**Assignment -3**

**Build CNN model for classification of Flowers**

| Assignment Date | 03 October 2022 |
| --- | --- |
| Team ID | PNT2022TMID27826 |
| Project Name | AI-powered Nutrition Analyzer for Fitness Enthusiasts |
| Student Name | SANJANA.S |
| Student Roll Number | 311519104050 |
| Maximum Marks | 2 Marks |

**Question-1.** Load the dataset

**Solution:**

**!unzip Flowers-Dataset.zip**

****

****

o #importing required libraries to build a CNN classification model with accuracy import  numpy as np

import tensonflow as tf

from tensorflow.keras impont layers

from tensorflow.keras.models import Sequential import

matplotlib.oyplot as plt

batch s1ze = 32 im height = 186 im

width = 180

data dir - ”/content/flowens“

**Question-2.** Image Augmentation

**Solution:**

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

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

**x\_train = train\_datagen.flow\_from\_directory(r"/content/flowers", target\_size = (64,64) , class\_mode = "categorical", batch\_size = 100)**

****

****

**Question-3.** Create model - Model Building and also Split dataset into training and testing sets **Solution:**

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

**from tensorflow.keras.layers import Convolution2D,MaxPooling2D,Flatten,Dense model = Sequential()**

**train\_ds = tf.keras.utils.image\_dataset\_from\_directory(data\_dir,**

**validation\_split=0.2,**

**subset="training", seed=123,**

**image\_size=(img\_height, img\_width), batch\_size=batch\_size)**

****

**val\_ds = tf.keras.utils.image\_dataset\_from\_directory( data\_dir,**

**validation\_split=0.2,**

**subset="validation", seed=123,**

**image\_size=(img\_height, img\_width),**

**batch\_size=batch\_size)**

****

**class\_names = train\_ds.class\_names**

**print(class\_names)**

****

**plt.figure(figsize=(10, 10))**

**for images, labels in train\_ds.take(1): for i in range(9):**

**ax = plt.subplot(3, 3, i + 1)**

**plt.imshow(images[i].numpy().astype("uint8"))**

**plt.title(class\_names[labels[i]])**

**plt.axis("off")**

**Question-4.** Add the layers (Convolution,MaxPooling,Flatten,Dense-(HiddenLayers),Output) **Solution:**

**model.add(Convolution2D(32, (3,3), activation = "relu", input\_shape = (64,64,3) )) model.add(MaxPooling2D(pool\_size = (2,2)))**

**model.add(Flatten()) model.add(Dense(300, activation =**

**"relu"))**

**model.add(Dense(150, activation = "relu")) #mulitple dense layers**

**model.add(Dense(5, activation = "softmax")) #output layer**

****

**Question-5.** Compile The Model

**Solution:**

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

****

***#Compile the model for further accuracy***

**model.compile(optimizer='adam',**

**loss=tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),**

**metrics=['accuracy'])**

**epochs=10**

**history = model.fit( train\_ds,**

**validation\_data=val\_ds, epochs=epochs**

**)**

***#To find the Training and Validation- Accuracy & Loss (Visualization)***

**acc = history.history['accuracy']**

**val\_acc = history.history['val\_accuracy']**

**loss = history.history['loss'] val\_loss =**

**history.history['val\_loss']**

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

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

**plt.subplot(1, 2, 1)**

**plt.plot(epochs\_range, acc, label='Training Accuracy')**

**plt.plot(epochs\_range, val\_acc, label='Validation Accuracy')**

**plt.legend(loc='lower right')**

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

**plt.subplot(1, 2, 2)**

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

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

**plt.legend(loc='upper right')**

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

**Question-6.** Fit The Model

**Solution:**

**model.fit(x\_train, epochs = 15, steps\_per\_epoch = len(x\_train))**

****

**Question-8.** Test The Model

**Solution:**

**from tensorflow.keras.models import load\_model from tensorflow.keras.preprocessing import image import numpy as np model = load\_model("/content/flowers.h1")**

***# Testing with a random rose image from Google***

**img = image.load\_img("/content/rose.gif", target\_size = (64,64) )**

**img**

****

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

****

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

****

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

**labels = ['daisy','dandelion','roses','sunflowers','tulips'] labels[np.argmax(pred)]**

****

**Question-7.** Save The Model

**Solution:**

**model.save("flowers.h1")**

**model.save("flowers.m5")#another model to show the accuracy**

***#Testing the alternative model with accuracy***

**sunflower\_url = "https://storage.googleapis.com/download.tensorflow.org/example\_images/592 px Red\_sunflower.jpg"**

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

**sunflower\_path, target\_size=(img\_height, img\_width)**

**)**

**img\_array = tf.keras.utils.img\_to\_array(img)**

**img\_array = tf.expand\_dims(img\_array, 0) *# Create a batch***

**predictions = model.predict(img\_array) score =**

**tf.nn.softmax(predictions[0]) print(**

**"This image most likely belongs to {} with a {:.2f} percent confidence."**

**.format(class\_names[np.argmax(score)], 100 \* np.max(score))**

**)**

****