Name: Flask API

Report date: 04-07-2021

Internship Batch: LISUM01 Version:1.0

Data intake by: Almudena Zhou Ramírez López

Data intake reviewer:

Model deployment

Before the model deployment, you need to train and save your model and upload to github with the name model.h5. This model will predict the label of an image with a single number. This part has been done in the model_training.py

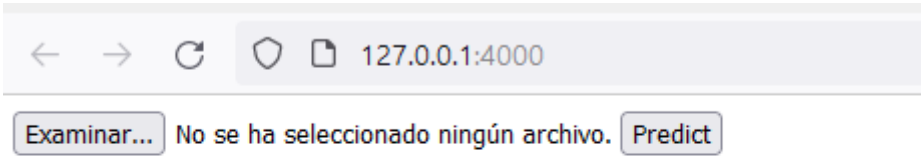
The image must have 28x28 pixel dimensions since I have used the mnist dataset. The script for the API Flask creation is in app.py.

```
@app.route('/')
def home():
    return render_template('app.html')


@app.route("/predict", methods=["POST"])
def predict():
    print("[+] request received")
    # get the data from the request and put ir under the right format
    req = request.get_json(force=True)
    image = decode_request(req)
    batch = preprocess(image)
    print(batch.shape)
    # actual prediction of the model
    predictions = model.predict(batch)[0]
    print(predictions)
    pred_list = [np.argmax(predictions), predictions[np.argmax(predictions)]]
    response = {"prediction": [str(pred_list[0]), str(pred_list[1])]}
    print("[+] results {}".format(response))

    return jsonify(response)


if __name__ == "__main__":
    app.run(debug=True, port=4000)
```

The last lines of code start the API in the port 4000, while the functions are defining how each part of the API will be presented. The home page will be shown as is indicated in app.html.

```html
<!DOCTYPE html>
<html>

    <head>
        <!-- title displayed on the browser tab -->
        <title>image classification app</title>
    </head>

    <!-- contains our front-end elements -->
    <body>
        <!--input to upload the image-->
        <input id="upload" type="file">
        <!--button to activate the script which call our Flask app-->
        <button id="predict-button">Predict</button>
        <h1>Predictions</h1>
        <p><span id="It must be an image 28x28 in png format"></span></p>
        <!--parapgraph where we will display the result-->
        <p><span id="prediction"></span></p>
        <!--display the uploaded image-->
        <img id="selected-image" src=""/>
    </body>

    <!--jquery lib-->
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <!--The script which call our Flask app-->
    <script>
        let base64Image;
        // when we upload an image
        $("#upload").change(function() {
            let reader = new FileReader();
            reader.onload = function(e) {
                let dataURL = reader.result;
                $('#selected-image').attr("src", dataURL);
                base64Image = dataURL.replace("","");
            }
            reader.readAsDataURL($("#upload")[0].files[0]);
            //reset pred text as empty
            $("#prediction").text("");
        });

        $("#predict-button").click(function(){
            let message = {image: base64Image}
            // you could also use 127.0.0.1 instead of 0.0.0.0
            $.post("http://127.0.0.1:4000/predict", JSON.stringify(message), function(response){
            $("#prediction").text("results: "+ response.prediction[0] + " with probability: " + response.prediction[1]);
                console.log(response);
            });
        });
    </script>
</html>
```

Home page:



Predictions

Once you send an image and click Predict, it will send the data to the /predict path, decode the image, process and scale the data, and send a prediction from there to the home page.

All the processes commented above for the image, are done through the functions in the image below.

```python
# decode the image coming from the request
def decode_request(req):
    encoded = req["image"]
    decoded = base64.b64decode(encoded)
    return decoded

def load_model1():
    new_model = keras.models.load_model('model.h5')
    return new_model

model = load_model1()

def preprocess(decoded):
    pil_image = cv2.imdecode(np.frombuffer(decoded, dtype=np.uint8), cv2.IMREAD_UNCHANGED)
    supp_pil = Image.fromarray(pil_image).convert("L")
    image = np.asarray(supp_pil).reshape((1, IMG_SHAPE[0], IMG_SHAPE[1], 1))

    return image/255
```

After that, the image goes to the /predict as we said and make the prediction with the model loaded here. After all the process, the image we see from the API/web page is the following: