Building CNN Model for Classification Of Flowers

Download the dataset here.

Unzip data

```
!unzip '/content/Flowers-Dataset.zip'
```

```
  inflating: flowers/daisy/3706420943_66f3214862_n.jpg
  inflating: flowers/daisy/3711723108_65247a3170.jpg
  inflating: flowers/daisy/3711892138_b8c953fdc1_z.jpg
  inflating: flowers/daisy/3713290261_8a66de23ab.jpg
  inflating: flowers/daisy/3717746329_53f515c6a6_m.jpg
  inflating: flowers/daisy/3720632920_93cf1cc7f3_m.jpg
  inflating: flowers/daisy/3750250718_eb61146c5f.jpg
  inflating: flowers/daisy/3750771898_cfd50090ba_n.jpg
  inflating: flowers/daisy/3758221664_b19116d61f.jpg
  inflating: flowers/daisy/3764116502_f394428ee0_n.jpg
  inflating: flowers/daisy/3773181799_5def396456.jpg
  inflating: flowers/daisy/3780380240_ef9ec1b737_m.jpg
  inflating: flowers/daisy/3848258315_ed2fde4fb4.jpg
  inflating: flowers/daisy/3861452393_14d2f95157_m.jpg
  inflating: flowers/daisy/3900172983_9312fdf39c_n.jpg
  inflating: flowers/daisy/3939135368_0af5c4982a_n.jpg
  inflating: flowers/daisy/3957488431_52a447c0e8_m.jpg
  inflating: flowers/daisy/3962240986_0661edc43a_n.jpg
  inflating: flowers/daisy/3963330924_6c6a3fa7be_n.jpg
  inflating: flowers/daisy/3975010332_3209f9f447_m.jpg
  inflating: flowers/daisy/3999978867_c67c79597f_m.jpg
  inflating: flowers/daisy/4065883015_4bb6010cb7_n.jpg
  inflating: flowers/daisy/4085794721_7cd88e0a6c_m.jpg
  inflating: flowers/daisy/4117918318_3c8935289b_m.jpg
```

```
inflating: flowers/daisy/4131565290_0585c4dd5a_n.jpg
inflating: flowers/daisy/413815348_764ae83088.jpg
inflating: flowers/daisy/4141147800_813f660b47.jpg
inflating: flowers/daisy/4144275653_7c02d47d9b.jpg
inflating: flowers/daisy/422094774_28acc69a8b_n.jpg
inflating: flowers/daisy/4222584034_8964cbd3de.jpg
inflating: flowers/daisy/4229503616_9b8a42123c_n.jpg
inflating: flowers/daisy/4258408909_b7cc92741c_m.jpg
inflating: flowers/daisy/4268817944_cdbdb226ae.jpg
inflating: flowers/daisy/4276898893_609d11db8b.jpg
inflating: flowers/daisy/4278442064_a5a598524b_m.jpg
inflating: flowers/daisy/4281102584_c548a69b81_m.jpg
inflating: flowers/daisy/4286053334_a75541f20b_m.jpg
inflating: flowers/daisy/4301689054_20519e5b68.jpg
inflating: flowers/daisy/4318007511_e9f4311936_n.jpg
inflating: flowers/daisy/4333085242_bbeb3e2841_m.jpg
inflating: flowers/daisy/43474673_7bb4465a86.jpg
inflating: flowers/daisy/435283392_72e4c5b5d6_m.jpg
inflating: flowers/daisy/437859108_173fb33c98.jpg
inflating: flowers/daisy/4407065098_ef25f1ccac_n.jpg
inflating: flowers/daisy/4413849849_b8d2f3bcf1_n.jpg
inflating: flowers/daisy/4432271543_01c56ca3a9.jpg
inflating: flowers/daisy/4434592930_6610d51fca_m.jpg
inflating: flowers/daisy/4440480869_632ce6aff3_n.jpg
inflating: flowers/daisy/446484749_4044affcaf_n.jpg
inflating: flowers/daisy/4482623536_b9fb5ae41f_n.jpg
inflating: flowers/daisy/4496202781_1d8e776ff5_n.jpg
inflating: flowers/daisy/450128527_fd35742d44.jpg
inflating: flowers/daisy/4511693548_20f9bd2b9c_m.jpg
inflating: flowers/daisy/4534460263_8e9611db3c_n.jpg
inflating: flowers/daisy/4538877108_3c793f7987_m.jpg
inflating: flowers/daisy/4540555191_3254dc4608_n.jpg
inflating: flowers/daisy/4544110929_a7de65d65f_n.jpg
inflating: flowers/daisy/4561871220_47f420ca59_m.jpg
```

1. Image Augmentation

```
#import lib.

from tensorflow.keras.preprocessing.image import ImageDataGenerator


#augmentation on flowers

rose_datagen=ImageDataGenerator(rescale=1./255,
                                zoom_range=0.2,
                                horizontal_flip=True)


tulip_datagen=ImageDataGenerator(rescale=1./255,
                                zoom_range=0.2,
                                horizontal_flip=True)


xrose = rose_datagen.flow_from_directory('/content/flowers',
                                         target_size=(64,64),
                                         class_mode='categorical',
                                         batch_size=100)
```

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

```
xtulip = tulip_datagen.flow_from_directory('/content/flowers',
                                         target_size=(64,64),
                                         class_mode='categorical',
                                         batch_size=100)
```

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

2. Creating a Model

```
#import lib.
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense
```

3. Add Layers (Convolution,MaxPooling,Flatten,Dense-(Hidden Layers),Output)

```
# Add a layers

model = Sequential() # Initializing sequential model
model.add(Convolution2D(32,(3,3),activation='relu',input_shape=(64,64,3))) # convolution layer
model.add(MaxPooling2D(pool_size=(2, 2))) # Max pooling layer
model.add(Flatten()) # Flatten layer
model.add(Dense(300,activation='relu')) # Hidden layer 1
model.add(Dense(150,activation='relu')) # Hidden layer 2
model.add(Dense(5,activation='softmax')) # Output layer
```

4. Compile The Model

```
# Compiling the model

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

5. Fit The Model

```
model.fit_generator(xrose,
                    steps_per_epoch=len(xrose),
                    epochs=10,
```

```
                validation_data=xtulip,
                validation_steps=len(xtulip))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: UserWarning: `Model.fit_generator` is deprecated and will be re
  """
Epoch 1/10
44/44 [==============================] - 50s 1s/step - loss: 1.5695 - accuracy: 0.4044 - val_loss: 1.1486 - val_accuracy: 0.520
Epoch 2/10
44/44 [==============================] - 49s 1s/step - loss: 1.1106 - accuracy: 0.5534 - val_loss: 1.0660 - val_accuracy: 0.577
Epoch 3/10
44/44 [==============================] - 48s 1s/step - loss: 1.0315 - accuracy: 0.5930 - val_loss: 0.9773 - val_accuracy: 0.618
Epoch 4/10
44/44 [==============================] - 48s 1s/step - loss: 0.9687 - accuracy: 0.6229 - val_loss: 0.9146 - val_accuracy: 0.655
Epoch 5/10
44/44 [==============================] - 48s 1s/step - loss: 0.9126 - accuracy: 0.6472 - val_loss: 0.9020 - val_accuracy: 0.651
Epoch 6/10
44/44 [==============================] - 48s 1s/step - loss: 0.8666 - accuracy: 0.6727 - val_loss: 0.8045 - val_accuracy: 0.687
Epoch 7/10
44/44 [==============================] - 48s 1s/step - loss: 0.8469 - accuracy: 0.6727 - val_loss: 0.8141 - val_accuracy: 0.692
Epoch 8/10
44/44 [==============================] - 48s 1s/step - loss: 0.7856 - accuracy: 0.7030 - val_loss: 0.7274 - val_accuracy: 0.720
Epoch 9/10
44/44 [==============================] - 48s 1s/step - loss: 0.7607 - accuracy: 0.7054 - val_loss: 0.7500 - val_accuracy: 0.712
Epoch 10/10
44/44 [==============================] - 49s 1s/step - loss: 0.7310 - accuracy: 0.7239 - val_loss: 0.7089 - val_accuracy: 0.726
<keras.callbacks.History at 0x7fdc896ca890>
```

6. Save The Model

```
model.save('rose.h5')
```

7. Test The Model

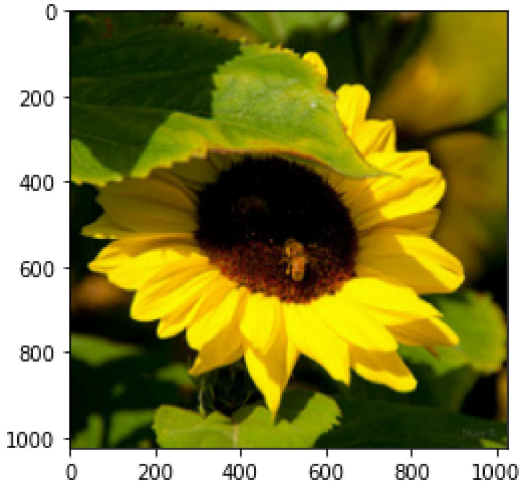
```
from tensorflow.keras.preprocessing import image
import numpy as np
import matplotlib.pyplot as plt
```

```
#testing 1
img = image.load_img('/content/flowers/sunflower/12471443383_b71e7a7480_m.jpg',target_size=(64,64)) # Reading image
x = image.img_to_array(img) # Converting image into array
x = np.expand_dims(x,axis=0) # expanding Dimensions
pred = np.argmax(model.predict(x)) # Predicting the higher probablity index
op = ['daisy','dandelion','rose','sunflower','tulip'] # Creating list
op[pred] # List indexing with output
```

```
'sunflower'
```

```
img = image.load_img('/content/flowers/sunflower/12471443383_b71e7a7480_m.jpg',target_size=(1024,1024))
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fdc88b06710>
```




```
img = image.load_img('/content/flowers/rose/14145188939_b4de638bd3_n.jpg',target_size=(1024,1024))
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7fdc887de490>
```