系统设计文档

# 1 引言

## 1.1设计目的

本文档将原本复杂的功能拆分为不同的模块，并逐步开发健壮的系统结构。使软件系统需求转化为未来设计，并进一步提高软件各方面的性能。本文档依据高内聚、低耦合的原则，将结构分解为不同的模块，各模块分别负责自己的功能，且模块与模块之间，模块与环境之间的关系尽可能简单。

本文档旨在将软件系统功能进一步细化，说明系统总体设计的技术方案，从程序系统的设计考虑，包括系统的基本处理流程、模块划分、功能分配、接口设计、运行设计、数据结构设计和出错处理设计等内容，以向整个设计期提供关于程序系统逻辑和数据功能实现方式的总体描述，从而作为程序详细设计或编码的基础。设计阶段将本文档为核心文档。

本文档的目标读者为本软件的开发人员、测试人员和其他相关人员。

## 1.2 参考资料

1.微信开发者文档

## 1.3 假定和约束

### 1.3.1 约束

l 发布时间：2021.6.18

l 开发条件：开发成员自备开发设备。

l 经费支持：无经费支持

l 设计成本：开发人员共同承担设计成本

l 开发效率：采用极限开发模型，开发期间团队成员结伴开发，选择宽敞明亮的图书馆。团队相互协作，每位成员都熟悉开发过程中涉及到的各项技术，保证项目的顺利推进以及每个迭代都朝着预定的目标并达到最⾼的质量。

l 小组合作：小组采用线上线下结合的方式，项目需求确立和项目设计通过线下沟通确立，项目相关分工通过conerstone确立分发，项目相关文档通过word合作完成，最大化利用成员的时间，保证开发效率。

l 代码托管：为了充分践⾏软件⼯程的思想，流程化、平台化地管理我们的整个项目，我们使用 Github 托管我们代码，每个成员在各自独立的分支上开发，开发完成之后各个分支间合并，产⽣新版本的代码。每次更新之后，最新的版本都将部署在服务器上。

### 1.3.2 术语表

l 垃圾减量分类：垃圾减量是指在产品设计、制造、流通和消费等过程中采用合理措施减少固体废物量，如避免过度包装、净菜进城、禁止使用一次性筷子等。垃圾分类是指按垃圾的不同成分、属性、利用价值以及对环境的影响，并根据不同的处置方式，分成属性不同的若干种类。

l 垃圾分类类别：居民小区垃圾分为可回收物、厨余垃圾、有害垃圾和其他垃圾四类；酒楼、宾馆、饭堂等餐饮场所的垃圾分为可回收物、餐厨垃圾、其他垃圾三类；机场、码头、火车站、地铁站、公交站、公园、旅游景区、加油等公共场所的垃圾分为可回收物、其他垃圾两类；政府机关、企事业单位办公区域、学校等场所的垃圾分为可回收物、其他垃圾、有害垃圾三类。可回收物可进一步细分为纸张类、饮料瓶罐类、其他塑料类。

l 可回收物：指回收后经过再加工可以成为生产原料或经过整理可以再利用的物品，主要包括废纸类、塑料类、金属类、织物类、电子废弃物类等。

l 厨余（餐厨）垃圾：指在食品加工和消费过程中产生的剩饭剩菜、菜帮菜叶、瓜果皮核、废弃食物、废弃食用油脂等易腐的垃圾。居民家庭产生的为厨余垃圾，酒楼、宾馆、饭堂等餐饮场所产生的为餐厨垃圾。

l 有害垃圾：指存在对人体健康有害的重金属、有害的物质或者对环境造成危害或者潜在危害的废弃物。包括电池、日光灯管、水银温度计、药品等。

l 其他垃圾：除上述可回收物、厨余（餐厨）垃圾、有害垃圾之外的垃圾。主要包括废弃食品袋（盒）、废弃保鲜膜、废弃纸巾、废弃瓶罐、灰土、烟头、宠物粪便等。

### 1.3.3 背景与依据

近年来，随着我国经济的快速发展，国家各项建设都蒸蒸日上，成绩显著。但与此同时，也让资源与环境受到了严重破坏。这种现象与垃圾分类投放时的不合理直接相关。人们日常生活中的垃圾主要包括有害垃圾、厨余垃圾、可回收垃圾以及其他垃圾这四类。合理地进行垃圾分类是有效进行垃圾处理、减少环境污染与资源再利用中的关键举措。

而对垃圾的分类首先是在图像识别的基础上的，因此我们的微信小程序想通过使用近几年来发展迅速的深度学习方法设计一个垃圾分类系统，从而实现对日常生活中常见垃圾进行智能识别分类，提高人们垃圾分类投放意识，同时避免人们错误投放而产生的环境污染。

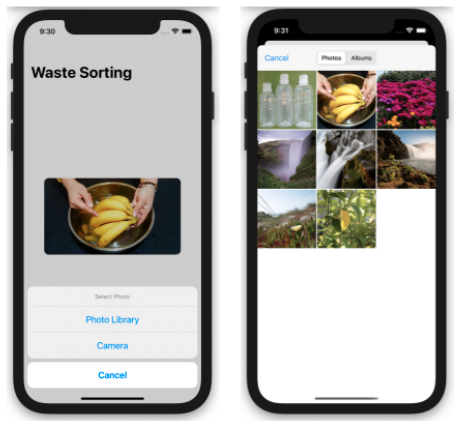
# 2 概要设计

## 2.1 系统总体架构设计

本系统整体采用了web系统常用的层次体系架构，在实际开发中，还结合了以数据为中心的架构设计，以微信云开发数据库为中心，数据作为客户端和管理员端的被访问对象。层次体系架构主要分为底层的运行环境、数据，中间主体的业务层，顶层的展示层和UI层。

## 2.2 页面设计

截图



## 2.3 系统维护设计

使用 git 进⾏版本控制，维护⼀条 master 主分支，开辟出其他3条分支，不同的开发⼈员独立地不同的分支上进⾏开发，有效避免多⼈开发导致程序版本控制的混乱，最后再merge 到 master 分支上。

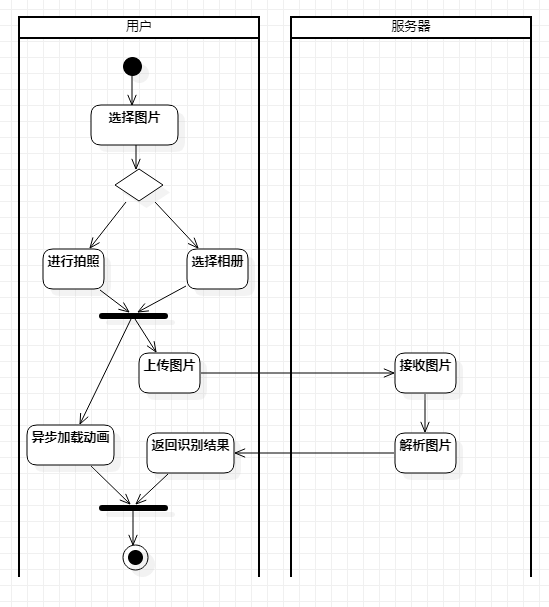
# 3 功能设计

在本项目中，将训练完成后的模型，使用FastAPI框架将后端部署到服务器中，微信小程序和ios app均通过相应接口调用完成功能。

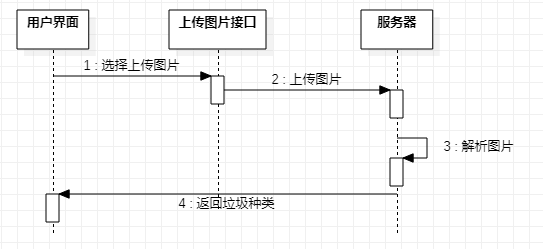
### 3.1 “垃圾图片识别”部分

#### 3.1.1 “垃圾图片识别”功能设计

活动图



**时序图**

****

|  |  |
| --- | --- |
| **动作序列** | **描述** |
| 用户上传图片 | 用户拍照或从相册选择图片，系统将图片上传至服务器。 |
| 服务器解析图片 | 服务器后端根据已训练的模型，分析图片的垃圾种类。 |
| 返回垃圾种类信息 | 向用户端返回分析结果，连同实际上传图片显示在小程序上。 |

#### 3.1.2 “垃圾图片识别”功能实现

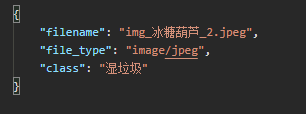


#### 3.1.3 “垃圾图片识别”接口描述

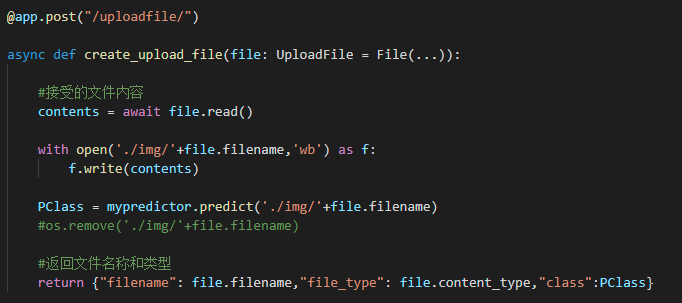
发送数据

|  |  |
| --- | --- |
| 上传图片 | /uploadfile/ |
| 方式 | POST |
| 端口 | 80 |
| 介绍 | 用户上传图片，如果上传成功返回Success以及图片识别结果，如果失败返回Error\_type |
| 参数 | [FormData]{  file : data  } |

返回数据



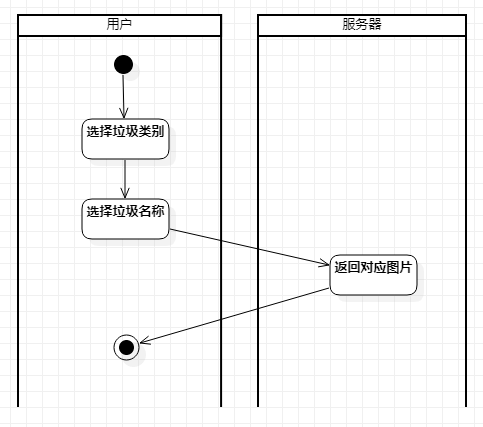
接口代码



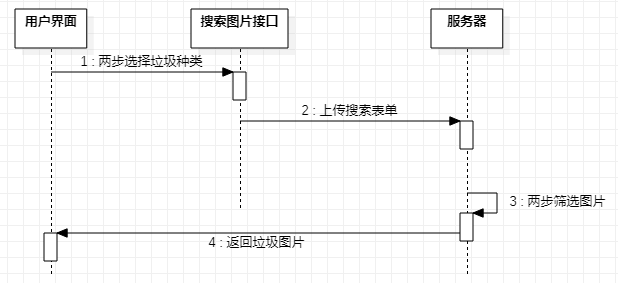
### 3.2 搜索垃圾图片

#### 3.2.1 “搜索垃圾图片”功能设计

活动图



时序图



|  |  |
| --- | --- |
| **动作序列** | **描述** |
| 用户选择垃圾类别（四选一） | 用户通过拖拉选单选择垃圾主要类别 |
| 用户选择垃圾名称（具体种类） | 用户点选垃圾名称 |
| 返回垃圾图片 | 服务器向用户端返回垃圾图片List，显示在小程序上。 |

#### 3.2.2 “搜索垃圾图片”功能实现



#### 3.2.3 “搜索垃圾图片”接口描述

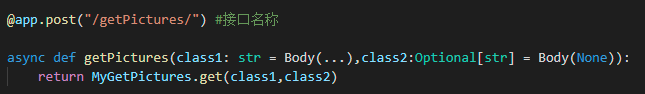
发送数据

|  |  |
| --- | --- |
| 上传图片 | /getPictures/ |
| 方式 | POST |
| 端口 | 81 |
| 介绍 | 用户选择垃圾种类（四大类），和具体垃圾名称，"class2"存在时服务器返回垃圾名称所对应的图片，"class2"不存在时服务器返回垃圾大类所涵盖的垃圾名称 |
| 参数 | {  "class1":"可回收垃圾",  "class2":"电风扇"  } |

返回数据



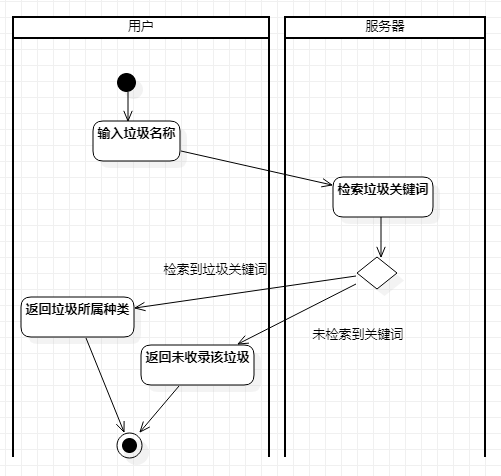
接口代码



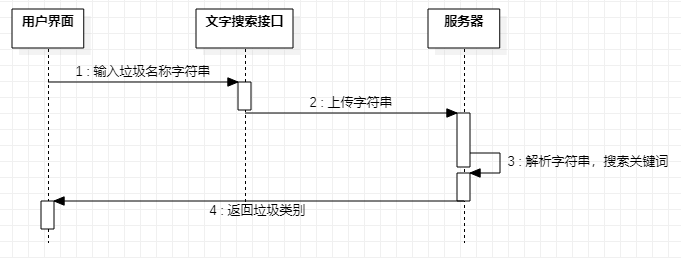
### 3.3 搜索垃圾类别

#### 3.3.1 “搜索垃圾类别”功能设计

活动图



时序图



|  |  |
| --- | --- |
| **动作序列** | **描述** |
| 用户输入垃圾名称 | 用户通过文字输入待查询垃圾的名称 |
| 服务器检索 | 服务器对用户输入文字串进行关键字解析 |
| 返回垃圾种类 | 服务器向用户端返回垃圾种类 |

#### 3.3.2 “搜索垃圾类别”功能实现



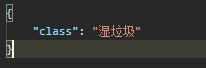


#### 3.3.3 “搜索垃圾类别”接口描述

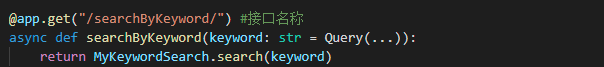
发送数据

|  |  |
| --- | --- |
| 上传图片 | /searchByKeyword/ |
| 方式 | GET |
| 端口 | 81 |
| 介绍 | 用户通过文字输入待查询垃圾的名称，服务器对用户输入文字串进行关键字解析，如果后台存在关键词则返回垃圾所属分类，否则返回“数据库未收录该种垃圾” |
| 学参数 | {  "keyword":"草莓"  } |

返回数据



接口代码



# 4 模型分析

## 4.1 模型概述

PyTorch是一个开源的Python机器学习库。Pytorch 底层和Torch框架一样，但是使用Python重新写了很多内容，简洁高效，能够实现强大的GPU加速，同时还支持动态神经网络。

在本项目中，图像识别部分使用pytorch框架，先导入预训练的ResNeXt101\_32x16d\_wsl卷积神经网络模型，模型一共定义五层，其中两层卷积层，两层池化层，最后一层为FC层进行分类输出。再经过微调得到我们需要的模型，然后用搜集到的数据集在GPU中训练模型。模型正确率在训练集中达到93%，测试集中达到81%。

训练完成后，使用FastAPI框架将后端部署到服务器中，通过文档中的接口调用。

## 4.2 模型训练

#### 4.2.1 制作需要的数据集

由于干垃圾和有害垃圾数据集较匮乏，以训练集为例，有害垃圾约有8500张，干垃圾约有9200张，远小于可回收垃圾和湿垃圾的数据集数量（这可能是因为不同国家垃圾分类政策不同，如日本分五大类、英国分三大类，大多说国家都有厨余垃圾和可回收垃圾两类，而其他垃圾分类细则却不甚统一，所以在网上较难找到相对应的开源数据集）。这样训练之后的模型，在面对有害垃圾和干垃圾图片时，正确率明显偏低。在经过一段时间的训练后，在测试集中正确率仅有72%左右，即使其中也有训练时间较短的原因。我们的解决方案是，用python爬虫从百度爬取这两类的一些图片，并将这两类垃圾图片进行图像增强，达到整体平衡的效果。最终有效地提高了模型总体正确率。

在搜集到数据集之后，需要创建数据集的索引txt文件。根据读取的文件夹目录，将所有图片一一对应在train.txt和test.txt文件中生成其相对路径。对于文件名中的空格，用下划线替换。遇到损坏的图片，则跳过。

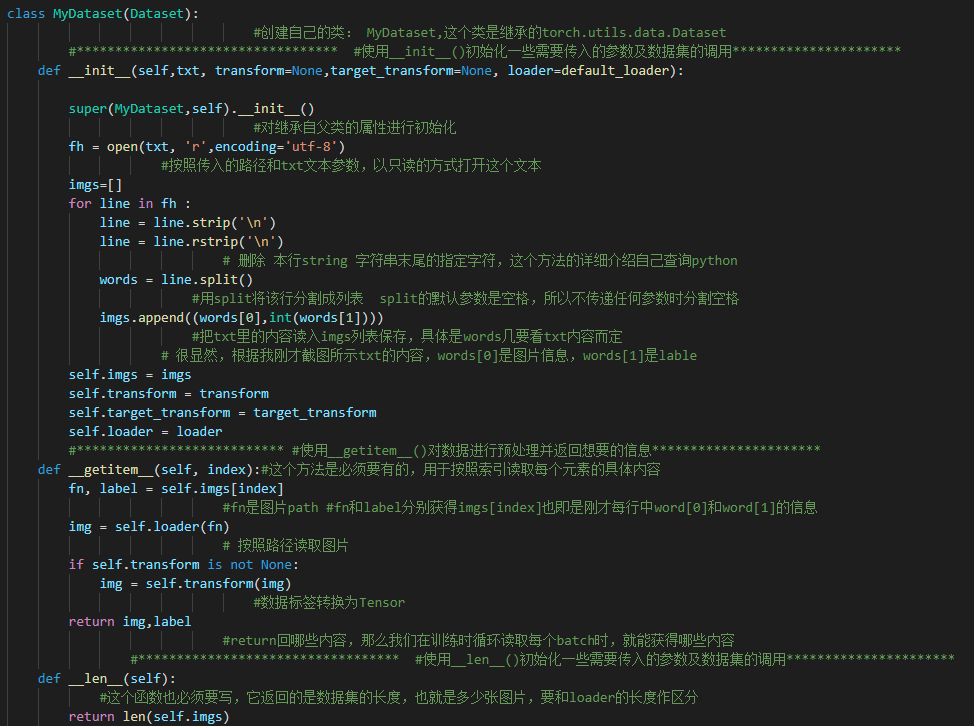




最终清洗得出的数据集，训练集数据集达到135000张，测试集数据集达到13800张。

#### 4.2.2 使用MyDatasets加载数据集准备训练

继承自Dataset类



#### 4.2.3 使用CUDA架构将模型迁移到GPU进行学习

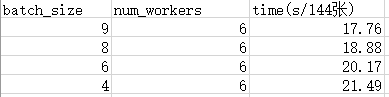
在模型训练过程中，我们着重于在避免超出显存的前提下，不断改变num\_works和batch\_size两个参数的值，以求达到最快训练速度。

关于batch\_size：表示每个minibatch的数据量，当batchsize过大时，会超出显存，过小则不能完全发挥显卡的性能。

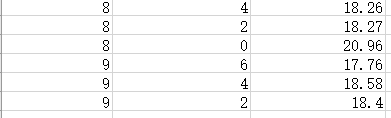
关于num\_workers：每个worker通过主进程获得自己需要采集的图片id。然后每个worker采集一个batch的数据存放在内存中。主进程收集完最后一个worker的batch。此时需要回去收集第一个worker产生的第二个batch。如果该worker此时没有采集完，主线程会卡在这里等待。

理论上，在不超出CPU占用的情况下，workers越多越好。但在实际中，我们发现num\_wokers需要和batchsize匹配。

在num\_workers均为6的情况下，batch\_size为9的时候训练144张图片耗时最短：



在batch\_size为8或9的情况下，改变num\_workers训练144张图片耗时如下所示：



通过这次实验，我们得出训练参数的最优解为batch\_size=9,num\_workers=6的组合，次优解为batch\_size=8,num\_workers=4的组合。但是考虑到显卡占用率过高的电脑散热问题，以及在个人电脑上进行训练的折旧问题，我们选择了batch\_size=8,num\_workers=4这一组合进行剩余时间的训练。