Date :07.10.2022

Name : KARTHIKEYAN

Roll No :611719104008

# Assignment3-BuildCNNModelForClassificationOfFlowers

1. Unzipdataset

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

Archive:/content/Flowers-Dataset.zip



inflating: flowers/daisy/100080576\_f52e8ee070\_n.jpginflating:flowers/daisy/10140303196\_b88d3d6cec.jpg

inflating: flowers/daisy/10172379554\_b296050f82\_n.jpginflating: flowers/daisy/10172567486\_2748826a8b.jpg

inflating: flowers/daisy/10172636503\_21bededa75\_n.jpginflating: flowers/daisy/102841525\_bd6628ae3c.jpg

inflating: flowers/daisy/10300722094\_28fa978807\_n.jpg

inflating: flowers/daisy/1031799732\_e7f4008c03.jpg

inflating: flowers/daisy/10391248763\_1d16681106\_n.jpginflating: flowers/daisy/10437754174\_22ec990b77\_m.jpginflating: flowers/daisy/10437770546\_8bb6f7bdd3\_m.jpginflating: flowers/daisy/10437929963\_bc13eebe0c.jpg

inflating: flowers/daisy/10466290366\_cc72e33532.jpginflating: flowers/daisy/10466558316\_a7198b87e2.jpginflating: flowers/daisy/10555749515\_13a12a026e.jpginflating:flowers/daisy/10555815624\_dc211569b0.jpg

inflating: flowers/daisy/10555826524\_423eb8bf71\_n.jpginflating: flowers/daisy/10559679065\_50d2b16f6d.jpg

inflating: flowers/daisy/105806915\_a9c13e2106\_n.jpginflating: flowers/daisy/10712722853\_5632165b04.jpginflating:flowers/daisy/107592979\_aaa9cdfe78\_m.jpg

inflating: flowers/daisy/10770585085\_4742b9dac3\_n.jpginflating: flowers/daisy/10841136265\_af473efc60.jpg

inflating: flowers/daisy/10993710036\_2033222c91.jpg

inflating: flowers/daisy/10993818044\_4c19b86c82.jpginflating: flowers/daisy/10994032453\_ac7f8d9e2e.jpginflating:flowers/daisy/11023214096\_b5b39fab08.jpg

inflating: flowers/daisy/11023272144\_fce94401f2\_m.jpginflating: flowers/daisy/11023277956\_8980d53169\_m.jpginflating: flowers/daisy/11124324295\_503f3a0804.jpg

inflating: flowers/daisy/1140299375\_3aa7024466.jpginflating:flowers/daisy/11439894966\_dca877f0cd.jpg

inflating: flowers/daisy/1150395827\_6f94a5c6e4\_n.jpginflating: flowers/daisy/11642632\_1e7627a2cc.jpg

inflating: flowers/daisy/11834945233\_a53b7a92ac\_m.jpg

inflating: flowers/daisy/11870378973\_2ec1919f12.jpg

inflating: flowers/daisy/11891885265\_ccefec7284\_n.jpginflating: flowers/daisy/12193032636\_b50ae7db35\_n.jpginflating: flowers/daisy/12348343085\_d4c396e5b5\_m.jpginflating: flowers/daisy/12585131704\_0f64b17059\_m.jpginflating: flowers/daisy/12601254324\_3cb62c254a\_m.jpginflating: flowers/daisy/1265350143\_6e2b276ec9.jpg



inflating: flowers/daisy/12701063955\_4840594ea6\_n.jpginflating: flowers/daisy/1285423653\_18926dc2c8\_n.jpginflating: flowers/daisy/1286274236\_1d7ac84efb\_n.jpginflating: flowers/daisy/12891819633\_e4c82b51e8.jpg

inflating: flowers/daisy/1299501272\_59d9da5510\_n.jpginflating: flowers/daisy/1306119996\_ab8ae14d72\_n.jpginflating: flowers/daisy/1314069875\_da8dc023c6\_m.jpginflating: flowers/daisy/1342002397\_9503c97b49.jpg

inflating: flowers/daisy/134409839\_71069a95d1\_m.jpginflating:flowers/daisy/1344985627\_c3115e2d71\_n.jpg

inflating: flowers/daisy/13491959645\_2cd9df44d6\_n.jpginflating: flowers/daisy/1354396826\_2868631432\_m.jpginflating: flowers/daisy/1355787476\_32e9f2a30b.jpg

inflating: flowers/daisy/13583238844\_573df2de8e\_m.jpginflating: flowers/daisy/1374193928 a52320eafa.jpg

# ImportingNecessaryLibraries

importwarnings

warnings.filterwarnings("ignore")

importnumpyasnp

import matplotlib.pyplot as pltimportpandas as pd

fromtensorflow.keras.modelsimportSequential

from tensorflow.keras.layers import Dense,Activation,Dropout,Conv2D,Flatten,MaxPool2D,Reshfromtensorflow.keras.applications.resnet50 importResNet50

from tensorflow.keras.applications.resnet50 import preprocess\_inputfromtensorflow.keras.preprocessing importimage

from tensorflow.keras.preprocessing.image import ImageDataGenerator,load\_img,img\_to\_arrayfromtensorflow.keras.callbacksimportEarlyStopping, ReduceLROnPlateau

# ImageAugumentation

Datasetconsistof5classes.

**Daisy**-EuropeanSpeciesofAsterfamily.

**Sunflower**-IdentifiedasthegenusofHelianthus.

**Tulip**-Itbelongstothespeciesofspringbloominggeophytes.

**Rose**-Itbelongstothefamilyofrosaceae.

**Dandelion**-IndentifiesasthegenusofAsterceae.

path ='flowers/'

train\_data\_gen=ImageDataGenerator(rescale =1./255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip = True,validation\_split=0.30)

test\_data\_gen=ImageDataGenerator(rescale =1./255,validation\_split=0.30)

training\_set= train\_data\_gen.flow\_from\_directory(path,

target\_size=(64,64),batch\_size=100,

class\_mode='categorical',shuffle=True,

color\_mode='rgb',

subset='training')

testing\_set= test\_data\_gen.flow\_from\_directory(path,

target\_size=(64,64),batch\_size=100,

class\_mode='categorical',shuffle=True,

color\_mode='rgb',

subset='validation')

Found 3024 images belonging to 5 classes.Found1293imagesbelongingto5classes.

# Createthemodel

model = Sequential()

# AddLayers(Convolution,MaxPooling,Flatten,Dense-HiddenLayers,Output)

#convolution and Pooling layer 1

model.add(Conv2D(filters=48,kernel\_size=3,activation='relu',input\_shape=(64,64,3)))model.add(MaxPool2D(pool\_size=2,strides=2))

model.add(Dropout(0.2))

#convolution and Pooling layer 2

model.add(Conv2D(filters=32,kernel\_size=3,activation='relu'))model.add(MaxPool2D(pool\_size=2,strides=2))

model.add(Dropout(0.2))

#Flattening the imagesmodel.add(Flatten())

#Fully Connected layers

model.add(Dense(64,activation='relu'))model.add(Dropout(0.2))

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

Model: "sequential"

Layer (type) Output Shape Param #

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| conv2d (Conv2D) | (None, 62, 62, 48) | | | 1344 |
| max\_pooling2d (MaxPooling2D | (None, 31, 31, 48) | | | 0 |
| ) |  | | |  |
| dropout (Dropout) | (None, 31, 31, 48) | | | 0 |
| conv2d\_1 (Conv2D) | (None, 29, 29, 32) | | | 13856 |
| max\_pooling2d\_1 (MaxPooling2D) | (None, 14, 14, 32) | | | 0 |
| dropout\_1 (Dropout) | (None, | 14, 14, | 32) | 0 |
| flatten (Flatten) | (None, | 6272) |  | 0 |
| dense (Dense) | (None, | 64) |  | 401472 |
| dropout\_2 (Dropout) | (None, | 64) |  | 0 |
| dense\_1 (Dense) | (None, | 5) |  | 325 |

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

Total params: 416,997

Trainable params: 416,997

Non-trainable params: 0

# CompilingtheModel

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

# FittingtheModel

early\_stop=EarlyStopping(monitor='val\_accuracy',

patience=5,verbose=1,mode='auto')

lr=ReduceLROnPlateau(monitor='val\_accuracy',

factor=0.2,patience=5,min\_lr=0.00001)

callback = [early\_stop,lr]

TrainingtheModel

result=model.fit(x=training\_set, validation\_data=testing\_set,epochs=10)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Epoch  31/31 | 1/10  [==============================] | - 30s | | 966ms/step | - loss: | 0.7625 | - accuracy: | 0 |
| Epoch | 2/10 |  | |  |  |  |  |  |
| 31/31  Epoch31/31 | [==============================]3/10  [==============================] | - 30s  - 31s | | 969ms/step  985ms/step | * loss: * loss: | 0.7454  0.7348 | * accuracy: * accuracy: | 0  0 |
| Epoch  31/31 | 4/10  [==============================] | - 30s | | 968ms/step | - loss: | 0.7144 | - accuracy: | 0 |
| Epoch | 5/10 |  | |  |  |  |  |  |
| 31/31  Epoch | [==============================]  6/10 | - 31s | | 992ms/step | - loss: | 0.7233 | - accuracy: | 0 |
| 31/31  Epoch | [==============================]7/10 | - | 32s | 1s/step - loss: 0.7017 - accuracy: 0.73 | | | | |
| 31/31 | [==============================] | - 30s | | 963ms/step | - loss: | 0.6715 | - accuracy: | 0 |
| Epoch  31/31 | 8/10  [==============================] | - 31s | | 978ms/step | - loss: | 0.6512 | - accuracy: | 0 |
| Epoch | 9/10 |  | |  |  |  |  |  |
| 31/31  Epoch31/31 | [==============================]10/10  [==============================] | - 31s  - 30s | | 982ms/step  974ms/step | * loss: * loss: | 0.6711  0.6481 | * accuracy: * accuracy: | 0  0 |

LossandAccuracycheckusingplot

#plot the loss

plt.plot(result.history['loss'],label='trainloss')

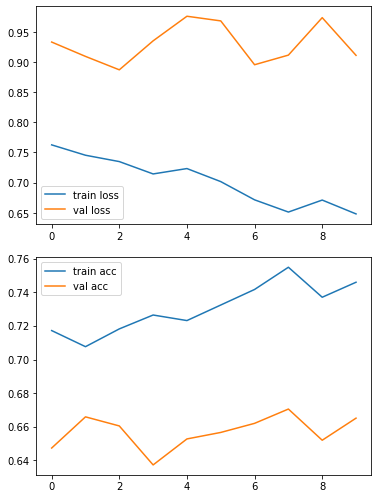
plt.plot(result.history['val\_loss'], label='val loss')plt.legend()

plt.show()

# plot the accuracy

plt.plot(result.history['accuracy'], label='train acc')plt.plot(result.history['val\_accuracy'], label='val acc')plt.legend()

plt.show()



7.SavingtheModel

model.save('daisy.h5')

8.TestingtheModel

training\_set.class\_indices

classes = ['Daisy','Dandelion','Rose','Sunflower','Tulip']deftesting(img):

img = image.load\_img(img,target\_size=(64,64))x = image.img\_to\_array(img)

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

pred = np.argmax(model.predict(x))

returnprint("Predictedclassas:",classes[pred])

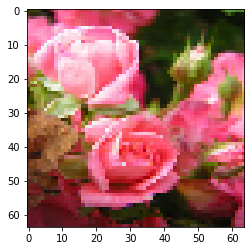
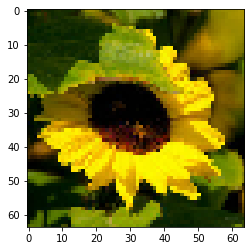
defimg\_show(img):

img1 = image.load\_img(img,target\_size=(64,64))plt.imshow(img1)

#test1

img\_show('/content/flowers/sunflower/12471443383\_b71e7a7480\_m.jpg')testing('/content/flowers/sunflower/12471443383\_b71e7a7480\_m.jpg')

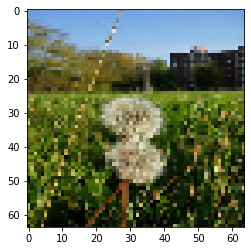
Predicted class as: Sunflower



#test3

img\_show('/content/flowers/dandelion/2116997627\_30fed84e53\_m.jpg')testing('/content/flowers/dandelion/2116997627\_30fed84e53\_m.jpg')

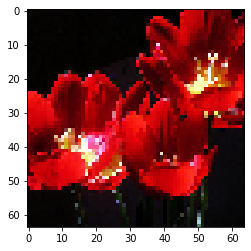
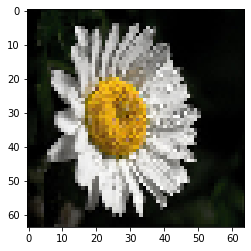
Predicted class as: Daisy



#test4

img\_show('/content/flowers/daisy/1314069875\_da8dc023c6\_m.jpg')testing('/content/flowers/daisy/1314069875\_da8dc023c6\_m.jpg')

Predicted class as: Daisy



**Conclusion:**

**Thedatasethasabout4317imagesfrom5differentclasses.**

* 1. Eachclasseshavemorethan500imagesfortrainingthedata.
  2. 30%ofthedatatakenforvalidation.
  3. Theaccuracyofthemodelisaround80%.
  4. Thevalidationaccuracyisaround70%.
  5. Themodelisbuiltwith2layeredconvolutionalnetworkconsidering1344trainableparameters.
  6. Testingthemodelwithunknownimagesgives95%accuracy.