532 Systems for Data Science
Project 2 ML Inference Report

Vishal Keshav
Kenneth Myers
Jessie Huo

Goal

This project aims to implement a server that classifies animal images by the pretrained machine learning model Densenet-121. Clients could make HTTP requests (by either command line or the user interface) that include images to the server, whose code is packed in a docker container and runs on local host to serve requests.

Docker setup and configuration

To spin up the server, simply run the shell script start_docker_script.sh, which contains only one sudo command 'docker-compose up'. This command reads the docker-compose.yaml file, which lists all the services needed for our web project (in this case only one service is needed), and follows the six steps specified in Dockerfile, as listed below, to build the image.

1. FROM python:3.7

   // launching point
2. WORKDIR /app
3. COPY . .
4. RUN pip install --upgrade -r requirements.txt

   // requirements.txt lists all the tools needed
5. ENTRYPOINT ["python"]
6. CMD ["app.py"]

   // a flask app that does inference

Interface to start the docker

To start the application in the docker, we have provided a script start_docker_script.sh. This acts as a starting point to start building the pre-specified docker image and then start running the docker. Once, the docker is up and running, the inference application will be served on the localhost at port 5000(https://0.0.0.0:5000)

We have implemented as web based application that starts serving a webpage on localhost. The user can interact with the app using a webbrowser.

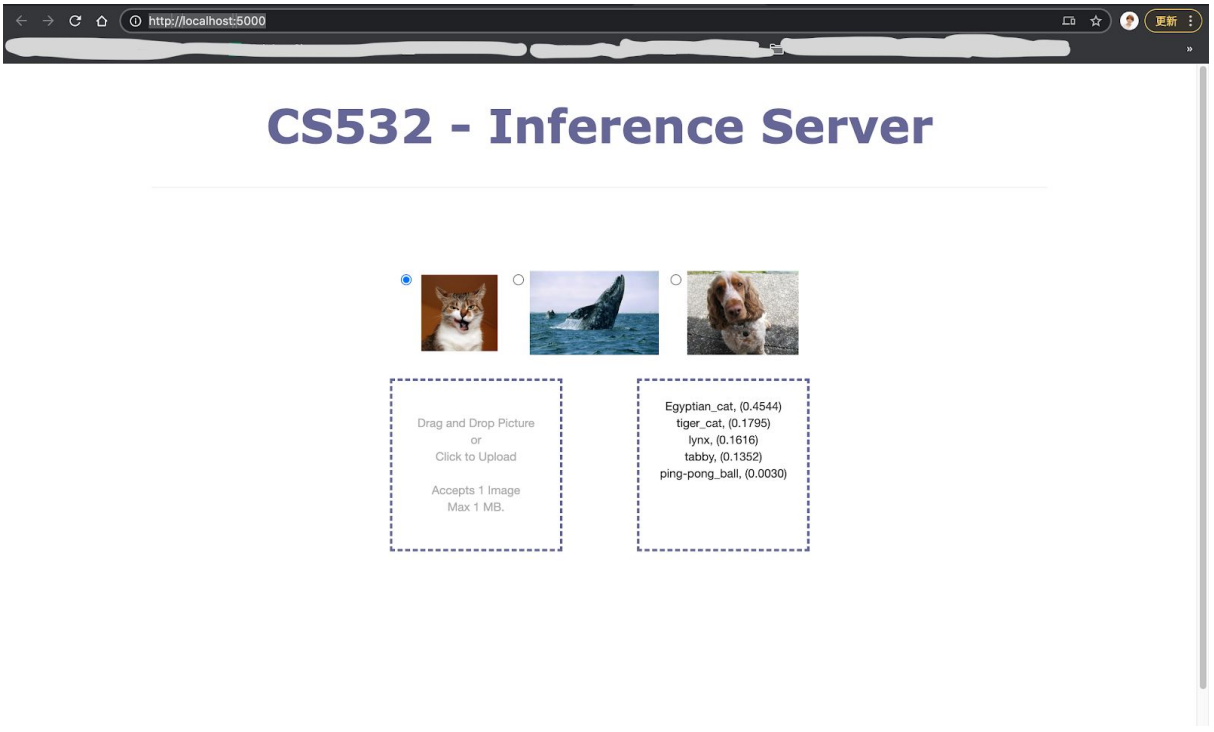
Alternatively, we also provide a command line interface that uses curl to send POST request to the server.

Using CLI to query the class of an image

An example using curl (curl -X POST -F "file=@upload/cat.jpg" http://0.0.0.0:5000/uploadajax) is shown in infer_image_script.sh. The client could upload an image under the /upload directory to the server for classification.

Using UI to query the class of an image

The following UI would appear after opening the browser and going to http://localhost:5000/.



The user could select from the three sample images or upload one according to his/her choices, and the top five output probability of labels would be ranked from highest to lowest in the right hand side column.

Implementation details for inference engine

The inference functionality is in inference_core.py, where the class ‘inference_engine’ is defined.

Implementation details for Flask web application

The flask library is used to develop the server app, that includes rendering the webpage for web UI.

Conclusion

In conclusion, we have experimented with pytorch inference engine, and developed a fully containerized app that has both UI and command line interface to do inference on user's image.