|  |  |
| --- | --- |
|  | # -\*- coding: utf-8 -\*- |
|  |  |
|  |  |
|  | from \_\_future\_\_ import print\_function |
|  |  |
|  |  |
|  | from \_\_future\_\_ import division |
|  |  |
|  | import os |
|  |  |
|  | import numpy as np |
|  | import tensorflow as tf |
|  | from PIL import Image |
|  | from flask import Flask, redirect, render\_template, request |
|  | from keras.applications.inception\_v3 import preprocess\_input |
|  | from keras.models import model\_from\_json, load\_model |
|  | from werkzeug.utils import secure\_filename |
|  | from keras.preprocessing import image |
|  |  |
|  | global graph |
|  | graph=tf.compat.v1.get\_default\_graph() |
|  | #this list is used to log the predictions in the server console |
|  | predictions = np.array(["Seneca White Deer", |
|  | "Pangolin", |
|  | "Lady's slipper orchid", |
|  | "Corpse Flower", |
|  | "Spoon Billed Sandpiper", |
|  | "Great Indian Bustard" |
|  | ]) |
|  |  |
|  | #this list contains the link to the predicted species |
|  | found = np.array([ |
|  | "Seneca White Deer", |
|  | "Pangolin", |
|  | "Lady's slipper orchid", |
|  | "Corpse Flower", |
|  | "Spoon Billed Sandpiper", |
|  | "Great Indian Bustard" |
|  | ]) |
|  | app = Flask(\_\_name\_\_) |
|  | model = load\_model("model.h5") |
|  |  |
|  | @app.route('/', methods=['GET']) |
|  | def index(): |
|  | # Home Page |
|  | return render\_template("index.html") |
|  | @app.route('/predict', methods=['GET', 'POST']) |
|  | def upload(): |
|  | if request.method== 'GET': |
|  | return("<h6 style=\"font-face:\"Courier New\";\">No GET request herd.....</h6 >") |
|  | if request.method== 'POST': |
|  | # fecting the uploaded image from the post request using the id 'uploadedimg' |
|  | f = request.files['uploadedimg'] |
|  | basepath = os.path.dirname(\_\_file\_\_) |
|  | #securing the file by creating a path in local storage |
|  | file\_path = os.path.join(basepath, 'uploads', secure\_filename(f.filename)) |
|  | #Saving the uploaded image locally |
|  | f.save(file\_path) |
|  | #loading the locally saved image |
|  | img = tf.keras.utils.load\_img(file\_path, target\_size=(224, 224)) |
|  | #converting the loaded image to image array |
|  | x = tf.keras.utils.img\_to\_array(img) |
|  | x = preprocess\_input(x) |
|  | #converting the preprocessed image to numpy array |
|  | inp = np.array([x]) |
|  | with graph.as\_default(): |
|  | #loading the saved model from training |
|  | json\_file = open('DigitalNaturalist.json') |
|  | loaded\_model\_json = json\_file.read() |
|  | json\_file.close() |
|  | loaded\_model = model\_from\_json(loaded\_model\_json) |
|  | #adding weights to the trained model |
|  | loaded\_model.load\_weights("model.h5") |
|  | #predecting the image |
|  | preds = np.argmax(loaded\_model.predict(inp),axis=1) |
|  |  |
|  | #logs are printed to the console |
|  | print("The predicted species is " , predictions[preds[0]]) |
|  | text = "The predicted species is " + found[preds[0]] |
|  | return render\_template("index.html", RESULT = text) |
|  |  |
|  |  |
|  | if \_\_name\_\_ == '\_\_main\_\_': |
|  | #Threads enabled so multiple users can request simutaneously |
|  | #debud is turned off, turn on during development to debug the errors |
|  | #applications is binded to port 8000 |
|  | app.run(threaded = True,debug=True,port="8000") |