Assignment-3

Problem Statement:- Build CNN Models for Classification Of Flowers

|  |  |
| --- | --- |
| Assignment Date | 14 November 2022 |
| Student Name | S.Abirami |
| Student Roll Number | 822019104001 |
| Maximum Marks | 2 Marks |

In [1]:

**import** splitfolders

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

**import** tensorflow **as** tf

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

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

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

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

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

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

**from** tensorflow.keras.applications.resnet50 **import** preprocess\_input, decode\_predictions

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

**import** matplotlib.pyplot **as** plt

**2. Image Augmentation**

In [2] :

rain\_datagen =ImageDataGenerator(rescale**=**1.**/**255,zoom\_range**=**0.2,horizontal\_flip**=True**,vertical\_flip**=False**) In [3]:

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

In  [4]:

input\_folder **=** 'C:\\Users\\manok\\Documents\\Sem\_7\\HX5001-HX6001\\Assignment\\Assignment\_3\\flowers'

In [5]:

splitfolders**.**ratio(input\_folder,output**=**"C:\\Users\\manok\\Documents\\Sem\_7\\HX5001-HX6001\\Assignment\\Assignment\_3\\flowersdataset",ratio**=**(.8,0,.2),group\_prefix**=None**)

Copying files: 4317 files [00:26, 166.01 files/s]

In [6]:

x\_train**=**train\_datagen**.**flow\_from\_directory(r"C:\Users\manok\Documents\Sem\_7\HX5001-HX6001\Assignment\Assignment\_3\flowersdataset\train",target\_size**=**(64,64),class\_mode**=**'categorical',batch\_size**=**24)

Found 3452 images belonging to 5 classes.

In [7]:

x\_test**=**test\_datagen**.**flow\_from\_directory(r"C:\Users\manok\Documents\Sem\_7\HX5001-HX6001\Assignment\Assignment\_3\flowersdataset\test",target\_size**=**(64,64),class\_mode**=**'categorical',batch\_size**=**24)

Found 865 images belonging to 5 classes.

In [8]:

x\_train**.**class\_indices

Out[8]:

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

3. Create Model In [9]:

model**=**Sequential()

**4. Add Layers**

4.1. Convolution Layer

In [10]:

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

4.2. MaxPooling Layer

In [11]:

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

4.3. Flatten Layer

In [12]:

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

4.4. Dense Layer

In [13]:

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

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

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

dense (Dense) (None, 300) 9225900

dense\_1 (Dense) (None, 150) 45150

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

Total params: 9,271,946

Trainable params: 9,271,946

Non-trainable params: 0

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

4.5. Output Layer

model**.**add(Dense(5,activation**=**'softmax') In [16]:

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

dense (Dense) (None, 300) 9225900

dense\_1 (Dense) (None, 150) 45150

dense\_2 (Dense) (None, 5) 755

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

Total params: 9,272,701

Trainable params: 9,272,701

Non-trainable params: 0

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

**5. Compile The Model**

In [17]:

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

len(x\_train)

Out[17]:

144

**6. Fit The Model**

In [18]:

epo**=**20

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

Epoch 1/20

144/144 [==============================] - 81s 552ms/step - loss: 1.4602 - accuracy: 0.4429 - val\_loss: 1.2587 - val\_accuracy: 0.5052

Epoch 2/20

144/144 [==============================] - 38s 264ms/step - loss: 1.0639 - accuracy: 0.5773 - val\_loss: 1.1310 - val\_accuracy: 0.5399

Epoch 3/20

144/144 [==============================] - 38s 263ms/step - loss: 0.9872 - accuracy: 0.6066 - val\_loss: 1.0271 - val\_accuracy: 0.5861

Epoch 4/20

144/144 [==============================] - 38s 263ms/step - loss: 0.9298 - accuracy: 0.6251 - val\_loss: 1.0208 - val\_accuracy: 0.6266

Epoch 5/20

144/144 [==============================] - 38s 264ms/step - loss: 0.8497 - accuracy: 0.6651 - val\_loss: 0.9911 - val\_accuracy: 0.6428

Epoch 6/20

144/144 [==============================] - 38s 263ms/step - loss: 0.8255 - accuracy: 0.6727 - val\_loss: 1.1223 - val\_accuracy: 0.6023

Epoch 7/20

144/144 [==============================] - 36s 253ms/step - loss: 0.7639 - accuracy: 0.7048 - val\_loss: 1.0702 - val\_accuracy: 0.6243

Epoch 8/20

144/144 [==============================] - 37s 254ms/step - loss: 0.7179 - accuracy: 0.7170 - val\_loss: 1.1313 - val\_accuracy: 0.5873

Epoch 9/20

144/144 [==============================] - 37s 254ms/step - loss: 0.6676 - accuracy: 0.7352 - val\_loss: 0.9532 - val\_accuracy: 0.6647

Epoch 10/20

144/144 [==============================] - 36s 252ms/step - loss: 0.6323 - accuracy: 0.7567 - val\_loss: 0.9810 - val\_accuracy: 0.6532

Epoch 11/20

144/144 [==============================] - 37s 253ms/step - loss: 0.6231 - accuracy: 0.7590 - val\_loss: 1.0481 - val\_accuracy: 0.6439

Epoch 12/20

144/144 [==============================] - 36s 254ms/step - loss: 0.5839 - accuracy: 0.7775 - val\_loss: 1.0738 - val\_accuracy: 0.6821

Epoch 13/20

144/144 [==============================] - 36s 253ms/step - loss: 0.5251 - accuracy: 0.8097 - val\_loss: 0.9613 - val\_accuracy: 0.6682

Epoch 14/20

144/144 [==============================] - 35s 245ms/step - loss: 0.4838 - accuracy: 0.8143 - val\_loss: 1.0360 - val\_accuracy: 0.6682

Epoch 15/20

144/144 [==============================] - 96s 667ms/step - loss: 0.4308 - accuracy: 0.8433 - val\_loss: 1.1060 - val\_accuracy: 0.6647

Epoch 16/20

144/144 [==============================] - 26s 180ms/step - loss: 0.4230 - accuracy: 0.8491 - val\_loss: 1.2172 - val\_accuracy: 0.6590

Epoch 17/20

144/144 [==============================] - 28s 192ms/step - loss: 0.4122 - accuracy: 0.8517 - val\_loss: 1.0914 - val\_accuracy: 0.6694

Epoch 18/20

144/144 [==============================] - 29s 199ms/step - loss: 0.3877 - accuracy: 0.8644 - val\_loss: 1.4504 - val\_accuracy: 0.5988

Epoch 19/20

144/144 [==============================] - 39s 271ms/step - loss: 0.3670 - accuracy: 0.8653 - val\_loss: 1.2226 - val\_accuracy: 0.6428

Epoch 20/20

144/144 [==============================] - 39s 272ms/step - loss: 0.3131 - accuracy: 0.8853 - val\_loss: 1.2005 - val\_accuracy: 0.6798 In [19]:

epochs\_range **=** range(epo)

plt**.**figure(figsize**=**(8, 8))

plt**.**plot(epochs\_range, history**.**history['accuracy'], label**=**'Training Accuracy')

plt**.**plot(epochs\_range, history**.**history['val\_accuracy'], label**=**'Validation Accuracy')

plt**.**legend()

plt**.**title('Training and Validation Accuracy')

plt**.**show()



plt**.**figure(figsize**=**(8, 8))

plt**.**plot(epochs\_range, history**.**history['loss'], label**=**'Training Loss')

plt**.**plot(epochs\_range, history**.**history['val\_loss'], label**=**'Validation Loss')

plt**.**legend()

plt**.**title('Training and Validation Loss')

plt**.**show()



**7. Save the Model**

In [21]:

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

**8. Test the Model**

In [22]:

img**=**image**.**load\_img(r"C:\Users\manok\Documents\Sem\_7\HX5001-HX6001\Assignment\Assignment\_3\flowersdataset\test\daisy\3706420943\_66f3214862\_n.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','dandellion','rose','sunflower','tulip']

index[y[0]]

1/1 [==============================] - 1s 661ms/step

Out[22]:

'daisy'

In [23]:

img\_url **=** "<https://storage.googleapis.com/download.tensorflow.org/example_images/592px-Red_sunflower.jpg>"

img\_path **=** tf**.**keras**.**utils**.**get\_file('Red\_sunflower', origin**=**img\_url)

img **=** image**.**load\_img(img\_path, target\_size**=**(224, 224))

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

img\_batch **=** np**.**expand\_dims(img\_array, axis**=**0)

img\_preprocessed **=** preprocess\_input(img\_batch)

model **=** tf**.**keras**.**applications**.**resnet50**.**ResNet50()

prediction **=** model**.**predict(img\_preprocessed)

print(decode\_predictions(prediction, top**=**3)[0])

score **=** tf**.**nn**.**softmax(prediction[0])

1/1 [==============================] - 2s 2s/step

[('n11939491', 'daisy', 0.57757616), ('n02206856', 'bee', 0.249383), ('n03991062', 'pot', 0.01181932)]