Nikita Agarwal, Varun Prasad, Ajinkya Ghadge

Design document : Project 2, COMPSCI 532

1. Project structure and instructions

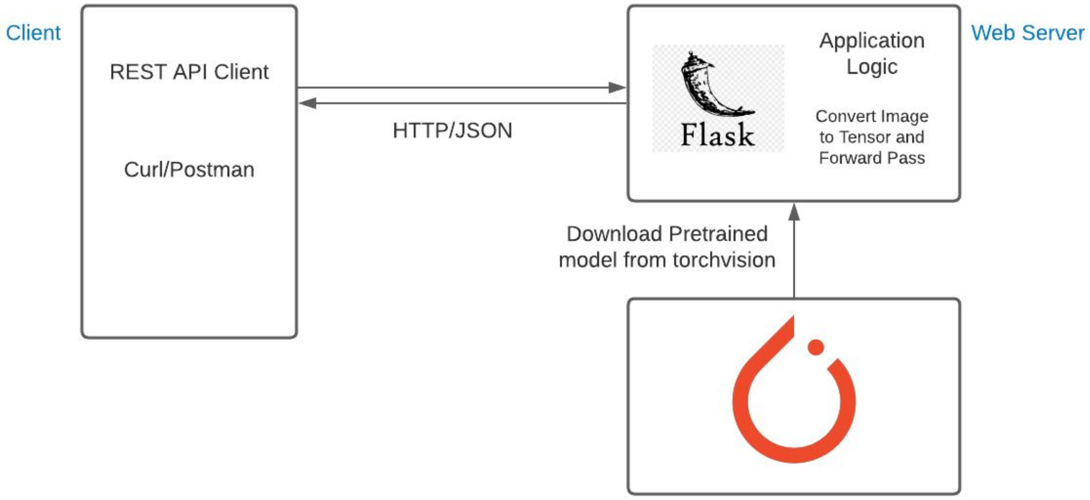
a. Instructions to test deployed container on Heroku

i. Place your images in the test folder and execute the python file testscript.py. Comment out the last line in the script and uncomment the second last line. It will generate results for all the files in the folder. Note that it may take upto a minute for the heroku container to wake up from sleep, after which it will work faster.

b. Instructions to build and test locally

i.
```
1. Run the command ```sudo docker build -t
flask-mlapplication:latest .``` and wait for the build
process to complete
```
ii.
```
2. Run the command ```sudo docker run --name
flask-mlapplication -p 5000:5000 flask-mlapplication```,
this will start the server.
```
iii.
```
3. Open the testfolder folder, and paste the images you
want to test
```
iv.
```
4. Open another terminal Run the python test script using
the command ```python testscript.py```
```

c. Architecture


Client
REST API Client
Curl/Postman
HTTP/JSON
Flask
Application Logic
Convert Image to Tensor and Forward Pass
Web Server
Download Pretrained model from torchvision

d.

e. Flask is a web application framework which is being used to serve the image classification model as a service. We use REST architecture as shown below

f. For containerization of the application we use Docker, which packages all dependencies together and can be exported into an image, which can be used as a plug and play system.

g. For deploying the container we use Heroku, which is a cloud compute and storage solution.

h. Execution flow of the program is as follows:

    i. An image is passed by the user, in the payload of a POST request.

    ii. The image is flattened out transformed before performing a forward pass.

    iii. We perform the forward pass and obtain the output from DenseNet

    iv. The output category is obtained by getting value for the output from a dictionary.

    v. The category is returned as an HTTP response

2. Design considerations

a. Using REST

    i. Other methods like SOAP protocol or RPC are slightly more complex and the given application can benefit from simple request and response based architecture used by REST architecture as it is a really simple system.

    ii. The Flask framework is primarily built for applications using the REST architecture.

b. Excluding pretrained-model from the program to limit size of docker container and give flexibility

    i. The size of the container gets really large due to installation of all dependencies. We thought of letting the server download the pretrained model so as to keep the size of the docker image small.

c. Deploying on Heroku vs Google cloud vs AWS

    i. All platforms are equally competitive with AWS offering great value. However, there are two areas where Heroku exceeds its competitors. Ease of working with the platform, and a very generous free tier. One can very easily push changes to a Heroku container.