from flask import Flask, request, render\_template

import tensorflow as tf

from tensorflow import keras

import numpy as np

from PIL import Image

app = Flask(\_\_name)

# Load a pre-trained model (e.g., MobileNetV2)

model = keras.applications.MobileNetV2(weights="imagenet")

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

def index():

if request.method == "POST":

if "file" not in request.files:

return render\_template("index.html", error="No file part")

file = request.files["file"]

if file.filename == "":

return render\_template("index.html", error="No selected file")

if file:

img = Image.open(file)

img = img.resize((224, 224))

img = keras.preprocessing.image.img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img = keras.applications.mobilenet\_v2.preprocess\_input(img)

predictions = model.predict(img)

decoded\_predictions = keras.applications.MobileNetV2.decode\_predictions(predictions, top=5)[0]

results = [{"label": label, "score": score} for \_, label, score in decoded\_predictions]

return render\_template("index.html", results=results)

return render\_template("index.html")

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

app.run(debug=True)

pip install Flask

from flask import Flask, request, render\_template

import tensorflow as tf

from tensorflow import keras

import numpy as np

from PIL import Image

app = Flask(\_\_name)

# Load a pre-trained model (e.g., MobileNetV2)

model = keras.applications.MobileNetV2(weights="imagenet")

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

def index():

if request.method == "POST":

if "file" not in request.files:

return render\_template("index.html", error="No file part")

file = request.files["file"]

if file.filename == "":

return render\_template("index.html", error="No selected file")

if file:

img = Image.open(file)

img = img.resize((224, 224))

img = keras.preprocessing.image.img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img = keras.applications.mobilenet\_v2.preprocess\_input(img)

predictions = model.predict(img)

decoded\_predictions = keras.applications.MobileNetV2.decode\_predictions(predictions, top=5)[0]

results = [{"label": label, "score": score} for \_, label, score in decoded\_predictions]

return render\_template("index.html", results=results)

return render\_template("index.html")

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

app.run(debug=True)

<!DOCTYPE html>

<html>

<head>

<title>Image Recognition</title>

<style>

body {

font-family: 'Arial', sans-serif;

background-image: url('your-background-image.jpg');

background-size: cover;

background-repeat: no-repeat;

margin: 0;

padding: 0;

}

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

background-color: rgba(255, 255, 255, 0.8);

box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);

border-radius: 5px;

}

.center-heading {

text-align: center;

margin: 20px 0;

padding: 0;

}

.upload-container {

text-align: center;

padding: 20px 0;

}

#imageUpload {

display: none;

}

label.upload-label {

display: block;

background-color: #3498db;

color: #fff;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

font-weight: bold;

font-size: 16px;

}

img#uploadedImage {

max-width: 100%;

display: none;

margin-top: 20px;

border: 1px solid #ccc;

margin: 0 auto;

display: block;

}

ul#classifications, p#caption {

text-align: center;

margin: 20px 0;

}

</style>

</head>

<body>

<div class="container">

<h1 class="center-heading">Image Recognition with AI Captions</h1>

<div class="upload-container">

<label for="imageUpload" class="upload-label">Upload Image</label>

<input type="file" accept="image/\*" id="imageUpload" onchange="processImage()">

<img id="uploadedImage">

</div>

<h2 class="center-heading">Classified Objects:</h2>

<ul id="classifications"></ul>

<h2 class="center-heading">Generated Caption:</h2>

<p id="caption"></p>

</div>

<script>

function processImage() {

const imageUpload = document.getElementById('imageUpload');

const uploadedImage = document.getElementById('uploadedImage');

const classifications = document.getElementById('classifications');

const caption = document.getElementById('caption');

const file = imageUpload.files[0];

if (!file) {

return;

}

// Display the uploaded image

const imageURL = URL.createObjectURL(file);

uploadedImage.src = imageURL;

uploadedImage.style.display = 'block';

// Perform image classification using IBM Cloud Visual Recognition

// You'll need to use the IBM Visual Recognition API here

// Generate captions for recognized objects

// You'll need to use your chosen natural language generation library

// Update the web page with the classifications and caption

classifications.innerHTML = "<li>Object 1: DOG</li><li>Object 2: Confidence 0.85</li>";

caption.innerHTML = "A generated caption for the recognized objects";

// Handle API calls and caption generation here

}

</script>

</body>

</html>

import tensorflow as tf

from tensorflow import keras

import numpy as np

import matplotlib.pyplot as plt

from PIL import Image

# Load a pre-trained model (e.g., MobileNetV2)

model = keras.applications.MobileNetV2(weights="imagenet")

# Load and preprocess the image

img\_path = "path\_to\_your\_image.jpg"

img = Image.open(img\_path)

img = img.resize((224, 224)) # Resize the image to match the model's input size

img = keras.preprocessing.image.img\_to\_array(img)

img = np.expand\_dims(img, axis=0)

img = keras.applications.mobilenet\_v2.preprocess\_input(img)

# Make predictions on the image

predictions = model.predict(img)

# Decode and display the top-5 predicted classes

decoded\_predictions = keras.applications.MobileNetV2.decode\_predictions(predictions, top=5)[0]

print("Top Predictions:")

for i, (imagenet\_id, label, score) in enumerate(decoded\_predictions):

print(f"{i + 1}: {label} ({score:.2f})")

# Show the image

img = Image.open(img\_path)

plt.imshow(img)

plt.axis("off")

plt.show()

from flask import Flask,request,render\_template,redirect

import os

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

app.config["IMAGE\_UPLOADS"] = "/Users/Carl/Desktop/Playlist/HTML Button 3/static/Images"

#app.config["ALLOWED\_IMAGE\_EXTENSIONS"] = ["PNG","JPG","JPEG"]

from werkzeug.utils import secure\_filename

@app.route('/home',methods = ["GET","POST"])

def upload\_image():

if request.method == "POST":

image = request.files['file']

if image.filename == '':

print("Image must have a file name")

return redirect(request.url)

filename = secure\_filename(image.filename)

basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

image.save(os.path.join(basedir,app.config["IMAGE\_UPLOADS"],filename))

return render\_template("main.html",filename=filename)

return render\_template('main.html')

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

def display\_image(filename):

return redirect(url\_for('static',filename = "/Images" + filename), code=301)

app.run(debug=True,port=2000)

import ibm\_watson

from ibm\_cloud\_sdk\_core.authenticators import IAMAuthenticator

# IBM Cloud Visual Recognition credentials

api\_key = "<YOUR\_API\_KEY>"

url = "<YOUR\_ENDPOINT>"

# Initialize Visual Recognition

authenticator = IAMAuthenticator(api\_key)

visual\_recognition = ibm\_watson.VisualRecognitionV3(

version="2018-03-19",

authenticator=authenticator

)

visual\_recognition.set\_service\_url(url)