算法部署

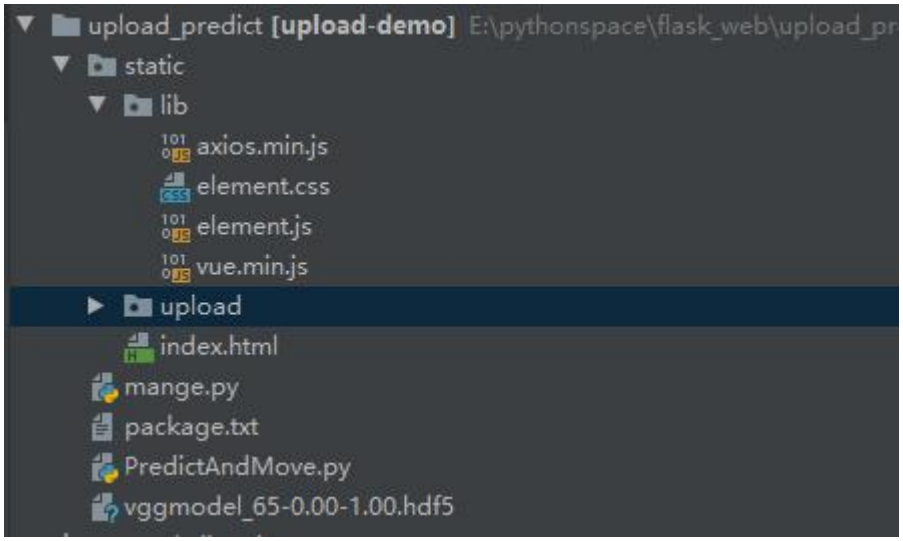
➢ 环境配置

系统环境：Ubuntu16.04 LTS、python3.x、Anaconda3 和相关的编辑器
所需 python 库：numpy、tensorflow、Keras、opencv、Flask 及相关依赖

➢ 必备知识

编辑器:Pycharm 和 HBuilder X
后端：python3.x、Flask
前端：Vue、Element、axios

➢ 项目结构




```
|——static      #静态资源目录，此处为存放前端上传的图片
        |——lib          #css、js 文件存放目录
        |——upload       #图片上传后的保存目录
        |——index.html  #前端页面代码
|——mange.py    #Flask 启动文件（主文件）
|——package.txt #记录运行此项目所需要的依赖包及版本信息(使用 pip
freeze>package.txt)命令打包
|——PredictAndMove.py  #封装后的模型预测文件
|——vggmodel_65-0.00-1.00.hdf5 #hdf5 模型文件
```

Flask 非常灵活，官方并没有给出一个固定的项目目录组织结构，此处的目录结构仅为个人习惯以及项目调用使用，也更加符合大型项目结构的工程化构建方法。

➢ 模型保存（以 keras 为例）

保存 Keras 模型有很多种方法，首先不推荐使用 pickle 或 cPickle 来保存 Keras 模型，此处使用 *model.save()*函数将 Keras 模型和权重保存在一个 HDF5 文件中。

HDF5 *模型保存结构*

- 模型的结构，刹车改造该模型
- 模型的权重
- 训练配置（损失函数，优化器等）
- 优化器的状态，煞于从上次训练中断的地方开始

➢ 前端页面

该项目采用前后端分离的架构，前端主要采用 Vue 和 Element，使用 axios 进行前后端数据的交互。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>vue上传demo</title>
        <link rel="stylesheet" href="./static/lib/element.css">
        <script type="text/javascript" src="./static/lib/vue.min.js"></script>
        <script type="text/javascript" src="./static/lib/element.js"></script>
        <script type="text/javascript" src="./static/lib/axios.min.js"></script>
    </head>

    <body>
        <div id="app">
            <input type="file" style="display: none;" ref="input" @change="fileChange" multiple= "multiple"></input>
            <el-button type="primary" round @click="handleClick" v-if="upload_awesome">上传图片</el-button>
            <div v-for="(file, index) in imageData">
                <el-tag type="success" style="margin:5px 5px;display: flex;justify-content:center;">识别结果为:{{index}}</el-tag>
                <el-row>
                    <el-col :span="6" v-for="img in file">
                        <div class="img-box" style="width: 100%;">
                            <img :src="img.path">

                        </div>
                    </el-col>
                </el-row>
            </div>
        </div>
    </body>
```

为保证项目可以正常运行，建议参照官方文档下载和安装完整的 Vue、Element、和 axios 包文件。这里为了压缩项目文件大小，所以只引入了用到的文件，并不是很好的做法。
官方文档地址：
Vue: https://cn.vuejs.org/v2/guide/
Element: https://element.eleme.cn/#/zh-CN/component/installation
Axios: http://www.axios-js.com/zh-cn/docs/#axios

1：必须 HTML 头部(head)使用 link 标签引入 element 的 css 样式文件，script

标签分别引入 vue.min.js、element.js、axios.min.js 文件。
2：页面主体使用 element 的 el-button 组件编写上传图片按钮，el-tag 组件渲染模型的识别结果，el-col 和 el-row 组件控制图片显示的布局风格。
3：使用@符号绑定了事件函数，用来在 JavaScript 中调用这些函数进行逻辑处理。

```
<script type="text/javascript">
    var vm = new Vue({
        el: '#app', // 创建Vue实例
        data: { //定义数据对象
            upload_awesome: true,
            imageData: {},
            file: '',
            retData: ''
        },
        methods: { //事件处理器，this将自动绑定到Vue实例上面

            handleClick() {
                this.$refs['input'].click();
            },
            fileChange(e) {
                this.file = e.target.files[0];
                this.uploads()
            },
            uploads() {
                let self = this;
                let param = new FormData();
                param.append('file', this.file);
                axios.post('/upload/', param, {}).then(function(response) {
                        if (response.status === 200) {
                            let data = response.data;
                            self.$set(self.$data, 'imageData', data)
                            self.$set(self.$data, 'upload_awesome', false)
                        }
                    })
                    .catch(function(error) {
                        console.log("上传图片失败!")
                    })
            },
        }
    })
</script>
```

这里编写前端逻辑代码，包括上传和展示，下面分别介绍自定义的三个函数 fileChange()、handleClick()和 uploads()。

handleClick()：

该函数主要是点击上传按钮之后，动态的去改变 input 属性，从而触发

fileChange()：

使用@change 对 input 的值进行监听,如果监听到值的改变，就截取上传文件的 name，并调用 upload()函数

uploads()：

该函数为主函数，即是通过该函数与服务器通信，以获取、交换数据。下面对该函数的核心代码进行解读

```
param.append('file', this.file);
```

将上传的文件信息加入到 param 变量

axios.post():
axios 中 post 请求实例方法的别名，创建该请求的基本配置选项如下
axios.post(url[, data[, config]])，即请求路径(url)、数据、和配置，只有 URL 是必需的，详情请访问下面网址
http://www.axios-js.com/zh-cn/docs/#axios-options-url-config-1
then 函数则接受该请求的响应信息，结构如下：

某个请求的响应包含以下信息

```
{
  // `data` 由服务器提供的响应
  data: {},

  // `status` 来自服务器响应的 HTTP 状态码
  status: 200,

  // `statusText` 来自服务器响应的 HTTP 状态信息
  statusText: 'OK',

  // `headers` 服务器响应的头
  headers: {},

   // `config` 是为请求提供的配置信息
  config: {},
 // 'request'
  // `request` is the request that generated this response
  // It is the last ClientRequest instance in node.js (in redirects)
  // and an XMLHttpRequest instance the browser
  request: {}
}
```

通过 if (response.status === 200) {}
判断响应状态，并更新相应的数据，同时视图会进行重渲染，这是 vue 的特性。
在使用 catch 时，或传递拒绝回调作为 then 的第二个参数时，响应可以通过 error 对象可被使用，主要作用是处理错误。

#源代码详见 index.html 文件

➢ 服务器端(flask 后端)

项目涉及到 flask 的内容较少，考虑到实用性的问题，此处并没有对核心代码进行路由、蓝图规划和拆分。

```
import os
import json
from flask import Flask, request, jsonify
from werkzeug.utils import secure_filename
```

导入 python 的第三方库文件，以及 flask 用到的依赖库文件

```python
# 文件上传
@app.route('/upload/', methods=['POST'])
def upload():
    if request.method == 'POST':
        f = request.files['file']
        if not (f and allowed_file(f.filename)):
            return jsonify({"error": 1001, "msg": "请检查上传的图片类型，仅限于jpg,jpeg,png,gif"})
        upload_path = os.path.join(path, secure_filename(f.filename))
        f.save(upload_path)

        # 图片展示
        files = os.listdir(path)
        fileList = {}
        for file in files:
            file_d = os.path.join(path, file)
            # 执行模型脚本
            res = os.popen("python ./PredictAndMove.py %s" % file_d)
            labels = res.read()
            label = str(labels).strip('\n')
            if label in fileList.keys():
                fileList[label].append({"filename": file, "path": file_d})
            else:
                fileList[label] = [{"filename": file, "path": file_d}]
            # 将字典形式的数据转化为字符串
    return json.dumps(fileList)
```

定义全局配置、判断后缀名函数以及路由注册
配置信息是以键值对的格式书写，判断文件格式的函数中使用 rsplit()方法通过指定分隔符对字符串进行分割并返回一个列表，路由注册通过 render_template()方法渲染前端模板（index.html）,flask 会在 templates 文件夹内寻找模板文件

该模块为该项目后端的核心模块，后端整体逻辑为：
保存前端上传的图片-->执行模型预测文件-->返回 json 数据给前端

```
  File "E:\pythonspace\flask_web\envs\lib\site-packages\flask\app.py", line 1820, in handle_user_exception
    reraise(exc_type, exc_value, tb)
  File "E:\pythonspace\flask_web\envs\lib\site-packages\flask\_compat.py", line 39, in reraise
    raise value
  File "E:\pythonspace\flask_web\envs\lib\site-packages\flask\app.py", line 1949, in full_dispatch_request
    rv = self.dispatch_request()
  File "E:\pythonspace\flask_web\envs\lib\site-packages\flask\app.py", line 1935, in dispatch_request
    return self.view_functions[rule.endpoint](**req.view_args)
  File "E:\pythonspace\flask_web\upload-demo\app.py", line 36, in upload
    f.save(upload_path)
  File "E:\pythonspace\flask_web\envs\lib\site-packages\werkzeug\datastructures.py", line 2800, in save
    dst = open(dst, "wb")
FileNotFoundError: [Errno 2] No such file or directory: './static/upload/40410814_fba3837226_n.jpg'
```

补充：若项目运行出现这种报错，是因为路径的问题所导致，建议使用 Ubuntu 系统运行 flask 程序，或者将用到的路径全部更换为绝对路径

➢ 运行程序

打开终端输入 python app.py

```
zy@ubuntu-B365M-DS3H:~/project/upload-demo$ ls
app.py  PredictAndMove.py  static  vggmodel_65-0.00-1.00.hdf5
zy@ubuntu-B365M-DS3H:~/project/upload-demo$ python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [12/Dec/2019 15:06:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [12/Dec/2019 15:06:00] "GET /static/lib/element.css HTTP/1.1" 304
-
127.0.0.1 - - [12/Dec/2019 15:06:00] "GET /static/lib/vue.min.js HTTP/1.1" 304 -
127.0.0.1 - - [12/Dec/2019 15:06:00] "GET /static/lib/element.js HTTP/1.1" 304 -
127.0.0.1 - - [12/Dec/2019 15:06:00] "GET /static/lib/axios.min.js HTTP/1.1" 304
-
```

此时服务器即为启动状态，打开浏览器，通过 127.0.0.1:5000 或者 ip:5000 的方式访问网页，上传图像