Building a CNN model for classification of Flowers

Load the dataset

```
!unzip Flowers-Dataset.zip
```

```
Archive:  Flowers-Dataset.zip
  inflating: flowers/daisy/100080576_f52e8ee070_n.jpg
  inflating: flowers/daisy/10140303196_b88d3d6cec.jpg
  inflating: flowers/daisy/10172379554_b296050f82_n.jpg
  inflating: flowers/daisy/10172567486_2748826a8b.jpg
  inflating: flowers/daisy/10172636503_21bededa75_n.jpg
  inflating: flowers/daisy/102841525_bd6628ae3c.jpg
  inflating: flowers/daisy/10300722094_28fa978807_n.jpg
  inflating: flowers/daisy/1031799732_e7f4008c03.jpg
  inflating: flowers/daisy/10391248763_1d16681106_n.jpg
  inflating: flowers/daisy/10437754174_22ec990b77_m.jpg
  inflating: flowers/daisy/10437770546_8bb6f7bdd3_m.jpg
  inflating: flowers/daisy/10437929963_bc13eebe0c.jpg
  inflating: flowers/daisy/10466290366_cc72e33532.jpg
  inflating: flowers/daisy/10466558316_a7198b87e2.jpg
  inflating: flowers/daisy/10555749515_13a12a026e.jpg
  inflating: flowers/daisy/10555815624_dc211569b0.jpg
  inflating: flowers/daisy/10555826524_423eb8bf71_n.jpg
  inflating: flowers/daisy/10559679065_50d2b16f6d.jpg
  inflating: flowers/daisy/105806915_a9c13e2106_n.jpg
  inflating: flowers/daisy/10712722853_5632165b04.jpg
  inflating: flowers/daisy/107592979_aaa9cdfe78_m.jpg
  inflating: flowers/daisy/10770585085_4742b9dac3_n.jpg
  inflating: flowers/daisy/10841136265_af473efc60.jpg
  inflating: flowers/daisy/10993710036_2033222c91.jpg
  inflating: flowers/daisy/10993818044_4c19b86c82.jpg
  inflating: flowers/daisy/10994032453_ac7f8d9e2e.jpg
  inflating: flowers/daisy/11023214096_b5b39fab08.jpg
  inflating: flowers/daisy/11023272144_fce94401f2_m.jpg
  inflating: flowers/daisy/11023277956_8980d53169_m.jpg
  inflating: flowers/daisy/11124324295_503f3a0804.jpg
  inflating: flowers/daisy/1140299375_3aa7024466.jpg
  inflating: flowers/daisy/11439894966_dca877f0cd.jpg
  inflating: flowers/daisy/1150395827_6f94a5c6e4_n.jpg
  inflating: flowers/daisy/11642632_1e7627a2cc.jpg
  inflating: flowers/daisy/11834945233_a53b7a92ac_m.jpg
  inflating: flowers/daisy/11870378973_2ec1919f12.jpg
  inflating: flowers/daisy/11891885265_ccefec7284_n.jpg
  inflating: flowers/daisy/12193032636_b50ae7db35_n.jpg
  inflating: flowers/daisy/12348343085_d4c396e5b5_m.jpg
  inflating: flowers/daisy/12585131704_0f64b17059_m.jpg
  inflating: flowers/daisy/12601254324_3cb62c254a_m.jpg
  inflating: flowers/daisy/1265350143_6e2b276ec9.jpg
```

```
inflating: flowers/daisy/12701063955_4840594ea6_n.jpg
inflating: flowers/daisy/1285423653_18926dc2c8_n.jpg
inflating: flowers/daisy/1286274236_1d7ac84efb_n.jpg
inflating: flowers/daisy/12891819633_e4c82b51e8.jpg
inflating: flowers/daisy/1299501272_59d9da5510_n.jpg
inflating: flowers/daisy/1306119996_ab8ae14d72_n.jpg
inflating: flowers/daisy/1314069875_da8dc023c6_m.jpg
inflating: flowers/daisy/1342002397_9503c97b49.jpg
inflating: flowers/daisy/134409839_71069a95d1_m.jpg
inflating: flowers/daisy/1344985627_c3115e2d71_n.jpg
inflating: flowers/daisy/13491959645_2cd9df44d6_n.jpg
inflating: flowers/daisy/1354396826_2868631432_m.jpg
inflating: flowers/daisy/1355787476_32e9f2a30b.jpg
inflating: flowers/daisy/13583238844_573df2de8e_m.jpg
inflating: flowers/daisy/1374193928_a52320eafa.jpg
```

```
#importing required libraries to build a CNN classification model with accuracy

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 = 32
img_height = 180
img_width = 180
data_dir = "/content/flowers"
```

Image Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True, vertical_flip =
```

```
x_train = train_datagen.flow_from_directory(r"/content/flowers", target_size = (64,64) , clas
```

```
Found 4317 images belonging to 5 classes.
```

```
#Image Augumentation accuracy
data_augmentation = Sequential(
  [
    layers.RandomFlip("horizontal",input_shape=(img_height, img_width, 3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)
```

Model Building and also Split dataset into training and testing sets

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense
model = Sequential()
```

```
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.
```

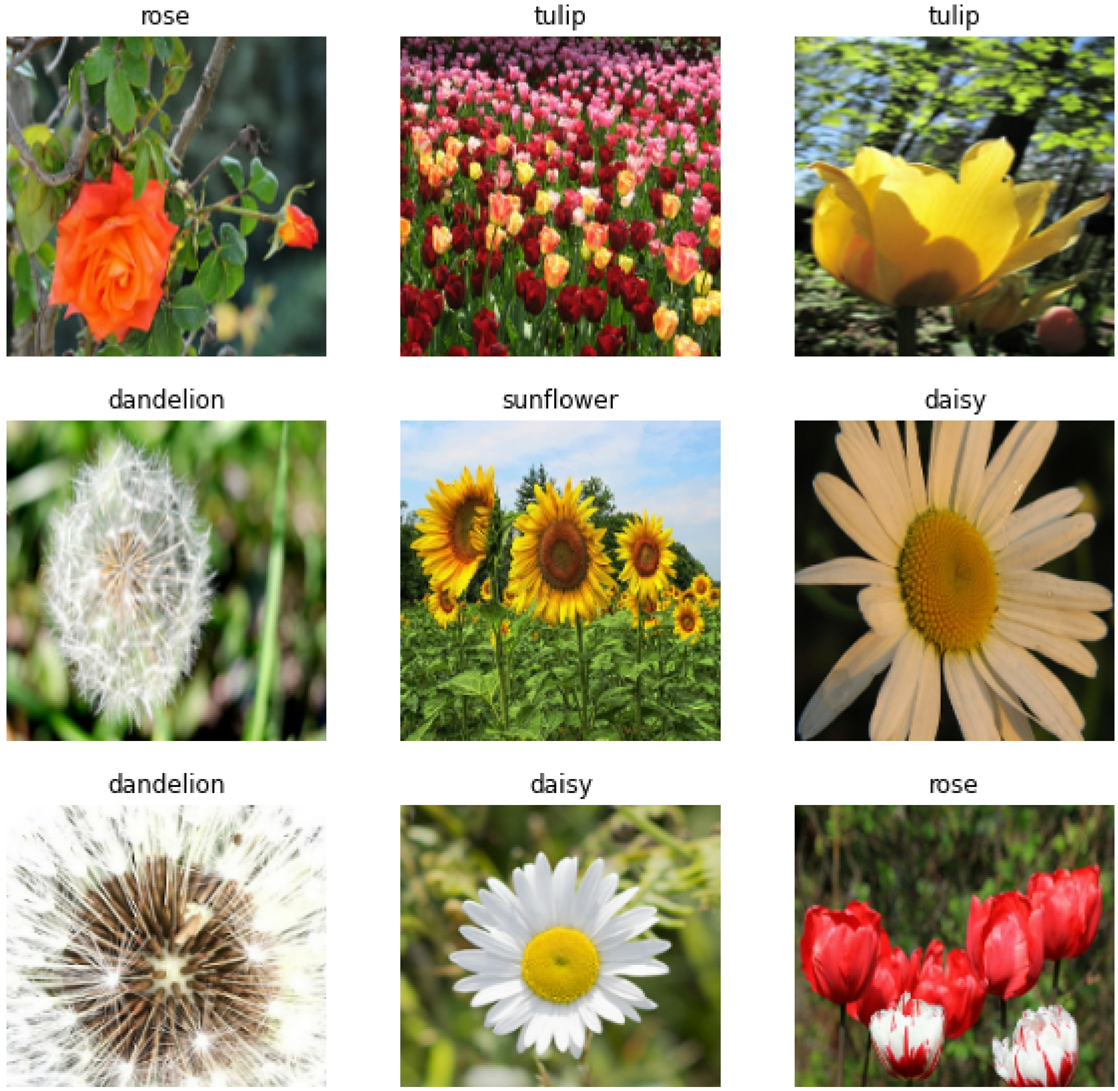
```
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

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

```
class_names = train_ds.class_names
print(class_names)
```

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

```
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")
```



Adding the layers (Convolution,MaxPooling,Flatten,Dense-(HiddenLayers),Output)

```
model.add(Convolution2D(32, (3,3), activation = "relu", input_shape = (64,64,3) ))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())
model.add(Dense(300, activation = "relu"))
model.add(Dense(150, activation = "relu")) #mulitple dense layers
model.add(Dense(5, activation = "softmax")) #output layer
```

```
#Adding the layers for accuracy
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)
])
```

Compile The Model

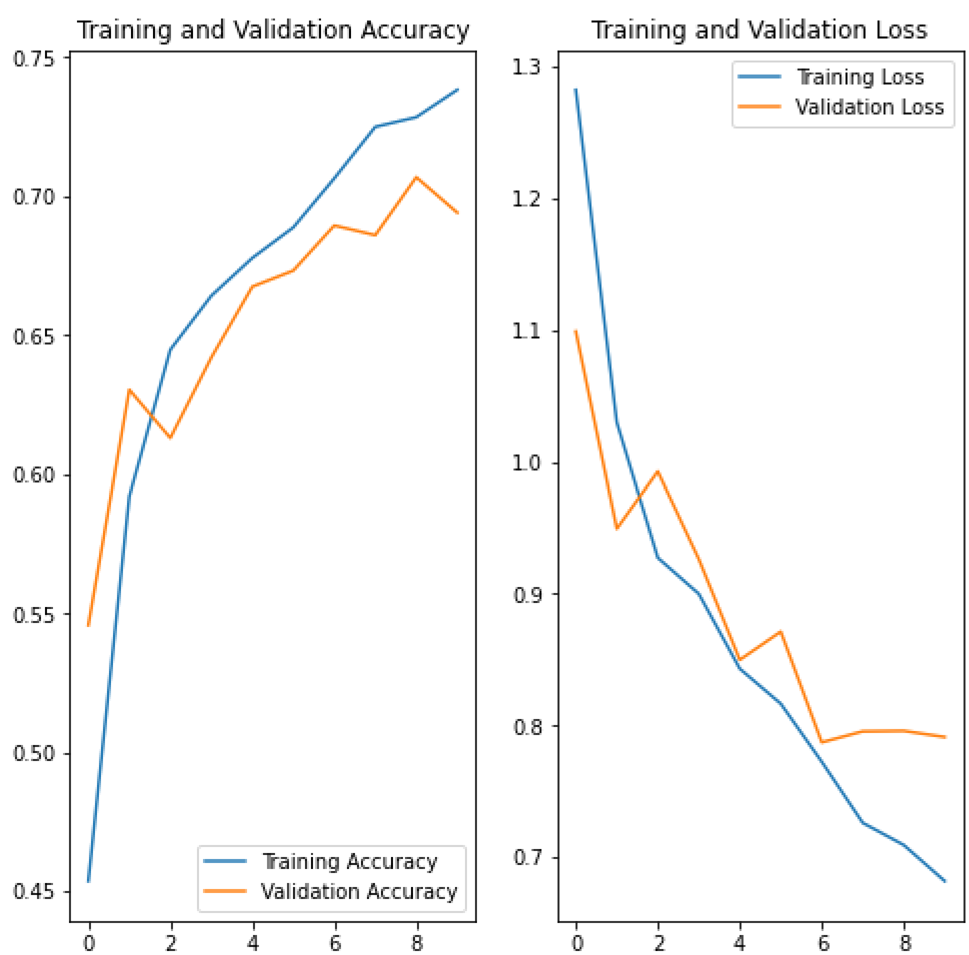
```
model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = "adam")
len(x_train)
```

```
44
```

```
#Compile the model for further accuracy
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/10
108/108 [==============================] - 132s 1s/step - loss: 1.2821 - accuracy: 0.45
Epoch 2/10
108/108 [==============================] - 130s 1s/step - loss: 1.0298 - accuracy: 0.59
Epoch 3/10
108/108 [==============================] - 129s 1s/step - loss: 0.9274 - accuracy: 0.64
Epoch 4/10
108/108 [==============================] - 129s 1s/step - loss: 0.9000 - accuracy: 0.66
Epoch 5/10
108/108 [==============================] - 136s 1s/step - loss: 0.8432 - accuracy: 0.67
Epoch 6/10
108/108 [==============================] - 130s 1s/step - loss: 0.8166 - accuracy: 0.68
Epoch 7/10
108/108 [==============================] - 130s 1s/step - loss: 0.7726 - accuracy: 0.70
Epoch 8/10
108/108 [==============================] - 130s 1s/step - loss: 0.7262 - accuracy: 0.72
Epoch 9/10
108/108 [==============================] - 128s 1s/step - loss: 0.7094 - accuracy: 0.72
Epoch 10/10
108/108 [==============================] - 130s 1s/step - loss: 0.6820 - accuracy: 0.73
```

```
#To find the Training and Validation- Accuracy & Loss (Visualization)

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()
```



Fit The Model

```
model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))
```

```
Epoch 1/15
44/44 [==============================] - 31s 684ms/step - loss: 1.7914 - accuracy: 0.358
Epoch 2/15
44/44 [==============================] - 29s 648ms/step - loss: 1.1730 - accuracy: 0.504
Epoch 3/15
44/44 [==============================] - 29s 650ms/step - loss: 1.0967 - accuracy: 0.552
Epoch 4/15
44/44 [==============================] - 29s 648ms/step - loss: 1.0351 - accuracy: 0.593
Epoch 5/15
44/44 [==============================] - 29s 645ms/step - loss: 0.9920 - accuracy: 0.612
Epoch 6/15
44/44 [==============================] - 30s 677ms/step - loss: 0.9659 - accuracy: 0.625
Epoch 7/15
44/44 [==============================] - 29s 648ms/step - loss: 0.9129 - accuracy: 0.642
Epoch 8/15
44/44 [==============================] - 29s 647ms/step - loss: 0.9085 - accuracy: 0.643
Epoch 9/15
44/44 [==============================] - 32s 717ms/step - loss: 0.8597 - accuracy: 0.662
Epoch 10/15
44/44 [==============================] - 30s 674ms/step - loss: 0.8350 - accuracy: 0.682
Epoch 11/15
44/44 [==============================] - 29s 648ms/step - loss: 0.8420 - accuracy: 0.671
Epoch 12/15
44/44 [==============================] - 29s 650ms/step - loss: 0.7857 - accuracy: 0.703
Epoch 13/15
44/44 [==============================] - 29s 649ms/step - loss: 0.7868 - accuracy: 0.706
Epoch 14/15
44/44 [==============================] - 29s 650ms/step - loss: 0.7542 - accuracy: 0.713
Epoch 15/15
44/44 [==============================] - 30s 676ms/step - loss: 0.7467 - accuracy: 0.716
<keras.callbacks.History at 0x7f602ce90090>
```

Save The Model

```
model.save("flowers.h1")
```

```
model.save("flowers.m5")#another model to show the accuracy
```

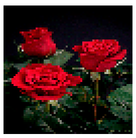
Test The Model

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
model = load_model("/content/flowers.h1")
```

```
#Testing with a random rose image from Google

img = image.load_img("/content/rose.gif", target_size = (64,64) )
```

```
img
```



```
x = image.img_to_array(img)
x.ndim
```

```
3
```

```
x = np.expand_dims(x,axis = 0)
x.ndim
```

```
4
```

```
pred = model.predict(x)
pred
```

```
array([[0., 0., 1., 0., 0.]], dtype=float32)
```

```
labels = ['daisy','dandelion','roses','sunflowers','tulips']
```

```
labels[np.argmax(pred)]
```

```
'roses'
```

```
#Testing the alternative model with accuracy

sunflower_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/592px-
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_ima
122880/117948 [===============================] - 0s 0us/step
131072/117948 [================================] - 0s 0us/step
This image most likely belongs to sunflower with a 99.85 percent confidence.
```