Grocery Recommendation System For Retail Markets.

**CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No** | **TOPIC** | **PAGE** **No** |
| **1** | **ABSTRACT** | **2** |
| **2** | **INTRODUCTION** | **3** |
| **3** | **LITERATURE SURVEY** | **3-4** |
| **4** | **THEORITICAL ANALYSIS** | **4-6** |
| **5** | **EXPERIMENTAL INVESTIGATIONS** | **6** |
| **6** | **FLOWCHART** | **7** |
| **7** | **RESULT** | **8-9** |
| **8** | **ADVANTAGES AND DISADVANTAGES** | **10** |
| **9** | **APPLICATIONS** | **10** |
| **10** | **CONCLUSION** | **11** |
| **11** | **FUTURE SCOPE** | **12** |
| **12** | **BIBILOGRAPHY** | **12** |

**1.ABSTRACT**

we present an approach for online grocery shopping recommendation, framed as a binary classification problem. we compare the results of our recommendation experiments with those of traditional and enhanced market basket analysis approaches. We use approximate recommendation precision to measure the success of our recommender, and Face recognition technology enables offline grocery stores to do what the online shopping recommendation systems have been doing for year’s i.e., cameras in offline grocery shops identifies their shoppers/customers, link them to past purchases and generate personalized product recommendations based on the data.

**2. INTRODUCTION**

**2.1 Overview:**

All online retail apps have proved that, when a retailer knows its customers, it can serve them more effectively. But how does that work in a physical, offline grocery retail environment? One way is to integrate a facial recognition system. Face recognition technology enables offline grocery stores to do what the online shopping recommendation systems have been doing for year’s i.e., cameras in offline grocery shops identifies their shoppers/customers, link them to past purchases and generate personalized product recommendations based on the data.

**2.2 Purpose:**

Using a combination of in-store cameras and facial recognition software, shops can now easily asses the purchase data of the customers and recommend products based on the history. This core data can be collected at each stage of the customer journey, tracking shoppers (and how they interact with a store) from entry to checkout. Not only can face recognition technology identify and classify customers, it can help retailers optimize and plan their product offerings.

**3. LITERATURE SURVEY**

**3.1 Existing problem :**

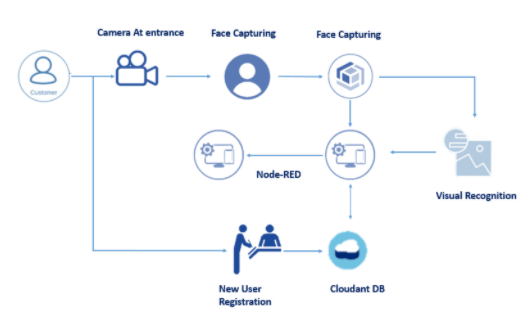
When transitioning between online and in-store experiences, customers not only want the same products to be available, they also want their experience to be seamless .This means, if they are a regular online customer, they want to be treated like a regular customer when they visit a brick-and-mortar location. If they made an online purchase earlier in the day, the in-store systems should already have a record of it. If retailers can create this type of fluid online/offline experience for their customers, they can cease pitting their channels against one another. Centralized customer data can help retailers build a seamless, fluid experience

**3.2 Proposed solution:**

Here we are going to detect the customer using a human trained model we trained our model using nearly 500 images and we got an accuracy of nearly 99%. Now when a customer comes the camera detects his/her face and gives the product recommendations and offers available in the store. Loyalty programs can help, by collecting relevant information and putting it to use. A retailer can use its loyalty program to not only reward customers, but deliver relevant content and integrate data across all interaction points – including online interactions, in-store sales, and home service technicians – creating an integrated, customer experience.

**4. THEORITICAL ANALYSIS**

**4.1 Block diagram:**



CNN TRAINING (training the images)

**4.2 Hardware / Software designing:**

**Software Requirements:**

Python 3 - We have used Python which is a statistical mathematical programming language like R instead of MATLAB due to the following reasons:

1. Python code is more compact and readable than MATLAB

2. The python data structure is superior to MATLAB

3. It is an open source and also provides more graphic packages and data sets Keras (with TensorFlow backend 2.3.0 version) - Keras is a neural network API consisting of TensorFlow, CNTk, Theano etc. Python packages like Numpy, Matplotlib, Pandas for mathematical computation and plotting graphs, SimpleITK for reading the images which were in .mha format and Mahotas for feature extraction of GLCM Kaggle was used to obtain the online dataset. GitHub and Stackoverflow was used for reference in case of programming syntax errors. OpenCV (Open Source Computer Vision) is a library of programming functions aimed at real time computer vision i.e. used for image processing and any operations relating to image like reading and writing images, modifying image quality, removing noise by using Gaussian Blur, performing binary thresholding on images, converting the original image consisting of pixel values into an array, changing the image from RGB to grayscale etc. It is free to use, simple to learn and supports C++, Java, C, Python. Its popular application lies in CamScanner or Instagram, GitHub or a web-based control repository. Google Colaboratory (open-source Jupyter Notebook interface with high GPU facility) - Google Colab /Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely on cloud. With Colab, one can write and execute code, save and share analyses, access powerful computing resources, all for free from browser.[Jupyter Notebook is a powerful way to iterate and write on your Python code for data analysis. Rather than writing and rewriting an entire code, one can write lines of code and run them at a time. It is built off of iPython which ©RCCIIT, DEPT. OF EE Page 20 is an interactive way of running Python code. It allows Jupyter notebook to support multiple languages as well as storing the code and writing own markdown.

**Hardware Requirements:**

Processor: Intel® Core™ i3-2350M CPU @ 2.30GHz

Installed memory (RAM):4.00GB

System Type: 64-bit Operating System

**5. EXPERIMENTAL INVESTIGATIONS**

**Artificial Intelligence**:

Artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems enabling it to even mimic human behaviour. Its applications lie in fields of Computer Vision, Natural Language Processing, Robotics, Speech Recognition, etc.

**Convolutional Neural Network:**

Convolutionalneural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

6**.Flow chart:**

**DATA COLLECTION.**

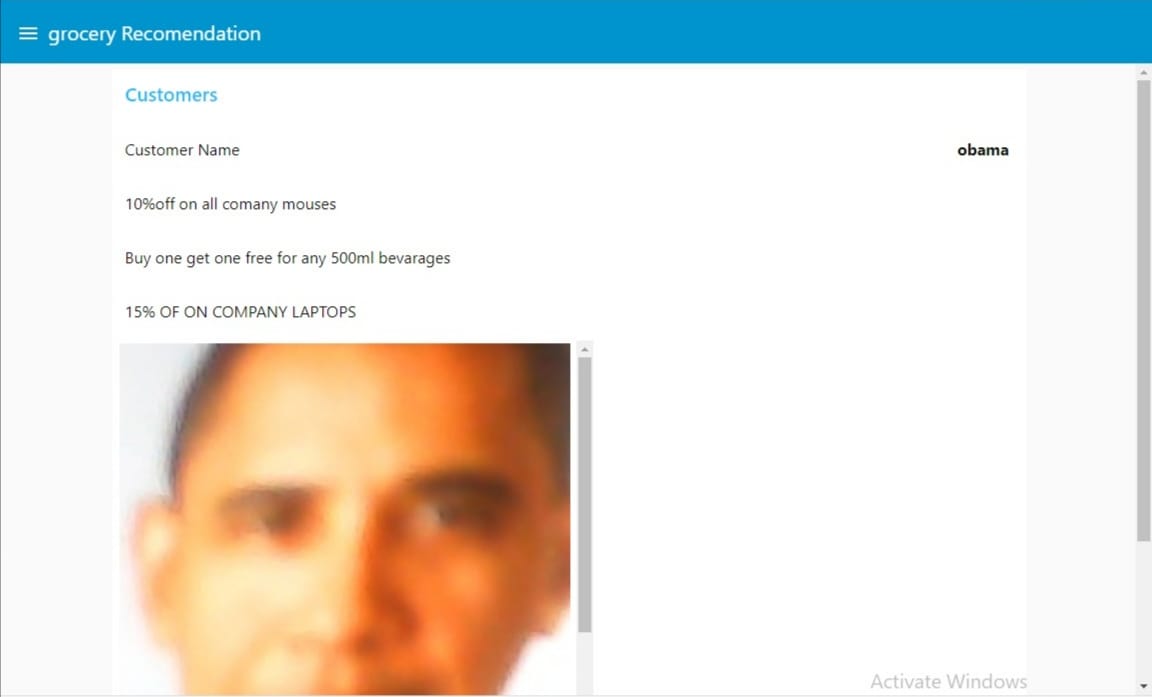
**IMAGE PREPROCESSING**

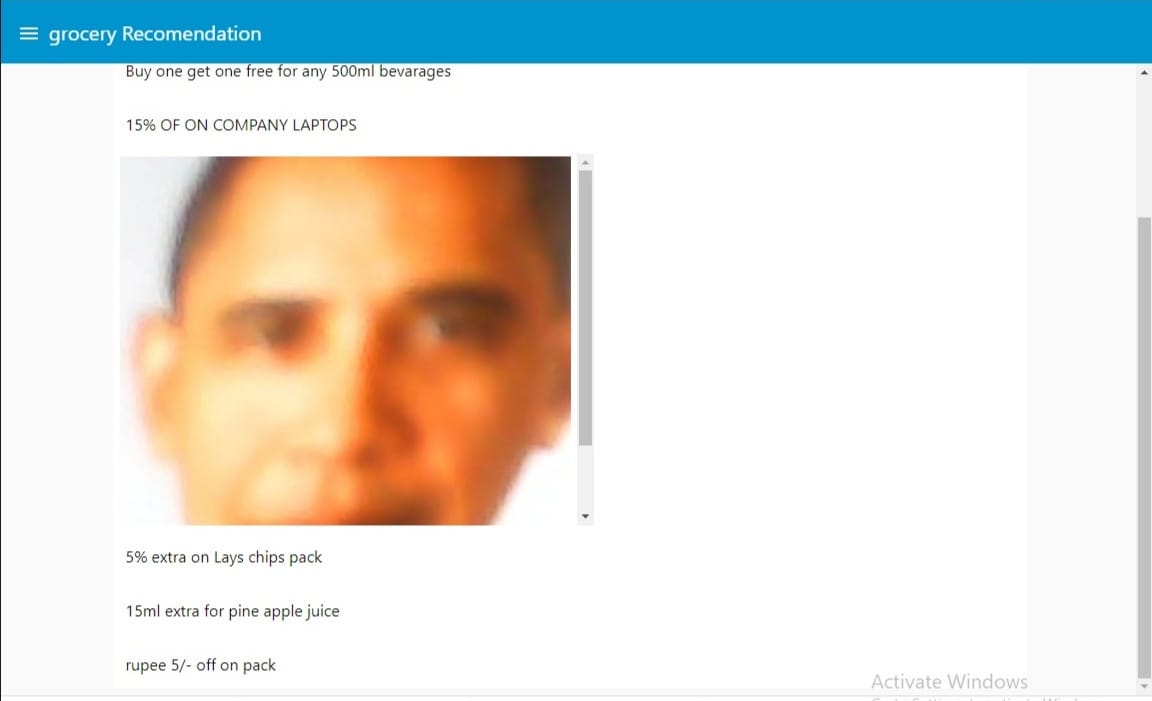
**CREATING BUCKETS AND DATABASES.**

**OPEN CV ANALYSIS**

**BUILD NODE-RED APPLICATION.**

**7.RESULT :**

****

****

**8. ADVANTAGES & DISADVANTAGES**

**Advantages:**

1. It is considered as the best ml technique for image classification due to high accuracy.

2. It is used over feed forward neural networks as it can be trained better in case of complex images to have higher accuracies.

3. It easy for the customers to shop.

4.The groceries are recommended just by capturing the image.

**Disadvantages**:

1. It requires a large training data.

2. It requires appropriate model.

3. It is time consuming.

4. It is a tedious and exhaustive procedure.

5. While convolutional networks have already existed for a long time, their success was limited due to the size of the considered network.

**9. APPLICATIONS**

The main application of this model is that it helps the customer by recommending the groceries to shop.  A technologic solution which can handle the huge amount of data being generated and ensure it is focused in a direction which best benefits – rather than overwhelms – marketing efforts.

**10. CONCLUSION**

This Project focuses on building a system where cameras at entry recognize the customer and link his previous purchase history stored in the database. Based on that purchase history he will get a recommendation of the offers on the product which he wishes to buy. And also displays on the screen at the entrance as soon as he steps into the store . Also only in the combining of streamlined un-siloed data science, seamless cross-channel customer service and marketing, and authentic personalization, can retailers expect to create buyer experiences which can combat the fickle nature of the modern consumer.

**11.FUTURE SCOPE**

Using a combination of in-store cameras and facial recognition software, shops can now easily asses the purchase data of the customers and recommend products based on the history. This core data can be collected at each stage of the customer journey, tracking shoppers (and how they interact with a store) from entry to checkout. Not only can face recognition technology identify and classify customers, it can help retailers optimize and plan their product offerings.

**12.BIBILOGRAPHY**

1. Guadagni, P.M., Little, J.D.C.. A logit model of brand choice, calibrated on scanner data. Marketing Science 1983;2:203–238.

2. Sano, N., Yada, K.. The influence of sales areas and bargain sales on customer behavior in a grocery store. Neural Computing and Applications 2015;26(2):355–361.

3. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.. Application of dimensionality reduction in recommender system –a case study. In: Proceedings of the ACM WebKDD Workshop. 2000, .

4. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.. Analysis of recommendation algorithms for ecommerce. In: Proceedings of the 2nd ACM conference on Electronic commerce. 2000, p. 158–167.

5. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.. Eigentaste : A constant time collaborative filtering algorithm. Information Retrieval Journal 2001;4:133–151.

6. Scholz, M., Fraunholz, M.,, , Selbig, J.. Nonlinear principal component analysis: Neural network models and applications. In: Principal Manifolds for Data Visualization and Dimension Reduction; vol. 58 of Lecture Notes in Computational Science and enginee. Heidelberg: Springer; 2008, p. 44–67.

7. Vozalis, M.G., Markos, A., Margaritis, K.G.. Collaborative filtering through svd-based and hierarchical nonlinear pca. In: Proceedings of 2010 International Conference on Artificial Neural Networks. 2010, p. 395–400

**Appendix**

**CNN code:**

{

"cells": [

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {},

"outputs": [],

"source": [

"from tensorflow.keras.models import Sequential\n",

"from tensorflow.keras.layers import Dense\n",

"from tensorflow.keras.layers import Convolution2D \n",

"from tensorflow.keras.layers import MaxPooling2D \n",

"from tensorflow.keras.layers import Flatten\n"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"Using TensorFlow backend.\n"

]

}

],

"source": [

"from keras.preprocessing.image import ImageDataGenerator\n",

"train\_datagen = ImageDataGenerator(rescale = 1./255,shear\_range =0.2,zoom\_range=0.2,horizontal\_flip= True)\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Found 415 images belonging to 4 classes.\n",

"Found 328 images belonging to 4 classes.\n"

]

}

],

"source": [

"x\_train = train\_datagen.flow\_from\_directory(r\"D:\\user im2\\train set\",target\_size=(64,64),batch\_size = 32,class\_mode =\"categorical\")\n",

"x\_test = test\_datagen.flow\_from\_directory(r\"D:\\user im2\\test set\",target\_size=(64,64),batch\_size = 32,class\_mode =\"categorical\")"

]

},

{

"cell\_type": "code",

"execution\_count": 4,

"metadata": {},

"outputs": [],

"source": [

"model = Sequential()"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"WARNING:tensorflow:From C:\\Users\\Vaishnavi Gandhi\\Anaconda3\\lib\\site-packages\\tensorflow\\python\\ops\\init\_ops.py:1251: calling VarianceScaling.\_\_init\_\_ (from tensorflow.python.ops.init\_ops) with dtype is deprecated and will be removed in a future version.\n",

"Instructions for updating:\n",

"Call initializer instance with the dtype argument instead of passing it to the constructor\n"

]

}

],

"source": [

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

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"metadata": {},

"outputs": [],

"source": [

"model.add(MaxPooling2D(2,2))"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"metadata": {},

"outputs": [],

"source": [

"model.add(Flatten())"

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"WARNING:tensorflow:From C:\\Users\\Vaishnavi Gandhi\\Anaconda3\\lib\\site-packages\\tensorflow\\python\\keras\\initializers.py:119: calling RandomUniform.\_\_init\_\_ (from tensorflow.python.ops.init\_ops) with dtype is deprecated and will be removed in a future version.\n",

"Instructions for updating:\n",

"Call initializer instance with the dtype argument instead of passing it to the constructor\n"

]

}

],

"source": [

"model.add(Dense(units = 512,activation = \"relu\",kernel\_initializer = \"random\_uniform\"))"

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"metadata": {},

"outputs": [],

"source": [

"model.add(Dense(units = 4,activation = \"softmax\",kernel\_initializer = \"random\_uniform\"))"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"metadata": {},

"outputs": [],

"source": [

"model.compile(\"sgd\",loss = \"categorical\_crossentropy\",metrics = [\"accuracy\"])"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Epoch 1/30\n",

"13/13 [==============================] - 7s 565ms/step - loss: 1.3677 - acc: 0.3205 - val\_loss: 1.3155 - val\_acc: 0.4146\n",

"Epoch 2/30\n",

"13/13 [==============================] - 4s 313ms/step - loss: 1.2585 - acc: 0.4458 - val\_loss: 1.1916 - val\_acc: 0.5427\n",

"Epoch 3/30\n",

"13/13 [==============================] - 4s 334ms/step - loss: 1.1865 - acc: 0.4988 - val\_loss: 1.1082 - val\_acc: 0.5823\n",

"Epoch 4/30\n",

"13/13 [==============================] - 4s 325ms/step - loss: 1.1550 - acc: 0.5229 - val\_loss: 1.0266 - val\_acc: 0.5884\n",

"Epoch 5/30\n",

"13/13 [==============================] - 4s 342ms/step - loss: 1.1106 - acc: 0.5639 - val\_loss: 1.1758 - val\_acc: 0.4543\n",

"Epoch 6/30\n",

"13/13 [==============================] - 4s 336ms/step - loss: 1.0415 - acc: 0.5855 - val\_loss: 0.9155 - val\_acc: 0.6799\n",

"Epoch 7/30\n",

"13/13 [==============================] - 4s 322ms/step - loss: 1.0126 - acc: 0.5831 - val\_loss: 1.1600 - val\_acc: 0.5610\n",

"Epoch 8/30\n",

"13/13 [==============================] - 4s 331ms/step - loss: 1.0161 - acc: 0.5711 - val\_loss: 0.8927 - val\_acc: 0.7012\n",

"Epoch 9/30\n",

"13/13 [==============================] - 5s 351ms/step - loss: 0.9817 - acc: 0.6120 - val\_loss: 0.8490 - val\_acc: 0.7287\n",

"Epoch 10/30\n",

"13/13 [==============================] - 4s 322ms/step - loss: 0.9513 - acc: 0.6361 - val\_loss: 0.9093 - val\_acc: 0.6128\n",

"Epoch 11/30\n",

"13/13 [==============================] - 4s 317ms/step - loss: 0.8633 - acc: 0.6771 - val\_loss: 0.8082 - val\_acc: 0.7195\n",

"Epoch 12/30\n",

"13/13 [==============================] - 4s 337ms/step - loss: 0.8971 - acc: 0.6337 - val\_loss: 0.7541 - val\_acc: 0.7317\n",

"Epoch 13/30\n",

"13/13 [==============================] - 5s 374ms/step - loss: 0.9010 - acc: 0.6410 - val\_loss: 0.7512 - val\_acc: 0.7591\n",

"Epoch 14/30\n",

"13/13 [==============================] - 4s 346ms/step - loss: 0.8116 - acc: 0.6988 - val\_loss: 0.7154 - val\_acc: 0.7439\n",

"Epoch 15/30\n",

"13/13 [==============================] - 5s 347ms/step - loss: 0.8090 - acc: 0.7012 - val\_loss: 0.7452 - val\_acc: 0.7226\n",

"Epoch 16/30\n",

"13/13 [==============================] - 4s 340ms/step - loss: 0.8222 - acc: 0.6916 - val\_loss: 0.6877 - val\_acc: 0.7774\n",

"Epoch 17/30\n",

"13/13 [==============================] - 4s 326ms/step - loss: 0.7934 - acc: 0.7036 - val\_loss: 0.6850 - val\_acc: 0.7805\n",

"Epoch 18/30\n",

"13/13 [==============================] - 4s 335ms/step - loss: 0.8095 - acc: 0.6867 - val\_loss: 0.6480 - val\_acc: 0.7866\n",

"Epoch 19/30\n",

"13/13 [==============================] - 5s 359ms/step - loss: 0.7737 - acc: 0.7012 - val\_loss: 0.6560 - val\_acc: 0.7896\n",

"Epoch 20/30\n",

"13/13 [==============================] - 4s 327ms/step - loss: 0.7836 - acc: 0.7253 - val\_loss: 0.6652 - val\_acc: 0.7652\n",

"Epoch 21/30\n",

"13/13 [==============================] - 4s 329ms/step - loss: 0.7731 - acc: 0.7133 - val\_loss: 0.5871 - val\_acc: 0.8201\n",

"Epoch 22/30\n",

"13/13 [==============================] - 4s 345ms/step - loss: 0.7410 - acc: 0.7253 - val\_loss: 0.5578 - val\_acc: 0.8323\n",

"Epoch 23/30\n",

"13/13 [==============================] - 4s 329ms/step - loss: 0.7469 - acc: 0.7229 - val\_loss: 0.5792 - val\_acc: 0.8323\n",

"Epoch 24/30\n",

"13/13 [==============================] - 4s 337ms/step - loss: 0.7361 - acc: 0.7277 - val\_loss: 0.5534 - val\_acc: 0.8323\n",

"Epoch 25/30\n",

"13/13 [==============================] - 4s 314ms/step - loss: 0.7438 - acc: 0.7277 - val\_loss: 0.5854 - val\_acc: 0.8110\n",

"Epoch 26/30\n",

"13/13 [==============================] - 4s 329ms/step - loss: 0.6951 - acc: 0.7590 - val\_loss: 0.5371 - val\_acc: 0.8171\n",

"Epoch 27/30\n",

"13/13 [==============================] - 4s 323ms/step - loss: 0.6789 - acc: 0.7277 - val\_loss: 0.7625 - val\_acc: 0.7165\n",

"Epoch 28/30\n",

"13/13 [==============================] - 4s 317ms/step - loss: 0.7172 - acc: 0.7253 - val\_loss: 0.5243 - val\_acc: 0.8232\n",

"Epoch 29/30\n",

"13/13 [==============================] - 4s 334ms/step - loss: 0.6690 - acc: 0.7446 - val\_loss: 0.4958 - val\_acc: 0.8537\n",

"Epoch 30/30\n",

"13/13 [==============================] - 4s 328ms/step - loss: 0.6977 - acc: 0.7181 - val\_loss: 0.5477 - val\_acc: 0.7866\n"

]

},

{

"data": {

"text/plain": [

"<tensorflow.python.keras.callbacks.History at 0x2292ac00f28>"

]

},

"execution\_count": 11,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"model.fit\_generator(x\_train ,steps\_per\_epoch = 13 ,epochs = 30,validation\_data= x\_test , validation\_steps = 11)"

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"metadata": {},

"outputs": [],

"source": [

"model.save(\"human face.h5\")"

]

},

{

"cell\_type": "code",

"execution\_count": 13,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"'D:\\\\externship'"

]

},

"execution\_count": 13,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"pwd"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.3"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}

**prediction code**

import datetime

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

import cv2

import random

import time

from cloudant.client import Cloudant

from cloudant.error import CloudantException

from cloudant.result import Result

from tensorflow.keras.models import load\_model

from keras.preprocessing import image

import numpy as np

face\_classifier=cv2.CascadeClassifier(r"C:\Users\Vaishnavi Gandhi\Desktop\New folder (2)\haarcascade\_frontalface\_default.xml")

COS\_ENDPOINT ="https://s3.jp-tok.cloud-object-storage.appdomain.cloud"

COS\_API\_KEY\_ID ="1GCuEbAa9nZclcKRvdS5FZNmn3p-BDmQ0BZr8ViGOkMn"

COS\_AUTH\_ENDPOINT ="https://iam.cloud.ibm.com/identity/token"

COS\_RESOURCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/b918a801e44a48e2b9dadc71f598c449:156f0326-184e-4e90-ba1a-996ad372b815::"

client = Cloudant("apikey-v2-2ww0fngmdbvvt0v6x3nlsat7xnpilxstg07wp9t6dwn1","855d1757a9673bbcfd8fe6edc6255a62",url= "https://apikey-v2-2ww0fngmdbvvt0v6x3nlsat7xnpilxstg07wp9t6dwn1:855d1757a9673bbcfd8fe6edc6255a62@d05cd9a9-22b1-46c9-b014-9833526ca651-bluemix.cloudantnosqldb.appdomain.cloud")

client.connect()

database\_name = "customer\_details"

picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")

picname=picname+".jpg"

pic=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")

model = load\_model(r"D:\externship\human face.h5")

cos = ibm\_boto3.resource("s3",

ibm\_api\_key\_id=COS\_API\_KEY\_ID,

ibm\_service\_instance\_id=COS\_RESOURCE\_CRN,

ibm\_auth\_endpoint=COS\_AUTH\_ENDPOINT,

config=Config(signature\_version="oauth"),

endpoint\_url=COS\_ENDPOINT

)

def multi\_part\_upload(bucket\_name, item\_name, file\_path):

try:

print("Starting file transfer for {0} to bucket: {1}\n".

format(item\_name, bucket\_name))

# set 5 MB chunks

part\_size = 1024 \* 1024 \* 5

# set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 15

# set the transfer threshold and chunk size

transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

multipart\_threshold=file\_threshold,

multipart\_chunksize=part\_size)

# the upload\_fileobj method will automatically execute a multi-part upload

# in 5 MB chunks for all files over 15 MB

with open(file\_path, "rb") as file\_data:

cos.Object(bucket\_name, item\_name).upload\_fileobj(

Fileobj=file\_data,

Config=transfer\_config

)

print("Transfer for {0} Complete!\n".format(item\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to complete multi-part upload: {0}".format(e))

video=cv2.VideoCapture(0)

while True:

#capture the first frame

print("hie")

check,frame=video.read()

print("check",check)

gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces=face\_classifier.detectMultiScale(gray,1.3,5)

cv2.imshow('Face detection', frame)

""""if cv2.waitKey(1) & 0xFF == ord('q'):

break"""

#drawing rectangle boundries for the detected face

for(x,y,w,h) in faces:

print("yes its face")

roi\_color = frame[y:y+h, x:x+w]

cv2.imwrite(picname, roi\_color)

cv2.rectangle(frame, (x,y), (x+w,y+h), (127,0,255), 2)

cv2.imshow('Face detection', frame)

person=1

#cloudant db database

my\_database = client.create\_database(database\_name)

#cloud object storage

multi\_part\_upload("vgimages",picname,pic+".jpg")

img =image.load\_img(picname,grayscale=False,

target\_size= (64,64))#loading of the image

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

print(x.shape)

x = np.expand\_dims(x,axis = 0)#changing the shape

pred = model.predict\_classes(x)

if(pred[0]==0):

pred = "allu arjun"

elif(pred[0]==1):

pred="modi"

elif(pred[0]==2):

pred="obama"

else:

pred = "selena"

if my\_database.exists():

print("'{database\_name}' successfully created.")

json\_document = {

"\_id": pic,

"link":COS\_ENDPOINT+"/vgimages/"+picname,

"prediction":pred

}

new\_document = my\_database.create\_document(json\_document)

if new\_document.exists():

print("Document '{new\_document}' successfully created.")

person=0

#waitKey(1)- for every 1 millisecond new frame will be captured

if cv2.waitKey(1) & 0xFF == ord('q'):

break

#release the camera

video.release()

#destroy all windows

cv2.destroyAllWindows()

client.disconnect()

END