COMPSCI 532: ML Inference

Technical Design Document

Dhruv Agarwal (dagarwal@umass.edu)
Shantam Shorewala (sshorewala@umass.edu)
Shubham Shetty (shubhamshett@umass.edu)

May 2021

1 Introduction

In this project, we implement an image classification server that leverages DenseNet-121 model. PyTorch library has been used to run inference on images whereas Flask, a web framework, is used to handle the HTTP requests. The application is hosted on a local server. The entire project has been containerized using Docker to standardize deployment on different machines/servers.

1.1 Project Overview

The main objective of the project is to create an application which provides the end-user the ability to generate HTTP requests for running image classification on RGB images. The major components of the application and its deployment have been described in following sections.

1.2 Technical Requirements

We have chosen to use Python 3 to develop our library for this project. Following Python libraries were used for our program -

- *Flask*
- *PyTorch*

Further, Docker is used to setup the environment configuration required to run the application.

2 System Design

2.1 Web server

- Flask framework has been used to implement the web server for this project.
- Port 5000 on localhost has been used to host the image classification application.
- We use POST requests in our system.
- The application returns a JSON which contains the predicted class ID and name.

2.2 Classification model using PyTorch

- Pre-trained DenseNet-121 model is used to run image classification.
- Images are resized to 255 × 255 before being passed to the model for inference.
- PyTorch supports both training and evaluation phase for the model. Since we use a pre-trained model, it is run in the eval mode where no gradients are computed or updated.
- The model returns the class corresponding to the highest probability (softmax).
- Images are stored in the /data folder.

2.3 Docker containerization

- Docker allows automation of environment configuration required to deploy the application on different servers/machines.
- Dockerfile is used to build an image of the environment configuration. In our case, we build our image on top of the *pytorch* image.
- Dockerfile installs the remaining application requirements (Flask, numpy) using the pytorch image as a base.
- It also exposes the port 5000 for the server to run on.

3 References

1. PyTorch (`https://pytorch.org/`)
2. Docker (`https://www.docker.com/`)
3. Flask (`https://flask.palletsprojects.com/en/1.1.x/`)