CS 532 Project II Report

Arvind Govindarajan

The objective of this project is to be able to serve the popular Densenet model on PyTorch as a server on a standalone Docker container that accepts image files to classify and returns the prediction to the user. In order to achieve this, I used Anaconda- a python package manager, Python 3.7, Flask, Torchvision, and docker itself.

To start, we shall use Docker from continuum:miniconda which assumes that the conda distribution is already present in the docker. This allows us to start working with python 2.7 on the fly. However, I used Python 3.7 for development and conda can create a new virtual environment to mimic the environment on the development machine. More details on this are mentioned later in the report.

Torchvision:

Used torchvision models to load densenet and used the predefined preprocessing specifications to convert images to required format. In addition, the script checks for if CUDA is available so that the model can be loaded onto the GPU if need be for faster inference.

In order to interpret the results, I use a json file that maps the results of the Densenet model to human readable labels. This is what is returned as prediction to the user.

Flask:

To host the server, I use Flask. The server is loaded onto a basic app with no static files or templates. The server is hosted on “0.0.0.0” on port 5000. The default route acts as a test for checking if the server is online, and the upload route is used for prediction. It uses the default REST methods of POST. If the uploaded file is of type image (checked via config for Flask app) and less than 5 MB (constraints to prevent malicious file uploads) with non-empty file name, the route triggers prediction from the model by passing the image as input. The image itself is preprocessed and then fed into the model for prediction. If the image cannot be read, the model fails and returns the result to the user.

Anaconda:

The choice of miniconda is well justified in that it is an easy way to manage packages for Python and Data Science. It is easy to export environments across conda as YAML files. The development of the model server requires the packages: flask, and torchvision. In addition, we also add a gunicorn dependency if the server were to be used in a production environment.

The environment on the docker container is built from the env_from-history.yml file. In doing so, we have all the pre-requisites to launch the flask app on the container.

Dockerfile:

The dockerfile is built from the latest miniconda distribution. The project directory is copied into container and the environment is built from YAML file followed by the reload of bashrc and the activation of the environment. Once that is done, the flask app is launched within the container and is exposed on port 5000.