

Assignment_3

November 17, 2022

Assignment Date	01 October 2022
Student Name	Shwetha L S
Student Roll Number	2019504586
Maximum Marks	2 Marks

1 Build CNN Model for Classification Of Flowers

```
[24]: import tensorflow as tf
      from tensorflow.keras import layers
      from tensorflow.keras.models import Sequential

      import numpy as np
```

```
[25]: batch_size = 32
      img_height = 180
      img_width = 180

      data_dir = "../input/flowers-ibmntp/flowers"
```

1.0.1 Image augmentation

```
[26]: data_augmentation = Sequential(
      [
          layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
          layers.RandomRotation(0.1),
          layers.RandomZoom(0.1),
      ]
      )
```

1.0.2 Split dataset into training and testing sets

```
[27]: train_ds = tf.keras.utils.image_dataset_from_directory(
      data_dir,
      validation_split=0.2,
      subset="training",
      seed=123,
```

```
image_size=(img_height, img_width),  
batch_size=batch_size)
```

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

```
[28]: val_ds = tf.keras.utils.image_dataset_from_directory(  
    data_dir,  
    validation_split=0.2,  
    subset="validation",  
    seed=123,  
    image_size=(img_height, img_width),  
    batch_size=batch_size)
```

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

```
[29]: class_names = train_ds.class_names  
print(class_names)
```

```
['daisy', 'dandelion', 'rose', 'sunflower', 'tulip']
```

```
[30]: import matplotlib.pyplot as plt  
  
plt.figure(figsize=(10, 10))  
for images, labels in train_ds.take(1):  
    for i in range(9):  
        ax = plt.subplot(3, 3, i + 1)  
        plt.imshow(images[i].numpy().astype("uint8"))  
        plt.title(class_names[labels[i]])  
        plt.axis("off")
```



```
[31]: normalization_layer = layers.Rescaling(1./255)
```

```
[32]: normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixel values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

```
0.0 0.97046
```

1.0.3 Create the model and adding layers

```
[33]: num_classes = len(class_names)

model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)
])
```

1.0.4 Compile the model

```
[34]: model.compile(optimizer='adam',
                    loss=tf.keras.losses.
                        SparseCategoricalCrossentropy(from_logits=True),
                    metrics=['accuracy'])
```

```
[35]: model.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
sequential_1 (Sequential)	(None, 180, 180, 3)	0
rescaling_3 (Rescaling)	(None, 180, 180, 3)	0
conv2d_3 (Conv2D)	(None, 180, 180, 16)	448
max_pooling2d_3 (MaxPooling2D)	(None, 90, 90, 16)	0
conv2d_4 (Conv2D)	(None, 90, 90, 32)	4640
max_pooling2d_4 (MaxPooling2D)	(None, 45, 45, 32)	0
conv2d_5 (Conv2D)	(None, 45, 45, 64)	18496
max_pooling2d_5 (MaxPooling2D)	(None, 22, 22, 64)	0

flatten_1 (Flatten)	(None, 30976)	0

dense_2 (Dense)	(None, 128)	3965056

dense_3 (Dense)	(None, 5)	645
=====		
Total params: 3,989,285		
Trainable params: 3,989,285		
Non-trainable params: 0		

1.0.5 Fit the model

```
[36]: epochs=10
      history = model.fit(
          train_ds,
          validation_data=val_ds,
          epochs=epochs
      )
```

```
Epoch 1/10
108/108 [=====] - 12s 98ms/step - loss: 1.3587 -
accuracy: 0.4311 - val_loss: 1.1208 - val_accuracy: 0.5481
Epoch 2/10
108/108 [=====] - 8s 63ms/step - loss: 1.0440 -
accuracy: 0.5926 - val_loss: 1.0209 - val_accuracy: 0.6083
Epoch 3/10
108/108 [=====] - 8s 57ms/step - loss: 0.9654 -
accuracy: 0.6283 - val_loss: 0.9353 - val_accuracy: 0.6431
Epoch 4/10
108/108 [=====] - 8s 66ms/step - loss: 0.8782 -
accuracy: 0.6726 - val_loss: 0.8580 - val_accuracy: 0.6582
Epoch 5/10
108/108 [=====] - 7s 61ms/step - loss: 0.8522 -
accuracy: 0.6699 - val_loss: 0.9512 - val_accuracy: 0.6211
Epoch 6/10
108/108 [=====] - 7s 61ms/step - loss: 0.8144 -
accuracy: 0.6957 - val_loss: 0.8153 - val_accuracy: 0.6651
Epoch 7/10
108/108 [=====] - 7s 58ms/step - loss: 0.7433 -
accuracy: 0.7215 - val_loss: 0.7417 - val_accuracy: 0.7115
Epoch 8/10
108/108 [=====] - 7s 65ms/step - loss: 0.7176 -
accuracy: 0.7336 - val_loss: 0.7869 - val_accuracy: 0.6813
Epoch 9/10
108/108 [=====] - 7s 62ms/step - loss: 0.6884 -
accuracy: 0.7360 - val_loss: 0.7605 - val_accuracy: 0.7080
Epoch 10/10
```

```
108/108 [=====] - 7s 62ms/step - loss: 0.6594 -  
accuracy: 0.7417 - val_loss: 0.7304 - val_accuracy: 0.7149
```

```
[37]: acc = history.history['accuracy']  
      val_acc = history.history['val_accuracy']  
  
      loss = history.history['loss']  
      val_loss = history.history['val_loss']  
  
      epochs_range = range(epochs)  
  
      plt.figure(figsize=(8, 8))  
      plt.subplot(1, 2, 1)  
      plt.plot(epochs_range, acc, label='Training Accuracy')  
      plt.plot(epochs_range, val_acc, label='Validation Accuracy')  
      plt.legend(loc='lower right')  
      plt.title('Training and Validation Accuracy')  
  
      plt.subplot(1, 2, 2)  
      plt.plot(epochs_range, loss, label='Training Loss')  
      plt.plot(epochs_range, val_loss, label='Validation Loss')  
      plt.legend(loc='upper right')  
      plt.title('Training and Validation Loss')  
      plt.show()
```



1.0.6 Save the model

```
[38]: model.save("flowers.m5")
```

1.0.7 Testing the model

```
[39]: sunflower_url = "https://storage.googleapis.com/download.tensorflow.org/
      ↪example_images/592px-Red_sunflower.jpg"
      sunflower_path = tf.keras.utils.get_file('Red_sunflower', origin=sunflower_url)

      img = tf.keras.utils.load_img(
          sunflower_path, target_size=(img_height, img_width)
      )
```

```

img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)

```

Downloading data from

https://storage.googleapis.com/download.tensorflow.org/example_images/592px-Red_sunflower.jpg

122880/117948 [=====] - 0s 0us/step

131072/117948 [=====] - 0s 0us/step

This image most likely belongs to sunflower with a 99.81 percent confidence.