**Importing the libraries**

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** Dense

**from** tensorflow.keras.layers **import** Convolution2D

**from** tensorflow.keras.layers **import** MaxPooling2D

**from** tensorflow.keras.layers **import** Flatten

*#import the preprocess library of image*

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

**Image Augmentation**

train\_datagen **=**

ImageDataGenerator(rescale**=**1.**/**255,shear\_range**=**0.2,zoom\_range**=**0.2,horizontal\_flip**=True**,vertical\_flip**=Tru e**)

*#rescale = pixel value rescaling to 0 to 1 from 0 to 255 #shear\_range => counter clock wise rotation(anti clock)*

test\_datagen **=** ImageDataGenerator(rescale**=**1.**/**255)

*#load your images data*

*#load your images data*

x\_train **=** train\_datagen**.**flow\_from\_directory(r"D:\IBM project\Flowers-Dataset\dataset\Training",target\_size**=**(128,128),batch\_size**=**32,class\_mode**=**"categorical")

Found 3457 images belonging to 5 classes.

x\_test **=** test\_datagen**.**flow\_from\_directory(r"D:\IBM project\Flowers-Dataset\dataset\Testing",target\_size**=**(128,128),batch\_size**=**32,class\_mode**=**"categorical")

Found 860 images belonging to 5 classes.

x\_train**.**class\_indices

{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}

**Create Model**

*#initialize the model*

model **=** Sequential()

**Add Layers**

**(Convolution,MaxPooling,Flatten,Dense-**

**(Hidden Layers),Output)**

*#add convlution layer*

model**.**add(Convolution2D(32,(3,3),input\_shape**=**(128,128,3),activation**=**'relu'))

*# 32 => no of feature detectors*

*#(3,3)=> kernel size(feature detector size => 3\*3 matrix)*

*#add maxpooling layer*

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

*# you can add more convolutiona and pooling layers*

model**.**add(Convolution2D(32,(3,3),input\_shape**=**(128,128,3),activation**=**'relu'))

model**.**add(MaxPooling2D(pool\_size**=**(2,2)))

*#flatten layer => input layer to your ANN*

model**.**add(Flatten())

*#hidden layers*

model**.**add(Dense(units**=**500,kernel\_initializer**=**"random\_uniform",activation**=**"relu"))

model**.**add(Dense(units**=**200,kernel\_initializer**=**"random\_uniform",activation**=**"relu"))

model**.**add(Dense(units**=**300,kernel\_initializer**=**"random\_uniform",activation**=**"relu"))

model**.**add(Dense(units**=**400,kernel\_initializer**=**"random\_uniform",activation**=**"relu"))

*#output layer*

model**.**add(Dense(units**=**5,kernel\_initializer**=**"random\_uniform",activation**=**"softmax"))

**Compile The Model**

*#compile the model*

model**.**compile(loss**=**"categorical\_crossentropy",optimizer**=**"adam",metrics**=**["accuracy"])

**Fit The Model**

*#train the model*

model**.**fit(x\_train,steps\_per\_epoch**=**109,epochs**=**25,validation\_data**=**x\_test,validation\_steps**=**27) *#steps\_per\_epoch = no of train images/batch size #validation\_steps = no of test images/batch size*

Epoch 1/25

109/109 [==============================] - 93s 822ms/step - loss: 1.3275 - accuracy: 0.3908 -

val\_loss: 1.7720 - val\_accuracy: 0.3081

Epoch 2/25

109/109 [==============================] - 83s 762ms/step - loss: 1.2066 - accuracy: 0.4631 -

val\_loss: 1.2472 - val\_accuracy: 0.4221

Epoch 3/25

109/109 [==============================] - 83s 762ms/step - loss: 1.0958 - accuracy: 0.5270 -

val\_loss: 1.1586 - val\_accuracy: 0.5256

Epoch 4/25

109/109 [==============================] - 92s 843ms/step - loss: 1.0594 - accuracy: 0.5733 -

val\_loss: 1.1411 - val\_accuracy: 0.5384

Epoch 5/25

109/109 [==============================] - 88s 803ms/step - loss: 0.9549 - accuracy: 0.6225 -

val\_loss: 1.1228 - val\_accuracy: 0.5430

Epoch 6/25

109/109 [==============================] - 84s 765ms/step - loss: 0.8991 - accuracy: 0.6402 -

val\_loss: 1.0456 - val\_accuracy: 0.5709

Epoch 7/25

109/109 [==============================] - 89s 811ms/step - loss: 0.8736 - accuracy: 0.6575 -

val\_loss: 1.1910 - val\_accuracy: 0.5802

Epoch 8/25

109/109 [==============================] - 87s 794ms/step - loss: 0.9027 - accuracy: 0.6468 -

val\_loss: 1.0589 - val\_accuracy: 0.6070

Epoch 9/25

109/109 [==============================] - 86s 788ms/step - loss: 0.8072 - accuracy: 0.6882 -

val\_loss: 1.0385 - val\_accuracy: 0.6163

Epoch 10/25

109/109 [==============================] - 90s 820ms/step - loss: 0.7578 - accuracy: 0.7102 -

val\_loss: 1.1246 - val\_accuracy: 0.5872

Epoch 11/25

109/109 [==============================] - 84s 772ms/step - loss: 0.7377 - accuracy: 0.7142 -

val\_loss: 1.0831 - val\_accuracy: 0.5872

Epoch 12/25

109/109 [==============================] - 94s 863ms/step - loss: 0.7545 - accuracy: 0.7067 -

val\_loss: 1.0106 - val\_accuracy: 0.5884

Epoch 13/25

109/109 [==============================] - 99s 905ms/step - loss: 0.7118 - accuracy: 0.7423 -

val\_loss: 1.0672 - val\_accuracy: 0.6058

Epoch 14/25

109/109 [==============================] - 91s 831ms/step - loss: 0.6494 - accuracy: 0.7547 -

val\_loss: 0.9917 - val\_accuracy: 0.6384

Epoch 15/25

109/109 [==============================] - 120s 1s/step - loss: 0.6305 - accuracy: 0.7605 - val\_loss:

1.2212 - val\_accuracy: 0.5930

Epoch 16/25

109/109 [==============================] - 96s 883ms/step - loss: 0.5863 - accuracy: 0.7787 -

val\_loss: 1.0767 - val\_accuracy: 0.6279

Epoch 17/25

109/109 [==============================] - 91s 832ms/step - loss: 0.5464 - accuracy: 0.7964 -

val\_loss: 1.1028 - val\_accuracy: 0.6360

Epoch 18/25

109/109 [==============================] - 91s 836ms/step - loss: 0.5666 - accuracy: 0.7935 -

val\_loss: 1.0856 - val\_accuracy: 0.6209

Epoch 19/25

109/109 [==============================] - 88s 808ms/step - loss: 0.5793 - accuracy: 0.7891 -

val\_loss: 1.0319 - val\_accuracy: 0.6512

Epoch 20/25

109/109 [==============================] - 84s 771ms/step - loss: 0.5085 - accuracy: 0.8117 -

val\_loss: 1.2402 - val\_accuracy: 0.6116

Epoch 21/25

109/109 [==============================] - 82s 754ms/step - loss: 0.5008 - accuracy: 0.8146 -

val\_loss: 1.0975 - val\_accuracy: 0.6221

Epoch 22/25

109/109 [==============================] - 82s 752ms/step - loss: 0.4399 - accuracy: 0.8423 -

val\_loss: 1.1795 - val\_accuracy: 0.6209

Epoch 23/25

109/109 [==============================] - 81s 741ms/step - loss: 0.4287 - accuracy: 0.8426 -

val\_loss: 1.3299 - val\_accuracy: 0.6267

Epoch 24/25

109/109 [==============================] - 85s 777ms/step - loss: 0.4200 - accuracy: 0.8455 -

val\_loss: 1.3333 - val\_accuracy: 0.6395

Epoch 25/25

109/109 [==============================] - 80s 731ms/step - loss: 0.4816 - accuracy: 0.8212 -

val\_loss: 1.1663 - val\_accuracy: 0.6500

<keras.callbacks.History at 0x1fc8ad5fa90>

**Save The Model**

model**.**save("flowers.h5")

**Test The Model**

**from** tensorflow.keras.models **import** load\_model

**from** tensorflow.keras.preprocessing **import** image

**import** numpy **as** np

model **=** load\_model("flowers.h5")

img **=** image**.**load\_img("sunflower.jpg",target\_size**=**(128,128))

Img



1. **=** image**.**img\_to\_array(img)

array([[[210., 222., 238.],

[186., 208., 221.],

[215., 224., 241.],

...,

[200., 212., 236.],

[192., 210., 230.],

[196., 213., 233.]],

[[190., 206., 222.],

[195., 214., 229.],

[191., 207., 223.],

...,

[186., 204., 224.],

[180., 200., 224.],

[184., 202., 224.]],

[[184., 205., 222.],

[201., 216., 235.],

[189., 210., 227.],

...,

[172., 196., 224.],

[171., 192., 219.],

[178., 198., 222.]],

...,

[[109., 133., 75.],

[111., 135., 77.],

[128., 152., 94.],

...,

[122., 128., 56.],

[ 69., 85., 12.],

[ 76., 93., 22.]],

[[104., 128., 70.],

[106., 130., 72.],

[107., 131., 73.],

...,

[ 92., 98., 36.],

[151., 166., 101.],

[ 43., 56., 13.]],

|  |  |
| --- | --- |
| [[103., 127., | 69.], |
| [112., 136., | 78.], |
| [118., 142., | 84.], |
| ..., |  |
| [115., 121., | 73.], |
| [122., 139., | 84.], |

[ 74., 86., 48.]]], dtype=float32)

x**.**shape

(128, 128, 3)

*#(1,64,64,3) to expand the dims*

1. **=** np**.**expand\_dims(x,axis**=**0)
2. shape

(1, 128, 128, 3)

pred\_prob **=** model**.**predict(x)

1/1 [==============================] - 0s 177ms/step pred\_prob

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

class\_name**=**['daisy','dandelion','rose','sunflower','tulip']

pred\_id **=** pred\_prob**.**argmax(axis**=**1)[0]

pred\_id

3

print("predicted animal is ",str(class\_name[pred\_id]))

predicted animal is sunflower