**Assignment 3**

**CNN MODEL FOR FLOWER CLASSIFICATION**

**Pre-Requisites**

In [1]:

**from** google.colab **import** drive

drive**.**mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force\_remount=True).

**STEP 1 UNZIP FILES**

In [2]:

cd**/**content**/**drive**/**MyDrive**/**AI\_IBM

/content/drive/MyDrive/AI\_IBM

In [3]:

!unzip Flowers**-**Dataset**.**zip

Archive: Flowers-Dataset.zip

replace flowers/daisy/100080576\_f52e8ee070\_n.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: N

**STEP 2 Image Augumentation**

In [4]:

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

In [5]:

train\_datagen**=**ImageDataGenerator(rescale**=**1.**/**255, zoom\_range**=**0.2,horizontal\_flip**=True**,vertical\_flip**=False**)

In [6]:

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

In [7]:

x\_train**=**train\_datagen**.**flow\_from\_directory(r"/content/drive/MyDrive/AI\_IBM/flowers",target\_size**=**(64,64),class\_mode**=**'categorical',batch\_size**=**24)

Found 4317 images belonging to 5 classes.

In [8]:

x\_test**=**test\_datagen**.**flow\_from\_directory(r"/content/drive/MyDrive/AI\_IBM/flowers",target\_size**=**(64,64),class\_mode**=**'categorical',batch\_size**=**24)

Found 4317 images belonging to 5 classes.

In [9]:

x\_train**.**class\_indices

Out[9]:

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

**Step -3 Initializing CNN And Create Model**

In [10]:

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

**from** tensorflow.keras.layers **import** Dense,Convolution2D,MaxPooling2D,Flatten

**Step -4 Add layers**

In [11]:

model**=**Sequential()

**4.1 Input Layers (Convolution ,MaxPooling,Flatten)**

In [12]:

model**.**add(Convolution2D(32,(3,3),input\_shape**=**(64,64,3),activation**=**'relu'))

In [13]:

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

In [14]:

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

In [15]:

model**.**summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 62, 62, 32) 896

max\_pooling2d (MaxPooling2D (None, 31, 31, 32) 0

)

flatten (Flatten) (None, 30752) 0

=================================================================

Total params: 896

Trainable params: 896

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4.2 Hidden Layers**

In [16]:

model**.**add(Dense(300,activation**=**'relu'))

model**.**add(Dense(150,activation**=**'relu'))

**4.3 Output Layers**

model**.**add(Dense(5,activation**=**'softmax'))

In [18]:

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

In [19]:

len(x\_train)

Out[19]:

180

**Step -5 Train the Model**

In [20]:

model**.**fit\_generator(x\_train,steps\_per\_epoch**=**len(x\_train), validation\_data**=**x\_test, validation\_steps**=**len(x\_test), epochs**=** 30)

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: UserWarning: `Model.fit\_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.

"""Entry point for launching an IPython kernel.

Epoch 1/30

180/180 [==============================] - 393s 2s/step - loss: 1.3213 - accuracy: 0.4714 - val\_loss: 1.1275 - val\_accuracy: 0.5532

Epoch 2/30

180/180 [==============================] - 74s 409ms/step - loss: 1.0600 - accuracy: 0.5854 - val\_loss: 0.9406 - val\_accuracy: 0.6301

Epoch 3/30

180/180 [==============================] - 73s 405ms/step - loss: 0.9678 - accuracy: 0.6247 - val\_loss: 0.9603 - val\_accuracy: 0.6203

Epoch 4/30

180/180 [==============================] - 77s 429ms/step - loss: 0.8884 - accuracy: 0.6546 - val\_loss: 0.8187 - val\_accuracy: 0.6938

Epoch 5/30

180/180 [==============================] - 76s 422ms/step - loss: 0.8358 - accuracy: 0.6787 - val\_loss: 0.7393 - val\_accuracy: 0.7225

Epoch 6/30

180/180 [==============================] - 75s 418ms/step - loss: 0.7924 - accuracy: 0.6965 - val\_loss: 0.8389 - val\_accuracy: 0.6928

Epoch 7/30

180/180 [==============================] - 73s 405ms/step - loss: 0.7521 - accuracy: 0.7158 - val\_loss: 0.8503 - val\_accuracy: 0.6789

Epoch 8/30

180/180 [==============================] - 74s 411ms/step - loss: 0.7048 - accuracy: 0.7313 - val\_loss: 0.6492 - val\_accuracy: 0.7521

Epoch 9/30

180/180 [==============================] - 72s 400ms/step - loss: 0.6502 - accuracy: 0.7521 - val\_loss: 0.6458 - val\_accuracy: 0.7438

Epoch 10/30

180/180 [==============================] - 74s 409ms/step - loss: 0.6182 - accuracy: 0.7684 - val\_loss: 0.5721 - val\_accuracy: 0.7818

Epoch 11/30

180/180 [==============================] - 72s 402ms/step - loss: 0.5662 - accuracy: 0.7931 - val\_loss: 0.5968 - val\_accuracy: 0.7725

Epoch 12/30

180/180 [==============================] - 72s 401ms/step - loss: 0.5600 - accuracy: 0.7908 - val\_loss: 0.6907 - val\_accuracy: 0.7612

Epoch 13/30

180/180 [==============================] - 72s 399ms/step - loss: 0.5064 - accuracy: 0.8138 - val\_loss: 0.5185 - val\_accuracy: 0.8117

Epoch 14/30

180/180 [==============================] - 71s 394ms/step - loss: 0.4830 - accuracy: 0.8249 - val\_loss: 0.3613 - val\_accuracy: 0.8673

Epoch 15/30

180/180 [==============================] - 71s 397ms/step - loss: 0.4650 - accuracy: 0.8196 - val\_loss: 0.3396 - val\_accuracy: 0.8768

Epoch 16/30

180/180 [==============================] - 71s 393ms/step - loss: 0.4117 - accuracy: 0.8559 - val\_loss: 0.3472 - val\_accuracy: 0.8738

Epoch 17/30

180/180 [==============================] - 71s 397ms/step - loss: 0.3892 - accuracy: 0.8631 - val\_loss: 0.3314 - val\_accuracy: 0.8826

Epoch 18/30

180/180 [==============================] - 70s 389ms/step - loss: 0.3441 - accuracy: 0.8726 - val\_loss: 0.4008 - val\_accuracy: 0.8589

Epoch 19/30

180/180 [==============================] - 73s 404ms/step - loss: 0.3467 - accuracy: 0.8719 - val\_loss: 0.2484 - val\_accuracy: 0.9060

Epoch 20/30

180/180 [==============================] - 72s 398ms/step - loss: 0.3327 - accuracy: 0.8758 - val\_loss: 0.2234 - val\_accuracy: 0.9210

Epoch 21/30

180/180 [==============================] - 73s 403ms/step - loss: 0.2807 - accuracy: 0.9009 - val\_loss: 0.2830 - val\_accuracy: 0.9036

Epoch 22/30

180/180 [==============================] - 70s 392ms/step - loss: 0.2751 - accuracy: 0.9013 - val\_loss: 0.2392 - val\_accuracy: 0.9141

Epoch 23/30

180/180 [==============================] - 73s 404ms/step - loss: 0.2549 - accuracy: 0.9097 - val\_loss: 0.2221 - val\_accuracy: 0.9189

Epoch 24/30

180/180 [==============================] - 72s 399ms/step - loss: 0.2412 - accuracy: 0.9243 - val\_loss: 0.2029 - val\_accuracy: 0.9291

Epoch 25/30

180/180 [==============================] - 72s 402ms/step - loss: 0.2360 - accuracy: 0.9199 - val\_loss: 0.1965 - val\_accuracy: 0.9307

Epoch 26/30

180/180 [==============================] - 72s 401ms/step - loss: 0.2199 - accuracy: 0.9201 - val\_loss: 0.1919 - val\_accuracy: 0.9331

Epoch 27/30

180/180 [==============================] - 72s 400ms/step - loss: 0.2008 - accuracy: 0.9363 - val\_loss: 0.1218 - val\_accuracy: 0.9560

Epoch 28/30

180/180 [==============================] - 73s 406ms/step - loss: 0.1889 - accuracy: 0.9310 - val\_loss: 0.2838 - val\_accuracy: 0.9108

Epoch 29/30

180/180 [==============================] - 70s 389ms/step - loss: 0.2046 - accuracy: 0.9275 - val\_loss: 0.2116 - val\_accuracy: 0.9307

Epoch 30/30

180/180 [==============================] - 70s 392ms/step - loss: 0.1886 - accuracy: 0.9372 - val\_loss: 0.2091 - val\_accuracy: 0.9280

Out[20]:

**Step -6 Save The model**

In [21]:

model**.**save('Flowers\_classification\_model1.h5')

**Step -7 Test The model**

In [22]:

ls

**flowers**/ Flowers\_classification\_model1.h5 Flowers-Dataset.zip video.mp4

In [23]:

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

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

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

In [24]:

*# Load the model*

model**=**load\_model('Flowers\_classification\_model1.h5')

In [38]:

img**=**image**.**load\_img(r"/content/s3.jpg",target\_size**=**(64,64))

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

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

y**=**np**.**argmax(model**.**predict(x),axis**=**1)

*# x\_train.class\_indices*

index**=**['daisy','dandelion','rose','sunflower','tulip']

index[y[0]]

Out[38]:

'sunflower'

**We Achieved 93 percent of accuracy with this model**