Week 4 assignment

Name: Flask implementation

Batch code: LISUM01

Submission date: 07/04/2021

Submitted to: Neil Mauricio (intern)

We downloaded a database with cats and dogs photos from Kaggle, the url:

[Cats -- VS -- Dogs | Kaggle](https://www.kaggle.com/pybear/cats-vs-dogs)

After, we utilize a model trained for recognition of dog photos from the code at the same url: (deployed in tensorflow)

The principal code file “app.py” contains the methods ‘GET’ and ‘POST’ from Flask and link the html files “show.html” and “index.html”.

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

from flask\_bootstrap import Bootstrap

import os

import inference

app = Flask(\_\_name\_\_)

Bootstrap(app)

"""

Routes

"""

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

def index():

    if request.method == 'POST':

        uploaded\_file = request.files['file']

        if uploaded\_file.filename != '':

            image\_path = os.path.join('static', uploaded\_file.filename)

            uploaded\_file.save(image\_path)

            class\_name = inference.get\_prediction(image\_path)

            print('CLASS NAME=', class\_name)

            result = {

                'class\_name': class\_name,

                'image\_path': image\_path,

            }

            return render\_template('show.html', result=result)

    return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

To implement the web pages using html we utilized flask templates and complement styles with CSS:

{% extends "base.html" %}

{% block content %}

 <div>Upload File</div>

 <hr>

 <form class="form-inline" action="{{ url\_for('index') }}" method="post" enctype="multipart/form-data">

     <div class="form-group">

        <input type="file" name="file" class="btn btn-default">

     </div>

     <div class="form-group">

        <input type="submit" value="Upload" class="btn btn-default">

     </div>

 </form>

{% endblock %}

Index.html

{% extends "base.html" %}

{% block content %}

    <div class="container">

        <h1>Predicted Class: {{ result.class\_name }}</h1>

        <hr>

        <div>

            <img src="{{ result.image\_path }}" class="img-rounded" width="400" height="auto">

        </div>

        <a ref="{{ url\_for('index') }}" class="btn btn-default">Back</a>

    </div>

{% endblock %}

Show.html

{% extends "bootstrap/base.html" %}

{% block title %}Cats, Dogs detector{% endblock %}

{% block content %}{% endblock %}

Base.html

The model calls the trained file of recognition, loads a photo and if the prediction lies upper from 0.5 the result is class Dog otherwise it’s a cat. Utilizing json and request libraries we can deserialize the image files to analysis.

import tensorflow as tf

import numpy as np

import json

import requests

SIZE=128

MODEL\_URI='http://localhost:8501/v1/models/pets:predict'

CLASSES = ['Cat', 'Dog']

def get\_prediction(image\_path):

    image = tf.keras.preprocessing.image.load\_img(

        image\_path, target\_size=(SIZE, SIZE)

    )

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

    image = tf.keras.applications.mobilenet\_v2.preprocess\_input(image)

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

    data = json.dumps({

        'instances': image.tolist()

    })

    response = requests.post(MODEL\_URI, data=data.encode())

    result = json.loads(response.text)

    prediction = np.squeeze(result['predictions'][0])

    class\_name = CLASSES[int(prediction > 0.5)]

    return class\_name

inference.py

to run the model, we utilize Docker:

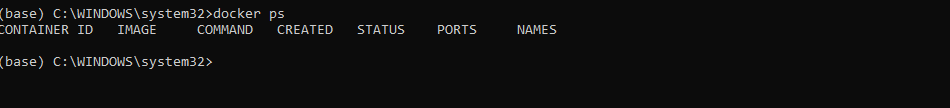


Fig. 1

Running the command:

docker run -p 8501:8501 –name=pets -v “local-route” -e MODEL\_NAME=pets tensorflow/serving

Texto

Descripción generada automáticamente

Fig. 2

We verified the correct creation of the environment:

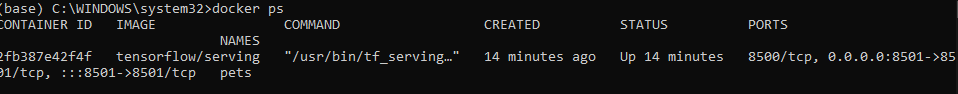


Fig. 3

After, we can run the python file and it begins to run the web server in localhost:

Texto

Descripción generada automáticamente

Fig. 4

Introducing this address to web browser we visualize the app running.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fig. 5

We select an image from the database and click upload.

Un perro con la boca abierta

Descripción generada automáticamente con confianza media

(a)

Un foto montaje de un gato

Descripción generada automáticamente

(b)

Fig. 6

After, the result for inference is displayed.