**Sprint - 3**

|  |  |
| --- | --- |
| **PROJECT TILE** | Nutrition Assistant Application |
| **TEAM ID** | PNT2022TMID42630 |

**Code:**

import tensorflow as tf

from tensorflow.keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense

from tensorflow.keras.models import Model, load\_model

from tensorflow.keras.callbacks import TensorBoard, ModelCheckpoint

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

import cv2

import imutils

import numpy as np

import matplotlib.pyplot as plt

import time

from os import listdir

def load\_data(dir\_list, image\_size):

# load all images in a directory

X = []

y = []

image\_width, image\_height = image\_size

for directory in dir\_list:

for filename in listdir(directory):

# load the image

image = cv2.imread(directory + '\\' + filename)

# resize image

image = cv2.resize(image, dsize=(image\_width, image\_height), interpolation=cv2.INTER\_CUBIC)

# normalize values

image = image / 255.

# convert image to numpy array and append it to X

X.append(image)

# append a value of 1 to the target array if the image

# is in the folder na

#print(directory[42:45])

if directory[42:45] == 'Egg':

y.append([0])

elif directory[42:45] == 'fis':

y.append([1])

elif directory[42:45] == 'Fri':

y.append([2])

elif directory[42:45] == 'mea':

y.append([3])

elif directory[42:45] == 'Noo':

y.append([4])

elif directory[42:45] == 'piz':

y.append([5])

elif directory[42:45] == 'Ric':

y.append([6])

X = np.array(X)

y = np.array(y)

# Shuffle the data

#X, y = shuffle(X, y)

return X, y

augmented\_path = 'C:/Users/sethu/Desktop/code/flaskapp/data/'

augmented\_Egg = augmented\_path + 'Egg'

augmented\_fish = augmented\_path + 'fish'

augmented\_Fries = augmented\_path + 'Fries'

augmented\_meat = augmented\_path + 'meat'

augmented\_Noodles = augmented\_path + 'Noodles'

augmented\_pizza = augmented\_path + 'pizza'

augmented\_Rice = augmented\_path + 'Rice'

IMG\_WIDTH, IMG\_HEIGHT = (256, 256)

X, y = load\_data([augmented\_Egg, augmented\_fish,augmented\_Fries,augmented\_meat,augmented\_Noodles,augmented\_pizza,augmented\_Rice], (IMG\_WIDTH, IMG\_HEIGHT))

def split\_data(X, y, test\_size=0.2):

X\_train, X\_test\_val, y\_train, y\_test\_val = train\_test\_split(X, y, test\_size=test\_size)

X\_test, X\_val, y\_test, y\_val = train\_test\_split(X\_test\_val, y\_test\_val, test\_size=0.5)

return X\_train, y\_train, X\_val, y\_val, X\_test, y\_test

X\_train, y\_train, X\_val, y\_val, X\_test, y\_test = split\_data(X, y, test\_size=0.1)

def hms\_string(sec\_elapsed):

h = int(sec\_elapsed / (60 \* 60))

m = int((sec\_elapsed % (60 \* 60)) / 60)

s = sec\_elapsed % 60

return f"{h}:{m}:{round(s,1)}"

def compute\_f1\_score(y\_true, prob):

# convert the vector of probabilities to a target vector

y\_pred = np.where(prob > 0.5, 1, 0)

score = f1\_score(y\_true, y\_pred)

return score

from tensorflow import keras

model = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(16, (3,3), activation='relu', input\_shape=(256, 256, 3)),

tf.keras.layers.MaxPooling2D(2, 2),

keras.layers.Dropout(rate=0.15), #adding dropout regularization throughout the model to deal with overfitting

# The second convolution

tf.keras.layers.Conv2D(32, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

keras.layers.Dropout(rate=0.1),

# The third convolution

tf.keras.layers.Conv2D(64, (3,3), activation='relu'),

tf.keras.layers.MaxPooling2D(2,2),

keras.layers.Dropout(rate=0.10),

# Flatten the results to feed into a DNN

tf.keras.layers.Flatten(),

# 512 neuron hidden layer

tf.keras.layers.Dense(512, activation='relu'),

# 3 output neuron for the 3 classes of Animal Images

tf.keras.layers.Dense(7, activation='softmax')

])

IMG\_SHAPE = (IMG\_WIDTH, IMG\_HEIGHT, 3)

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

#model.fit(x=X, y=y, batch\_size=64, epochs=20, validation\_data=(X\_val, y\_val)) need to show training epochs uncomment this line

from keras.models import load\_model

model = load\_model('modelnutrition.h5')

import numpy as np

from keras.preprocessing import image

from tkinter import filedialog

filepath=filedialog.askopenfilename()

test\_image = cv2.imread(filepath,3)

test\_image = np.expand\_dims(test\_image, axis=0)

result = model.predict(test\_image)

result = np.round(result)

print(result)

if result[0][0] == 1:

food ="Egg"

print("Egg")

fn='egg.csv'

elif result[0][1] == 1:

food ="Fish"

print("Fish")

fn='fish.csv'

elif result[0][2] == 1:

food ="Fries"

print("Fries")

fn='fries.csv'

elif result[0][3] == 1:

food ="Meat"

print("Meat")

fn='meat.csv'

elif result[0][4] == 1:

food ="Noodles"

print("Noodles")

fn='noodle.csv'

elif result[0][5] == 1:

food ="Pizza"

print("Pizza")

fn='pizza.csv'

elif result[0][6] == 1:

food ="Rice"

print("Rice")

fn='rice.csv'

import pandas as pd

# x axis values

x1 = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]

# corresponding y axis values

y2 = [0.125,0.25,0.2,0.35,0.325,0.625,0.75,0.825,0.8,0.95,0.975,0.975,0.95,0.96, 0.975,0.98]

y1 = [0.155,0.1575,0.25,0.32,0.4275,0.5412,0.6,0.7212,0.793,0.8662,0.90,0.95,0.9688,0.99,0.987,0.995]

plt.plot(x1, y1, label = "Training accuracy")

plt.plot(x1, y2, label = "Validation accuracy")

plt.xlabel('Number of epochs')

plt.ylabel('Accuracy(%)')

plt.title('Accuracy')

plt.legend()

plt.show()

y1 = [2.313,1.912,1.838,1.686,1.151,1.217,1.10,0.84,0.627,0.4515,0.312,0.190,0.1395,0.07,0.08,0.06]

y2=[1.947,1.88,1.769,1.658,1.630,1.143,1.003,0.721,0.6114,0.401,0.205,0.093,0.2,0.08,0.05,0.01]

plt.plot(x1, y1, label = "Training loss")

plt.plot(x1, y2, label = "Validation loss")

plt.xlabel('Number of epochs')

plt.ylabel('Loss')

plt.title('Loss')

plt.legend()

plt.show()

from keras.models import load\_model

from flask import Flask, render\_template

from PIL import Image

import base64

import io

from io import BytesIO

from tkinter import filedialog

im = Image.open(filepath)

data = BytesIO()

im.save(data, format=im.format)

encoded\_img\_data = base64.b64encode(data.getvalue())

from flask import Flask, render\_template

from PIL import Image

import base64

import io

app = Flask(\_\_name\_\_)

df = pd.read\_csv(fn)

df.to\_csv(fn, index=None)

@app.route('/')

def hello\_world():

# Full Script.

im = Image.open(filepath)

data = io.BytesIO()

im.save(data, format=im.format)

encoded\_img\_data = base64.b64encode(data.getvalue())

data = pd.read\_csv(fn)

foodr =food

return render\_template("index.html", img\_data=encoded\_img\_data.decode('utf-8'),tables=[data.to\_html()], titles=[''],dataToRender=foodr)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0')