Name	Kaviraj Y
Roll No	210419104082
Assignment No	3

```
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt

batch_size = 16
```

IMAGE AUGMENTATION

```
data_augmentation = Sequential(
[
layers.RandomFlip("horizontal",input_shape=(180, 180, 3)),
layers.RandomRotation(0.1),
layers.RandomZoom(0.1),
]
)
```

Spliting dataset into training and test

```
train_data_set = tf.keras.utils.image_dataset_from_directory(
"flowers",
validation_split=0.25,
subset="training",
seed=132,
image_size=(180, 180),
batch_size=batch_size)
```

```
Found 4317 files belonging to 5 classes.
Using 3238 files for training.
```

```
val_data_set = tf.keras.utils.image_dataset_from_directory(
"flowers",
validation_split=0.25,
subset="validation",
seed=132,
image_size=(180, 180),
batch_size=batch_size)
```

```
Found 4317 files belonging to 5 classes.
Using 1079 files for validation.
```

```
class_names = train_data_set.class_names

plt.figure(figsize=(15, 15))
for images, labels in train_data_set.take(1):
```

Name	Kaviraj Y
Roll No	210419104082
Assignment No	3

```
4. for i in range(6):
5. ax = plt.subplot(3, 3, i + 1)
6. plt.imshow(images[i].numpy().astype("uint8"))
7. plt.title(class_names[labels[i]])
```

Normalizing pixel value from 0 - 255 to 0 - 1

```
1. normalization_layer = layers.Rescaling(1./255)
```

```
2. dataset_normalized = train_data_set.map(lambda x, y:
   (normalization_layer(x), y))
3. image_batch, labels_batch = next(iter(dataset_normalized))
4. first_image = image_batch[0]
5. print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```

MODEL CREATION AND ADDITION OF LAYERS

Name	Kaviraj Y
Roll No	210419104082
Assignment No	3

```
num_classes = len(class_names)

model = Sequential([
data_augmentation,
layers.Rescaling(1./255, input_shape=(180, 180, 3)),
# adding convolutional layer
layers.Conv2D(16, 3, padding='same', activation='relu'),
# adding maxpooling layer
layers.MaxPooling2D(),
layers.Conv2D(32, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 layers.Conv2D(64, 3, padding='same', activation='relu'),
 layers.MaxPooling2D(),
 # adding flatten
 layers.Flatten(),
 # adding dense hidden layer
 layers.Dense(128, activation='relu'),
 # adding dense output layer
 layers.Dense(num_classes)
 ])
```

COMPLIATION OF MODEL

```
# compiling model with categorical cross entropy and adam optimizer
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy'])
```

FITTING THE MODEL

```
epochs=15
history =
model.fit(train_data_set,validation_data=val_data_set,epochs=epochs)
```

```
Epoch 1/15
203/203 [==============================] - 113s 554ms/step - loss: 1.2347 -
accuracy: 0.4700 - val_loss: 1.0847 - val_accuracy: 0.5672
Epoch 2/15
203/203 [==============================] - 109s 537ms/step - loss: 0.9885 -
accuracy: 0.6124 - val_loss: 1.0040 - val_accuracy: 0.5996
Epoch 3/15
203/203 [==============================] - 88s 432ms/step - loss: 0.8784 -
accuracy: 0.6563 - val_loss: 1.0471 - val_accuracy: 0.6154
Epoch 4/15
203/203 [==============================] - 97s 479ms/step - loss: 0.8347 -
accuracy: 0.6776 - val_loss: 0.8886 - val_accuracy: 0.6793
```

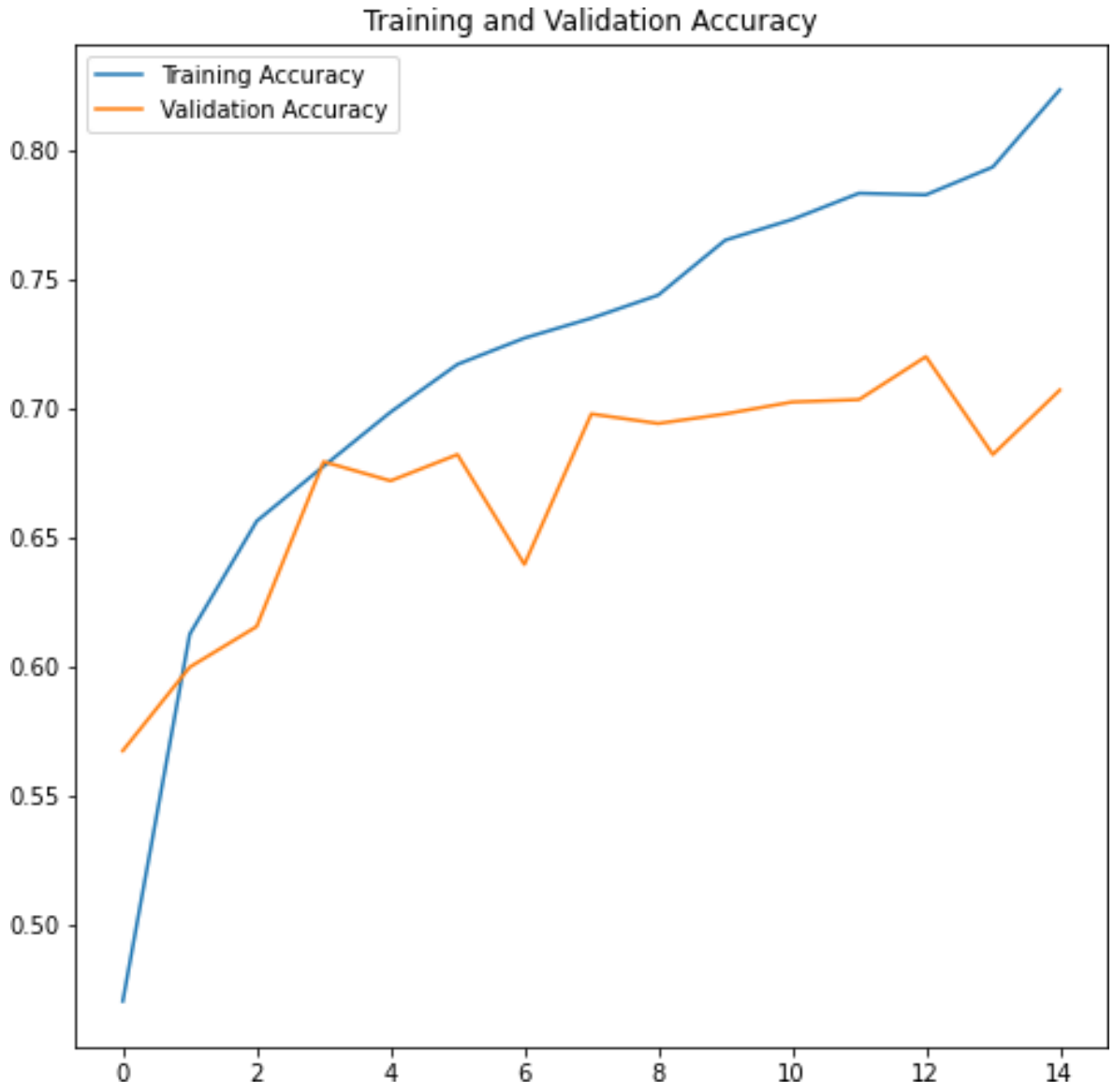
Name	Kaviraj Y
Roll No	210419104082
Assignment No	3

```
Epoch 5/15
203/203 [==============================] - 98s 482ms/step - loss: 0.7949 -
accuracy: 0.6986 - val_loss: 0.9212 - val_accuracy: 0.6719
Epoch 6/15
203/203 [==============================] - 92s 454ms/step - loss: 0.7518 -
accuracy: 0.7171 - val_loss: 0.8591 - val_accuracy: 0.6821
Epoch 7/15
203/203 [==============================] - 82s 402ms/step - loss: 0.7336 -
accuracy: 0.7273 - val_loss: 1.0130 - val_accuracy: 0.6395
Epoch 8/15
203/203 [==============================] - 95s 465ms/step - loss: 0.7143 -
accuracy: 0.7350 - val_loss: 0.8428 - val_accuracy: 0.6979
Epoch 9/15
203/203 [==============================] - 97s 478ms/step - loss: 0.6683 -
accuracy: 0.7440 - val_loss: 0.8703 - val_accuracy: 0.6942
Epoch 10/15
203/203 [==============================] - 91s 448ms/step - loss: 0.6359 -
accuracy: 0.7653 - val_loss: 0.8908 - val_accuracy: 0.6979
Epoch 11/15
203/203 [==============================] - 90s 444ms/step - loss: 0.6027 -
accuracy: 0.7733 - val_loss: 0.8223 - val_accuracy: 0.7025
Epoch 12/15
203/203 [==============================] - 78s 383ms/step - loss: 0.5753 -
accuracy: 0.7835 - val_loss: 0.9151 - val_accuracy: 0.7034
Epoch 13/15
203/203 [==============================] - 76s 373ms/step - loss: 0.5684 -
accuracy: 0.7829 - val_loss: 0.8659 - val_accuracy: 0.7201
Epoch 14/15
203/203 [==============================] - 76s 373ms/step - loss: 0.5285 -
accuracy: 0.7937 - val_loss: 1.0205 - val_accuracy: 0.6821
Epoch 15/15
203/203 [==============================] - 75s 371ms/step - loss: 0.4781 -
accuracy: 0.8237 - val_loss: 0.9877 - val_accuracy: 0.7071
```

```
1. epochs_range = range(epochs)

2. plt.figure(figsize=(8, 8))
3. plt.plot(epochs_range, history.history['accuracy'], label='Training
   Accuracy')
4. plt.plot(epochs_range, history.history['val_accuracy'],
   label='Validation Accuracy')
5. plt.legend()
6. plt.title('Training and Validation Accuracy')
7. plt.show()
```

Name	**Kaviraj Y**
Roll No	**210419104082**
Assignment No	**3**



In [35]:

```
1. plt.figure(figsize=(8, 8))
2. plt.plot(epochs_range, history.history['loss'], label='Training
   Loss')
3. plt.plot(epochs_range, history.history['val_loss'], label='Validation
   Loss')
4. plt.legend()
5. plt.title('Training and Validation Loss')
6. plt.show()
```

Name	**Kaviraj Y**
Roll No	**210419104082**
Assignment No	**3**

SAVING THE MODEL

```
1. model.save("CNN Model for Classification Of Flowers.h5")

2. model.load_weights('CNN Model for Classification Of Flowers.h5')
```

TESTING THE MODEL

Name	**Kaviraj Y**
Roll No	**210419104082**
Assignment No	**3**

```
1. sunflower_url =
   "https://storage.googleapis.com/download.tensorflow.org/example_image
   s/592px-Red_sunflower.jpg"
2. sunflower_path = tf.keras.utils.get_file('Red_sunflower',
   origin=sunflower_url)

3. img = tf.keras.utils.load_img(
4. sunflower_path, target_size=(180, 180)
5. )
6. img_array = tf.keras.utils.img_to_array(img)
7. img_array = tf.expand_dims(img_array, 0) # Create a batch

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

10.  print(class_names[np.argmax(score)],100 * np.max(score))
```

```
sunflower 98.68776202201843
```