MIT Big Data Remote Course

Handwritten Number Recognition stored by Cassandra and Encapsulated by Docker

Qiangqiang Liu
Computer Science and Software Engineering
Xi'an Jiaotong-Liverpool University
Su Zhou China
Qiangqiang.liu17@student.xjtlu.edu.cn

***Abstract*—Docker, an open platform that can be used for building and distributing, are wildly used in Computer Science field. By using docker, a programmer can program ignoring the differences of Operating System. The aim of this report is to build a docker file which can realize handwritten number recognition (from 0 to 9) and results storage. This report includes an introduction stating the outline of this remote course. Literature review will be provided to recall some background knowledge needed for this course. After that operating principles depicting how that docker file is established and how it works will be given. Finally, an introspection about the program and myself will be given.**

Keywords—Docker, Handwritten Recognition, MNIST, Cassandra, Python

I. Introduction

Last winter holiday, I joined the MIT Big Data Remote Course to do some research with my teacher Mr. Zhang. The purpose of this course is to build up a website which allows the user to upload a handwritten number picture with 28*28 pixels formats. After uploading that picture, the back-end of the website can specify what the number is by using python program. The results of that picture will be stored in Cassandra database. Different from others who arrange their program directly in their computer or server, this course require us to encapsulate the program using Docker so that the program can run in different operating system. During the program, distributed version control system such as GitHub is used for a better programming experience. This report will firstly give a literature review to introduce the background needed for this course. After that operating principles about how these functions established will be also provided. Finally, an introspection will be given to reflect my disadvantages.

II. Background Information

A. GitHub

During this course, the tool I used most often is GitHub. GitHub is founded on Git which is a source management system [1]. GitHub is a web-based distributed version control system in which both the codes and the history of the codes are mirrored in every developer's computer. The difference between GitHub and Git is that Git is used for local code management while GitHub provides online code control system. One feature of GitHub is its cooperating performance. For example, if there is a big project which has been published on GitHub by Tom but not finished. Tom encountered a problem and Tom opened his project to everyone for help. Then Jerry who wants to help Tom can access to that project by fork. Fork allows Jerry to have a copy of Tom's project in his GitHub. To load Tom's project from web(GitHub) to local disk, Jerry needs to use Git to clone the project online to local computer disk. After Jerry finished his code, he has to pull a request to Tom to ask for a change on Tom's project. Once Tom agrees Jerry's request, changes can be made on his project. Another important feature of GitHub is its convenience of bug fixing. When you started a project or you fork a project from others for the first time. A branch called master branch will be automatedly created by Git. When you have bug in the master branch, you can use git branch to create another branch. After that, you can fix your bug on the new branch. You can get back to the original code using git check out master whenever you want. After you fixing the bug on the new branch, you can use git merge to merge the new branch and master branch together. But sometimes there are always conflicts in this step. When conflicts occur, we need to deal this conflict firstly and then merge the branches together. Therefore, GitHub is a powerful code management tool used by most programmers due to its easy cooperation and bug fixing feature.

B. Docker

The most important software in this big data remote course is Docker. Docker is an open platform for developers to build, ship and run distributed applications by using container. Docker containers are based on images [2]. A Docker image is an executable package, which contains everything needed to run a program, such as the code, a runtime, libraries, environment variables, and configuration files. In addition, constructing a container requires a Dockerfile in advance, which is used to run some commands and ensure some statements automatically. Nowadays, Docker becomes quite popular mainly because of flexibility and lightweight. Unlike traditional virtual machines, Docker can containerize most of applications and leverage the host kernel. Virtual machines require installing all the software and application code manually, which is rather time-consuming. However, in the Docker world, it just takes seconds to build a container and fulfill the same function [3]. As a result, Docker is convenient for peoples life and work.

C. *Cassandra*

Apache Cassandra is an open source distributed database management system, aimed to handle very large amounts of data spread out across many commodity severs while providing a highly available service with no single point of failure. Its core is NoSQL solution, initially developed by Facebook for reliability and scalability [4].

III. Implementation

There are mainly three steps in the implementation of this project. The first step, which is the core part , is building a Deep Natural Network to recognize the handwriting number. The second step is building up a website to allow the upload of handwriting picture. The third step is store the results in Cassandra and encapsulate the whole project into Docker.

A. *DNN Module for MNIST*

a. *Structure of DNN*

The DNN model I designed is a 64*32*10 dense model with 3 layers.

The ***1st layer*** is an input layer which contains 64 input units (each unit represent one feature of the input iamge). The activation function for first layer is Sigmoid function.

The ***2nd layer*** is a fully connected hidden layer to the first layer which contains 32 activation units. The activation function method for second layer is Sigmoid function.

The ***3rd layer*** is a fully connected output layer which contains 10 activation units. The activation function for this output layer is SoftMax. Each unit in the output layer has a probability which represents their confidence to be positive.

b. *Optimizer, Loss Function, Metric*

Optimizer is the algorithm which is used to minimize the loss function. Different optimizer has different performance on minimization. In this DNN model, I choose to use Adaptive Moment Estimation algorithm (Adam).

Loss function is used to measure the cost of this model. The aim of machine learning is to minimize this cost. In this DNN model, I choose to use Categorical Cross Entropy as loss function because it is more suitable for multi-class classification.

Metric is used to measure the goodness of the model. In this DNN model, I choose to use categorical accuracy to metric this model.

c. *Train this DNN model* ***<u>(Additional Requirement)</u>***

In the Python source file, I have designed a ***create_dense_model(X_train, y_train, num_class=10)***. By calling this function, you can create a DNN model and train on the training dataset.

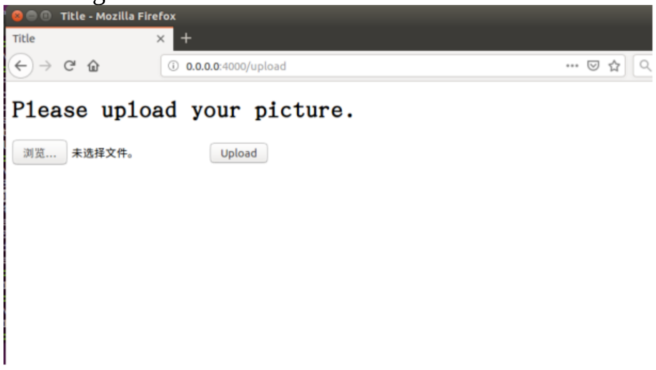
B. *Website Building*

Flask is used to build up the website. The detailed code is showing below:

```
import os
from flask import Flask, flash, request, redirect, url_for
from werkzeug.utils import secure_filename
UPLOAD_FOLDER = '/app/upload_picture'
ALLOWED_EXTENSIONS = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
def allowed_file(filename):
  return '.' in filename and \
  filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
@app.route('/', methods=['GET', 'POST'])
    def upload_file():
      if request.method == 'POST':
        # check if the post request has the file part
        if 'file' not in request.files:
          flash('No file part')
          return redirect(request.url)
        file = request.files['file']
        # if user does not select file, browser also
        # submit an empty part without filename
        if file.filename == '':
          flash('No selected file')
          return redirect(request.url)
        if file and allowed_file(file.filename):
          filename = secure_filename(file.filename)
          file.save(os.path.join(app.config['UPLOAD_FOLDER'],
    filename))
          return redirect(url_for('uploaded_file',
                        filename=filename))
      return '''
      <!doctype html>
      <title>Upload new File</title>
      <h1>Upload new File</h1>
      <form method=post enctype=multipart/form-data>
       <input type=file name=file>
       <input type=submit value=Upload>
```

```
        </form>
    '''
```

The image below shows what the website looks like:



Please upload your picture.

C. *Docker Encapulation and Result Storing*

a. Result Storing

Cassandra is used to store the result of the handwriting number. Instead of download Cassandra directly, it is more convenient to manipulate Cassandra using Docker. The specific codes is showing below:

*(1)**Download Cassandra***

docker pull cassandra;

*(2)**Start an instance***

docker run –name some-cassandra –network some-network -d cassandra:tag

*(3)**Connect handwriting container to Cassandra Container***

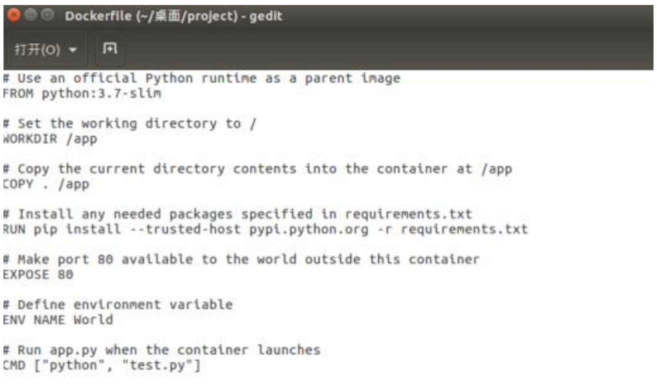
docker run -d –name mnist -p 4000:80 test

docker run -d –name hchen-cassandra –network container:mnist cassandra

b. Docker Encapsulation

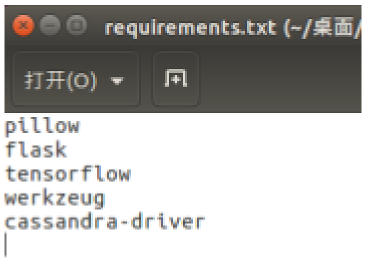
From the Docker official documentation, building a container needs three parts: a Dock a Docker-file, the app itself and a requirement text. The Dockerfile is used to define the environment in the container.

Docker File:



```
# Use an official Python runtime as a parent image
FROM python:3.7-slim

# Set the working directory to /
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "test.py"]
```

Requirement text:



```
pillow
flask
tensorflow
werkzeug
cassandra-driver
```

IV. Conclusion

A. *Summary*

The final purpose of this course is to build up a website which can let user upload a handwriting picture of number (what I said is something about MNIST). After uploading the picture, the website can return what the number is and store the result in Cassandra which is a database.

References

[1] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," IEEE Cloud Comput., vol. 1, no. 3, pp. 81-84, 2014.

[2] C. Anderson, "Docker," Softw. Eng., pp. 102-104, 2015.

[3] J. James, "Cassandra," Notes Queries, vol. s2-X, no. 241, p. 111, 1860.