SOURCE CODE

**Cnn.py**

#Importing keras libraries and packages

pip install tensorflow

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.layers import MaxPooling2D

#step1 Initializing CNN

classifier = Sequential()

# step2 adding 1st Convolution layer and Pooling layer

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

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# step3 adding 2nd convolution layer and polling layer

classifier.add(Convolution2D(32,(3,3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#step4 Flattening the layers

classifier.add(Flatten())

#step5 Full\_Connection

classifier.add(Dense(units=32,activation = 'relu'))

classifier.add(Dense(units=64,activation = 'relu'))

classifier.add(Dense(units=128,activation = 'relu'))

classifier.add(Dense(units=256,activation = 'relu'))

classifier.add(Dense(units=256,activation = 'relu'))

classifier.add(Dense(units=6,activation = 'softmax'))

#step6 Compiling CNN

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

#step7 Fitting CNN to images

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255, # To rescaling the image in range of [0,1]

                                   shear\_range = 0.2, # To randomly shear the images

                                   zoom\_range = 0.2, # To randomly zoom the images

                                   horizontal\_flip = True) #  for randomly flipping half of the images horizontally

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

print("\nTraining the data...\n")

training\_set = train\_datagen.flow\_from\_directory('train',

                                                target\_size=(64,64),

                                                batch\_size=12, #Total no. of batches

                                                class\_mode='categorical')

test\_set = test\_datagen.flow\_from\_directory('test',

                                            target\_size=(64,64),

                                            batch\_size=12,

                                            class\_mode='categorical')

classifier.fit\_generator(training\_set,

                         samples\_per\_epoch=1212, # Total training images

                         nb\_epoch = 20, # Total no. of epochs

                         validation\_data = test\_set,

                         nb\_val\_samples = 300) # Total testing images

#step8 saving model

classifier.save("model.h5")

**App.py**

import os

from uuid import uuid4

from flask import Flask, request, render\_template, send\_from\_directory

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

# app = Flask(\_\_name\_\_, static\_folder="images")

APP\_ROOT = os.path.dirname(os.path.abspath(\_\_file\_\_))

classes = ['Fresh Apple','Fresh Banana','Fresh Orange','Rotten Apple','Rotten Banana','Rotten Orange']

@app.route("/")

def index():

    return render\_template("index.html")

@app.route("/upload", methods=["POST"])

def upload():

    target = os.path.join(APP\_ROOT, 'images/')

    # target = os.path.join(APP\_ROOT, 'static/')

    print(target)

    if not os.path.isdir(target):

            os.mkdir(target)

    else:

        print("Couldn't create upload directory: {}".format(target))

    print(request.files.getlist("file"))

    for upload in request.files.getlist("file"):

        print(upload)

        print("{} is the file name".format(upload.filename))

        filename = upload.filename

        destination = "/".join([target, filename])

        print ("Accept incoming file:", filename)

        print ("Save it to:", destination)

        upload.save(destination)

        #import tensorflow as tf

        import numpy as np

        from keras.preprocessing import image

        from keras.models import load\_model

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

        new\_model.summary()

        test\_image = image.load\_img('images\\'+filename,target\_size=(64,64))

        test\_image = image.img\_to\_array(test\_image)

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

        result = new\_model.predict(test\_image)

        result1 = result[0]

        for i in range(6):

            if result1[i] == 1.:

                break;

        prediction = classes[i]

    # return send\_from\_directory("images", filename, as\_attachment=True)

    return render\_template("template.html",image\_name=filename, text=prediction)

@app.route('/upload/<filename>')

def send\_image(filename):

    return send\_from\_directory("images", filename)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=False)

**CNN.ipynb**

#Importing keras libraries and packages

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.layers import MaxPooling2D

#step1 Initializing CNN

classifier = Sequential()

# step2 adding 1st Convolution layer and Pooling layer

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

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

# step3 adding 2nd convolution layer and polling layer

classifier.add(Convolution2D(32,(3,3), activation = 'relu'))

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#step4 Flattening the layers

classifier.add(Flatten())

#step5 Full\_Connection

classifier.add(Dense(units=32,activation = 'relu'))

classifier.add(Dense(units=64,activation = 'relu'))

classifier.add(Dense(units=128,activation = 'relu'))

classifier.add(Dense(units=256,activation = 'relu'))

classifier.add(Dense(units=256,activation = 'relu'))

classifier.add(Dense(units=6,activation = 'softmax'))

#step6 Compiling CNN

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

#step7 Fitting CNN to images

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale = 1./255, # To rescaling the image in range of [0,1]

                                   shear\_range = 0.2, # To randomly shear the images

                                   zoom\_range = 0.2, # To randomly zoom the images

                                   horizontal\_flip = True) #  for randomly flipping half of the images horizontally

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

print("\nTraining the data...\n")

training\_set = train\_datagen.flow\_from\_directory('train',

                                                target\_size=(64,64),

                                                batch\_size=12, #Total no. of batches

                                                class\_mode='categorical')

test\_set = test\_datagen.flow\_from\_directory('test',

                                            target\_size=(64,64),

                                            batch\_size=12,

                                            class\_mode='categorical')

classifier.fit\_generator(training\_set,

                         epochs = 50,

                         verbose = 2,

                         validation\_data = test\_set) # Total testing images

#step8 saving model

classifier.save("model.h5")

**Style.css**

@import url(https://fonts.googleapis.com/css?family=Open+Sans);

.btn {

display: inline-block; \*display: inline; \*zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13px; line-height: 18px; color: #333333; text-align: center;text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75);

vertical-align: middle;

background-color: #f5f5f5;

background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6);

background-image: -ms-linear-gradient(top, #ffffff, #e6e6e6);

background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff), to(#e6e6e6));

background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6);

background-image: -o-linear-gradient(top, #ffffff, #e6e6e6);

background-image: linear-gradient(top, #ffffff, #e6e6e6);

background-repeat: repeat-x;

filter: progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff, endColorstr=#e6e6e6, GradientType=0);

border-color: #e6e6e6 #e6e6e6 #e6e6e6;

border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);

border: 1px solid #e6e6e6;

-webkit-border-radius: 4px; -moz-border-radius: 4px; border-radius: 4px;

-webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);

-moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);

box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);

cursor: pointer; \*margin-left: .3em;

}

.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] {

background-color: #e6e6e6;

}

.btn-large {

padding: 9px 14px;

font-size: 15px;

line-height: normal;

-webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius: 5px;

}

.btn:hover {

color: #333333;

text-decoration: none;

background-color: #e6e6e6;

background-position: 0 -15px;

-webkit-transition: background-position 0.1s linear;

-moz-transition: background-position 0.1s linear;

-ms-transition: background-position 0.1s linear;

-o-transition: background-position 0.1s linear;

transition: background-position 0.1s linear;

}

.btn-primary, .btn-primary:hover {

text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff;

}

.btn-primary.active {

color: rgba(255, 255, 255, 0.75);

}

.btn-primary {

background-color: #4a77d4;

background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4);

background-image: -ms-linear-gradient(top, #6eb6de, #4a77d4);

background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#6eb6de), to(#4a77d4));

background-image: -webkit-linear-gradient(top, #6eb6de, #4a77d4);

background-image: -o-linear-gradient(top, #6eb6de, #4a77d4);

background-image: linear-gradient(top, #6eb6de, #4a77d4);

background-repeat: repeat-x; filter: progid:dximagetransform.microsoft.gradient(startColorstr=#6eb6de, endColorstr=#4a77d4, GradientType=0);

border: 1px solid #3762bc;

text-shadow: 1px 1px 1px rgba(0,0,0,0.4);

box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.5);

}

.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] {

filter: none; background-color: #4a77d4;

}

.btn-block {

width: 100%; display:block;

}

\* {

-webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box; -o-box-sizing:border-box; box-sizing:border-box;

}

html {

width: 100%; height:100%; overflow:hidden;

}

body {

width: 100%;

height:100%;

font-family: 'Open Sans', sans-serif;

color: #fff;

font-size: 18px;

text-align:center;

letter-spacing:1.2px;

}

.login {

position: absolute;

top: 40%;

left: 50%;

margin: -150px 0 0 -150px;

width:400px;

height:400px;

}

.heading{

width: 100%;

}

.fileupload {

width: 100%;

margin-bottom: 10px;

background: rgba(252, 250, 250, 0.938);

border: none;

outline: none;

padding: 10px;

font-size: 13px;

color: #000000;

text-shadow: 1px 1px 1px rgb(0,0,0);

border: 1px solid rgba(0,0,0,0.3);

border-radius: 4px;

box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);

-webkit-transition: box-shadow .5s ease;

-moz-transition: box-shadow .5s ease;

-o-transition: box-shadow .5s ease;

-ms-transition: box-shadow .5s ease;

transition: box-shadow .5s ease;

}

::placeholder {

color: red;

opacity: 1; /\* Firefox \*/

}

input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px rgba(255,255,255,0.2);

}

**Index.html**

<!DOCTYPE html>

<html >

<!--From https://codepen.io/frytyler/pen/EGdtg-->

<head>

<meta charset="UTF-8">

<title>Fruit Classifier</title>

<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

<style>

::placeholder {

color:rgb(13, 13, 14);

opacity:0.8; /\* Firefox \*/

font-weight: bold;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>

<script>

$("#file-picker").change(function(){

var input = document.getElementById('file-picker');

for (var i=0; i<input.files.length; i++)

{

var ext= input.files[i].name.substring(input.files[i].name.lastIndexOf('.')+1).toLowerCase()

if ((ext == 'jpg') || (ext == 'png'))

{

$("#msg").text("Files are supported")

}

else

{

$("#msg").text("Files are NOT supported")

document.getElementById("file-picker").value ="";

}

}

} );

</script>

</head>

<body style="background-image: url('{{ url\_for('static', filename='bg2.png') }}');

background-repeat:no-repeat;

background-size:cover;

height:100vh;

display:flex;

">

<marquee direction="left" style="color:orangered;"><h3>Fruit Classifier</h3></marquee>

<form id="upload-form" action="{{ url\_for('upload') }}" method="POST" enctype="multipart/form-data">

<div class="login card" style="color: orangered;">

<pre>

<h1 style="width: fit-content;color: orangered;">Upload Image To Classify<br> Fruit</h1>

<strong>Choose Image:</strong>

<input class="btn btn-primary btn-block btn-large" type="file" name="file" accept="image/\*" multiple>

<input type="submit" value="Predict!" class="btn btn-primary btn-block btn-large">

</pre>

</div>

</form>

</body>

</html>

**Template.html**

<!DOCTYPE html>

<html >

<!--From https://codepen.io/frytyler/pen/EGdtg-->

<head>

<meta charset="UTF-8">

<title>Fruit Classifier</title>

<link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>

<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

<style>

::placeholder {

color:rgb(13, 13, 14);

opacity:0.8; /\* Firefox \*/

font-weight: bold;

}

</style>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>

<script>

$("#file-picker").change(function(){

var input = document.getElementById('file-picker');

for (var i=0; i<input.files.length; i++)

{

var ext= input.files[i].name.substring(input.files[i].name.lastIndexOf('.')+1).toLowerCase()

if ((ext == 'jpg') || (ext == 'png'))

{

$("#msg").text("Files are supported")

}

else

{

$("#msg").text("Files are NOT supported")

document.getElementById("file-picker").value ="";

}

}

} );

</script>

</head>

<body style="background-image: url('{{ url\_for('static', filename='bg2.png') }}');

background-repeat:no-repeat;

background-size:cover;

height:100vh;

display:flex;

">

<marquee direction="left" style="color: orangered;"><h3>Fruit---Classifying Fruit image</h3></marquee>

<div class="login card" style="color: orangered;">

Uploaded Image:

<br><br>

<img src=" {{url\_for('send\_image', filename=image\_name)}}" style="width:300px;height:300px;">

<br>

{{text}}

<a style="text-decoration: none; display: block; margin: 20px auto;" class="btn btn-primary btn-block btn-large" href="/" role="button">Predict Another image</a>

</div>

</body>

</html>