Double-click (or enter) to edit

Unzipping the data

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

```
Archive:  /content/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
```

Data Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator # Import necessary lib.


# Data augmentation on training variable
```

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


# Data augmentation on testing variable

test_datagen = ImageDataGenerator(rescale=1./255)


# Data augmentation on training data

xtrain = train_datagen.flow_from_directory('/content/flowers',
                                           target_size=(64,64),
                                           class_mode='categorical',
                                           batch_size=100)



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

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

CNN Model Creation

```
# Importing req. lib.

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

```
 # Build a CNN block

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#
```

Compiling the model

```
# Compiling the model

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

Fit The Model

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau


early_stopping = EarlyStopping(monitor='val_accuracy',
                        patience=5)
reduce_lr = ReduceLROnPlateau(monitor='val_accuracy',
                        patience=5,
                        factor=0.5,min_lr=0.00001)

callback = [reduce_lr,early_stopping]


#Fit The Model
```

```
model.fit_generator(xtrain,
                    steps_per_epoch=len(xtrain),
                    epochs=100,
                    callbacks=callback,
                    validation_data=xtest,
                    validation_steps=len(xtest))
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: UserWarning: `Model.fit_generator` is deprecated and will be

Epoch 1/100
44/44 [==============================] - 46s 1s/step - loss: 1.8332 - accuracy: 0.3433 - val_loss: 1.2083 - val_accuracy: 0.4
Epoch 2/100
44/44 [==============================] - 44s 1s/step - loss: 1.1639 - accuracy: 0.5140 - val_loss: 1.1734 - val_accuracy: 0.5
Epoch 3/100
44/44 [==============================] - 46s 1s/step - loss: 1.0839 - accuracy: 0.5650 - val_loss: 0.9902 - val_accuracy: 0.6
Epoch 4/100
44/44 [==============================] - 44s 1s/step - loss: 0.9972 - accuracy: 0.6229 - val_loss: 1.1430 - val_accuracy: 0.5
Epoch 5/100
44/44 [==============================] - 44s 1s/step - loss: 0.9589 - accuracy: 0.6331 - val_loss: 1.0079 - val_accuracy: 0.6
Epoch 6/100
44/44 [==============================] - 46s 1s/step - loss: 0.8961 - accuracy: 0.6574 - val_loss: 0.8191 - val_accuracy: 0.6
Epoch 7/100
44/44 [==============================] - 46s 1s/step - loss: 0.8802 - accuracy: 0.6688 - val_loss: 0.7923 - val_accuracy: 0.7
Epoch 8/100
44/44 [==============================] - 44s 1s/step - loss: 0.8121 - accuracy: 0.6880 - val_loss: 0.7668 - val_accuracy: 0.7
Epoch 9/100
44/44 [==============================] - 45s 1s/step - loss: 0.7726 - accuracy: 0.7074 - val_loss: 0.7821 - val_accuracy: 0.7
Epoch 10/100
44/44 [==============================] - 46s 1s/step - loss: 0.7751 - accuracy: 0.7058 - val_loss: 0.8353 - val_accuracy: 0.7
Epoch 11/100
44/44 [==============================] - 44s 1s/step - loss: 0.7253 - accuracy: 0.7241 - val_loss: 0.7078 - val_accuracy: 0.7
Epoch 12/100
44/44 [==============================] - 45s 1s/step - loss: 0.6879 - accuracy: 0.7475 - val_loss: 0.6531 - val_accuracy: 0.7
Epoch 13/100
44/44 [==============================] - 45s 1s/step - loss: 0.6957 - accuracy: 0.7392 - val_loss: 0.6171 - val_accuracy: 0.7
Epoch 14/100
44/44 [==============================] - 44s 1s/step - loss: 0.6813 - accuracy: 0.7457 - val_loss: 0.5893 - val_accuracy: 0.7
Epoch 15/100
44/44 [==============================] - 45s 1s/step - loss: 0.6403 - accuracy: 0.7582 - val_loss: 0.5694 - val_accuracy: 0.7
```

```
Epoch 16/100
44/44 [==============================] - 45s 1s/step - loss: 0.6283 - accuracy: 0.7607 - val_loss: 0.5750 - val_accuracy: 0.7
Epoch 17/100
44/44 [==============================] - 44s 1s/step - loss: 0.6053 - accuracy: 0.7739 - val_loss: 0.6091 - val_accuracy: 0.7
Epoch 18/100
44/44 [==============================] - 45s 1s/step - loss: 0.5883 - accuracy: 0.7774 - val_loss: 0.5886 - val_accuracy: 0.7
Epoch 19/100
44/44 [==============================] - 46s 1s/step - loss: 0.5642 - accuracy: 0.7924 - val_loss: 0.6080 - val_accuracy: 0.7
Epoch 20/100
44/44 [==============================] - 44s 1s/step - loss: 0.5659 - accuracy: 0.7941 - val_loss: 0.5321 - val_accuracy: 0.8
Epoch 21/100
44/44 [==============================] - 44s 1s/step - loss: 0.5088 - accuracy: 0.8151 - val_loss: 0.4650 - val_accuracy: 0.8
Epoch 22/100
44/44 [==============================] - 45s 1s/step - loss: 0.5435 - accuracy: 0.7996 - val_loss: 0.5500 - val_accuracy: 0.8
Epoch 23/100
44/44 [==============================] - 44s 1s/step - loss: 0.4878 - accuracy: 0.8219 - val_loss: 0.5577 - val_accuracy: 0.7
Epoch 24/100
44/44 [==============================] - 44s 1s/step - loss: 0.4627 - accuracy: 0.8339 - val_loss: 0.3762 - val_accuracy: 0.8
Epoch 25/100
44/44 [==============================] - 44s 1s/step - loss: 0.4362 - accuracy: 0.8453 - val_loss: 0.4182 - val_accuracy: 0.8
Epoch 26/100
44/44 [==============================] - 45s 1s/step - loss: 0.4231 - accuracy: 0.8464 - val_loss: 0.3928 - val_accuracy: 0.8
Epoch 27/100
44/44 [==============================] - 44s 1s/step - loss: 0.4101 - accuracy: 0.8517 - val_loss: 0.3149 - val_accuracy: 0.8
Epoch 28/100
```

Saving the Model

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

Testing the Model

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

#Testing
img = image.load_img('/content/flowers/rose/5349865018_99cd7f985a_n.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
```

```
'rose'
```

```
img
```

```
#Testing 2
img = image.load_img('/content/flowers/dandelion/34351602790_37234e2dae_n.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
```

```
#Testing 3
img = image.load_img('/content/flowers/tulip/13974542496_e4b5d1c913_n.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'
```

```
#Testing Google Image
img = image.load_img('/content/marguerite-5959944__340.webp',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
```

```
'daisy'
```

```
img
```