DATA IMAGE PROCESSING

SUBJECT CODE: CSDA303

Project: Image Classification Using Machine Learning

Submitted By:

ANJALI NAZRIN(Msccs2307)

ANAIDA SUNIL (Msccs2306)

SHAIK SANA ALLANUR(Msccs2327)

IMAGE CLASSIFICATION

Abstract

The project focuses on building a model to recognize different types of flowers from images using machine learning techniques. The goal is to classify flower images into various categories based on their features. The flower recognition project aims to classify images of flowers into various species using machine learning techniques. This involves training a model on a labeled dataset of flower images to recognize and predict the species of flowers in new images.

About Dataset

The dataset contains 5 types of flower type put in separate folders. The flowers are Rose, Daisy, Tulip, Sunflower and Dandelion. We have 764 images of Daisy,1052 images of Dandelion, 784 images of Rose,733 images of Sunflower and 984 images of Tulip. Total we have 4,317 in which we perform image classification using CNN.

Algorithm

Data Preprocessing:

- Image resizing and normalization.
- Data augmentation techniques like rotation, flipping, and zooming to increase dataset variability.

Model Selection:

- Convolutional Neural Networks (CNNs): These are the most widely used models for image classification tasks.
- Transfer Learning: Utilizing pre-trained models like VGG16, ResNet, or InceptionV3 and fine-tuning them for the specific flower dataset.

Training:

- Splitting the dataset into training, validation, and test sets.
- Using a loss function (e.g., categorical cross-entropy) and an optimizer (e.g., Adam) to train the model.
- Monitoring performance on the validation set to avoid overfitting.

Code

*Importing the Libraries

```python
#Importing Libraries
import os
import numpy as np

import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import load_img, ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
```

[1] ✓ 22.1s Python

*Fetching the count of images from the folders

```
#Fetch Images count from Fodlers
```

```
count = 0
dirs = os.listdir('Images/')
for dir in dirs:
    files = list(os.listdir('Images/'+dir))
    print( dir +' Folder has '+ str(len(files)) + ' Images')
    count = count + len(files)
print( 'Images Folder has '+ str(count) + ' Images')
```

```
daisy-20240801T040543Z-001 Folder has 1 Images
dandelion-20240801T040546Z-001 Folder has 1 Images
rose-20240801T040552Z-001 Folder has 1 Images
sunflower-20240801T040557Z-001 Folder has 1 Images
tulip-20240801T040606Z-001 Folder has 1 Images
Images Folder has 5 Images
```

*Loading the images into the arrays as Dataset

```
#Load Images into Arrays as Dataset
```

```
base_dir = 'Images/'
img_size = 180
batch = 32
```

```
train_ds = tf.keras.utils.image_dataset_from_directory( base_dir,
                                                       seed = 123,
                                                       validation_split=0.2,
                                                       subset = 'training',
                                                       batch_size=batch,
                                                       image_size=(img_size,img_size))

val_ds = tf.keras.utils.image_dataset_from_directory( base_dir,
                                                       seed = 123,
                                                       validation_split=0.2,
                                                       subset = 'validation',
                                                       batch_size=batch,
                                                       image_size=(img_size,img_size))
```

```
Found 4317 files belonging to 5 classes.
Using 3454 files for training.
Found 4317 files belonging to 5 classes.
Using 863 files for validation.
```

```python
flower_names = train_ds.class_names
flower_names
```

[5] ✓ 0.0s Python

```
['daisy-20240801T040543Z-001',
 'dandelion-20240801T040546Z-001',
 'rose-20240801T040552Z-001',
 'sunflower-20240801T040557Z-001',
 'tulip-20240801T040606Z-001']
```

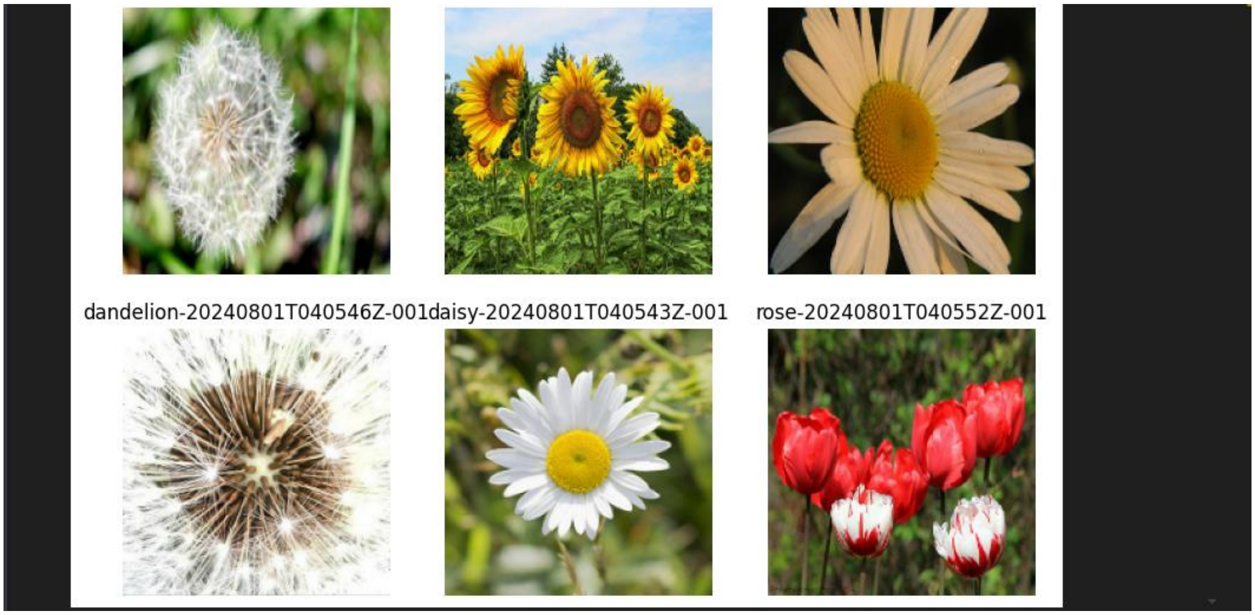
```python
import matplotlib.pyplot as plt
```

[6] ✓ 3.4s Python

```python
i = 0
plt.figure(figsize=(10,10))

for images, labels in train_ds.take(1):
    for i in range(9):
        plt.subplot(3,3, i+1)
        plt.imshow(images[i].numpy().astype('uint8'))
        plt.title(flower_names[labels[i]])
        plt.axis('off')
```

[7] ✓ 2.8s Python



*Doing the data augmentation

```
[8]
AUTOTUNE = tf.data.AUTOTUNE
✓ 0.0s                                                        Python

[9]
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size = AUTOTUNE)
✓ 0.0s                                                        Python

[11]
val_ds = val_ds.cache().prefetch(buffer_size = AUTOTUNE)
✓ 0.0s                                                        Python

[81]
#Data Augmentation
                                                              Python

[12]
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape = (img_size,img_size,3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])
✓ 0.0s                                                        Python

... c:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\preprocessing\tf_data_layer.py:19: User
  super().__init__(**kwargs)
```

```python
i = 0
plt.figure(figsize=(10,10))

for images, labels in train_ds.take(1):
    for i in range(9):
        images = data_augmentation(images)
        plt.subplot(3,3, i+1)
        plt.imshow(images[0].numpy().astype('uint8'))
        plt.axis('off')
```
[13] ✓ 21.5s Python

*Creating model for prediction

```python
#MOdel Creation
```
[84] Python

```python
model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    Conv2D(16, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(32, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Conv2D(64, 3, padding='same', activation='relu'),
    MaxPooling2D(),
    Dropout(0.2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(5)
])
```
[14] ✓ 0.2s Python

```python
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```
[15] ✓ 0.1s Python

```
model.summary()
```

[18] ✓ 0.4s Python

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 180, 180, 3)	0
rescaling (Rescaling)	(None, 180, 180, 3)	0
conv2d (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_1 (Conv2D)	(None, 90, 90, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_2 (Conv2D)	(None, 45, 45, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 22, 22, 64)	0
dropout (Dropout)	(None, 22, 22, 64)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 128)	3,965,056
dense_1 (Dense)	(None, 5)	645

```
Total params: 11,967,857 (45.65 MB)

Trainable params: 3,989,285 (15.22 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 7,978,572 (30.44 MB)
```

```python
history = model.fit(train_ds, epochs=15, validation_data=val_ds)
```

[17] ✓ 20m 1.8s Python

```
Epoch 1/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 90s 778ms/step - accuracy: 0.3259 - loss: 1.5923 - val_accuracy: 0.5759 - val_loss: 1.0868
Epoch 2/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 87s 809ms/step - accuracy: 0.5342 - loss: 1.1167 - val_accuracy: 0.5910 - val_loss: 1.0579
Epoch 3/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 89s 823ms/step - accuracy: 0.6024 - loss: 0.9832 - val_accuracy: 0.5771 - val_loss: 1.0203
Epoch 4/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 84s 781ms/step - accuracy: 0.6255 - loss: 0.9475 - val_accuracy: 0.6211 - val_loss: 0.9960
Epoch 5/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 82s 758ms/step - accuracy: 0.6529 - loss: 0.9022 - val_accuracy: 0.6570 - val_loss: 0.8800
Epoch 6/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 78s 725ms/step - accuracy: 0.6860 - loss: 0.8196 - val_accuracy: 0.6640 - val_loss: 0.8562
Epoch 7/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 77s 711ms/step - accuracy: 0.7052 - loss: 0.7670 - val_accuracy: 0.6883 - val_loss: 0.8067
Epoch 8/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 76s 706ms/step - accuracy: 0.6941 - loss: 0.8059 - val_accuracy: 0.6895 - val_loss: 0.8113
Epoch 9/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 78s 725ms/step - accuracy: 0.7358 - loss: 0.7063 - val_accuracy: 0.7022 - val_loss: 0.7646
Epoch 10/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 79s 729ms/step - accuracy: 0.7463 - loss: 0.7046 - val_accuracy: 0.6964 - val_loss: 0.7720
Epoch 11/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 76s 702ms/step - accuracy: 0.7423 - loss: 0.6737 - val_accuracy: 0.7138 - val_loss: 0.7606
Epoch 12/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 76s 707ms/step - accuracy: 0.7487 - loss: 0.6618 - val_accuracy: 0.7254 - val_loss: 0.7109
Epoch 13/15
...
Epoch 14/15
```

```
Epoch 14/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 75s 691ms/step - accuracy: 0.7745 - loss: 0.6066 - val_accuracy: 0.7231 - val_loss: 0.7140
Epoch 15/15
108/108 ━━━━━━━━━━━━━━━━━━━━ 78s 725ms/step - accuracy: 0.7868 - loss: 0.5765 - val_accuracy: 0.7416 - val_loss: 0.6881
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
def classify_images(image_path):
    input_image = tf.keras.utils.load_img(image_path, target_size=(180,180))
    input_image_array = tf.keras.utils.img_to_array(input_image)
    input_image_exp_dim = tf.expand_dims(input_image_array,0)

    predictions = model.predict(input_image_exp_dim)
    result = tf.nn.softmax(predictions[0])
    outcome = 'The Image belongs to ' + flower_names[np.argmax(result)] + ' with a score of '+ str(np.max(result)*100)
    return outcome
```

[19] ✓ 0.0s Python

```python
classify_images('D:/Jupyter Notebook/Sample/tulip.jfif')
```

[20] ✓ 3.7s Python

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
```

```
'The Image belongs to tulip-20240801T040606Z-001 with a score of 54.11316156387329'
```

*Compiling the model

```
model = Sequential([
    Flatten(input_shape=(224, 224, 3)),
    Dense(128, activation='relu'),
    Dense(5, activation='softmax')  # Assuming you have 5 classes
])

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

# Example: Training the model (assuming you have your own training data)
# datagen = ImageDataGenerator(rescale=1./255)
# train_generator = datagen.flow_from_directory(
#     'path_to_train_data',
#     target_size=(224, 224),
#     batch_size=32,
#     class_mode='sparse'
# )

# model.fit(train_generator, epochs=10)

# Save the model
model.save('Flower_Recog_Model.h5')
```

```
[22]  ✓ 2.2s                                                                    Python
c:\Users\Admin\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\reshaping\flatten.py:37: UserWarning: D
  super().__init__(**kwargs)
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format i
```

```
classify_images('Sample/rose.jpg')
[102]                                                                            Python
1/1 [==============================] - 0s 27ms/step
'The Image belongs to rose with a score of 93.2832658290863'
```

OUTPUT

Here when we gave the sample image it was predicted correctly with an accuracy of 93.28%.