**CSCIGA.3033 HW 4**

Chenmeinian Guo

This report details the process of setting up a Docker environment for running the MNIST digit recognition task using PyTorch. The purpose of this homework is to demonstrate an understanding of container technologies and gain hands-on experience on using with container technologies (for machine learning tasks).

**Files:**

`main.py`: source code from the example code set; the training is implemented using PyTorch

`requirement.txt`: dependencies (packages) for running the code

`Dockerfile`: contains scripts to create and setup the container

**Dockerfile**

```

# Use an official Python runtime as a base image

FROM python:3.8-slim

# Set the working directory to /app

WORKDIR /app

# Install the pre-reqs

ADD requirements.txt /app

# Install any needed packages specified in requirements.txt

RUN pip install -r requirements.txt

# Add the code as the last Docker layer because it changes the most

ADD main.py /app/main.py

# Run main.py when the container launches

CMD ["python", "main.py", "--epochs", "1"]

```

Since I’m using a M2 Mac Air to do this homework assignment, the default epoch number=14 is too heavy for my machine. Therefore, I changed the parameter to epoch=1 in the Dockerfie because the main purpose for this assignment is not to gain competitive result.

After finishing the Dockerfile, the nexr step is to build the Docker Image:

With the command `**docker build -t mnist** .`, we are able to build the Docker image tagged as mnist from the Dockerfile in the current directory. Then, we could run the mnist image in a new container named mnist\_run with this command `**docker run -it --name mnist\_run mnist**

`.

Since Docker containers are isolated environments, we guarantee that the MNIST model training can be reproduced on any system with Docker installed, regardless of underlying differences in OS or installed libraries.

**Dockerfile Runtime Screenshot:**

A screenshot of a computer program

Description automatically generated

…



**Singularity**

From my experience when doing this homework, this is much more complicated to use then the Docker, but the procedures are similar.

1. Download Vagrant from the official website: <https://developer.hashicorp.com/vagrant/install>
2. Download VirtualBox from the official website:

https://www.virtualbox.org/wiki/Download\_Old\_Builds\_7\_0

1. Create a new Vagrantfile (in this particular case, I followed the sample code and created an Ubuntu 20.04 VM).
2. Once VirtualBox is installed, navigate to the project directory where the Vagrantfile is located and run:

`vagrant up --provider=virtualbox`

1. Now, we could just run the main.py as usual.

**Discussion**

VirtualBox/Vagrant and Docker are both tools that facilitate the creation of isolated environments for development and deployment, but they operate at different levels of the system and serve slightly different purposes. VirtualBox is a hypervisor that creates virtual machines (VMs). Each VM includes a full copy of an operating system, a virtual copy of the hardware that the OS requires to run, and complete isolation from the host system. The setup of VirtualBox/Vagrant is suitable for testing across different operating systems and environments where full isolation is necessary and is more resource-intensive and less efficient compared to Docker. On the other hand, Docker is a containerization platform that encapsulates an application and its dependencies into a container that runs on the same operating system kernel as the host. It is therefore much more resource-efficient and faster, especially for application deployment and development. It is ideal for continuous integration and deployment environments.