大数据分析总结报告

图片识别服务

黄艺冉

2020-2-16—2020-4-12

目录

第一章　项目简介

本项目基于 tensorflow 机器学习图片识别的模型，完成一个简单的图片识别应用服务，该服务可以让别人使用，即从本地选择图片并上传。同时，当前预测时间、上传的文件名以及预测结果会写入 cassandra 数据库中。

此外，以上所有均在 docker 容器内部运行，并实现了两个不同容器的相互正常通信，增加了项目的难度。

第二章 知识点简要描述

2.1　机器学习：tensorflow tutorials

https://tensorflow.google.cn/tutorials/keras/classification?hl=zh_cn

`Basic classification: Classify images of clothing`

2.2　虚拟化的容器技术：docker

将服务放在容器里面运行，Dockerfile 的编写。

https://docs.docker.com/get-started/

2.3　服务部署：flask

文件上传 flask 应用。

http://flask.pocoo.org/docs/0.12/quickstart/#a-minimal-application

2.4　大数据存储技术：cassandra

https://hub.docker.com/_/cassandra/

2.4.1 Cassandra-driver

使 python 代码连接 cassandra 数据库。

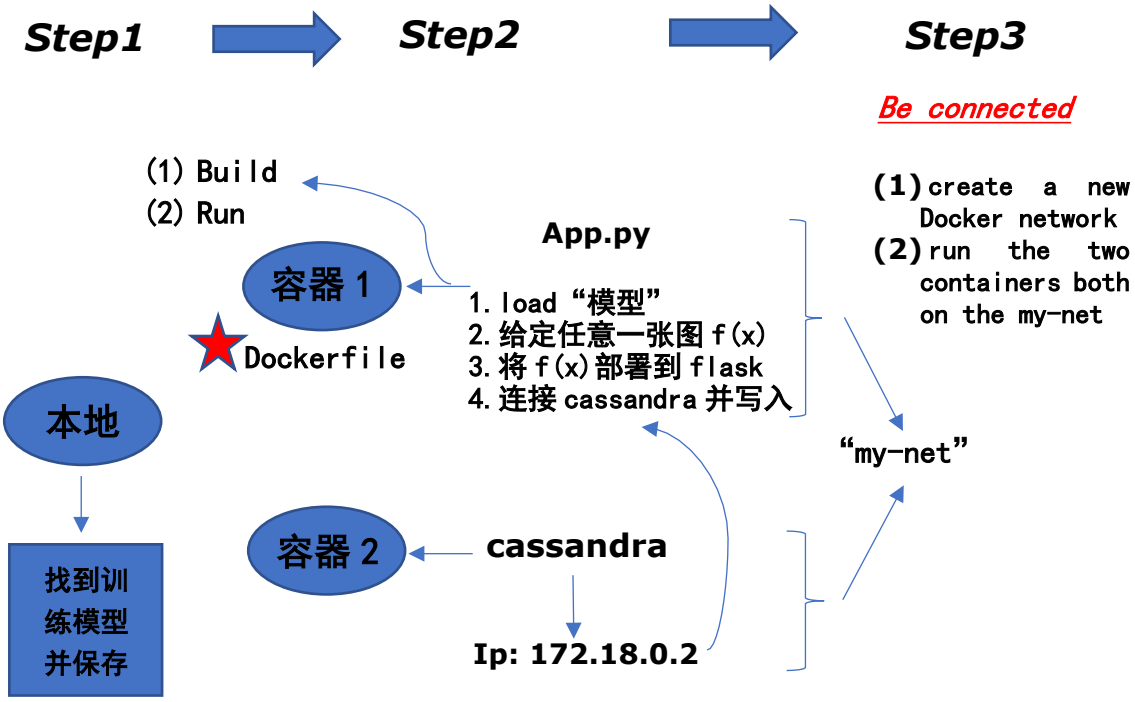
https://datastax.github.io/python-driver/

2.4.2 Cqlsh

cql 语句创建 keyspace, table，写入以及查询数据。

http://abiasforaction.net/a-practical-introduction-to-cassandra-query-language/

第三章 项目步骤



图片描述

文字描述：

3.1 本地

3.1.1 找到训练模型并保存

3.2 flask 预测服务+写入 cassandra（app.py）

3.2.1 load 模型

3.2.2 给定任意一张图 f(x)，并进行相应的格式处理

3.2.3 将 f(x)部署到 flask

3.2.4 连接 cassandra 并写入数据

3.3 将代码在 docker 中运行

step1:create a requirements.txt file to import them
step2:Define a container with Dockerfile

```
FROM python:3.5.6
COPY . /
WORKDIR /
EXPOSE 5000
RUN pip install -r requirements.txt
ENTRYPOINT ["python"]
CMD ["app.py"]
```

step3:Build and test my image with my app.py code

```
docker build -t web:latest .
```

step4:Run

3.4 docker 和 cassandra 通信

Step1: Let's first create a new Docker network

```
docker network create my-net
```

Step2: Run my image as a container and connected container

```
docker run -it -d --name app --network my-net -p 5000:5000 web:latest
docker run --name cassandra -p 9042:9042 --network my-net -d
cassandra:latest
```

*检查是否连接成功：
分别进入两个容器，输入 ping <container>

```
docker exec -it cassandra cqlsh
docker exec -it app /bin/bash
root@119ab8db4e0e:/app# ping Cassandra
```

看到数据更新即成功

第四章 程序设计

4.1 本地

4.1.1 找到训练模型并保存

```
model.fit(train_images, train_labels, epochs=10)
model.save('my_model.h5')
```

4.2 flask 预测服务+写入 cassandra（app.py）

4.2.1 load 模型

```
new_model = keras.models.load_model('my_model.h5')
```

4.2.2 给定一张图 f(x)，并进行格式处理

```
def ImageToMatrix(filename):
    im = Image.open(filename)    #获取图像
    im = im.resize((28, 28))    #转换图像尺寸
    width,height = im.size     #获取图像宽高，即（28,28）
    im = im.convert("L")    #转成灰度图
    data = im.getdata()    #将图像以 ImagingCore 形式赋给 data
    data = np.reshape(data,(height,width))    #转成矩阵
    data=255-data     #反色
    data = data / 255.0
    return data
```

4.2.3 将 f(x)部署到 flask

```
import os
from flask import Flask, request, redirect, url_for
from werkzeug.utils import secure_filename
UPLOAD_FOLDER = './images'
```

```
#允许的文件扩展名
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 a empty part without filename
        if file.filename == '':
            flash('No selected file')
            return redirect(request.url)
        if file and allowed_file(file.filename):
            # Grab an image from the test dataset.
            filename = secure_filename(file.filename)
            data = ImageToMatrix(file)
            # 概率模型
            probability_model = tf.keras.Sequential([new_model,
            tf.keras.layers.Softmax()])
            # Add the image to a batch where it's the only member.
            data = (np.expand_dims(data, 0))
            # Predict the correct label for this image
            predictions_single = probability_model.predict(data)
            def insertTable(): # 插入表置信度最高的的标签值
           {……}
            insertTable();
            return redirect(url_for('uploaded_file', filename=filename))
    return '''
    <!doctype html>
    <title>Upload new File</title>
    <h1>Welcome to the Picture Recognition System</h1>
    <form method=post enctype=multipart/form-data>
    <p><input type=file name=file>
```

```
      <input type=submit value=Upload>
    </form>
    '''

#返回该名称的文件
from flask import send_from_directory
@app.route('/images/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

4.2.4 连接 cassandra

```
import logging
from cassandra.cluster import Cluster
from cassandra.query import SimpleStatement

log = logging.getLogger()
log.setLevel('INFO')
handler = logging.StreamHandler()
handler.setFormatter(logging.Formatter("%(asctime)s
[%(levelname)s] %(name)s: %(message)s"))
log.addHandler(handler)
KEYSPACE = "mykeyspace2"

#创建 KeySpace 和 table
def createKeySpace():
    cluster = Cluster(contact_points=['172.18.0.2'], port=9042)
    session = cluster.connect()
    #session.execute('drop keyspace mykeyspace1;')
    log.info("Creating keyspace...")
    try:
        session.execute("""
          CREATE KEYSPACE %s
          WITH replication = { 'class': 'SimpleStrategy',
                               'replication_factor': '2' }
          """ % KEYSPACE)

        log.info("setting keyspace...")
        session.set_keyspace(KEYSPACE)
```

```
        log.info("creating table...")
        session.execute("""
           CREATE TABLE mytable (
               time varchar,
               name varchar,
               value varchar,
               PRIMARY KEY (time, name)
           )
           """)
    except Exception as e:
        log.error("Unable to create keyspace")
        log.error(e)
    cluster.shutdown()

#插入表置信度最高的的标签值
def insertTable():
                cluster = Cluster(contact_points=['172.18.0.2'], port=9042)
                session = cluster.connect()
                session.set_keyspace(KEYSPACE)
                start = datetime.datetime.now()
                session.execute('insert into mytable (time, name, value) '
                                'values (%s, %s, %s);',
                                [str(start), filename,
                                class_names[np.argmax(predictions_single)]])
                rows = session.execute("select * from mytable;")
                for row in rows:
                    print(row)
createKeySpace();
insertTable();
```

第五章 结果分析

1. Visit my application at 127.0.0.1:5000 or localhost:5000.
2. Choose the file you want to predict and upload it.
3. Enter the cassandra database to make sure the result has been already in there. Use cqlsh language to inquire:

   docker exec -it cassandra cqlsh
   Connected to Test Cluster at *127.0.0.1*:9042.
   [cqlsh 5.0.1 | Cassandra 3.11.6 | CQL spec 3.4.4 | Native protocol v4]
   Use HELP for help.
   cqlsh> use mykeyspace8;
   cqlsh:mykeyspace8> Select * from mytable;

```
time                            | name        | value
--------------------------------+-------------+---------
2020-04-17 15:47:33.149668 | Sneaker.jpg | Sneaker
(1 rows)
```

这里我之前发现一个问题：就是我的 cassandra 的 ip 地址已经改为 `172.18.0.2`，但是当我进入容器是仍然显示连接的是本地的容器。

其实，因为 `172.18.0.2` 这个 ip 地址是从容器里面获得的，就好比你在屋里看到这个屋的 ip 地址 `127.0.0.1`，你在外面看到这个屋的 ip 就是一个别的。既然 `127.0.0.1` 是本地的 ip，那么我在本地看到的就是 `127.0.0.1`。

第六章 主要问题和解决

6.1 将代码放在容器中运行

*问题：当我想修改容器里面的代码时，应该怎么办？

*解决：

方法一：运用 docker cp 命令将本地文件拷到容器中

```
docker cp C:\PY\fashion\docker\app1.py app:/app      #docker cp
docker exec -it app /bin/bash            #进入 app 容器
root@119ab8db4e0e:/app# ls -l        #查看 app 工作目录下文件的详细信息，可以
                                          看到我新拷贝进去的文件的修改时间
total 1244
-rwxr-xr-x 1 root root      134 Apr   4 16:56 Dockerfile
drwxr-xr-x 2 root root     4096 Apr   3 06:54 __pycache__
-rwxr-xr-x 1 root root     3101 Apr   4 17:34 app1.py
drwxr-xr-x 2 root root     4096 Apr   4 16:44 images
-rwxr-xr-x 1 root root     1128 Apr   7 03:10 log.py
-rwxr-xr-x 1 root root 1247680 Mar 29 02:14 my_model.h5
-rwxr-xr-x 1 root root      979 Apr   4 05:19 requirements.txt
root@119ab8db4e0e:/app# cat log.py      #在终端查看文件 log.py，看是否已修改
```

方法二：只好再 build 一遍

6.2 容器通信

*问题：

当我的 app.py 第一次尝试来连接 cassandra 时，无论如何都连接不上，不管是本地连接还是容器连接。

*解决：

找了半天发现是装 cassandra 的容器运行起来了但我的 cassandra 却没有运行。

遇到问题要懂得运用以下命令来寻找错误：

Docker logs	查看容器运行日志，能够详细看到容器运行的进程或者错误信息
Docker inspect	查看容器详细信息（例如查看 ip 地址、网络）
Docker ps	查看正在运行的容器，容器的端口映射
Docker kill	停止一个正在运行的容器
Docker rm	移除一个容器

6.3 写入 cassandra 数据库

*问题：

起初当我看到数据已经写入 table 中，我便觉得已经可以了。但当我一次又一次测试的时候发现，我写的程序只能允许一条数据的写入，但当我想要预测很多条数据该怎么办呢？

*解决：

于是我发现我之前是把 create 和 insert 写入到一个函数了，所以以后每当我想要再写入一条数据时，我必须要再创建一个新的 keyspace 和 table。所以我将 create 和 insert 写成两个函数，每次只用调用 insert 就可以了。

6.4 Cassandra 中登录的时间差异问题

*问题：

终端的登录时间和北京时间相差七个小时。

*解决：

容器使用的时间是格林威治时间，就是英国的时间，因为中国是 UTC+8，所以领先容器时间八小时。

第七章 项目心得

首先，通过一个月的上课和学习，收获了很多知识：

1. 虚拟化的容器技术：docker
2. 服务部署->调用他人服务->暴露服务给他人调用

(1) 服务部署：flask

http://flask.pocoo.org/docs/0.12/quickstart/#a-minimal-application

(2) 调用他人服务：RESTful API

https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

(3) 将服务放在容器里面运行

https://docs.docker.com/get-started/

3. 大数据存储技术 mongoDB + Cassandra(noSQL)

(1) https://hub.docker.com/_/cassandra/

(2) Python 代码连接 cassandra，创建 keyspace，table，写入并查询。

(3) Cassandra-driver https://datastax.github.io/python-driver/

(4) Cqlsh
http://abiasforaction.net/a-practical-introduction-to-cassandra-query-language/
4. 大数据处理技术 mapreduce + Hadpop + spark + fink
https://spark.apache.org/
https://spark.apache.org/docs/2.2.0/rdd-programming-guide.html
5. 大数据可视化技术
(1) amCharts:https://www.amcharts.com/
(2) d3js:https://d3js.org/
(3) playground:http://playground.tensorflow.org/
6. Python + Anaconda
(1) 刷题：LeetCode
(2) Practice: https://github.com/fanzhang15/pythoncourse
(3) Conda virtual environment
7. 人工智能算法：tensorflow tutorials
8. Github
Practice: https://try.github.io
9. 半结构化数据 Json、XML、CSV、Yaml
10. Big data
11. Cloud Native Landscape
http://redmonk.com/fryan/files/2017/04/CloudNativeLandscape_v0.9.4.jpg

其次，要知道的是，完成这个项目并不代表着真正理解透彻每个知识点，要学习的东西远远要超过这个项目所涉及到知识点的深度和广度。

然后，这次项目重要的不仅仅是知识，更是在遇到问题的时候找到问题突破口的思路和思考方法，有的可以是自己在网上搜索大量的相似问题；有的可以是自己不断试错，定位到具体出错的代码，思考可能是什么原因会导致这个错误，不断排除缩小范围；还有的可以是向老师请教，不仅可以帮助你解决问题，更重要的是学会解决问题的思路，学习到很多新的方法和技巧。

最后，我要感谢帮助过我的老师和项目伙伴，他们让我学到的是知识，更是如何解决应对不同的问题和珍贵的科研技能。