNN

January 20, 2025

Difference Between CNNs and Pre-trained Models 1. CNNs (Convolutional Neural Networks): Definition: A type of neural network specifically designed to process grid-like data such as images. It is built from scratch using components like convolutional layers, pooling layers, and fully connected layers for classification. Advantages: Highly flexible for designing custom architectures tailored to specific datasets. Useful for understanding the impact of architectural modifications on model performance. Challenges: Requires training from scratch, which demands large datasets and significant computational resources. 2. Pre-trained Models: Definition: Models that have been trained on large datasets (e.g., ImageNet) and come with pre-learned weights that can be reused for similar tasks. Advantages: Reduces the need for large amounts of data. Provides a quick way to achieve high performance without training a model from scratch. Challenges: May not perform optimally if the new task is very different from the original task the model was trained on.

Key Difference:

Custom CNNs: Built and trained from scratch, offering complete flexibility but requiring substantial resources.

Pre-trained Models: Start with pre-learned weights, saving time and computational effort, and can be fine-tuned for the specific task.

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

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

So lets try to read some data and start with **CNN**

1 Split data in drive into Train,Test and Validation

[ ]:
```
import os
import shutil
from sklearn.model_selection import train_test_split

def split_data(data_dir, output_dir, train_ratio=0.7, val_ratio=0.25,␣
 ↪test_ratio=0.05):
    """
    Split data into train, validation, and test directories.

    Parameters:
        data_dir (str): Path to the dataset directory.
```

```
        output_dir (str): Path to save the split dataset.
        train_ratio (float): Ratio of training data (default is 0.7).
        val_ratio (float): Ratio of validation data (default is 0.2).
        test_ratio (float): Ratio of test data (default is 0.1).
    """
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    classes = os.listdir(data_dir)
    for class_name in classes:
        class_dir = os.path.join(data_dir, class_name)
        if not os.path.isdir(class_dir):
            continue

        # Get all files in the class directory
        files = os.listdir(class_dir)
        train_files, temp_files = train_test_split(files, test_size=(1 - train_ratio), random_state=42)
        val_files, test_files = train_test_split(temp_files, test_size=(test_ratio / (val_ratio + test_ratio)), random_state=42)

        # Define output subdirectories
        train_dir = os.path.join(output_dir, 'train', class_name)
        val_dir = os.path.join(output_dir, 'val', class_name)
        test_dir = os.path.join(output_dir, 'test', class_name)
        os.makedirs(train_dir, exist_ok=True)
        os.makedirs(val_dir, exist_ok=True)
        os.makedirs(test_dir, exist_ok=True)

        # Move files to respective directories
        for file in train_files:
            shutil.copy(os.path.join(class_dir, file), os.path.join(train_dir, file))
        for file in val_files:
            shutil.copy(os.path.join(class_dir, file), os.path.join(val_dir, file))
        for file in test_files:
            shutil.copy(os.path.join(class_dir, file), os.path.join(test_dir, file))

    print("Data split completed!")
```

[ ]:
```
'''
data_dir = '/content/drive/MyDrive/Flowers/train'  # Path to your dataset
output_dir = '/content/drive/MyDrive/Flowers_Divided'  # Path to save split data
split_data(data_dir, output_dir)'''
```

[ ]:
```
"\ndata_dir = '/content/drive/MyDrive/Flowers/train'  # Path to your
dataset\noutput_dir = '/content/drive/MyDrive/Flowers_Divided'  # Path to save
split data\nsplit_data(data_dir, output_dir)"
```

[ ]:
```
import os

def dataset_info(data_dir):
    """
    Display information about the dataset: folder structure and number of␣
 ↪images per class.
    This function will now be applied to the train, validation, and test␣
 ↪directories.

    Parameters:
        data_dir (str): Path to the dataset directory (train, val, or test).
    """
    total_images = 0
    class_counts = {}

    # Traverse dataset directory (train, val, or test)
    for class_name in sorted(os.listdir(data_dir)):
        class_dir = os.path.join(data_dir, class_name)
        if not os.path.isdir(class_dir):
            continue  # Skip non-directory files

        # Count images in the current class folder
        num_images = len([f for f in os.listdir(class_dir) if os.path.isfile(os.
 ↪path.join(class_dir, f))])
        class_counts[class_name] = num_images
        total_images += num_images

    # Print results
    print(f"Dataset Directory: {data_dir}")
    print(f"Total Classes: {len(class_counts)}")
    print(f"Total Images: {total_images}\n")

    print("Images per Class:")
    for class_name, count in class_counts.items():
        print(f"  {class_name}: {count} images")

# Example usage
train_dir = '/content/drive/MyDrive/divided_mask_dataset/train'  # Path to your␣
 ↪train data
val_dir = '/content/drive/My Drive/divided_mask_dataset/val'      # Path to␣
 ↪your validation data
test_dir = '/content/drive/My Drive/divided_mask_dataset/test'    # Path to␣
 ↪your test data
```

```
print("Train Dataset Info:")
dataset_info(train_dir)

print("\nValidation Dataset Info:")
dataset_info(val_dir)

print("\nTest Dataset Info:")
dataset_info(test_dir)
```

```
Train Dataset Info:
Dataset Directory: /content/drive/MyDrive/divided_mask_dataset/train
Total Classes: 2
Total Images: 962

Images per Class:
  with mask: 482 images
  without mask: 480 images

Validation Dataset Info:
Dataset Directory: /content/drive/My Drive/divided_mask_dataset/val
Total Classes: 2
Total Images: 275

Images per Class:
  with mask: 138 images
  without mask: 137 images

Test Dataset Info:
Dataset Directory: /content/drive/My Drive/divided_mask_dataset/test
Total Classes: 2
Total Images: 139

Images per Class:
  with mask: 70 images
  without mask: 69 images
```

plot some images

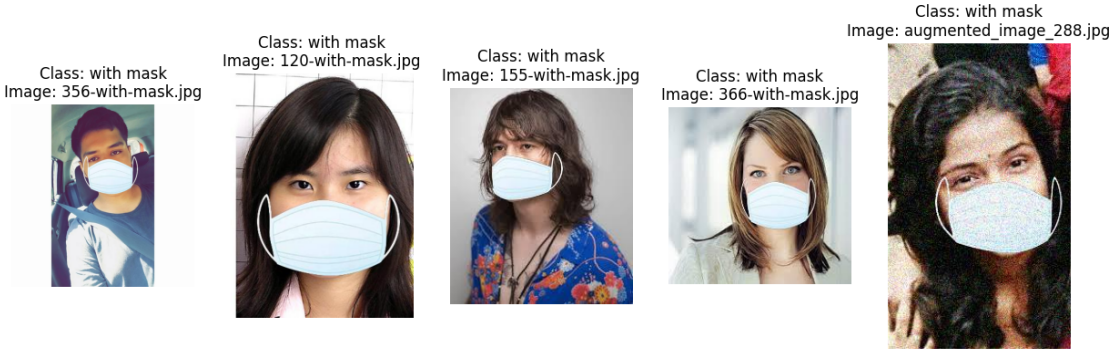
[ ]:
```
import os
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def plot_random_images(data_dir, num_images=5):
    """
    Plot random images from the specified dataset directory.
```

```
    Parameters:
        data_dir (str): Path to the dataset directory (train, val, or test).
        num_images (int): Number of random images to display.
    """
    class_names = sorted(os.listdir(data_dir))
    class_name = random.choice(class_names)  # Choose a random class to display␣
↪images
    class_dir = os.path.join(data_dir, class_name)

    # Get all image file names in the chosen class folder
    image_files = [f for f in os.listdir(class_dir) if os.path.isfile(os.path.
↪join(class_dir, f))]

    # Choose random images to display
    selected_images = random.sample(image_files, num_images)

    # Plot the selected images
    plt.figure(figsize=(15, 10))
    for i, img_file in enumerate(selected_images):
        img_path = os.path.join(class_dir, img_file)
        img = mpimg.imread(img_path)

        plt.subplot(1, num_images, i+1)
        plt.imshow(img)
        plt.axis('off')  # Hide axis
        plt.title(f"Class: {class_name}\nImage: {img_file}")

    plt.show()


print("Random Images from Train Dataset:")
plot_random_images(train_dir)

print("Random Images from Validation Dataset:")
plot_random_images(val_dir)

print("Random Images from Test Dataset:")
plot_random_images(test_dir)
```

```
Random Images from Train Dataset:
```

Random Images from Validation Dataset:

Random Images from Test Dataset:



Create ImageDataGenerators for train, validation, and test sets note : Val and test have no augmentation

[ ]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the target size for the CNN model
```

```
target_size = (224, 224)  # Target size for models like MobileNetV2

# Define the batch size
batch_size = 32

# Create ImageDataGenerators for train, validation, and test sets
train_datagen = ImageDataGenerator(
    rescale=1./255,  # Rescale pixel values to [0, 1]
    rotation_range=20,  # Optional: Data augmentation
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)  # No augmentation for␣
 ↪validation/test

# Create generators to load and resize images
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=target_size,  # Resize images to 224x224
    batch_size=batch_size,
    class_mode='categorical',  # For multi-class classification
    shuffle=True
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=target_size,  # Resize images to 224x224
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=target_size,  # Resize images to 224x224
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 962 images belonging to 2 classes.
Found 275 images belonging to 2 classes.
Found 139 images belonging to 2 classes.
```

2 Build the CNN

[ ]:
```
# Build a simple CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224,␣
 ↪224, 3)),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')  # 2 classes for CIFAR-10
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

[ ]:
```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)
model_checkpoint = ModelCheckpoint('/content/drive/MyDrive/divided_mask_dataset/
 ↪best_advanced_model.keras', monitor='val_loss', save_best_only=True,␣
 ↪verbose=1)
```

[ ]:
```
# Train the model using the generators
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=50,
    validation_data=val_generator,
    validation_steps=val_generator.samples // batch_size,
    callbacks=[early_stopping, model_checkpoint]
)
```

```
Epoch 1/50

/usr/local/lib/python3.10/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()

30/30            0s 476ms/step -
accuracy: 0.5332 - loss: 3.6778
Epoch 1: val_loss improved from inf to 0.64849, saving model to
```

```
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30            30s 638ms/step -
accuracy: 0.5326 - loss: 3.6211 - val_accuracy: 0.5469 - val_loss: 0.6485
Epoch 2/50
 1/30            1s 52ms/step - accuracy:
0.5312 - loss: 0.6448

/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out of data;
interrupting training. Make sure that your dataset or generator can generate at
least `steps_per_epoch * epochs` batches. You may need to use the `.repeat()`
function when building your dataset.
  self.gen.throw(typ, value, traceback)


Epoch 2: val_loss did not improve from 0.64849
30/30            2s 57ms/step -
accuracy: 0.5312 - loss: 0.6448 - val_accuracy: 0.0000e+00 - val_loss: 0.9708
Epoch 3/50
30/30            0s 435ms/step -
accuracy: 0.7363 - loss: 0.5029
Epoch 3: val_loss improved from 0.64849 to 0.40754, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30            20s 545ms/step -
accuracy: 0.7380 - loss: 0.5018 - val_accuracy: 0.8125 - val_loss: 0.4075
Epoch 4/50
 1/30            1s 52ms/step - accuracy:
0.8438 - loss: 0.2785
Epoch 4: val_loss did not improve from 0.40754
30/30            2s 64ms/step -
accuracy: 0.8438 - loss: 0.2785 - val_accuracy: 0.7368 - val_loss: 0.7172
Epoch 5/50
29/30            0s 412ms/step -
accuracy: 0.8509 - loss: 0.3546
Epoch 5: val_loss improved from 0.40754 to 0.17103, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30            37s 513ms/step -
accuracy: 0.8517 - loss: 0.3541 - val_accuracy: 0.9492 - val_loss: 0.1710
Epoch 6/50
 1/30            1s 52ms/step - accuracy:
0.7812 - loss: 0.4487
Epoch 6: val_loss did not improve from 0.17103
30/30            2s 60ms/step -
accuracy: 0.7812 - loss: 0.4487 - val_accuracy: 0.8947 - val_loss: 0.3407
Epoch 7/50
30/30            0s 456ms/step -
accuracy: 0.8883 - loss: 0.3253
Epoch 7: val_loss improved from 0.17103 to 0.15580, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30            20s 564ms/step -
```

```
accuracy: 0.8884 - loss: 0.3248 - val_accuracy: 0.9688 - val_loss: 0.1558
Epoch 8/50
 1/30            0s 13ms/step - accuracy:
0.5000 - loss: 0.7200
Epoch 8: val_loss did not improve from 0.15580
30/30            0s 1ms/step -
accuracy: 0.5000 - loss: 0.7200 - val_accuracy: 0.8947 - val_loss: 0.5462
Epoch 9/50
29/30            0s 432ms/step -
accuracy: 0.9014 - loss: 0.3270
Epoch 9: val_loss improved from 0.15580 to 0.15452, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30            21s 534ms/step -
accuracy: 0.9013 - loss: 0.3277 - val_accuracy: 0.9570 - val_loss: 0.1545
Epoch 10/50
 1/30            1s 52ms/step - accuracy:
0.9062 - loss: 0.3866
Epoch 10: val_loss did not improve from 0.15452
30/30            0s 1ms/step -
accuracy: 0.9062 - loss: 0.3866 - val_accuracy: 0.8947 - val_loss: 0.4882
Epoch 11/50
29/30            0s 373ms/step -
accuracy: 0.9258 - loss: 0.2594
Epoch 11: val_loss improved from 0.15452 to 0.13850, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30            40s 495ms/step -
accuracy: 0.9252 - loss: 0.2604 - val_accuracy: 0.9609 - val_loss: 0.1385
Epoch 12/50
 1/30            1s 51ms/step - accuracy:
0.9062 - loss: 0.2503
Epoch 12: val_loss did not improve from 0.13850
30/30            0s 887us/step -
accuracy: 0.9062 - loss: 0.2503 - val_accuracy: 0.8947 - val_loss: 0.3852
Epoch 13/50
29/30            0s 462ms/step -
accuracy: 0.9118 - loss: 0.2450
Epoch 13: val_loss improved from 0.13850 to 0.11348, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30            21s 578ms/step -
accuracy: 0.9121 - loss: 0.2439 - val_accuracy: 0.9648 - val_loss: 0.1135
Epoch 14/50
 1/30            1s 52ms/step - accuracy:
0.9688 - loss: 0.0880
Epoch 14: val_loss did not improve from 0.11348
30/30            1s 43ms/step -
accuracy: 0.9688 - loss: 0.0880 - val_accuracy: 0.8947 - val_loss: 0.3012
Epoch 15/50
30/30            0s 437ms/step -
```

```
accuracy: 0.9468 - loss: 0.1869
Epoch 15: val_loss did not improve from 0.11348
30/30             18s 492ms/step -
accuracy: 0.9466 - loss: 0.1875 - val_accuracy: 0.9648 - val_loss: 0.1456
Epoch 16/50
 1/30             1s 45ms/step - accuracy:
0.9375 - loss: 0.1846
Epoch 16: val_loss improved from 0.11348 to 0.10635, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30             3s 90ms/step -
accuracy: 0.9375 - loss: 0.1846 - val_accuracy: 1.0000 - val_loss: 0.1063
Epoch 17/50
29/30             0s 455ms/step -
accuracy: 0.9342 - loss: 0.2232
Epoch 17: val_loss improved from 0.10635 to 0.07669, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30             19s 546ms/step -
accuracy: 0.9339 - loss: 0.2233 - val_accuracy: 0.9766 - val_loss: 0.0767
Epoch 18/50
 1/30             1s 53ms/step - accuracy:
0.9688 - loss: 0.0839
Epoch 18: val_loss did not improve from 0.07669
30/30             0s 947us/step -
accuracy: 0.9688 - loss: 0.0839 - val_accuracy: 0.8947 - val_loss: 0.3288
Epoch 19/50
29/30             0s 659ms/step -
accuracy: 0.9308 - loss: 0.2163
Epoch 19: val_loss improved from 0.07669 to 0.06738, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30             26s 761ms/step -
accuracy: 0.9311 - loss: 0.2153 - val_accuracy: 0.9844 - val_loss: 0.0674
Epoch 20/50
 1/30             1s 52ms/step - accuracy:
1.0000 - loss: 0.1187
Epoch 20: val_loss did not improve from 0.06738
30/30             0s 851us/step -
accuracy: 1.0000 - loss: 0.1187 - val_accuracy: 0.8947 - val_loss: 0.2825
Epoch 21/50
29/30             0s 451ms/step -
accuracy: 0.9420 - loss: 0.1719
Epoch 21: val_loss did not improve from 0.06738
30/30             18s 484ms/step -
accuracy: 0.9419 - loss: 0.1729 - val_accuracy: 0.9766 - val_loss: 0.0897
Epoch 22/50
 1/30             1s 40ms/step - accuracy:
1.0000 - loss: 0.0962
Epoch 22: val_loss did not improve from 0.06738
30/30             0s 2ms/step -
```

```
accuracy: 1.0000 - loss: 0.0962 - val_accuracy: 0.8947 - val_loss: 0.1552
Epoch 23/50
30/30             0s 381ms/step -
accuracy: 0.9231 - loss: 0.1930
Epoch 23: val_loss improved from 0.06738 to 0.06149, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
30/30             19s 512ms/step -
accuracy: 0.9228 - loss: 0.1947 - val_accuracy: 0.9805 - val_loss: 0.0615
Epoch 24/50
 1/30             1s 57ms/step - accuracy:
0.9688 - loss: 0.1414
Epoch 24: val_loss did not improve from 0.06149
30/30             1s 43ms/step -
accuracy: 0.9688 - loss: 0.1414 - val_accuracy: 0.8947 - val_loss: 0.1520
Epoch 25/50
29/30             0s 453ms/step -
accuracy: 0.9036 - loss: 0.2319
Epoch 25: val_loss did not improve from 0.06149
30/30             18s 486ms/step -
accuracy: 0.9044 - loss: 0.2310 - val_accuracy: 0.9688 - val_loss: 0.0989
Epoch 26/50
 1/30             1s 39ms/step - accuracy:
1.0000 - loss: 0.1066
Epoch 26: val_loss did not improve from 0.06149
30/30             1s 39ms/step -
accuracy: 1.0000 - loss: 0.1066 - val_accuracy: 0.8947 - val_loss: 0.3627
Epoch 27/50
29/30             0s 415ms/step -
accuracy: 0.9354 - loss: 0.1990
Epoch 27: val_loss did not improve from 0.06149
30/30             19s 450ms/step -
accuracy: 0.9356 - loss: 0.1984 - val_accuracy: 0.9883 - val_loss: 0.0640
Epoch 28/50
 1/30             1s 40ms/step - accuracy:
1.0000 - loss: 0.1027
Epoch 28: val_loss did not improve from 0.06149
30/30             1s 39ms/step -
accuracy: 1.0000 - loss: 0.1027 - val_accuracy: 0.8947 - val_loss: 0.1296
```

2.1 Evaluate

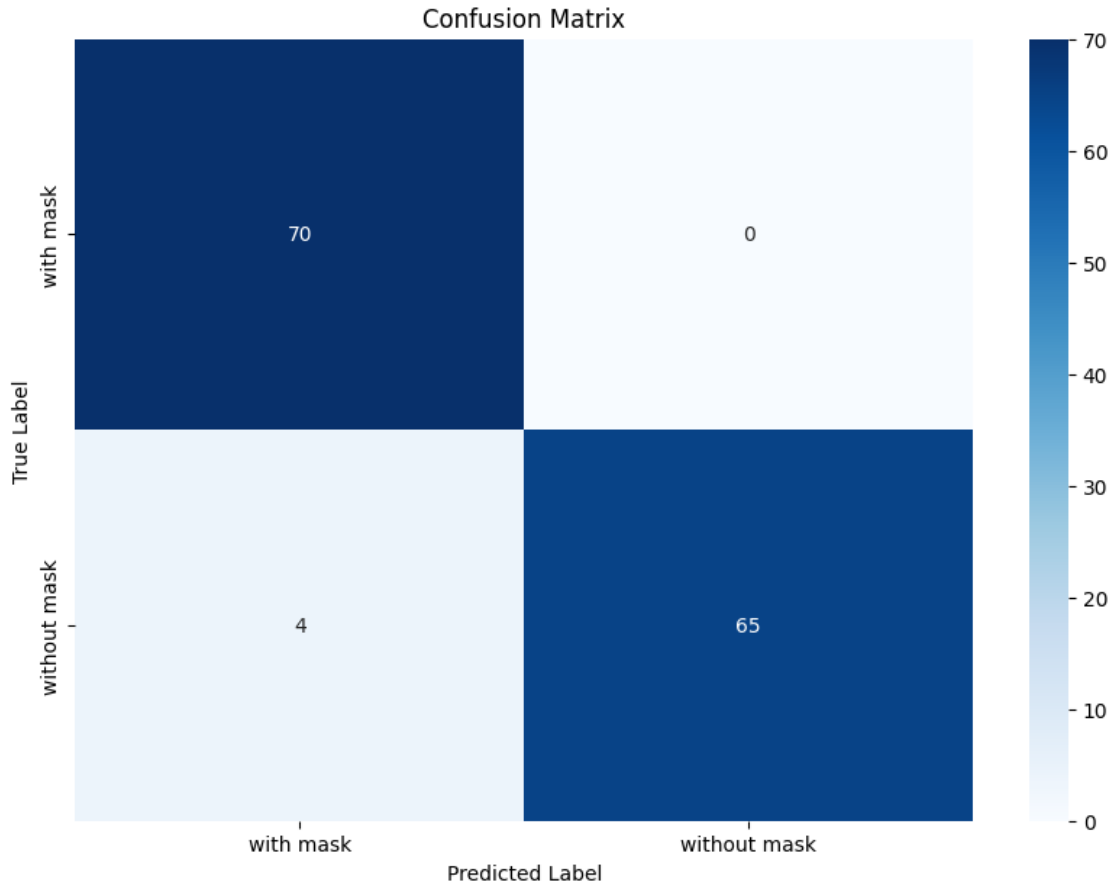
[ ]:
```python
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_generator.
 ↪samples // batch_size)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4             1s 246ms/step -
```

```
accuracy: 0.9885 - loss: 0.0422
Test Loss: 0.07171785086393356
Test Accuracy: 0.9765625
```

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import load_model

# Generate predictions for the test data without specifying 'steps'
test_predictions = model.predict(test_generator, verbose=1)

# Get the predicted class labels
predicted_labels = np.argmax(test_predictions, axis=1)

# Get the true class labels from the test generator
true_labels = test_generator.classes

# Generate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=test_generator.
 ↪class_indices.keys(), yticklabels=test_generator.class_indices.keys())
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
5/5             4s 778ms/step
```



i expected complexed data than what i see but lets try

another confused data to see how CNN dealing with

Confused classes

3 Try confused data

Flowers data may be difficult for Simple CNN to Classifing

[ ]:
```
import os

def dataset_info(data_dir):
    """
    Display information about the dataset: folder structure and number of␣
 ↪images per class.
    This function will now be applied to the train, validation, and test␣
 ↪directories.

    Parameters:
```

```
        data_dir (str): Path to the dataset directory (train, val, or test).
    """
    total_images = 0
    class_counts = {}

    # Traverse dataset directory (train, val, or test)
    for class_name in sorted(os.listdir(data_dir)):
        class_dir = os.path.join(data_dir, class_name)
        if not os.path.isdir(class_dir):
            continue  # Skip non-directory files

        # Count images in the current class folder
        num_images = len([f for f in os.listdir(class_dir) if os.path.isfile(os.
 ↪path.join(class_dir, f))])
        class_counts[class_name] = num_images
        total_images += num_images

    # Print results
    print(f"Dataset Directory: {data_dir}")
    print(f"Total Classes: {len(class_counts)}")
    print(f"Total Images: {total_images}\n")

    print("Images per Class:")
    for class_name, count in class_counts.items():
        print(f"  {class_name}: {count} images")

# Example usage
train_dir = '/content/drive/MyDrive/Flowers_Divided/train'  # Path to your␣
 ↪train data
val_dir = '/content/drive/MyDrive/Flowers_Divided/val'      # Path to your␣
 ↪validation data
test_dir = '/content/drive/MyDrive/Flowers_Divided/test'    # Path to your test␣
 ↪data

print("Train Dataset Info:")
dataset_info(train_dir)

print("\nValidation Dataset Info:")
dataset_info(val_dir)

print("\nTest Dataset Info:")
dataset_info(test_dir)
```

```
Train Dataset Info:
Dataset Directory: /content/drive/MyDrive/Flowers_Divided/train
Total Classes: 5
Total Images: 1919
```

```
Images per Class:
  daisy: 350 images
  dandelion: 452 images
  rose: 347 images
  sunflower: 346 images
  tulip: 424 images

Validation Dataset Info:
Dataset Directory: /content/drive/MyDrive/Flowers_Divided/val
Total Classes: 5
Total Images: 686

Images per Class:
  daisy: 125 images
  dandelion: 161 images
  rose: 124 images
  sunflower: 124 images
  tulip: 152 images

Test Dataset Info:
Dataset Directory: /content/drive/MyDrive/Flowers_Divided/test
Total Classes: 5
Total Images: 141

Images per Class:
  daisy: 26 images
  dandelion: 33 images
  rose: 26 images
  sunflower: 25 images
  tulip: 31 images
```

[ ]:
```python
import os
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def plot_random_images(data_dir, num_images=5):
    """
    Plot random images from the specified dataset directory.

    Parameters:
        data_dir (str): Path to the dataset directory (train, val, or test).
        num_images (int): Number of random images to display.
    """
    class_names = sorted(os.listdir(data_dir))
```

```
    class_name = random.choice(class_names)  # Choose a random class to display␣
↪images
    class_dir = os.path.join(data_dir, class_name)

    # Get all image file names in the chosen class folder
    image_files = [f for f in os.listdir(class_dir) if os.path.isfile(os.path.
↪join(class_dir, f))]

    # Choose random images to display
    selected_images = random.sample(image_files, num_images)

    # Plot the selected images
    plt.figure(figsize=(15, 10))
    for i, img_file in enumerate(selected_images):
        img_path = os.path.join(class_dir, img_file)
        img = mpimg.imread(img_path)

        plt.subplot(1, num_images, i+1)
        plt.imshow(img)
        plt.axis('off')  # Hide axis
        plt.title(f"Class: {class_name}")

    plt.show()


print("Random Images from Train Dataset:")
plot_random_images(train_dir)

print("Random Images from Validation Dataset:")
plot_random_images(val_dir)

print("Random Images from Test Dataset:")
plot_random_images(test_dir)
```

```
Random Images from Train Dataset:
```

Random Images from Validation Dataset:

Random Images from Test Dataset:

[ ]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the target size for the CNN model
target_size = (224, 224)  # Target size for models like MobileNetV2

# Define the batch size
batch_size = 32

# Create ImageDataGenerators for train, validation, and test sets
train_datagen = ImageDataGenerator(
    rescale=1./255,  # Rescale pixel values to [0, 1]
    rotation_range=20,  # Optional: Data augmentation
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)  # No augmentation for␣
 ↪validation/test
```

```
# Create generators to load and resize images
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=target_size,  # Resize images to 224x224
    batch_size=batch_size,
    class_mode='categorical',  # For multi-class classification (CIFAR-10)
    shuffle=True
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=target_size,  # Resize images to 224x224
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=target_size,  # Resize images to 224x224
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 1919 images belonging to 5 classes.
Found 686 images belonging to 5 classes.
Found 141 images belonging to 5 classes.
```

[ ]:
```
# Build a simple CNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224,␣
 ↪224, 3)),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(5, activation='softmax')  # 10 classes for CIFAR-10
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
```

```
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

[ ]:
```
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)
model_checkpoint = ModelCheckpoint('/content/drive/MyDrive/divided_mask_dataset/
 ↪best_advanced_model.keras', monitor='val_loss', save_best_only=True,␣
 ↪verbose=1)
```

[ ]:
```
# Train the model using the generators
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=50,
    validation_data=val_generator,
    validation_steps=val_generator.samples // batch_size,
    callbacks=[early_stopping, model_checkpoint]
)
```

```
Epoch 1/50

/usr/local/lib/python3.10/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()

59/59             0s 618ms/step -
accuracy: 0.2674 - loss: 4.8076
Epoch 1: val_loss improved from inf to 1.26450, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
59/59             64s 893ms/step -
accuracy: 0.2685 - loss: 4.7680 - val_accuracy: 0.4628 - val_loss: 1.2645
Epoch 2/50
 1/59             4:07 4s/step - accuracy:
0.5806 - loss: 1.2086

/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out of data;
interrupting training. Make sure that your dataset or generator can generate at
least `steps_per_epoch * epochs` batches. You may need to use the `.repeat()`
function when building your dataset.
  self.gen.throw(typ, value, traceback)
```

```
Epoch 2: val_loss improved from 1.26450 to 0.97379, saving model to
/content/drive/MyDrive/divided_mask_dataset/best_advanced_model.keras
59/59             28s 405ms/step -
accuracy: 0.5806 - loss: 1.2086 - val_accuracy: 0.3571 - val_loss: 0.9738
Epoch 3/50
58/59             0s 496ms/step -
accuracy: 0.4667 - loss: 1.2440
Epoch 3: val_loss did not improve from 0.97379
59/59             87s 549ms/step -
accuracy: 0.4665 - loss: 1.2448 - val_accuracy: 0.5536 - val_loss: 1.1431
Epoch 4/50
 1/59             3s 52ms/step - accuracy:
0.5625 - loss: 0.9831
Epoch 4: val_loss did not improve from 0.97379
59/59             0s 487us/step -
accuracy: 0.5625 - loss: 0.9831 - val_accuracy: 0.4286 - val_loss: 1.1856
Epoch 5/50
59/59             0s 474ms/step -
accuracy: 0.5264 - loss: 1.1673
Epoch 5: val_loss did not improve from 0.97379
59/59             36s 545ms/step -
accuracy: 0.5268 - loss: 1.1669 - val_accuracy: 0.5759 - val_loss: 1.1152
Epoch 6/50
 1/59             3s 53ms/step - accuracy:
0.5625 - loss: 1.1106
Epoch 6: val_loss did not improve from 0.97379
59/59             0s 488us/step -
accuracy: 0.5625 - loss: 1.1106 - val_accuracy: 0.5000 - val_loss: 0.9987
Epoch 7/50
59/59             0s 490ms/step -
accuracy: 0.5754 - loss: 1.0838
Epoch 7: val_loss did not improve from 0.97379
59/59             41s 558ms/step -
accuracy: 0.5754 - loss: 1.0836 - val_accuracy: 0.6101 - val_loss: 1.0163
```

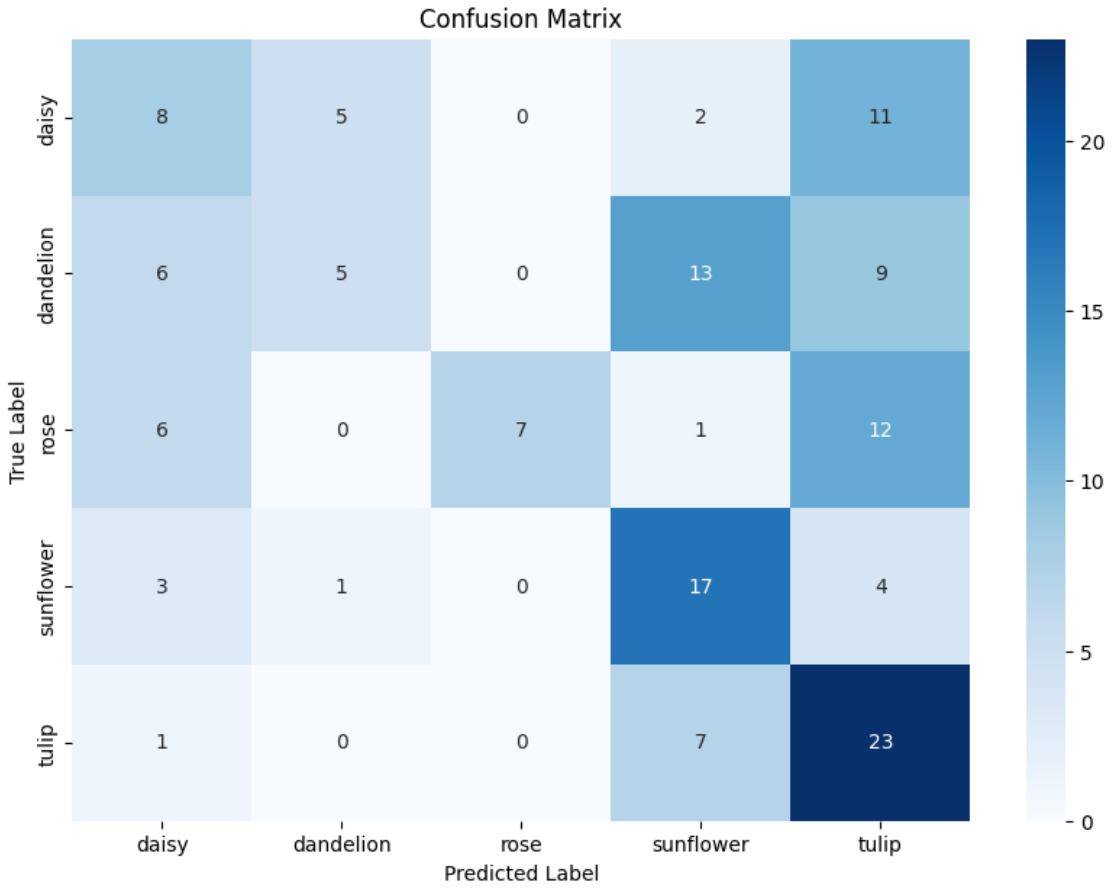
[ ]:
```python
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_generator.
 ↪samples // batch_size)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4               1s 217ms/step -
accuracy: 0.3146 - loss: 1.4061
Test Loss: 1.2935124635696411
Test Accuracy: 0.40625
```

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import load_model

# Generate predictions for the test data without specifying 'steps'
test_predictions = model.predict(test_generator, verbose=1)

# Get the predicted class labels
predicted_labels = np.argmax(test_predictions, axis=1)

# Get the true class labels from the test generator
true_labels = test_generator.classes

# Generate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=test_generator.
 ↪class_indices.keys(), yticklabels=test_generator.class_indices.keys())
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
5/5             2s 319ms/step
```



As expected it was so hard to know from the first time

so i will not waste time in tring find the best CNN model

but i will try **Pretrained models**

3.1 Pre-Trained MobileNetV2

[ ]:
```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model

# Define the number of classes in your dataset
num_classes = 5

# Load the pre-trained MobileNetV2 model
```

```
base_model = MobileNetV2(weights='imagenet', include_top=False,␣
 ↪input_shape=(224, 224, 3))

# Freeze the base model's layers to prevent them from being trained
base_model.trainable = False

# Add custom classification head
x = base_model.output
x = GlobalAveragePooling2D()(x)  # Reduce dimensions to prevent overfitting
x = Dropout(0.3)(x)  # Dropout for regularization
x = Dense(128, activation='relu')(x)  # Add a fully connected layer
x = Dropout(0.3)(x)  # Dropout for regularization
output = Dense(num_classes, activation='softmax')(x)  # Final classification␣
 ↪layer

# Create the model
model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])

# Display the model summary
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
ns/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h
5
9406464/9406464            2s
0us/step
```

Model: "functional_2"

Layer (type)	Output Shape	Param #	Connected to
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0	-
Conv1 (Conv2D)	(None, 112, 112, 32)	864	input_layer_2[0][0]
bn_Conv1	(None, 112, 112, 32)	128	Conv1[0][0]

```
(BatchNormalization)                                                    ␣
↪

Conv1_relu (ReLU)           (None, 112, 112, 32)              0         ␣
↪bn_Conv1[0][0]

expanded_conv_depthwise     (None, 112, 112, 32)              288       ␣
↪Conv1_relu[0][0]
(DepthwiseConv2D)                                                       ␣
↪

expanded_conv_depthwise_…   (None, 112, 112, 32)              128       ␣
↪expanded_conv_depthwi…
(BatchNormalization)                                                    ␣
↪

expanded_conv_depthwise_…   (None, 112, 112, 32)              0         ␣
↪expanded_conv_depthwi…
(ReLU)                                                                  ␣
↪

expanded_conv_project       (None, 112, 112, 16)              512       ␣
↪expanded_conv_depthwi…
(Conv2D)                                                                ␣
↪

expanded_conv_project_BN    (None, 112, 112, 16)              64        ␣
↪expanded_conv_project…
(BatchNormalization)                                                    ␣
↪

block_1_expand (Conv2D)     (None, 112, 112, 96)              1,536     ␣
↪expanded_conv_project…

block_1_expand_BN           (None, 112, 112, 96)              384       ␣
↪block_1_expand[0][0]
(BatchNormalization)                                                    ␣
↪

block_1_expand_relu         (None, 112, 112, 96)              0         ␣
↪block_1_expand_BN[0][…
(ReLU)                                                                  ␣
↪

block_1_pad                 (None, 113, 113, 96)              0         ␣
↪block_1_expand_relu[0…
```

```
(ZeroPadding2D)                                                                 ␣
↪

block_1_depthwise           (None, 56, 56, 96)                  864 ␣
↪block_1_pad[0][0]
(DepthwiseConv2D)                                                               ␣
↪

block_1_depthwise_BN        (None, 56, 56, 96)                  384 ␣
↪block_1_depthwise[0][…
(BatchNormalization)                                                            ␣
↪

block_1_depthwise_relu      (None, 56, 56, 96)                    0 ␣
↪block_1_depthwise_BN[…
(ReLU)                                                                          ␣
↪

block_1_project (Conv2D)    (None, 56, 56, 24)                2,304 ␣
↪block_1_depthwise_rel…

block_1_project_BN          (None, 56, 56, 24)                   96 ␣
↪block_1_project[0][0]
(BatchNormalization)                                                            ␣
↪

block_2_expand (Conv2D)     (None, 56, 56, 144)               3,456 ␣
↪block_1_project_BN[0]…

block_2_expand_BN           (None, 56, 56, 144)                 576 ␣
↪block_2_expand[0][0]
(BatchNormalization)                                                            ␣
↪

block_2_expand_relu         (None, 56, 56, 144)                   0 ␣
↪block_2_expand_BN[0][…
(ReLU)                                                                          ␣
↪

block_2_depthwise           (None, 56, 56, 144)               1,296 ␣
↪block_2_expand_relu[0…
(DepthwiseConv2D)                                                               ␣
↪

block_2_depthwise_BN        (None, 56, 56, 144)                 576 ␣
↪block_2_depthwise[0][…
```

```
(BatchNormalization)                                                    ␣
↪
block_2_depthwise_relu      (None, 56, 56, 144)          0              ␣
↪block_2_depthwise_BN[…
(ReLU)                                                                  ␣
↪
block_2_project (Conv2D)    (None, 56, 56, 24)           3,456          ␣
↪block_2_depthwise_rel…
block_2_project_BN          (None, 56, 56, 24)           96             ␣
↪block_2_project[0][0]
(BatchNormalization)                                                    ␣
↪
block_2_add (Add)           (None, 56, 56, 24)           0              ␣
↪block_1_project_BN[0]…
                                                                        ␣
↪block_2_project_BN[0]…
block_3_expand (Conv2D)     (None, 56, 56, 144)          3,456          ␣
↪block_2_add[0][0]
block_3_expand_BN           (None, 56, 56, 144)          576            ␣
↪block_3_expand[0][0]
(BatchNormalization)                                                    ␣
↪
block_3_expand_relu         (None, 56, 56, 144)          0              ␣
↪block_3_expand_BN[0][…
(ReLU)                                                                  ␣
↪
block_3_pad                 (None, 57, 57, 144)          0              ␣
↪block_3_expand_relu[0…
(ZeroPadding2D)                                                         ␣
↪
block_3_depthwise           (None, 28, 28, 144)          1,296          ␣
↪block_3_pad[0][0]
(DepthwiseConv2D)                                                       ␣
↪
block_3_depthwise_BN        (None, 28, 28, 144)          576            ␣
↪block_3_depthwise[0][…
```

```
(BatchNormalization)                                                    ␣
↪
block_3_depthwise_relu      (None, 28, 28, 144)          0              ␣
↪block_3_depthwise_BN[…
(ReLU)                                                                  ␣
↪
block_3_project (Conv2D)    (None, 28, 28, 32)           4,608          ␣
↪block_3_depthwise_rel…
block_3_project_BN          (None, 28, 28, 32)           128            ␣
↪block_3_project[0][0]
(BatchNormalization)                                                    ␣
↪
block_4_expand (Conv2D)     (None, 28, 28, 192)          6,144          ␣
↪block_3_project_BN[0]…
block_4_expand_BN           (None, 28, 28, 192)          768            ␣
↪block_4_expand[0][0]
(BatchNormalization)                                                    ␣
↪
block_4_expand_relu         (None, 28, 28, 192)          0              ␣
↪block_4_expand_BN[0][…
(ReLU)                                                                  ␣
↪
block_4_depthwise           (None, 28, 28, 192)          1,728          ␣
↪block_4_expand_relu[0…
(DepthwiseConv2D)                                                       ␣
↪
block_4_depthwise_BN        (None, 28, 28, 192)          768            ␣
↪block_4_depthwise[0][…
(BatchNormalization)                                                    ␣
↪
block_4_depthwise_relu      (None, 28, 28, 192)          0              ␣
↪block_4_depthwise_BN[…
(ReLU)                                                                  ␣
↪
block_4_project (Conv2D)    (None, 28, 28, 32)           6,144          ␣
↪block_4_depthwise_rel…
```

```
block_4_project_BN          (None, 28, 28, 32)          128 ␣
↪block_4_project[0][0]
(BatchNormalization)                                        ␣
↪

block_4_add (Add)           (None, 28, 28, 32)            0 ␣
↪block_3_project_BN[0]…
                                                            ␣
↪block_4_project_BN[0]…

block_5_expand (Conv2D)     (None, 28, 28, 192)       6,144 ␣
↪block_4_add[0][0]

block_5_expand_BN           (None, 28, 28, 192)         768 ␣
↪block_5_expand[0][0]
(BatchNormalization)                                        ␣
↪

block_5_expand_relu         (None, 28, 28, 192)           0 ␣
↪block_5_expand_BN[0][…
(ReLU)                                                      ␣
↪

block_5_depthwise           (None, 28, 28, 192)       1,728 ␣
↪block_5_expand_relu[0…
(DepthwiseConv2D)                                           ␣
↪

block_5_depthwise_BN        (None, 28, 28, 192)         768 ␣
↪block_5_depthwise[0][…
(BatchNormalization)                                        ␣
↪

block_5_depthwise_relu      (None, 28, 28, 192)           0 ␣
↪block_5_depthwise_BN[…
(ReLU)                                                      ␣
↪

block_5_project (Conv2D)    (None, 28, 28, 32)        6,144 ␣
↪block_5_depthwise_rel…

block_5_project_BN          (None, 28, 28, 32)          128 ␣
↪block_5_project[0][0]
(BatchNormalization)                                        ␣
↪
```

```
block_5_add (Add)           (None, 28, 28, 32)          0 ␣
↪block_4_add[0][0],
                                                        ␣
↪block_5_project_BN[0]…

block_6_expand (Conv2D)     (None, 28, 28, 192)         6,144 ␣
↪block_5_add[0][0]

block_6_expand_BN           (None, 28, 28, 192)         768 ␣
↪block_6_expand[0][0]
(BatchNormalization)                                        ␣
↪

block_6_expand_relu         (None, 28, 28, 192)         0 ␣
↪block_6_expand_BN[0][…
(ReLU)                                                      ␣
↪

block_6_pad                 (None, 29, 29, 192)         0 ␣
↪block_6_expand_relu[0…
(ZeroPadding2D)                                             ␣
↪

block_6_depthwise           (None, 14, 14, 192)         1,728 ␣
↪block_6_pad[0][0]
(DepthwiseConv2D)                                           ␣
↪

block_6_depthwise_BN        (None, 14, 14, 192)         768 ␣
↪block_6_depthwise[0][…
(BatchNormalization)                                        ␣
↪

block_6_depthwise_relu      (None, 14, 14, 192)         0 ␣
↪block_6_depthwise_BN[…
(ReLU)                                                      ␣
↪

block_6_project (Conv2D)    (None, 14, 14, 64)          12,288 ␣
↪block_6_depthwise_rel…

block_6_project_BN          (None, 14, 14, 64)          256 ␣
↪block_6_project[0][0]
(BatchNormalization)                                        ␣
↪
```

```
block_7_expand (Conv2D)      (None, 14, 14, 384)          24,576 ␣
↪block_6_project_BN[0]…

block_7_expand_BN            (None, 14, 14, 384)           1,536 ␣
↪block_7_expand[0][0]
(BatchNormalization)                                             ␣
↪

block_7_expand_relu          (None, 14, 14, 384)               0 ␣
↪block_7_expand_BN[0][…
(ReLU)                                                           ␣
↪

block_7_depthwise            (None, 14, 14, 384)           3,456 ␣
↪block_7_expand_relu[0…
(DepthwiseConv2D)                                                ␣
↪

block_7_depthwise_BN         (None, 14, 14, 384)           1,536 ␣
↪block_7_depthwise[0][…
(BatchNormalization)                                             ␣
↪

block_7_depthwise_relu       (None, 14, 14, 384)               0 ␣
↪block_7_depthwise_BN[…
(ReLU)                                                           ␣
↪

block_7_project (Conv2D)     (None, 14, 14, 64)           24,576 ␣
↪block_7_depthwise_rel…

block_7_project_BN           (None, 14, 14, 64)              256 ␣
↪block_7_project[0][0]
(BatchNormalization)                                             ␣
↪

block_7_add (Add)            (None, 14, 14, 64)                0 ␣
↪block_6_project_BN[0]…

                                                                 ␣
↪block_7_project_BN[0]…

block_8_expand (Conv2D)      (None, 14, 14, 384)          24,576 ␣
↪block_7_add[0][0]
```

```
block_8_expand_BN          (None, 14, 14, 384)          1,536 ␣
↪block_8_expand[0][0]
(BatchNormalization)                                         ␣
↪

block_8_expand_relu        (None, 14, 14, 384)              0 ␣
↪block_8_expand_BN[0][…
(ReLU)                                                       ␣
↪

block_8_depthwise          (None, 14, 14, 384)          3,456 ␣
↪block_8_expand_relu[0…
(DepthwiseConv2D)                                            ␣
↪

block_8_depthwise_BN       (None, 14, 14, 384)          1,536 ␣
↪block_8_depthwise[0][…
(BatchNormalization)                                         ␣
↪

block_8_depthwise_relu     (None, 14, 14, 384)              0 ␣
↪block_8_depthwise_BN[…
(ReLU)                                                       ␣
↪

block_8_project (Conv2D)   (None, 14, 14, 64)          24,576 ␣
↪block_8_depthwise_rel…

block_8_project_BN         (None, 14, 14, 64)              256 ␣
↪block_8_project[0][0]
(BatchNormalization)                                         ␣
↪

block_8_add (Add)          (None, 14, 14, 64)               0 ␣
↪block_7_add[0][0],
                                                             ␣
↪block_8_project_BN[0]…

block_9_expand (Conv2D)    (None, 14, 14, 384)          24,576 ␣
↪block_8_add[0][0]

block_9_expand_BN          (None, 14, 14, 384)          1,536 ␣
↪block_9_expand[0][0]
(BatchNormalization)                                         ␣
↪
```

```
block_9_expand_relu          (None, 14, 14, 384)               0 ␣
↪block_9_expand_BN[0][…
(ReLU)                                                         ␣
↪

block_9_depthwise            (None, 14, 14, 384)           3,456 ␣
↪block_9_expand_relu[0…
(DepthwiseConv2D)                                              ␣
↪

block_9_depthwise_BN         (None, 14, 14, 384)           1,536 ␣
↪block_9_depthwise[0][…
(BatchNormalization)                                           ␣
↪

block_9_depthwise_relu       (None, 14, 14, 384)               0 ␣
↪block_9_depthwise_BN[…
(ReLU)                                                         ␣
↪

block_9_project (Conv2D)     (None, 14, 14, 64)           24,576 ␣
↪block_9_depthwise_rel…

block_9_project_BN           (None, 14, 14, 64)              256 ␣
↪block_9_project[0][0]
(BatchNormalization)                                           ␣
↪

block_9_add (Add)            (None, 14, 14, 64)                0 ␣
↪block_8_add[0][0],
                                                               ␣
↪block_9_project_BN[0]…

block_10_expand (Conv2D)     (None, 14, 14, 384)          24,576 ␣
↪block_9_add[0][0]

block_10_expand_BN           (None, 14, 14, 384)           1,536 ␣
↪block_10_expand[0][0]
(BatchNormalization)                                           ␣
↪

block_10_expand_relu         (None, 14, 14, 384)               0 ␣
↪block_10_expand_BN[0]…
(ReLU)                                                         ␣
↪
```

```
block_10_depthwise          (None, 14, 14, 384)          3,456 ␣
↪block_10_expand_relu[…
(DepthwiseConv2D)                                              ␣
↪

block_10_depthwise_BN       (None, 14, 14, 384)          1,536 ␣
↪block_10_depthwise[0]…
(BatchNormalization)                                           ␣
↪

block_10_depthwise_relu     (None, 14, 14, 384)              0 ␣
↪block_10_depthwise_BN…
(ReLU)                                                         ␣
↪

block_10_project (Conv2D)   (None, 14, 14, 96)          36,864 ␣
↪block_10_depthwise_re…

block_10_project_BN         (None, 14, 14, 96)             384 ␣
↪block_10_project[0][0]
(BatchNormalization)                                           ␣
↪

block_11_expand (Conv2D)    (None, 14, 14, 576)         55,296 ␣
↪block_10_project_BN[0…

block_11_expand_BN          (None, 14, 14, 576)          2,304 ␣
↪block_11_expand[0][0]
(BatchNormalization)                                           ␣
↪

block_11_expand_relu        (None, 14, 14, 576)              0 ␣
↪block_11_expand_BN[0]…
(ReLU)                                                         ␣
↪

block_11_depthwise          (None, 14, 14, 576)          5,184 ␣
↪block_11_expand_relu[…
(DepthwiseConv2D)                                              ␣
↪

block_11_depthwise_BN       (None, 14, 14, 576)          2,304 ␣
↪block_11_depthwise[0]…
(BatchNormalization)                                           ␣
↪
```

```
block_11_depthwise_relu     (None, 14, 14, 576)               0 ␣
↪block_11_depthwise_BN…
(ReLU)                                                        ␣
↪

block_11_project (Conv2D)   (None, 14, 14, 96)           55,296 ␣
↪block_11_depthwise_re…

block_11_project_BN         (None, 14, 14, 96)              384 ␣
↪block_11_project[0][0]
(BatchNormalization)                                          ␣
↪

block_11_add (Add)          (None, 14, 14, 96)                0 ␣
↪block_10_project_BN[0…
                                                              ␣
↪block_11_project_BN[0…

block_12_expand (Conv2D)    (None, 14, 14, 576)          55,296 ␣
↪block_11_add[0][0]

block_12_expand_BN          (None, 14, 14, 576)           2,304 ␣
↪block_12_expand[0][0]
(BatchNormalization)                                          ␣
↪

block_12_expand_relu        (None, 14, 14, 576)               0 ␣
↪block_12_expand_BN[0]…
(ReLU)                                                        ␣
↪

block_12_depthwise          (None, 14, 14, 576)           5,184 ␣
↪block_12_expand_relu[…
(DepthwiseConv2D)                                             ␣
↪

block_12_depthwise_BN       (None, 14, 14, 576)           2,304 ␣
↪block_12_depthwise[0]…
(BatchNormalization)                                          ␣
↪

block_12_depthwise_relu     (None, 14, 14, 576)               0 ␣
↪block_12_depthwise_BN…
(ReLU)                                                        ␣
↪
```

```
block_12_project (Conv2D)   (None, 14, 14, 96)              55,296 ␣
↪block_12_depthwise_re…

block_12_project_BN         (None, 14, 14, 96)                 384 ␣
↪block_12_project[0][0]
(BatchNormalization)                                                ␣
↪

block_12_add (Add)          (None, 14, 14, 96)                   0 ␣
↪block_11_add[0][0],
                                                                    ␣
↪block_12_project_BN[0…

block_13_expand (Conv2D)    (None, 14, 14, 576)             55,296 ␣
↪block_12_add[0][0]

block_13_expand_BN          (None, 14, 14, 576)              2,304 ␣
↪block_13_expand[0][0]
(BatchNormalization)                                                ␣
↪

block_13_expand_relu        (None, 14, 14, 576)                  0 ␣
↪block_13_expand_BN[0]…
(ReLU)                                                              ␣
↪

block_13_pad                (None, 15, 15, 576)                  0 ␣
↪block_13_expand_relu[…
(ZeroPadding2D)                                                     ␣
↪

block_13_depthwise          (None, 7, 7, 576)                5,184 ␣
↪block_13_pad[0][0]
(DepthwiseConv2D)                                                   ␣
↪

block_13_depthwise_BN       (None, 7, 7, 576)                2,304 ␣
↪block_13_depthwise[0]…
(BatchNormalization)                                                ␣
↪

block_13_depthwise_relu     (None, 7, 7, 576)                    0 ␣
↪block_13_depthwise_BN…
(ReLU)                                                              ␣
↪
```

```
block_13_project (Conv2D)   (None, 7, 7, 160)              92,160 ␣
↪block_13_depthwise_re…

block_13_project_BN         (None, 7, 7, 160)                 640 ␣
↪block_13_project[0][0]
(BatchNormalization)                                              ␣
↪

block_14_expand (Conv2D)    (None, 7, 7, 960)             153,600 ␣
↪block_13_project_BN[0…

block_14_expand_BN          (None, 7, 7, 960)               3,840 ␣
↪block_14_expand[0][0]
(BatchNormalization)                                              ␣
↪

block_14_expand_relu        (None, 7, 7, 960)                   0 ␣
↪block_14_expand_BN[0]…
(ReLU)                                                            ␣
↪

block_14_depthwise          (None, 7, 7, 960)               8,640 ␣
↪block_14_expand_relu[…
(DepthwiseConv2D)                                                 ␣
↪

block_14_depthwise_BN       (None, 7, 7, 960)               3,840 ␣
↪block_14_depthwise[0]…
(BatchNormalization)                                              ␣
↪

block_14_depthwise_relu     (None, 7, 7, 960)                   0 ␣
↪block_14_depthwise_BN…
(ReLU)                                                            ␣
↪

block_14_project (Conv2D)   (None, 7, 7, 160)             153,600 ␣
↪block_14_depthwise_re…

block_14_project_BN         (None, 7, 7, 160)                 640 ␣
↪block_14_project[0][0]
(BatchNormalization)                                              ␣
↪

block_14_add (Add)          (None, 7, 7, 160)                   0 ␣
↪block_13_project_BN[0…
```

```
↪block_14_project_BN[0…                                                  ␣

block_15_expand (Conv2D)    (None, 7, 7, 960)          153,600  ␣
↪block_14_add[0][0]

block_15_expand_BN          (None, 7, 7, 960)            3,840  ␣
↪block_15_expand[0][0]
(BatchNormalization)                                                     ␣
↪

block_15_expand_relu        (None, 7, 7, 960)                0  ␣
↪block_15_expand_BN[0]…
(ReLU)                                                                   ␣
↪

block_15_depthwise          (None, 7, 7, 960)            8,640  ␣
↪block_15_expand_relu[…
(DepthwiseConv2D)                                                        ␣
↪

block_15_depthwise_BN       (None, 7, 7, 960)            3,840  ␣
↪block_15_depthwise[0]…
(BatchNormalization)                                                     ␣
↪

block_15_depthwise_relu     (None, 7, 7, 960)                0  ␣
↪block_15_depthwise_BN…
(ReLU)                                                                   ␣
↪

block_15_project (Conv2D)   (None, 7, 7, 160)          153,600  ␣
↪block_15_depthwise_re…

block_15_project_BN         (None, 7, 7, 160)              640  ␣
↪block_15_project[0][0]
(BatchNormalization)                                                     ␣
↪

block_15_add (Add)          (None, 7, 7, 160)                0  ␣
↪block_14_add[0][0],
                                                                         ␣
↪block_15_project_BN[0…

block_16_expand (Conv2D)    (None, 7, 7, 960)          153,600  ␣
↪block_15_add[0][0]
```

```
block_16_expand_BN          (None, 7, 7, 960)          3,840 ␣
↪block_16_expand[0][0]
(BatchNormalization)                                      ␣
↪

block_16_expand_relu        (None, 7, 7, 960)              0 ␣
↪block_16_expand_BN[0]…
(ReLU)                                                    ␣
↪

block_16_depthwise          (None, 7, 7, 960)          8,640 ␣
↪block_16_expand_relu[…
(DepthwiseConv2D)                                         ␣
↪

block_16_depthwise_BN       (None, 7, 7, 960)          3,840 ␣
↪block_16_depthwise[0]…
(BatchNormalization)                                      ␣
↪

block_16_depthwise_relu     (None, 7, 7, 960)              0 ␣
↪block_16_depthwise_BN…
(ReLU)                                                    ␣
↪

block_16_project (Conv2D)   (None, 7, 7, 320)        307,200 ␣
↪block_16_depthwise_re…

block_16_project_BN         (None, 7, 7, 320)          1,280 ␣
↪block_16_project[0][0]
(BatchNormalization)                                      ␣
↪

Conv_1 (Conv2D)             (None, 7, 7, 1280)       409,600 ␣
↪block_16_project_BN[0…

Conv_1_bn                   (None, 7, 7, 1280)         5,120 ␣
↪Conv_1[0][0]
(BatchNormalization)                                      ␣
↪

out_relu (ReLU)             (None, 7, 7, 1280)             0 ␣
↪Conv_1_bn[0][0]
```

```
global_average_pooling2d…   (None, 1280)                    0  ␣
↪out_relu[0][0]
(GlobalAveragePooling2D)                                       ␣
↪

dropout_4 (Dropout)         (None, 1280)                    0  ␣
↪global_average_poolin…

dense_4 (Dense)             (None, 128)                163,968  ␣
↪dropout_4[0][0]

dropout_5 (Dropout)         (None, 128)                     0  ␣
↪dense_4[0][0]

dense_5 (Dense)             (None, 5)                     645  ␣
↪dropout_5[0][0]


Total params: 2,422,597 (9.24 MB)

Trainable params: 164,613 (643.02 KB)

Non-trainable params: 2,257,984 (8.61 MB)
```

[ ]:
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator


train_dir = '/content/drive/MyDrive/Flowers_Divided/train'  # Path to your␣
 ↪train data
val_dir = '/content/drive/MyDrive/Flowers_Divided/val'      # Path to your␣
 ↪validation data

# Create ImageDataGenerators
train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=20,␣
 ↪width_shift_range=0.2, height_shift_range=0.2, horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir,␣
 ↪target_size=(224, 224), batch_size=32, class_mode='categorical')
val_generator = val_datagen.flow_from_directory(val_dir, target_size=(224,␣
 ↪224), batch_size=32, class_mode='categorical')
```

```
Found 1919 images belonging to 5 classes.
Found 686 images belonging to 5 classes.
```

[ ]:
```
# Train the model
history = model.fit(train_generator, epochs=10, validation_data=val_generator)
```

```
Epoch 1/10
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
```

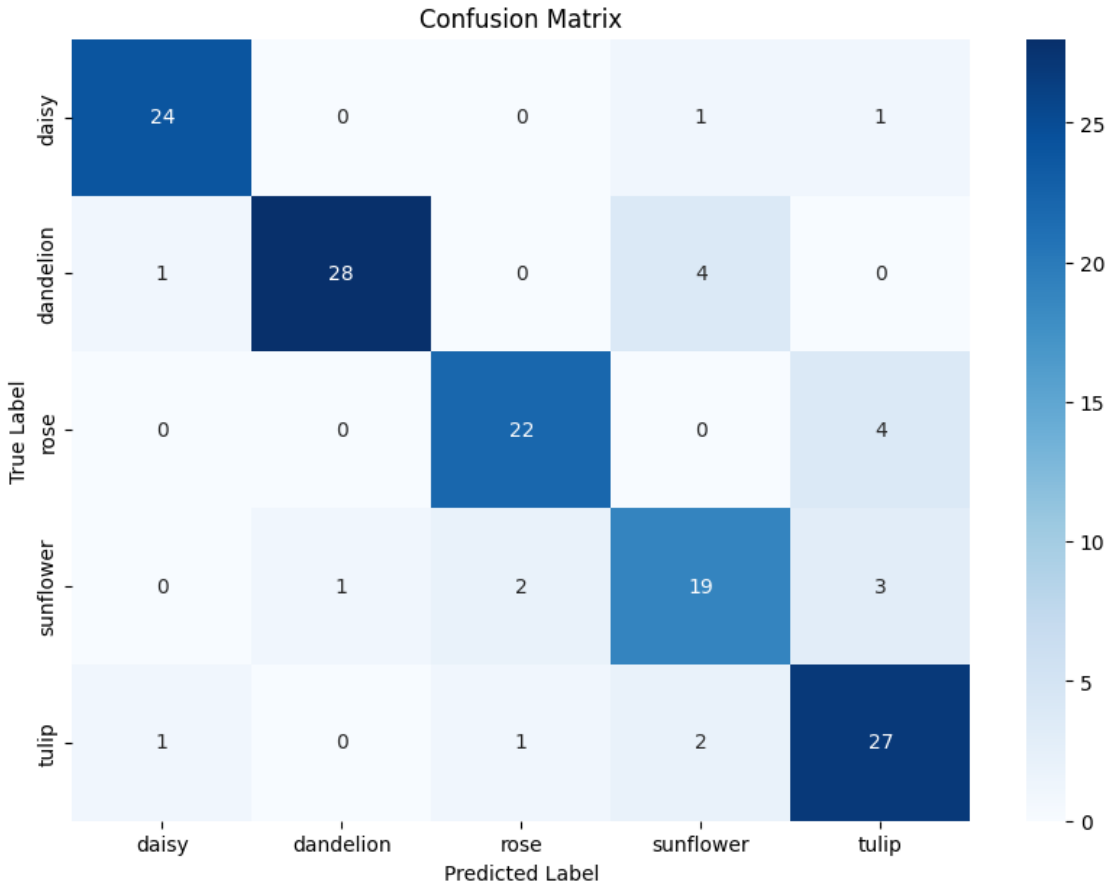
```
60/60             57s 730ms/step -
accuracy: 0.5499 - loss: 1.1626 - val_accuracy: 0.8397 - val_loss: 0.4573
Epoch 2/10
60/60             35s 523ms/step -
accuracy: 0.7890 - loss: 0.5565 - val_accuracy: 0.8703 - val_loss: 0.3778
Epoch 3/10
60/60             35s 531ms/step -
accuracy: 0.8240 - loss: 0.4652 - val_accuracy: 0.8557 - val_loss: 0.3841
Epoch 4/10
60/60             34s 512ms/step -
accuracy: 0.8439 - loss: 0.3984 - val_accuracy: 0.8907 - val_loss: 0.3281
Epoch 5/10
60/60             41s 521ms/step -
accuracy: 0.8489 - loss: 0.4177 - val_accuracy: 0.8717 - val_loss: 0.3581
Epoch 6/10
60/60             36s 536ms/step -
accuracy: 0.8642 - loss: 0.3809 - val_accuracy: 0.8950 - val_loss: 0.3194
Epoch 7/10
60/60             40s 528ms/step -
accuracy: 0.8739 - loss: 0.3457 - val_accuracy: 0.8950 - val_loss: 0.3073
Epoch 8/10
60/60             34s 515ms/step -
accuracy: 0.8950 - loss: 0.2812 - val_accuracy: 0.8965 - val_loss: 0.3214
Epoch 9/10
60/60             36s 543ms/step -
accuracy: 0.9021 - loss: 0.2899 - val_accuracy: 0.8834 - val_loss: 0.3295
Epoch 10/10
60/60             34s 520ms/step -
accuracy: 0.9072 - loss: 0.2762 - val_accuracy: 0.8980 - val_loss: 0.2977
```

[ ]:
```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_generator.
  ↪samples // batch_size)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4             1s 254ms/step -
accuracy: 0.8615 - loss: 0.4090
Test Loss: 0.44765931367874146
Test Accuracy: 0.8515625
```

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import load_model

# Generate predictions for the test data without specifying 'steps'
test_predictions = model.predict(test_generator, verbose=1)

# Get the predicted class labels
predicted_labels = np.argmax(test_predictions, axis=1)

# Get the true class labels from the test generator
true_labels = test_generator.classes

# Generate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=test_generator.
 ↪class_indices.keys(), yticklabels=test_generator.class_indices.keys())
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
5/5             8s 1s/step
```



Nice one

[ ]:
```
model.save("/content/drive/MyDrive/Flowers_Divided/MobileNetV2.keras")
```

3.2 fine-tune the MobileNetV2 model

Steps for Fine-tuning MobileNetV2 Train the Custom Head: Initially, freeze the base model and train only the custom head (already done in the previous step).

Unfreeze Layers: Unfreeze some of the deeper layers of the base model while keeping the earlier layers frozen (to preserve general features).

Lower Learning Rate: Use a smaller learning rate for fine-tuning to avoid destroying the pre-trained weights.

Re-train the Model: Train the model with the updated trainable layers.

[ ]:
```
# Unfreeze some layers of the base model
base_model.trainable = True  # Unfreeze the entire model for fine-tuning␣
↪(optional: unfreeze specific layers)
```

```
# Optionally, unfreeze specific layers (e.g., last 20 layers)
for layer in base_model.layers[:-20]:  # Freeze all layers except the last 20
    layer.trainable = False

# Compile the model with a lower learning rate
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),  # Lower␣
 ↪learning rate
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "functional_1"
```

Layer (type)	Output Shape	Param #	Connected␣ ↪to
input_layer_1 ↪ (InputLayer) ↪	(None, 224, 224, 3)	0	- ␣ ␣
Conv1 (Conv2D) ↪input_layer_1[0][0]	(None, 112, 112, 32)	864	␣
bn_Conv1 ↪Conv1[0][0] (BatchNormalization) ↪	(None, 112, 112, 32)	128	␣ ␣
Conv1_relu (ReLU) ↪bn_Conv1[0][0]	(None, 112, 112, 32)	0	␣
expanded_conv_depthwise ↪Conv1_relu[0][0] (DepthwiseConv2D) ↪	(None, 112, 112, 32)	288	␣ ␣
expanded_conv_depthwise_… ↪expanded_conv_depthwi… (BatchNormalization) ↪	(None, 112, 112, 32)	128	␣ ␣
expanded_conv_depthwise_… ↪expanded_conv_depthwi…	(None, 112, 112, 32)	0	␣

```
(ReLU)                                                                  ␣
↪
expanded_conv_project       (None, 112, 112, 16)          512 ␣
↪expanded_conv_depthwi…
(Conv2D)                                                                ␣
↪
expanded_conv_project_BN    (None, 112, 112, 16)           64 ␣
↪expanded_conv_project…
(BatchNormalization)                                                    ␣
↪
block_1_expand (Conv2D)     (None, 112, 112, 96)        1,536 ␣
↪expanded_conv_project…
block_1_expand_BN           (None, 112, 112, 96)          384 ␣
↪block_1_expand[0][0]
(BatchNormalization)                                                    ␣
↪
block_1_expand_relu         (None, 112, 112, 96)            0 ␣
↪block_1_expand_BN[0][…
(ReLU)                                                                  ␣
↪
block_1_pad                 (None, 113, 113, 96)            0 ␣
↪block_1_expand_relu[0…
(ZeroPadding2D)                                                         ␣
↪
block_1_depthwise           (None, 56, 56, 96)            864 ␣
↪block_1_pad[0][0]
(DepthwiseConv2D)                                                       ␣
↪
block_1_depthwise_BN        (None, 56, 56, 96)            384 ␣
↪block_1_depthwise[0][…
(BatchNormalization)                                                    ␣
↪
block_1_depthwise_relu      (None, 56, 56, 96)              0 ␣
↪block_1_depthwise_BN[…
(ReLU)                                                                  ␣
↪
```

```
block_1_project (Conv2D)     (None, 56, 56, 24)            2,304 ␣
↪block_1_depthwise_rel…

block_1_project_BN           (None, 56, 56, 24)               96 ␣
↪block_1_project[0][0]
(BatchNormalization)                                             ␣
↪

block_2_expand (Conv2D)      (None, 56, 56, 144)           3,456 ␣
↪block_1_project_BN[0]…

block_2_expand_BN            (None, 56, 56, 144)             576 ␣
↪block_2_expand[0][0]
(BatchNormalization)                                             ␣
↪

block_2_expand_relu          (None, 56, 56, 144)               0 ␣
↪block_2_expand_BN[0][…
(ReLU)                                                           ␣
↪

block_2_depthwise            (None, 56, 56, 144)           1,296 ␣
↪block_2_expand_relu[0…
(DepthwiseConv2D)                                                ␣
↪

block_2_depthwise_BN         (None, 56, 56, 144)             576 ␣
↪block_2_depthwise[0][…
(BatchNormalization)                                             ␣
↪

block_2_depthwise_relu       (None, 56, 56, 144)               0 ␣
↪block_2_depthwise_BN[…
(ReLU)                                                           ␣
↪

block_2_project (Conv2D)     (None, 56, 56, 24)            3,456 ␣
↪block_2_depthwise_rel…

block_2_project_BN           (None, 56, 56, 24)               96 ␣
↪block_2_project[0][0]
(BatchNormalization)                                             ␣
↪

block_2_add (Add)            (None, 56, 56, 24)                0 ␣
↪block_1_project_BN[0]…
```

```
↪block_2_project_BN[0]…                                                 ␣

block_3_expand (Conv2D)      (None, 56, 56, 144)           3,456       ␣
↪block_2_add[0][0]

block_3_expand_BN            (None, 56, 56, 144)             576       ␣
↪block_3_expand[0][0]
(BatchNormalization)                                                    ␣
↪

block_3_expand_relu          (None, 56, 56, 144)               0       ␣
↪block_3_expand_BN[0][…
(ReLU)                                                                  ␣
↪

block_3_pad                  (None, 57, 57, 144)               0       ␣
↪block_3_expand_relu[0…
(ZeroPadding2D)                                                         ␣
↪

block_3_depthwise            (None, 28, 28, 144)           1,296       ␣
↪block_3_pad[0][0]
(DepthwiseConv2D)                                                       ␣
↪

block_3_depthwise_BN         (None, 28, 28, 144)             576       ␣
↪block_3_depthwise[0][…
(BatchNormalization)                                                    ␣
↪

block_3_depthwise_relu       (None, 28, 28, 144)               0       ␣
↪block_3_depthwise_BN[…
(ReLU)                                                                  ␣
↪

block_3_project (Conv2D)     (None, 28, 28, 32)            4,608       ␣
↪block_3_depthwise_rel…

block_3_project_BN           (None, 28, 28, 32)              128       ␣
↪block_3_project[0][0]
(BatchNormalization)                                                    ␣
↪

block_4_expand (Conv2D)      (None, 28, 28, 192)           6,144       ␣
↪block_3_project_BN[0]…
```

```
block_4_expand_BN          (None, 28, 28, 192)          768 ␣
↪block_4_expand[0][0]
(BatchNormalization)                                        ␣
↪

block_4_expand_relu        (None, 28, 28, 192)            0 ␣
↪block_4_expand_BN[0][…
(ReLU)                                                      ␣
↪

block_4_depthwise          (None, 28, 28, 192)        1,728 ␣
↪block_4_expand_relu[0…
(DepthwiseConv2D)                                           ␣
↪

block_4_depthwise_BN       (None, 28, 28, 192)          768 ␣
↪block_4_depthwise[0][…
(BatchNormalization)                                        ␣
↪

block_4_depthwise_relu     (None, 28, 28, 192)            0 ␣
↪block_4_depthwise_BN[…
(ReLU)                                                      ␣
↪

block_4_project (Conv2D)   (None, 28, 28, 32)         6,144 ␣
↪block_4_depthwise_rel…

block_4_project_BN         (None, 28, 28, 32)           128 ␣
↪block_4_project[0][0]
(BatchNormalization)                                        ␣
↪

block_4_add (Add)          (None, 28, 28, 32)             0 ␣
↪block_3_project_BN[0]…
                                                            ␣
↪block_4_project_BN[0]…

block_5_expand (Conv2D)    (None, 28, 28, 192)        6,144 ␣
↪block_4_add[0][0]

block_5_expand_BN          (None, 28, 28, 192)          768 ␣
↪block_5_expand[0][0]
(BatchNormalization)                                        ␣
↪
```

```
block_5_expand_relu          (None, 28, 28, 192)          0  ␣
↪block_5_expand_BN[0][…
(ReLU)                                                     ␣
↪

block_5_depthwise            (None, 28, 28, 192)          1,728  ␣
↪block_5_expand_relu[0…
(DepthwiseConv2D)                                          ␣
↪

block_5_depthwise_BN         (None, 28, 28, 192)          768  ␣
↪block_5_depthwise[0][…
(BatchNormalization)                                       ␣
↪

block_5_depthwise_relu       (None, 28, 28, 192)          0  ␣
↪block_5_depthwise_BN[…
(ReLU)                                                     ␣
↪

block_5_project (Conv2D)     (None, 28, 28, 32)           6,144  ␣
↪block_5_depthwise_rel…

block_5_project_BN           (None, 28, 28, 32)           128  ␣
↪block_5_project[0][0]
(BatchNormalization)                                       ␣
↪

block_5_add (Add)            (None, 28, 28, 32)           0  ␣
↪block_4_add[0][0],
                                                           ␣
↪block_5_project_BN[0]…

block_6_expand (Conv2D)      (None, 28, 28, 192)          6,144  ␣
↪block_5_add[0][0]

block_6_expand_BN            (None, 28, 28, 192)          768  ␣
↪block_6_expand[0][0]
(BatchNormalization)                                       ␣
↪

block_6_expand_relu          (None, 28, 28, 192)          0  ␣
↪block_6_expand_BN[0][…
(ReLU)                                                     ␣
↪
```

```
block_6_pad                  (None, 29, 29, 192)                0 ␣
↪block_6_expand_relu[0…
(ZeroPadding2D)                                                   ␣
↪

block_6_depthwise            (None, 14, 14, 192)            1,728 ␣
↪block_6_pad[0][0]
(DepthwiseConv2D)                                                 ␣
↪

block_6_depthwise_BN         (None, 14, 14, 192)              768 ␣
↪block_6_depthwise[0][…
(BatchNormalization)                                              ␣
↪

block_6_depthwise_relu       (None, 14, 14, 192)                0 ␣
↪block_6_depthwise_BN[…
(ReLU)                                                            ␣
↪

block_6_project (Conv2D)     (None, 14, 14, 64)            12,288 ␣
↪block_6_depthwise_rel…

block_6_project_BN           (None, 14, 14, 64)               256 ␣
↪block_6_project[0][0]
(BatchNormalization)                                              ␣
↪

block_7_expand (Conv2D)      (None, 14, 14, 384)           24,576 ␣
↪block_6_project_BN[0]…

block_7_expand_BN            (None, 14, 14, 384)            1,536 ␣
↪block_7_expand[0][0]
(BatchNormalization)                                              ␣
↪

block_7_expand_relu          (None, 14, 14, 384)                0 ␣
↪block_7_expand_BN[0][…
(ReLU)                                                            ␣
↪

block_7_depthwise            (None, 14, 14, 384)            3,456 ␣
↪block_7_expand_relu[0…
(DepthwiseConv2D)                                                 ␣
↪
```

```
block_7_depthwise_BN        (None, 14, 14, 384)              1,536 ␣
↪block_7_depthwise[0][…
(BatchNormalization)                                               ␣
↪

block_7_depthwise_relu      (None, 14, 14, 384)                  0 ␣
↪block_7_depthwise_BN[…
(ReLU)                                                             ␣
↪

block_7_project (Conv2D)    (None, 14, 14, 64)              24,576 ␣
↪block_7_depthwise_rel…

block_7_project_BN          (None, 14, 14, 64)                 256 ␣
↪block_7_project[0][0]
(BatchNormalization)                                               ␣
↪

block_7_add (Add)           (None, 14, 14, 64)                   0 ␣
↪block_6_project_BN[0]…
                                                                   ␣
↪block_7_project_BN[0]…

block_8_expand (Conv2D)     (None, 14, 14, 384)             24,576 ␣
↪block_7_add[0][0]

block_8_expand_BN           (None, 14, 14, 384)              1,536 ␣
↪block_8_expand[0][0]
(BatchNormalization)                                               ␣
↪

block_8_expand_relu         (None, 14, 14, 384)                  0 ␣
↪block_8_expand_BN[0][…
(ReLU)                                                             ␣
↪

block_8_depthwise           (None, 14, 14, 384)              3,456 ␣
↪block_8_expand_relu[0…
(DepthwiseConv2D)                                                  ␣
↪

block_8_depthwise_BN        (None, 14, 14, 384)              1,536 ␣
↪block_8_depthwise[0][…
(BatchNormalization)                                               ␣
↪
```

```
block_8_depthwise_relu      (None, 14, 14, 384)               0 ␣
↪block_8_depthwise_BN[…
(ReLU)                                                        ␣
↪

block_8_project (Conv2D)    (None, 14, 14, 64)           24,576 ␣
↪block_8_depthwise_rel…

block_8_project_BN          (None, 14, 14, 64)              256 ␣
↪block_8_project[0][0]
(BatchNormalization)                                          ␣
↪

block_8_add (Add)           (None, 14, 14, 64)                0 ␣
↪block_7_add[0][0],
                                                              ␣
↪block_8_project_BN[0]…

block_9_expand (Conv2D)     (None, 14, 14, 384)          24,576 ␣
↪block_8_add[0][0]

block_9_expand_BN           (None, 14, 14, 384)           1,536 ␣
↪block_9_expand[0][0]
(BatchNormalization)                                          ␣
↪

block_9_expand_relu         (None, 14, 14, 384)               0 ␣
↪block_9_expand_BN[0][…
(ReLU)                                                        ␣
↪

block_9_depthwise           (None, 14, 14, 384)           3,456 ␣
↪block_9_expand_relu[0…
(DepthwiseConv2D)                                             ␣
↪

block_9_depthwise_BN        (None, 14, 14, 384)           1,536 ␣
↪block_9_depthwise[0][…
(BatchNormalization)                                          ␣
↪

block_9_depthwise_relu      (None, 14, 14, 384)               0 ␣
↪block_9_depthwise_BN[…
(ReLU)                                                        ␣
↪
```

```
block_9_project (Conv2D)    (None, 14, 14, 64)              24,576 ␣
↪block_9_depthwise_rel…

block_9_project_BN          (None, 14, 14, 64)                 256 ␣
↪block_9_project[0][0]
(BatchNormalization)                                                ␣
↪

block_9_add (Add)           (None, 14, 14, 64)                   0 ␣
↪block_8_add[0][0],
                                                                    ␣
↪block_9_project_BN[0][0]

block_10_expand (Conv2D)    (None, 14, 14, 384)             24,576 ␣
↪block_9_add[0][0]

block_10_expand_BN          (None, 14, 14, 384)              1,536 ␣
↪block_10_expand[0][0]
(BatchNormalization)                                                ␣
↪

block_10_expand_relu        (None, 14, 14, 384)                  0 ␣
↪block_10_expand_BN[0][0]
(ReLU)                                                              ␣
↪

block_10_depthwise          (None, 14, 14, 384)              3,456 ␣
↪block_10_expand_relu[0]
(DepthwiseConv2D)                                                   ␣
↪

block_10_depthwise_BN       (None, 14, 14, 384)              1,536 ␣
↪block_10_depthwise[0][0]
(BatchNormalization)                                                ␣
↪

block_10_depthwise_relu     (None, 14, 14, 384)                  0 ␣
↪block_10_depthwise_BN[0]
(ReLU)                                                              ␣
↪

block_10_project (Conv2D)   (None, 14, 14, 96)              36,864 ␣
↪block_10_depthwise_re…
```

```
block_10_project_BN          (None, 14, 14, 96)              384 ␣
↪block_10_project[0][0]
(BatchNormalization)                                             ␣
↪

block_11_expand (Conv2D)     (None, 14, 14, 576)           55,296 ␣
↪block_10_project_BN[0…

block_11_expand_BN           (None, 14, 14, 576)            2,304 ␣
↪block_11_expand[0][0]
(BatchNormalization)                                             ␣
↪

block_11_expand_relu         (None, 14, 14, 576)                0 ␣
↪block_11_expand_BN[0]…
(ReLU)                                                           ␣
↪

block_11_depthwise           (None, 14, 14, 576)            5,184 ␣
↪block_11_expand_relu[…
(DepthwiseConv2D)                                                ␣
↪

block_11_depthwise_BN        (None, 14, 14, 576)            2,304 ␣
↪block_11_depthwise[0]…
(BatchNormalization)                                             ␣
↪

block_11_depthwise_relu      (None, 14, 14, 576)                0 ␣
↪block_11_depthwise_BN…
(ReLU)                                                           ␣
↪

block_11_project (Conv2D)    (None, 14, 14, 96)            55,296 ␣
↪block_11_depthwise_re…

block_11_project_BN          (None, 14, 14, 96)              384 ␣
↪block_11_project[0][0]
(BatchNormalization)                                             ␣
↪

block_11_add (Add)           (None, 14, 14, 96)                0 ␣
↪block_10_project_BN[0…
                                                                 ␣
↪block_11_project_BN[0…
```

```
block_12_expand (Conv2D)    (None, 14, 14, 576)          55,296 ␣
↪block_11_add[0][0]

block_12_expand_BN          (None, 14, 14, 576)           2,304 ␣
↪block_12_expand[0][0]
(BatchNormalization)                                            ␣
↪

block_12_expand_relu        (None, 14, 14, 576)               0 ␣
↪block_12_expand_BN[0]…
(ReLU)                                                          ␣
↪

block_12_depthwise          (None, 14, 14, 576)           5,184 ␣
↪block_12_expand_relu[…
(DepthwiseConv2D)                                               ␣
↪

block_12_depthwise_BN       (None, 14, 14, 576)           2,304 ␣
↪block_12_depthwise[0]…
(BatchNormalization)                                            ␣
↪

block_12_depthwise_relu     (None, 14, 14, 576)               0 ␣
↪block_12_depthwise_BN…
(ReLU)                                                          ␣
↪

block_12_project (Conv2D)   (None, 14, 14, 96)           55,296 ␣
↪block_12_depthwise_re…

block_12_project_BN         (None, 14, 14, 96)              384 ␣
↪block_12_project[0][0]
(BatchNormalization)                                            ␣
↪

block_12_add (Add)          (None, 14, 14, 96)                0 ␣
↪block_11_add[0][0],
                                                                ␣
↪block_12_project_BN[0…

block_13_expand (Conv2D)    (None, 14, 14, 576)          55,296 ␣
↪block_12_add[0][0]

block_13_expand_BN          (None, 14, 14, 576)           2,304 ␣
↪block_13_expand[0][0]
```

```
(BatchNormalization)                                                    ␣
↪
block_13_expand_relu        (None, 14, 14, 576)              0          ␣
↪block_13_expand_BN[0]…
(ReLU)                                                                  ␣
↪
block_13_pad                (None, 15, 15, 576)              0          ␣
↪block_13_expand_relu[…
(ZeroPadding2D)                                                         ␣
↪
block_13_depthwise          (None, 7, 7, 576)                5,184      ␣
↪block_13_pad[0][0]
(DepthwiseConv2D)                                                       ␣
↪
block_13_depthwise_BN       (None, 7, 7, 576)                2,304      ␣
↪block_13_depthwise[0]…
(BatchNormalization)                                                    ␣
↪
block_13_depthwise_relu     (None, 7, 7, 576)                0          ␣
↪block_13_depthwise_BN…
(ReLU)                                                                  ␣
↪
block_13_project (Conv2D)   (None, 7, 7, 160)                92,160     ␣
↪block_13_depthwise_re…
block_13_project_BN         (None, 7, 7, 160)                640        ␣
↪block_13_project[0][0]
(BatchNormalization)                                                    ␣
↪
block_14_expand (Conv2D)    (None, 7, 7, 960)                153,600    ␣
↪block_13_project_BN[0…
block_14_expand_BN          (None, 7, 7, 960)                3,840      ␣
↪block_14_expand[0][0]
(BatchNormalization)                                                    ␣
↪
block_14_expand_relu        (None, 7, 7, 960)                0          ␣
↪block_14_expand_BN[0]…
```

```
(ReLU)                                                                  ␣
↪
block_14_depthwise          (None, 7, 7, 960)             8,640       ␣
↪block_14_expand_relu[…
(DepthwiseConv2D)                                                       ␣
↪
block_14_depthwise_BN       (None, 7, 7, 960)             3,840       ␣
↪block_14_depthwise[0]…
(BatchNormalization)                                                    ␣
↪
block_14_depthwise_relu     (None, 7, 7, 960)                 0       ␣
↪block_14_depthwise_BN…
(ReLU)                                                                  ␣
↪
block_14_project (Conv2D)   (None, 7, 7, 160)           153,600       ␣
↪block_14_depthwise_re…
block_14_project_BN         (None, 7, 7, 160)               640       ␣
↪block_14_project[0][0]
(BatchNormalization)                                                    ␣
↪
block_14_add (Add)          (None, 7, 7, 160)                 0       ␣
↪block_13_project_BN[0…
                                                                        ␣
↪block_14_project_BN[0…
block_15_expand (Conv2D)    (None, 7, 7, 960)           153,600       ␣
↪block_14_add[0][0]
block_15_expand_BN          (None, 7, 7, 960)             3,840       ␣
↪block_15_expand[0][0]
(BatchNormalization)                                                    ␣
↪
block_15_expand_relu        (None, 7, 7, 960)                 0       ␣
↪block_15_expand_BN[0]…
(ReLU)                                                                  ␣
↪
block_15_depthwise          (None, 7, 7, 960)             8,640       ␣
↪block_15_expand_relu[…
```

```
(DepthwiseConv2D)                                                              ␣
↪
block_15_depthwise_BN       (None, 7, 7, 960)                3,840 ␣
↪block_15_depthwise[0]…
(BatchNormalization)                                                           ␣
↪
block_15_depthwise_relu     (None, 7, 7, 960)                    0 ␣
↪block_15_depthwise_BN…
(ReLU)                                                                         ␣
↪
block_15_project (Conv2D)   (None, 7, 7, 160)              153,600 ␣
↪block_15_depthwise_re…
block_15_project_BN         (None, 7, 7, 160)                  640 ␣
↪block_15_project[0][0]
(BatchNormalization)                                                           ␣
↪
block_15_add (Add)          (None, 7, 7, 160)                    0 ␣
↪block_14_add[0][0],
                                                                               ␣
↪block_15_project_BN[0…
block_16_expand (Conv2D)    (None, 7, 7, 960)              153,600 ␣
↪block_15_add[0][0]
block_16_expand_BN          (None, 7, 7, 960)                3,840 ␣
↪block_16_expand[0][0]
(BatchNormalization)                                                           ␣
↪
block_16_expand_relu        (None, 7, 7, 960)                    0 ␣
↪block_16_expand_BN[0]…
(ReLU)                                                                         ␣
↪
block_16_depthwise          (None, 7, 7, 960)                8,640 ␣
↪block_16_expand_relu[…
(DepthwiseConv2D)                                                              ␣
↪
block_16_depthwise_BN       (None, 7, 7, 960)                3,840 ␣
↪block_16_depthwise[0]…
```

```
(BatchNormalization)                                                    ␣
↪

block_16_depthwise_relu     (None, 7, 7, 960)                 0       ␣
↪block_16_depthwise_BN…
(ReLU)                                                                  ␣
↪

block_16_project (Conv2D)   (None, 7, 7, 320)               307,200   ␣
↪block_16_depthwise_re…

block_16_project_BN         (None, 7, 7, 320)               1,280     ␣
↪block_16_project[0][0]
(BatchNormalization)                                                    ␣
↪

Conv_1 (Conv2D)             (None, 7, 7, 1280)              409,600   ␣
↪block_16_project_BN[0…

Conv_1_bn                   (None, 7, 7, 1280)              5,120     ␣
↪Conv_1[0][0]
(BatchNormalization)                                                    ␣
↪

out_relu (ReLU)             (None, 7, 7, 1280)                0       ␣
↪Conv_1_bn[0][0]

global_average_pooling2d    (None, 1280)                      0       ␣
↪out_relu[0][0]
(GlobalAveragePooling2D)                                                ␣
↪

dropout (Dropout)           (None, 1280)                      0       ␣
↪global_average_poolin…

dense_2 (Dense)             (None, 128)                     163,968   ␣
↪dropout[0][0]

dropout_1 (Dropout)         (None, 128)                       0       ␣
↪dense_2[0][0]

dense_3 (Dense)             (None, 5)                         645     ␣
↪dropout_1[0][0]


Total params: 2,422,597 (9.24 MB)
```

Trainable params: 1,370,693 (5.23 MB)

Non-trainable params: 1,051,904 (4.01 MB)

[ ]:
```
for i, layer in enumerate(model.layers):
    print(f"Layer {i}: {layer.name}, Trainable: {layer.trainable}")
```

```
Layer 0: input_layer_1, Trainable: False
Layer 1: Conv1, Trainable: False
Layer 2: bn_Conv1, Trainable: False
Layer 3: Conv1_relu, Trainable: False
Layer 4: expanded_conv_depthwise, Trainable: False
Layer 5: expanded_conv_depthwise_BN, Trainable: False
Layer 6: expanded_conv_depthwise_relu, Trainable: False
Layer 7: expanded_conv_project, Trainable: False
Layer 8: expanded_conv_project_BN, Trainable: False
Layer 9: block_1_expand, Trainable: False
Layer 10: block_1_expand_BN, Trainable: False
Layer 11: block_1_expand_relu, Trainable: False
Layer 12: block_1_pad, Trainable: False
Layer 13: block_1_depthwise, Trainable: False
Layer 14: block_1_depthwise_BN, Trainable: False
Layer 15: block_1_depthwise_relu, Trainable: False
Layer 16: block_1_project, Trainable: False
Layer 17: block_1_project_BN, Trainable: False
Layer 18: block_2_expand, Trainable: False
Layer 19: block_2_expand_BN, Trainable: False
Layer 20: block_2_expand_relu, Trainable: False
Layer 21: block_2_depthwise, Trainable: False
Layer 22: block_2_depthwise_BN, Trainable: False
Layer 23: block_2_depthwise_relu, Trainable: False
Layer 24: block_2_project, Trainable: False
Layer 25: block_2_project_BN, Trainable: False
Layer 26: block_2_add, Trainable: False
Layer 27: block_3_expand, Trainable: False
Layer 28: block_3_expand_BN, Trainable: False
Layer 29: block_3_expand_relu, Trainable: False
Layer 30: block_3_pad, Trainable: False
Layer 31: block_3_depthwise, Trainable: False
Layer 32: block_3_depthwise_BN, Trainable: False
Layer 33: block_3_depthwise_relu, Trainable: False
Layer 34: block_3_project, Trainable: False
Layer 35: block_3_project_BN, Trainable: False
Layer 36: block_4_expand, Trainable: False
Layer 37: block_4_expand_BN, Trainable: False
```

```
Layer 38: block_4_expand_relu, Trainable: False
Layer 39: block_4_depthwise, Trainable: False
Layer 40: block_4_depthwise_BN, Trainable: False
Layer 41: block_4_depthwise_relu, Trainable: False
Layer 42: block_4_project, Trainable: False
Layer 43: block_4_project_BN, Trainable: False
Layer 44: block_4_add, Trainable: False
Layer 45: block_5_expand, Trainable: False
Layer 46: block_5_expand_BN, Trainable: False
Layer 47: block_5_expand_relu, Trainable: False
Layer 48: block_5_depthwise, Trainable: False
Layer 49: block_5_depthwise_BN, Trainable: False
Layer 50: block_5_depthwise_relu, Trainable: False
Layer 51: block_5_project, Trainable: False
Layer 52: block_5_project_BN, Trainable: False
Layer 53: block_5_add, Trainable: False
Layer 54: block_6_expand, Trainable: False
Layer 55: block_6_expand_BN, Trainable: False
Layer 56: block_6_expand_relu, Trainable: False
Layer 57: block_6_pad, Trainable: False
Layer 58: block_6_depthwise, Trainable: False
Layer 59: block_6_depthwise_BN, Trainable: False
Layer 60: block_6_depthwise_relu, Trainable: False
Layer 61: block_6_project, Trainable: False
Layer 62: block_6_project_BN, Trainable: False
Layer 63: block_7_expand, Trainable: False
Layer 64: block_7_expand_BN, Trainable: False
Layer 65: block_7_expand_relu, Trainable: False
Layer 66: block_7_depthwise, Trainable: False
Layer 67: block_7_depthwise_BN, Trainable: False
Layer 68: block_7_depthwise_relu, Trainable: False
Layer 69: block_7_project, Trainable: False
Layer 70: block_7_project_BN, Trainable: False
Layer 71: block_7_add, Trainable: False
Layer 72: block_8_expand, Trainable: False
Layer 73: block_8_expand_BN, Trainable: False
Layer 74: block_8_expand_relu, Trainable: False
Layer 75: block_8_depthwise, Trainable: False
Layer 76: block_8_depthwise_BN, Trainable: False
Layer 77: block_8_depthwise_relu, Trainable: False
Layer 78: block_8_project, Trainable: False
Layer 79: block_8_project_BN, Trainable: False
Layer 80: block_8_add, Trainable: False
Layer 81: block_9_expand, Trainable: False
Layer 82: block_9_expand_BN, Trainable: False
Layer 83: block_9_expand_relu, Trainable: False
Layer 84: block_9_depthwise, Trainable: False
Layer 85: block_9_depthwise_BN, Trainable: False
```

```
Layer 86: block_9_depthwise_relu, Trainable: False
Layer 87: block_9_project, Trainable: False
Layer 88: block_9_project_BN, Trainable: False
Layer 89: block_9_add, Trainable: False
Layer 90: block_10_expand, Trainable: False
Layer 91: block_10_expand_BN, Trainable: False
Layer 92: block_10_expand_relu, Trainable: False
Layer 93: block_10_depthwise, Trainable: False
Layer 94: block_10_depthwise_BN, Trainable: False
Layer 95: block_10_depthwise_relu, Trainable: False
Layer 96: block_10_project, Trainable: False
Layer 97: block_10_project_BN, Trainable: False
Layer 98: block_11_expand, Trainable: False
Layer 99: block_11_expand_BN, Trainable: False
Layer 100: block_11_expand_relu, Trainable: False
Layer 101: block_11_depthwise, Trainable: False
Layer 102: block_11_depthwise_BN, Trainable: False
Layer 103: block_11_depthwise_relu, Trainable: False
Layer 104: block_11_project, Trainable: False
Layer 105: block_11_project_BN, Trainable: False
Layer 106: block_11_add, Trainable: False
Layer 107: block_12_expand, Trainable: False
Layer 108: block_12_expand_BN, Trainable: False
Layer 109: block_12_expand_relu, Trainable: False
Layer 110: block_12_depthwise, Trainable: False
Layer 111: block_12_depthwise_BN, Trainable: False
Layer 112: block_12_depthwise_relu, Trainable: False
Layer 113: block_12_project, Trainable: False
Layer 114: block_12_project_BN, Trainable: False
Layer 115: block_12_add, Trainable: False
Layer 116: block_13_expand, Trainable: False
Layer 117: block_13_expand_BN, Trainable: False
Layer 118: block_13_expand_relu, Trainable: False
Layer 119: block_13_pad, Trainable: False
Layer 120: block_13_depthwise, Trainable: False
Layer 121: block_13_depthwise_BN, Trainable: False
Layer 122: block_13_depthwise_relu, Trainable: False
Layer 123: block_13_project, Trainable: False
Layer 124: block_13_project_BN, Trainable: False
Layer 125: block_14_expand, Trainable: False
Layer 126: block_14_expand_BN, Trainable: False
Layer 127: block_14_expand_relu, Trainable: False
Layer 128: block_14_depthwise, Trainable: False
Layer 129: block_14_depthwise_BN, Trainable: False
Layer 130: block_14_depthwise_relu, Trainable: False
Layer 131: block_14_project, Trainable: False
Layer 132: block_14_project_BN, Trainable: False
Layer 133: block_14_add, Trainable: False
```

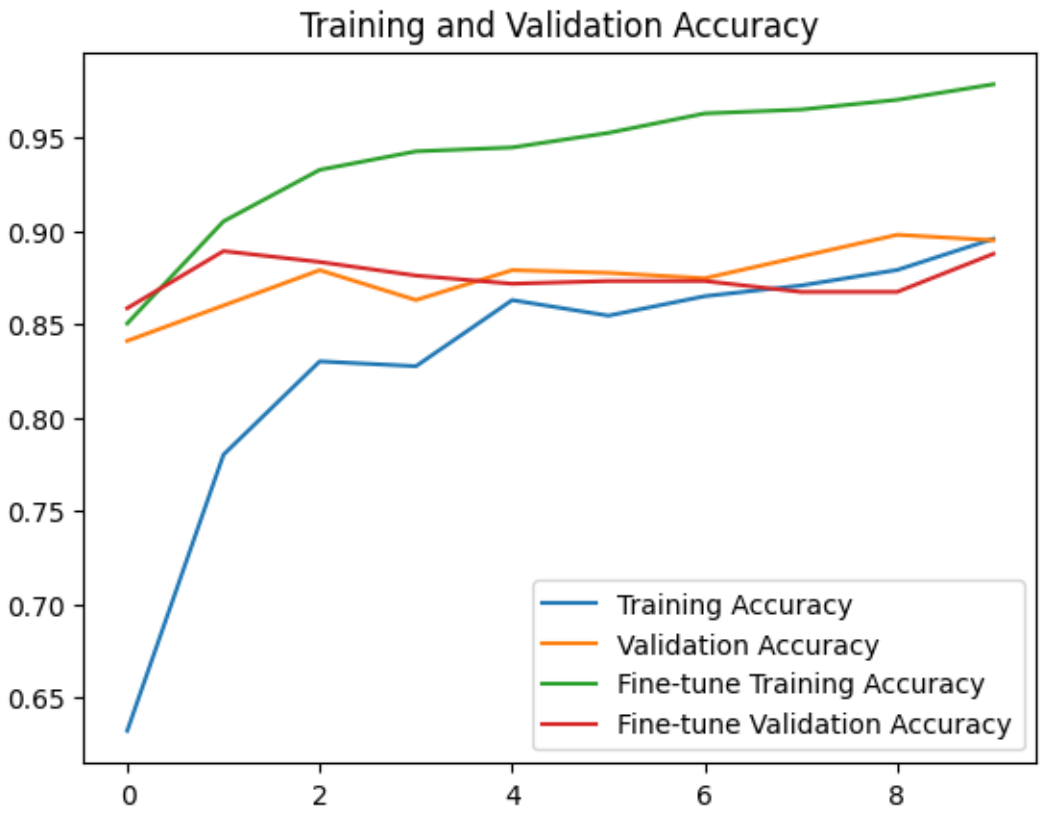
```
Layer 134: block_15_expand, Trainable: True
Layer 135: block_15_expand_BN, Trainable: True
Layer 136: block_15_expand_relu, Trainable: True
Layer 137: block_15_depthwise, Trainable: True
Layer 138: block_15_depthwise_BN, Trainable: True
Layer 139: block_15_depthwise_relu, Trainable: True
Layer 140: block_15_project, Trainable: True
Layer 141: block_15_project_BN, Trainable: True
Layer 142: block_15_add, Trainable: True
Layer 143: block_16_expand, Trainable: True
Layer 144: block_16_expand_BN, Trainable: True
Layer 145: block_16_expand_relu, Trainable: True
Layer 146: block_16_depthwise, Trainable: True
Layer 147: block_16_depthwise_BN, Trainable: True
Layer 148: block_16_depthwise_relu, Trainable: True
Layer 149: block_16_project, Trainable: True
Layer 150: block_16_project_BN, Trainable: True
Layer 151: Conv_1, Trainable: True
Layer 152: Conv_1_bn, Trainable: True
Layer 153: out_relu, Trainable: True
Layer 154: global_average_pooling2d, Trainable: True
Layer 155: dropout, Trainable: True
Layer 156: dense_2, Trainable: True
Layer 157: dropout_1, Trainable: True
Layer 158: dense_3, Trainable: True
```

[ ]:
```
# Re-train the model
history_fine_tune = model.fit(train_generator,
                              epochs=10,  # Add more epochs for fine-tuning
                              validation_data=val_generator)
```

```
Epoch 1/10
60/60             65s 791ms/step -
accuracy: 0.8386 - loss: 0.4489 - val_accuracy: 0.8586 - val_loss: 0.4909
Epoch 2/10
60/60             35s 534ms/step -
accuracy: 0.9119 - loss: 0.2598 - val_accuracy: 0.8892 - val_loss: 0.3831
Epoch 3/10
60/60             41s 541ms/step -
accuracy: 0.9332 - loss: 0.2067 - val_accuracy: 0.8834 - val_loss: 0.4602
Epoch 4/10
60/60             36s 542ms/step -
accuracy: 0.9444 - loss: 0.1809 - val_accuracy: 0.8761 - val_loss: 0.4500
Epoch 5/10
60/60             42s 560ms/step -
accuracy: 0.9521 - loss: 0.1361 - val_accuracy: 0.8717 - val_loss: 0.4385
Epoch 6/10
60/60             37s 572ms/step -
```

```
accuracy: 0.9581 - loss: 0.1139 - val_accuracy: 0.8732 - val_loss: 0.5043
Epoch 7/10
60/60             40s 546ms/step -
accuracy: 0.9586 - loss: 0.1278 - val_accuracy: 0.8732 - val_loss: 0.5187
Epoch 8/10
60/60             37s 566ms/step -
accuracy: 0.9682 - loss: 0.1029 - val_accuracy: 0.8673 - val_loss: 0.5314
Epoch 9/10
60/60             35s 538ms/step -
accuracy: 0.9698 - loss: 0.0847 - val_accuracy: 0.8673 - val_loss: 0.5903
Epoch 10/10
60/60             41s 537ms/step -
accuracy: 0.9820 - loss: 0.0596 - val_accuracy: 0.8878 - val_loss: 0.5326
```

[ ]:
```python
import matplotlib.pyplot as plt

# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.plot(history_fine_tune.history['accuracy'], label='Fine-tune Training␣
↪Accuracy')
plt.plot(history_fine_tune.history['val_accuracy'], label='Fine-tune Validation␣
↪Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')
plt.show()
```



[ ]:
```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_generator, steps=test_generator.
  ↪samples // batch_size)
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
4/4             1s 233ms/step -
accuracy: 0.8521 - loss: 0.5830
Test Loss: 0.7566991448402405
Test Accuracy: 0.8125
```

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import load_model

# Generate predictions for the test data without specifying 'steps'
test_predictions = model.predict(test_generator, verbose=1)

# Get the predicted class labels
```

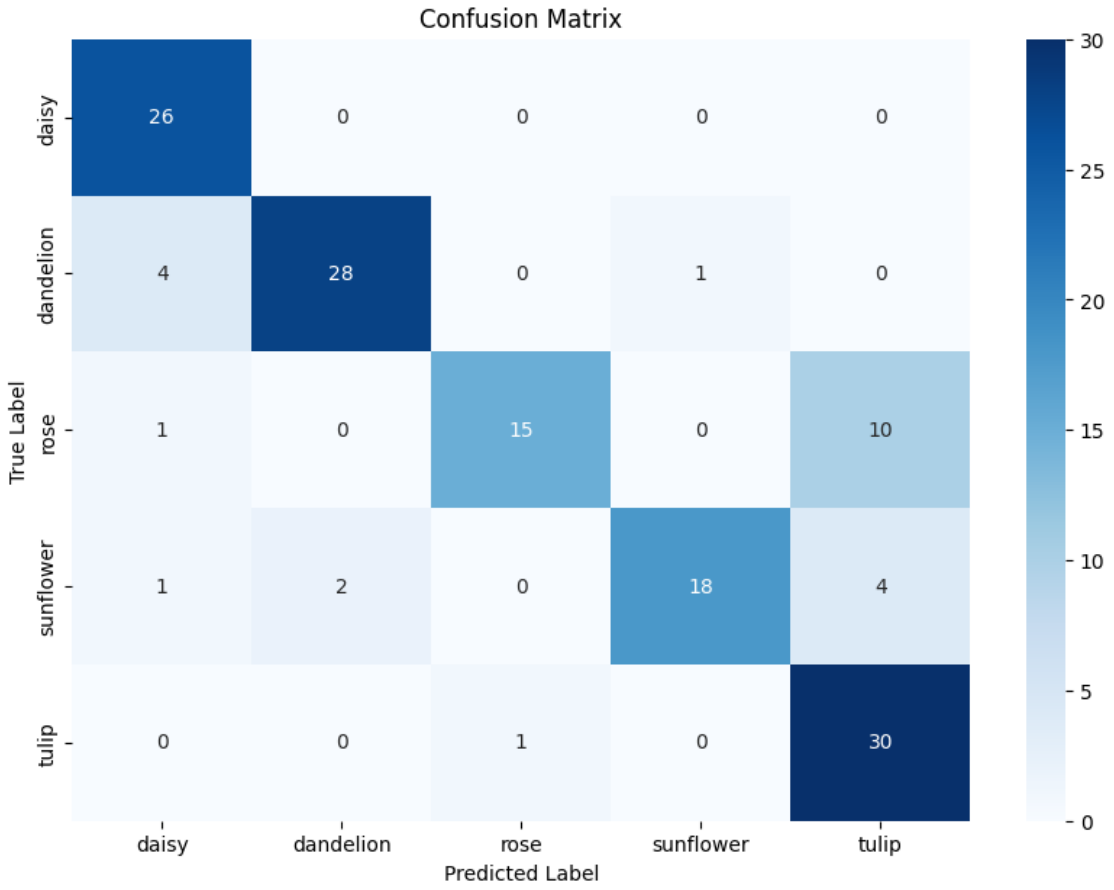
```
predicted_labels = np.argmax(test_predictions, axis=1)

# Get the true class labels from the test generator
true_labels = test_generator.classes

# Generate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=test_generator.
 ↪class_indices.keys(), yticklabels=test_generator.class_indices.keys())
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
WARNING:tensorflow:5 out of the last 11 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x7e0cec2b75b0> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

```
4/5            0s 113ms/step
```

```
WARNING:tensorflow:5 out of the last 11 calls to <function
TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at
0x7e0cec2b75b0> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has reduce_retracing=True option that
can avoid unnecessary retracing. For (3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

```
5/5            6s 785ms/step
```



[ ]:
```
model.save("/content/drive/MyDrive/Flowers_Divided/MobileNetV2_fine_tune.keras")
```

3.3 Pre-trained ResNet50

[ ]:
```
train_dir = '/content/drive/MyDrive/Flowers_Divided/train'  # Path to your␣
 ↪train data
val_dir = '/content/drive/MyDrive/Flowers_Divided/val'      # Path to your␣
 ↪validation data
test_dir = '/content/drive/MyDrive/Flowers_Divided/test'
```

[ ]:
```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import preprocess_input as␣
 ↪resnet_preprocess
# Data generators
train_datagen = ImageDataGenerator(
```

```
    preprocessing_function=resnet_preprocess,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

val_generator = train_datagen.flow_from_directory(
    val_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 1919 images belonging to 5 classes.
Found 686 images belonging to 5 classes.
```

[ ]:
```
# Model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224,␣
 ↪224, 3))
base_model.trainable = False

model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(5, activation='softmax')
])

model.compile(optimizer=Adam(learning_rate=1e-4),␣
 ↪loss='categorical_crossentropy', metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_2"
```

```
Layer (type)                    Output Shape
↪Param #

resnet50 (Functional)           (None, 7, 7, 2048)
↪23,587,712

global_average_pooling2d_2      (None, 2048)
↪  0
(GlobalAveragePooling2D)
↪

dense_4 (Dense)                 (None, 256)
↪524,544

dropout_2 (Dropout)             (None, 256)
↪  0

dense_5 (Dense)                 (None, 5)
↪1,285


Total params: 24,113,541 (91.99 MB)

Trainable params: 525,829 (2.01 MB)

Non-trainable params: 23,587,712 (89.98 MB)
```

3.3.1 Train

[ ]:
```
# Train
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,␣
 ↪ModelCheckpoint

LEARNING_RATE = 1e-4
early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.
 ↪2, patience=3, min_lr=1e-6)
checkpoint = ModelCheckpoint(
    '/content/drive/MyDrive/flower_data/ResNet50.keras',              # Filepath␣
 ↪to save the model
    monitor='val_accuracy',      # Metric to monitor (e.g., validation accuracy)
    save_best_only=True,         # Save only the best model
```

```
    mode='max',                    # Maximize the monitored metric (e.g.,␣
 ↪accuracy)
    verbose=1                      # Print a message when saving the model
)
history = model.fit(train_generator, validation_data=val_generator,␣
 ↪epochs=25,callbacks=[early_stopping, reduce_lr,checkpoint])
```

```
Epoch 1/25
60/60             0s 476ms/step -
accuracy: 0.8191 - loss: 0.5075
Epoch 1: val_accuracy improved from -inf to 0.86880, saving model to
/content/drive/MyDrive/flower_data/ResNet50.keras
60/60             45s 702ms/step -
accuracy: 0.8193 - loss: 0.5071 - val_accuracy: 0.8688 - val_loss: 0.4113 -
learning_rate: 1.0000e-04
Epoch 2/25
60/60             0s 491ms/step -
accuracy: 0.8240 - loss: 0.4618
Epoch 2: val_accuracy did not improve from 0.86880
60/60             81s 684ms/step -
accuracy: 0.8242 - loss: 0.4613 - val_accuracy: 0.8659 - val_loss: 0.3927 -
learning_rate: 1.0000e-04
Epoch 3/25
60/60             0s 481ms/step -
accuracy: 0.8673 - loss: 0.3743
Epoch 3: val_accuracy did not improve from 0.86880
60/60             83s 696ms/step -
accuracy: 0.8673 - loss: 0.3744 - val_accuracy: 0.8601 - val_loss: 0.3866 -
learning_rate: 1.0000e-04
Epoch 4/25
60/60             0s 481ms/step -
accuracy: 0.8929 - loss: 0.3320
Epoch 4: val_accuracy improved from 0.86880 to 0.88192, saving model to
/content/drive/MyDrive/flower_data/ResNet50.keras
60/60             55s 844ms/step -
accuracy: 0.8928 - loss: 0.3324 - val_accuracy: 0.8819 - val_loss: 0.3532 -
learning_rate: 1.0000e-04
Epoch 5/25
60/60             0s 480ms/step -
accuracy: 0.8795 - loss: 0.3154
Epoch 5: val_accuracy did not improve from 0.88192
60/60             71s 675ms/step -
accuracy: 0.8795 - loss: 0.3157 - val_accuracy: 0.8732 - val_loss: 0.3577 -
learning_rate: 1.0000e-04
Epoch 6/25
60/60             0s 469ms/step -
accuracy: 0.8858 - loss: 0.3238
Epoch 6: val_accuracy did not improve from 0.88192
```

```
60/60            43s 663ms/step -
accuracy: 0.8860 - loss: 0.3233 - val_accuracy: 0.8790 - val_loss: 0.3439 -
learning_rate: 1.0000e-04
Epoch 7/25
60/60            0s 470ms/step -
accuracy: 0.8855 - loss: 0.3061
Epoch 7: val_accuracy improved from 0.88192 to 0.89504, saving model to
/content/drive/MyDrive/flower_data/ResNet50.keras
60/60            44s 681ms/step -
accuracy: 0.8856 - loss: 0.3058 - val_accuracy: 0.8950 - val_loss: 0.3212 -
learning_rate: 1.0000e-04
Epoch 8/25
60/60            0s 482ms/step -
accuracy: 0.9042 - loss: 0.2672
Epoch 8: val_accuracy did not improve from 0.89504
60/60            44s 674ms/step -
accuracy: 0.9042 - loss: 0.2673 - val_accuracy: 0.8834 - val_loss: 0.3257 -
learning_rate: 1.0000e-04
Epoch 9/25
60/60            0s 470ms/step -
accuracy: 0.9109 - loss: 0.2470
Epoch 9: val_accuracy did not improve from 0.89504
60/60            91s 817ms/step -
accuracy: 0.9110 - loss: 0.2470 - val_accuracy: 0.8776 - val_loss: 0.3272 -
learning_rate: 1.0000e-04
Epoch 10/25
60/60            0s 472ms/step -
accuracy: 0.9217 - loss: 0.2233
Epoch 10: val_accuracy improved from 0.89504 to 0.89942, saving model to
/content/drive/MyDrive/flower_data/ResNet50.keras
60/60            73s 684ms/step -
accuracy: 0.9216 - loss: 0.2235 - val_accuracy: 0.8994 - val_loss: 0.3191 -
learning_rate: 1.0000e-04
Epoch 11/25
60/60            0s 479ms/step -
accuracy: 0.9143 - loss: 0.2328
Epoch 11: val_accuracy did not improve from 0.89942
60/60            44s 673ms/step -
accuracy: 0.9144 - loss: 0.2326 - val_accuracy: 0.8878 - val_loss: 0.3172 -
learning_rate: 1.0000e-04
Epoch 12/25
60/60            0s 466ms/step -
accuracy: 0.9209 - loss: 0.2190
Epoch 12: val_accuracy did not improve from 0.89942
60/60            42s 656ms/step -
accuracy: 0.9209 - loss: 0.2190 - val_accuracy: 0.8848 - val_loss: 0.3188 -
learning_rate: 1.0000e-04
Epoch 13/25
```

```
60/60            0s 493ms/step -
accuracy: 0.9367 - loss: 0.1847
Epoch 13: val_accuracy improved from 0.89942 to 0.90525, saving model to
/content/drive/MyDrive/flower_data/ResNet50.keras
60/60            45s 701ms/step -
accuracy: 0.9367 - loss: 0.1848 - val_accuracy: 0.9052 - val_loss: 0.2887 -
learning_rate: 1.0000e-04
Epoch 14/25
60/60            0s 484ms/step -
accuracy: 0.9361 - loss: 0.1874
Epoch 14: val_accuracy did not improve from 0.90525
60/60            54s 833ms/step -
accuracy: 0.9361 - loss: 0.1876 - val_accuracy: 0.8950 - val_loss: 0.2860 -
learning_rate: 1.0000e-04
Epoch 15/25
60/60            0s 478ms/step -
accuracy: 0.9294 - loss: 0.1967
Epoch 15: val_accuracy did not improve from 0.90525
60/60            43s 653ms/step -
accuracy: 0.9294 - loss: 0.1966 - val_accuracy: 0.8907 - val_loss: 0.2909 -
learning_rate: 1.0000e-04
Epoch 16/25
60/60            0s 472ms/step -
accuracy: 0.9516 - loss: 0.1527
Epoch 16: val_accuracy did not improve from 0.90525
60/60            81s 663ms/step -
accuracy: 0.9515 - loss: 0.1529 - val_accuracy: 0.9023 - val_loss: 0.2874 -
learning_rate: 1.0000e-04
Epoch 17/25
60/60            0s 472ms/step -
accuracy: 0.9512 - loss: 0.1589
Epoch 17: val_accuracy did not improve from 0.90525
60/60            83s 681ms/step -
accuracy: 0.9512 - loss: 0.1588 - val_accuracy: 0.9038 - val_loss: 0.2810 -
learning_rate: 1.0000e-04
Epoch 18/25
60/60            0s 470ms/step -
accuracy: 0.9476 - loss: 0.1537
Epoch 18: val_accuracy did not improve from 0.90525
60/60            43s 660ms/step -
accuracy: 0.9476 - loss: 0.1538 - val_accuracy: 0.8878 - val_loss: 0.2996 -
learning_rate: 1.0000e-04
Epoch 19/25
60/60            0s 468ms/step -
accuracy: 0.9405 - loss: 0.1688
Epoch 19: val_accuracy did not improve from 0.90525
60/60            92s 816ms/step -
accuracy: 0.9405 - loss: 0.1688 - val_accuracy: 0.8907 - val_loss: 0.3142 -
```

```
learning_rate: 1.0000e-04
Epoch 20/25
60/60             0s 466ms/step -
accuracy: 0.9485 - loss: 0.1590
Epoch 20: val_accuracy did not improve from 0.90525
60/60             71s 658ms/step -
accuracy: 0.9484 - loss: 0.1590 - val_accuracy: 0.8892 - val_loss: 0.3077 -
learning_rate: 1.0000e-04
Epoch 21/25
60/60             0s 495ms/step -
accuracy: 0.9503 - loss: 0.1477
Epoch 21: val_accuracy did not improve from 0.90525
60/60             83s 683ms/step -
accuracy: 0.9503 - loss: 0.1477 - val_accuracy: 0.8950 - val_loss: 0.2890 -
learning_rate: 2.0000e-05
Epoch 22/25
60/60             0s 470ms/step -
accuracy: 0.9484 - loss: 0.1452
Epoch 22: val_accuracy did not improve from 0.90525
60/60             81s 654ms/step -
accuracy: 0.9484 - loss: 0.1452 - val_accuracy: 0.8994 - val_loss: 0.2899 -
learning_rate: 2.0000e-05
```

[ ]:
```python
test_datagen = ImageDataGenerator(preprocessing_function=resnet_preprocess)  #␣
↪Replace with your model's preprocessing

test_generator = test_datagen.flow_from_directory(
    test_dir,  # Path to your test data
    target_size=(224, 224),  # Same as training
    batch_size=32,  # Adjust as needed
    class_mode='categorical',  # Same as training
    shuffle=False  # Do not shuffle for evaluation
)
```

```
Found 141 images belonging to 5 classes.
```

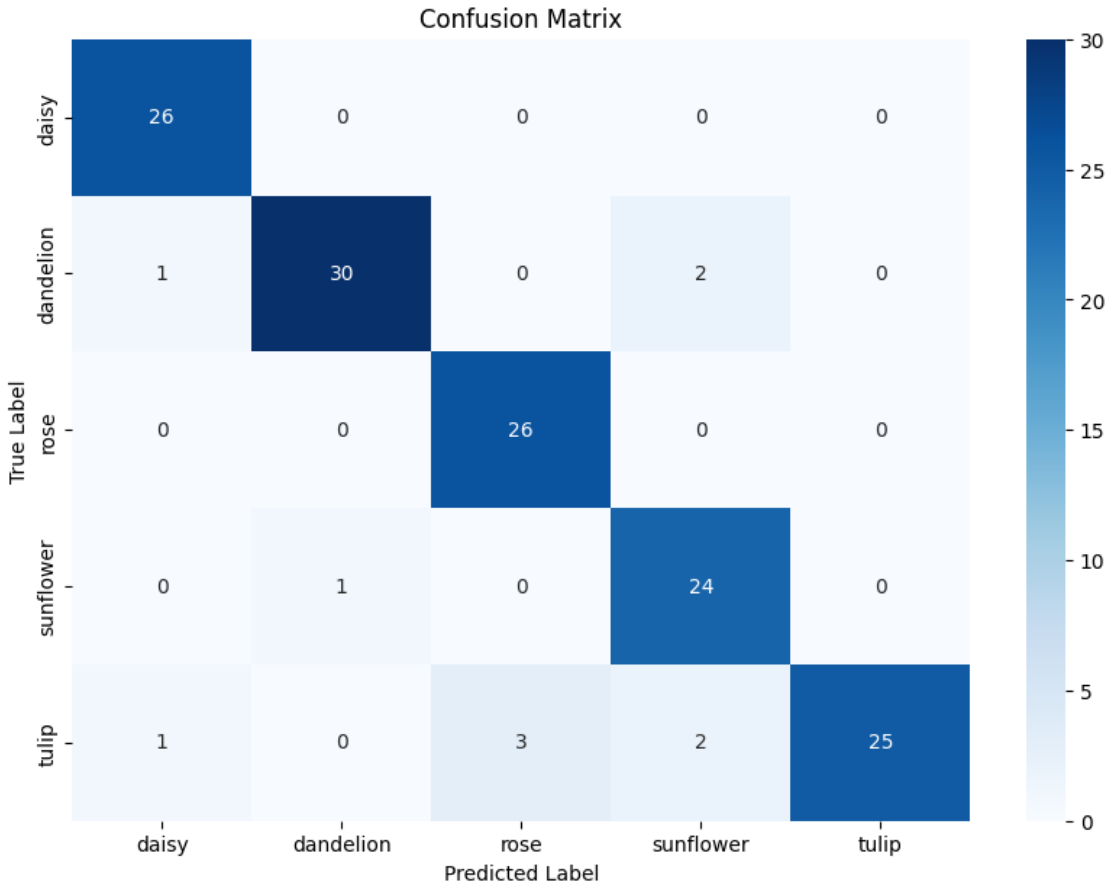
[ ]:
```python
# Evaluate the model
results = model.evaluate(test_generator)
print(f"Test Loss: {results[0]}")
print(f"Test Accuracy: {results[1]}")
```

```
/usr/local/lib/python3.10/dist-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
```

```
5/5             1s 186ms/step -
accuracy: 0.9525 - loss: 0.1839
Test Loss: 0.264686644077301
Test Accuracy: 0.9290780425071716
```

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
from tensorflow.keras.models import load_model

# Generate predictions for the test data without specifying 'steps'
test_predictions = model.predict(test_generator, verbose=1)

# Get the predicted class labels
predicted_labels = np.argmax(test_predictions, axis=1)

# Get the true class labels from the test generator
true_labels = test_generator.classes

# Generate confusion matrix
cm = confusion_matrix(true_labels, predicted_labels)

# Plot the confusion matrix
plt.figure(figsize=(10, 7))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=test_generator.
 ↪class_indices.keys(), yticklabels=test_generator.class_indices.keys())
plt.title("Confusion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
5/5             2s 121ms/step
```



4 Comparison of Pre-trained Models

Detailed Analysis 1. Complexity MobileNetV2 and EfficientNet-B0 are lightweight, making them suitable for resource-constrained devices like mobile phones or embedded systems.

ResNet-50 and InceptionV3 offer a balance between complexity and performance.

VGG-16 and DenseNet-121 are heavier models with higher memory and computational requirements.

Vision Transformers (ViT) are complex and require substantial resources for training but excel in handling large datasets and structured data.

2. Parameters VGG-16 has the highest number of parameters, making it memory-intensive and slow during inference.

MobileNetV2 and NASNetMobile are highly efficient with fewer parameters, making them ideal for edge devices.

ResNet-50 and InceptionV3 strike a good balance, providing robust performance without being excessively large.

3. Key Features MobileNetV2: Depthwise separable convolutions drastically reduce computation without sacrificing performance.

EfficientNet-B0: Uses a compound scaling method to scale depth, width, and resolution effectively.

ResNet-50: Residual connections alleviate the vanishing gradient problem, enabling deep architectures.

DenseNet-121: Reuses features, reducing redundancy and improving gradient flow.

InceptionV3: Factorized convolutions and auxiliary classifiers enhance learning.

4. Problems to Solve Simple Classification Tasks: MobileNetV2, EfficientNet-B0, and NASNet-Mobile are ideal for classification tasks on small or medium datasets.

High-performance Applications: ResNet, Inception, and DenseNet perform well for larger and more complex datasets.

Sequential/Structured Data: Vision Transformers (ViT) excel in structured or sequential image data processing, like video analytics or multi-task learning.

5. Use Cases Mobile and Embedded Devices: MobileNetV2, NASNetMobile, and EfficientNet-B0 are optimized for low-power environments.

Feature Extraction: ResNet-50 and DenseNet-121 are commonly used as backbone models for feature extraction in custom models.

Research and Development: VGG-16 and Vision Transformers are favored in academic and experimental scenarios.

Object Detection and Segmentation: InceptionV3 and DenseNet-121 are often used for advanced tasks like object detection and semantic segmentation.

When to Choose Which Model? Resource Constraints: Use MobileNetV2 or EfficientNet-B0 for devices with limited computation or memory.

Large Datasets: Use ResNet-50, InceptionV3, or Vision Transformers for tasks requiring high accuracy on complex datasets.

Real-time Applications: Use MobileNetV2 or NASNetMobile for applications like real-time image recognition.

Advanced Research: Use Vision Transformers or DenseNet-121 for innovative applications in structured data processing or semantic segmentation.

[ ]: