**SAMPLE CODE**

**User side views.py**

import os

from django.conf import settings

from django.contrib import messages

from django.core.files.storage import FileSystemStorage

from django.shortcuts import render

from .forms import UserRegistrationForm

from .models import UserRegistrationModel

# Create your views here.

def UserRegisterActions(request):

if request.method == 'POST':

form = UserRegistrationForm(request.POST)

if form.is\_valid():

print('Data is Valid')

form.save()

messages.success(request, 'You have been successfully registered')

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

else:

messages.success(request, 'Email or Mobile Already Existed')

print("Invalid form")

else:

form = UserRegistrationForm()

return render(request, 'UserRegistrations.html', {'form': form})

def UserLoginCheck(request):

if request.method == "POST":

loginid = request.POST.get('loginid')

pswd = request.POST.get('pswd')

print("Login ID = ", loginid, ' Password = ', pswd)

try:

check = UserRegistrationModel.objects.get(loginid=loginid, password=pswd)

status = check.status

print('Status is = ', status)

if status == "activated":

request.session['id'] = check.id

request.session['loggeduser'] = check.name

request.session['loginid'] = loginid

request.session['email'] = check.email

print("User id At", check.id, status)

return render(request, 'users/UserHome.html', {})

else:

messages.success(request, 'Your Account Not at activated')

return render(request, 'UserLogin.html')

except Exception as e:

print('Exception is ', str(e))

pass

messages.success(request, 'Invalid Login id and password')

return render(request, 'UserLogin.html', {})

def UserHome(request):

return render(request, 'users/UserHome.html', {})

def user\_predict\_food(request):

if request.method == 'POST':

myfile = request.FILES['file']

fs = FileSystemStorage()

filename = fs.save(myfile.name, myfile)

uploaded\_file\_url = fs.url(filename)

from .utility.Load\_Food\_Models import start\_process

results = start\_process(filename)

return render(request, "users/which\_food\_it\_is.html", {"path": uploaded\_file\_url, "results": results})

else:

return render(request, "users/which\_food\_it\_is.html", {})

def train\_fun(request):

return render(request, "users/train\_func.html", {})

def loss\_fun(request):

return render(request, "users/loss\_func.html", {})

def prediction\_classes(request):

label\_path = os.path.join(settings.MEDIA\_ROOT, 'models', 'labels.txt')

with open(label\_path, 'r') as f:

food101 = [l.strip().lower() for l in f]

print(type(food101))

return render(request, "users/prediction\_class.html", {"labels": food101})

Train Models.py

from keras.layers import Convolution2D

from keras.layers import Dense

from keras.layers import Flatten

from keras.layers import MaxPooling2D

# Importing the Keras libraries and packages

from keras.models import Sequential

# Initialising the CNN

classifier = Sequential()

# Step 1 - Convolution

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

# Step 2 - Pooling

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

# Adding a second convolutional layer

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

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

# Step 3 - Flattening

classifier.add(Flatten())

# Step 4 - Full connection

classifier.add(Dense(output\_dim=128, activation='relu'))

classifier.add(Dense(output\_dim=128, activation='relu'))

classifier.add(Dense(output\_dim=128, activation='relu'))

classifier.add(Dense(output\_dim=3, activation='softmax'))

# Compiling the CNN

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

# Part 2 - Fitting the CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(rescale=1. / 255,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True)

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

training\_set = train\_datagen.flow\_from\_directory('UPMC\_Food101/train',

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical')

test\_set = test\_datagen.flow\_from\_directory('UPMC\_Food101/test',

target\_size=(64, 64),

batch\_size=32,

class\_mode='categorical')

classifier.fit\_generator(training\_set,

samples\_per\_epoch=2030,

nb\_epoch=20,

validation\_data=test\_set,

nb\_val\_samples=679)

classifier.summary()

from keras.models import load\_model

classifier.save('my\_model.h5') # creates a HDF5 file 'my\_model.h5'

# del classifier # deletes the existing model

# returns a compiled model

# identical to the previous one

classifier2 = load\_model('my\_model.h5')

classifier2.summary()

res = classifier.predict(k)

Load\_Models.py

import os

import cv2

import matplotlib.pyplot as plt

import numpy as np

from django.conf import settings

from keras.applications.inception\_v3 import preprocess\_input

from keras.preprocessing import image

def path\_to\_tensor(img\_path):

# loads RGB image as PIL.Image.Image type

img = image.load\_img(img\_path, target\_size=(299, 299))

# convert PIL.Image.Image type to 3D tensor with shape (299, 299, 3)

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

# convert 3D tensor to 4D tensor with shape (1, 299, 299, 3) and return 4D tensor

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

label\_path = os.path.join(settings.MEDIA\_ROOT, 'models', 'labels.txt')

with open(label\_path, 'r') as f:

food101 = [l.strip().lower() for l in f]

# weights\_path = "food101\_final\_model.h5" # orginal weights converted from caffe

weights\_path = os.path.join(settings.MEDIA\_ROOT, "models", "food101\_final\_model.h5")

from keras.applications.inception\_v3 import InceptionV3

from keras.layers import AveragePooling2D, Dropout, Dense, Flatten

from keras.models import Model

n\_classes = 101

# base model is inception\_v3 weights pre-trained on ImageNet

base\_model = InceptionV3(

weights='imagenet',

include\_top=False,

input\_shape=(299, 299, 3)

)

x = base\_model.output

# added layers to the base model

x = AveragePooling2D(pool\_size=(8, 8))(x)

x = Dropout(.4)(x)

x = Flatten()(x)

# add softmax activation

predictions = Dense(n\_classes, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

model.load\_weights(weights\_path)

def start\_process(img\_path, plot=False):

img\_path = os.path.join(settings.MEDIA\_ROOT, img\_path)

# model\_path = os.path.join(settings.MEDIA\_ROOT, "models", "food101\_final\_model.h5")

img = path\_to\_tensor(img\_path)

img = preprocess\_input(img)

# make prediction

predicted\_vec = model.predict(img)

predicted\_label = food101[np.argmax(predicted\_vec)]

# show predicted image

img = cv2.imread(img\_path)

rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

plt.imshow(rgb)

plt.axis('off')

plt.title("yummy! It looks like {}".format(predicted\_label))

plt.show()

# show top 5 predictions with probability

if plot:

# take top 5 probable pics

top5\_probs = np.sort(predicted\_vec)[0][-5:]

top5\_labels = np.argsort(predicted\_vec)[0][-5:]

# plot bar graph

x\_pos = np.arange(len(top5\_labels))

plt.bar(x\_pos, top5\_probs)

plt.title("top 5 predictions")

plt.xticks(x\_pos, [food101[int(idx)] for idx in top5\_labels], rotation=20)

plt.yticks(np.arange(0, 1.1, 0.1))

plt.show()

return predicted\_label

Models.py

from django.db import models

# Create your models here.

class UserRegistrationModel(models.Model):

name = models.CharField(max\_length=100)

loginid = models.CharField(unique=True, max\_length=100)

password = models.CharField(max\_length=100)

mobile = models.CharField(unique=True, max\_length=100)

email = models.CharField(unique=True, max\_length=100)

locality = models.CharField(max\_length=100)

address = models.CharField(max\_length=1000)

city = models.CharField(max\_length=100)

state = models.CharField(max\_length=100)

status = models.CharField(max\_length=100)

def \_\_str\_\_(self):

return self.loginid

class Meta:

db\_table = 'UserRegistrations'

from keras.applications.inception\_v3 import preprocess\_input

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(

rotation\_range=90,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=[.8, 1],

channel\_shift\_range=30,

fill\_mode='reflect',

preprocessing\_function=preprocess\_input,

horizontal\_flip=True,

vertical\_flip=True

)

test\_datagen = ImageDataGenerator(

preprocessing\_function=preprocess\_input,

)

# set path to the data sets

train\_path = 'data/train'

valid\_path = 'data/valid'

test\_path = 'data/test'

# set input dimention to 299x299 (default for inception\_v3)

dimentions = (299, 299)

batch\_size = 32

train\_generator = train\_datagen.flow\_from\_directory(

train\_path,

target\_size=dimentions,

batch\_size=batch\_size

)

validation\_generator = test\_datagen.flow\_from\_directory(

valid\_path,

target\_size=dimentions,

batch\_size=batch\_size

)

test\_generator = test\_datagen.flow\_from\_directory(

test\_path,

target\_size=dimentions,

batch\_size=batch\_size

)

from keras.applications.inception\_v3 import InceptionV3

from keras.layers import AveragePooling2D, Dropout, Dense, Flatten

from keras.models import Model

n\_classes = 101

# base model is inception\_v3 weights pre-trained on ImageNet

base\_model = InceptionV3(

weights='imagenet',

include\_top=False,

input\_shape=(299, 299, 3)

)

x = base\_model.output

# added layers to the base model

x = AveragePooling2D(pool\_size=(8, 8))(x)

x = Dropout(.4)(x)

x = Flatten()(x)

# add softmax activation

predictions = Dense(n\_classes, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=predictions)

def step\_decay(epoch):

initial\_lr = 1e-2

if epoch < 9:

return initial\_lr

else:

return 0.00008

import matplotlib.pyplot as plt

x = [i for i in range(20)]

y = [step\_decay(i) for i in range(20)]

plt.plot(x, y)

from keras.optimizers import SGD

from keras.callbacks import LearningRateScheduler

from keras.metrics import top\_k\_categorical\_accuracy

# optimizer

opt = SGD(lr=.01, momentum=.9)

# add LearningRateScheduler to update it

lr\_scheduler = LearningRateScheduler(step\_decay)

# calculate top\_5\_accuracy to evalute the model

def top\_5\_accuracy(y\_true, y\_pred):

return top\_k\_categorical\_accuracy(y\_true, y\_pred, k=5)

# putting them together and compile the model

model.compile(

optimizer=opt,

loss='categorical\_crossentropy',

metrics=['accuracy', top\_5\_accuracy]

)

from keras.callbacks import ModelCheckpoint

model\_file = 'saved\_models/weights.best.InceptionV3\_final\_model.hdf5'

checkpointer = ModelCheckpoint(filepath=model\_file,

verbose=1, save\_best\_only=True)

# %%time

epochs = 16

history = model.fit\_generator(train\_generator,

validation\_data=validation\_generator,

epochs=epochs,

callbacks=[checkpointer, lr\_scheduler],

steps\_per\_epoch=train\_generator.samples // batch\_size,

validation\_steps=validation\_generator.samples // batch\_size

)

plt.plot(history.history['acc'])

plt.plot(history.history['val\_acc'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'])

plt.show()

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'])

plt.show()

model.load\_weights(model\_file)

model.evaluate\_generator(test\_generator, steps=10, verbose=1)

### saving the trained model

# serialize model to JSON

model\_json = model.to\_json()

with open("saved\_models/food101\_final\_model.json", "w") as json\_file:

json\_file.write(model\_json)

# serialize weights to HDF5

model.save\_weights("saved\_models/food101\_final\_model.h5")

## Load the class labels (which are indexes are the same as the ones from generator)

with open('data/labels.txt', 'r') as f:

food101 = [l.strip().lower() for l in f]

from keras.preprocessing import image

import numpy as np

def path\_to\_tensor(img\_path):

# loads RGB image as PIL.Image.Image type

img = image.load\_img(img\_path, target\_size=(299, 299))

# convert PIL.Image.Image type to 3D tensor with shape (299, 299, 3)

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

# convert 3D tensor to 4D tensor with shape (1, 299, 299, 3) and return 4D tensor

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

import cv2

def img\_analysis(img\_path, plot=False):

# process image

img = path\_to\_tensor(img\_path)

img = preprocess\_input(img)

# make prediction

predicted\_vec = model.predict(img)

predicted\_label = food101[np.argmax(predicted\_vec)]

# show predicted image

img = cv2.imread(img\_path)

rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

plt.imshow(rgb)

plt.axis('off')

plt.title("yummy! It looks like {}".format(predicted\_label))

plt.show()

# show top 5 predictions with probability

if plot:

# take top 5 probable pics

top5\_probs = np.sort(predicted\_vec)[0][-5:]

top5\_labels = np.argsort(predicted\_vec)[0][-5:]

# plot bar graph

x\_pos = np.arange(len(top5\_labels))

plt.bar(x\_pos, top5\_probs)

plt.title("top 5 predictions")

plt.xticks(x\_pos, [food101[int(idx)] for idx in top5\_labels], rotation=20)

plt.yticks(np.arange(0, 1.1, 0.1))

plt.show()

img\_analysis('test\_imgs/sushi.jpg')

Base.html

{%load static %}

<!DOCTYPE html>

<html lang="en">

<head>

<!-- basic -->

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<!-- mobile metas -->

<meta name="viewport" content="width=device-width, initial-scale=1">

<meta name="viewport" content="initial-scale=1, maximum-scale=1">

<!-- site metas -->

<title>Food Classification</title>

<meta name="keywords" content="">

<meta name="description" content="">

<meta name="author" content="">

<!-- bootstrap css -->

<link rel="stylesheet" href="{%static 'css/bootstrap.min.css'%}">

<!-- owl css -->

<link rel="stylesheet" href="{%static 'css/owl.carousel.min.css'%}">

<!-- style css -->

<link rel="stylesheet" href="{%static 'css/style.css'%}">

<!-- responsive-->

<link rel="stylesheet" href="{%static 'css/responsive.css'%}">

<!-- awesome fontfamily -->

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script><![endif]-->

</head>

<!-- body -->

<body class="main-layout">

<!-- loader -->

<div class="loader\_bg">

<div class="loader"><img src="{%static 'images/loading.gif'%}" alt=""/></div>

</div>

<div id="content">

<!-- header -->

<header>

<div class="container-fluid">

<div class="row">

<div class="col-md-3">

<div class="full">

<a class="logo" href="index.html" style="color:WHITE;font-size: 20px">Food image

Classification</a>

</div>

</div>

<div class="col-md-9">

<div class="full">

<div class="right\_header\_info">

<ul>

<li class="button\_user">

<a class="button active" href="{%url 'index'%}">Home</a>

<a class="button active" href="{%url 'AdminLogin'%}">Admin</a>

<a class="button active" href="{%url 'UserLogin'%}">User</a>

<a class="button active" href="{%url 'UserRegister'%}">Register</a>

</li>

</ul>

</div>

</div>

</div>

</div>

</div>

</header>

{%block contents%}

{%endblock%}

<section class="resip\_section">

<div class="container">

<div class="row">

<div class="col-md-12">

<div class="ourheading">

<h2>Recipes</h2>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-md-12">

<div class="owl-carousel owl-theme">

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs1.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Homemade</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs2.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Noodles</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs3.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Egg</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs4.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Sushi Dizzy</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs5.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>The Coffee Break</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs1.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Homemade</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs2.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Noodles</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs3.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Egg</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs4.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>Sushi Dizzy</h3>

</div>

</div>

<div class="item">

<div class="product\_blog\_img">

<img src="{%static 'images/rs5.png'%}" alt="#"/>

</div>

<div class="product\_blog\_cont">

<h3>The Coffee Break</h3>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</section>

<div class="bg\_bg">

<!-- footer -->

<fooetr>

<div class="footer">

<div class="copyright">

<div class="container">

<p>© 2022 All Rights Reserved. Design by Alex Corporation Html Templates</a></p>

</div>

</div>

</div>

</fooetr>

<!-- end footer -->

<!-- Javascript files-->

<script src="{%static 'js/jquery.min.js'%}"></script>

<script src="{%static 'js/popper.min.js'%}"></script>

<script src="{%static 'js/bootstrap.bundle.min.js'%}"></script>

<script src="{%static 'js/owl.carousel.min.js'%}"></script>

<script src="{%static 'js/custom.js'%}"></script>

<script src="{%static 'js/jquery.mCustomScrollbar.concat.min.js'%}"></script>

<script src="{%static 'js/jquery-3.0.0.min.js'%}"></script>

<script type="text/javascript">

$(document).ready(function() {

$("#sidebar").mCustomScrollbar({

theme: "minimal"

});

$('#dismiss, .overlay').on('click', function() {

$('#sidebar').removeClass('active');

$('.overlay').removeClass('active');

});

$('#sidebarCollapse').on('click', function() {

$('#sidebar').addClass('active');

$('.overlay').addClass('active');

$('.collapse.in').toggleClass('in');

$('a[aria-expanded=true]').attr('aria-expanded', 'false');

});

});

</script>

<style>

#owl-demo .item{

margin: 3px;

}

#owl-demo .item img{

display: block;

width: 100%;

height: auto;

}

</style>

<script>

$(document).ready(function() {

var owl = $('.owl-carousel');

owl.owlCarousel({

margin: 10,

nav: true,

loop: true,

responsive: {

0: {

items: 1

},

600: {

items: 2

},

1000: {

items: 5

}

}

})

})

</script>

</body>

</html>