

**《软件项目管理》课程项目报告**



题 目  **林业害虫识别**

学 院  **软件学院**

专 业  **软件工程**

学生姓名  **刘桐源**

学 号  **2018141501067** 年级 **2018级**

二Ο二一 年 六 月

1 绪论

* 1. 项目背景

在我国社会经济快速发展的背景下，生态保护越来越重要，而林业有害生物防治工作在推动林业的可持续发展方面有非常重要的作用。

根据软件项目管理课程要求，本项目致力于通过图像识别进行智能分析，从而对林业有害生物进行侦查来促进林业生态环境建设。

1.2 项目的主要工作

项目的主要工作是搭建一个基于微信小程序的害虫识别系统，它主要服务于林业管理人员，方便他们在工作中识别一些虫类，判断林区的健康状况，同时也可以由管理员管理害虫信息库，项目具备在线学习的能力。

项目的预期结果是完成一个可供使用的微信小程序。为了完成项目，需要进行以下三个阶段的工作，首先，搭建一个具备害虫目标检测和识别功能的深度学习框架，其次，完成后端与数据库系统的设计搭建，最后，完成前端系统设计和代码编写，并将其与后端系统和深度学习板块进行连接调试。

1.3 项目组成员及分工

|  |  |  |
| --- | --- | --- |
| 姓名 | 角色 | 工作描述 |
| 王心雨 | 项目管理，前期分析，设计 | 分析系统需求，项目计划，项目团队管理，进度把握 |
| 王璐瑶 | 分析，设计，编码，文档编写 | 分析需求，代码模块编写，软件使用手册编写 |
| 李萌 | 设计，编码，文档编写，测试 | 代码模块编写，软件使用手册编写，软件测试 |
| 刘桐源 | 分析，设计，编码，测试 | 需求分析，代码模块编写，软件测试 |

沟通方式如下：

|  |  |
| --- | --- |
| 沟通对象 | 团队内部成员，包括项目经理，设计和编码组，测试组所有成员 |
| 沟通方式 | 通过面对面会议，腾讯会议（网络），QQ聊天群和私聊方式开展沟通活动 |
| 协作模式 | 团队在不同时期有不同的分工，其中项目经理为王心雨同学担任，在设计阶段王璐瑶和刘桐源同学为设计和编码组成员，在编码阶段王璐瑶，李萌和刘桐源同学都为设计和编码组成员，在测试阶段团队所有成员都参与软件测试，由于团队人数较少，因此设计为扁平化管理 |
| 沟通内容 | 沟通内容包括角色临时分配，项目阶段现状沟通，测试内容沟通等 |

2 相关技术介绍（个人工作内容相关）

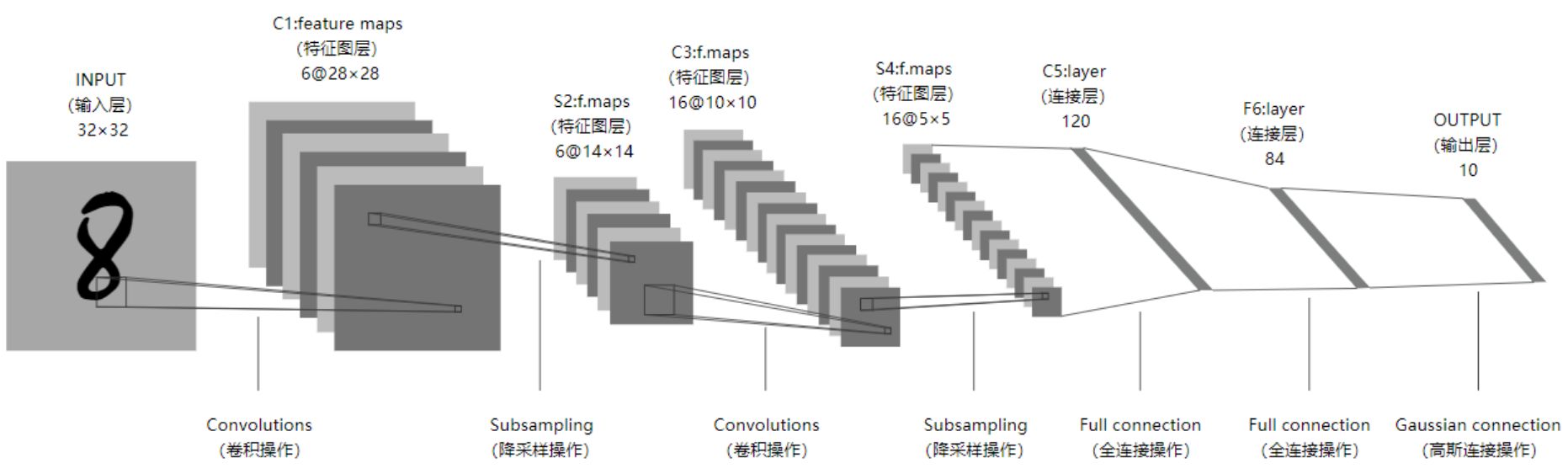
**2.1卷积神经网络**

卷积神经网络是近年发展起来的，并引起广泛重视的一种高效识别方法，20世纪60年代，Hubel和Wiesel在研究猫脑皮层中用于局部敏感和方向选择的神经元时发现其独特的网络结构可以有效地降低反馈神经网络的复杂性，继而提出了卷积神经网络（Convolutional Neural Networks-简称CNN）。现在，CNN已经成为众多科学领域的研究热点之一，特别是在模式分类领域，由于该网络避免了对图像的复杂前期预处理，可以直接输入原始图像，因而得到了更为广泛的应用。 K.Fukushima在1980年提出的新识别机是卷积神经网络的第一个实现网络。随后，更多的科研工作者对该网络进行了改进。其中，具有代表性的研究成果是Alexander和Taylor提出的“改进认知机”，该方法综合了各种改进方法的优点并避免了耗时的误差反向传播。

这听起来像是一个奇怪的生物学和数学的结合，但是这些网络已经成为计算机视觉领域最具影响力的创新之一。2012年是神经网络成长的第一年，Alex Krizhevsky用它们赢得了当年的ImageNet竞赛（基本上是计算机视觉年度奥运会），把分类错误记录从26％降到了15％，这个惊人的提高从那以后，许多公司一直在以服务为核心进行深度学习。Facebook使用自动标记算法的神经网络，谷歌的照片搜索，亚马逊的产品推荐，Pinterest的家庭饲料个性化和Instagram的搜索基础设施。

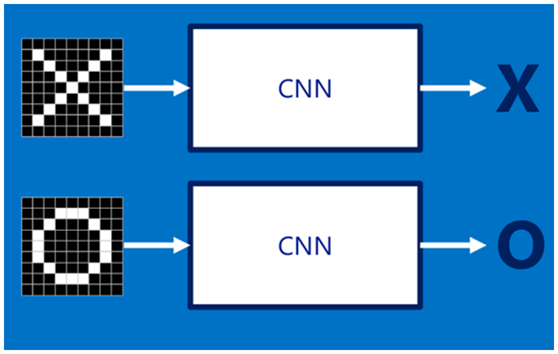
一般的，CNN的基本结构包括两层，其一为特征提取层，每个神经元的输入与前一层的局部接受域相连，并提取该局部的特征。一旦该局部特征被提取后，它与其它特征间的位置关系也随之确定下来；其二是特征映射层，网络的每个计算层由多个特征映射组成，每个特征映射是一个平面，平面上所有神经元的权值相等。特征映射结构采用影响函数核小的sigmoid函数作为卷积网络的激活函数，使得特征映射具有位移不变性。此外，由于一个映射面上的神经元共享权值，因而减少了网络自由参数的个数。卷积神经网络中的每一个卷积层都紧跟着一个用来求局部平均与二次提取的计算层，这种特有的两次特征提取结构减小了特征分辨率。

CNN主要用来识别位移、缩放及其他形式扭曲不变性的二维图形，该部分功能主要由池化层实现。由于CNN的特征检测层通过训练数据进行学习，所以在使用CNN时，避免了显式的特征抽取，而隐式地从训练数据中进行学习；再者由于同一特征映射面上的神经元权值相同，所以网络可以并行学习，这也是卷积网络相对于神经元彼此相连网络的一大优势。卷积神经网络以其局部权值共享的特殊结构在语音识别和图像处理方面有着独特的优越性，其布局更接近于实际的生物神经网络，权值共享降低了网络的复杂性，特别是多维输入向量的图像可以直接输入网络这一特点避免了特征提取和分类过程中数据重建的复杂度。



2.1.1

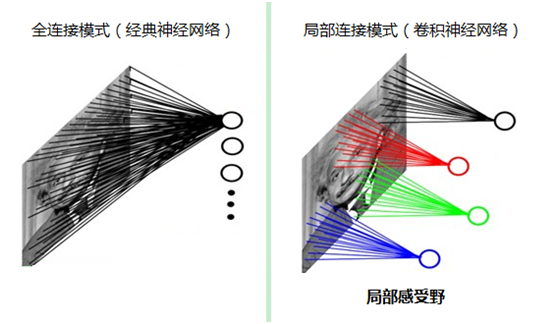
假设给定一张图（可能是字母X或者字母O），通过CNN即可识别出是X还是O，如下图所示，那怎么做到的呢



2.1.2 图像输入

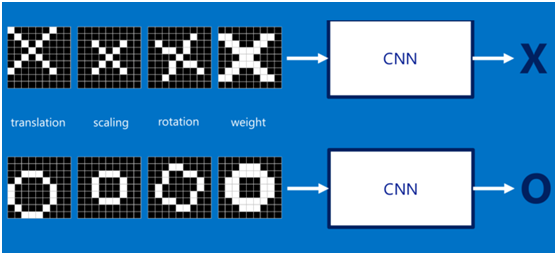
如果采用经典的神经网络模型，则需要读取整幅图像作为神经网络模型的输入（即全连接的方式），当图像的尺寸越大时，其连接的参数将变得很多，从而导致计算量非常大。

而我们人类对外界的认知一般是从局部到全局，先对局部有感知的认识，再逐步对全体有认知，这是人类的认识模式。在图像中的空间联系也是类似，局部范围内的像素之间联系较为紧密，而距离较远的像素则相关性较弱。因而，每个神经元其实没有必要对全局图像进行感知，只需要对局部进行感知，然后在更高层将局部的信息综合起来就得到了全局的信息。这种模式就是卷积神经网络中降低参数数目的重要神器：局部感受野。



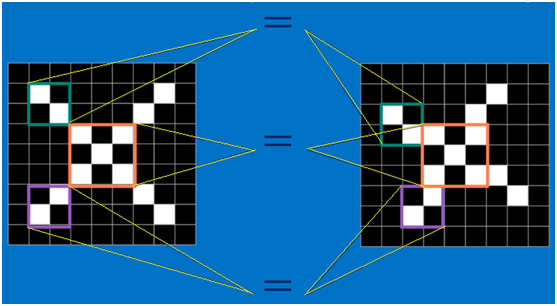
2.1.3 提取特征

如果字母X、字母O是固定不变的，那么最简单的方式就是图像之间的像素一一比对就行，但在现实生活中，字体都有着各个形态上的变化（例如手写文字识别），例如平移、缩放、旋转、微变形等等，如下图所示：

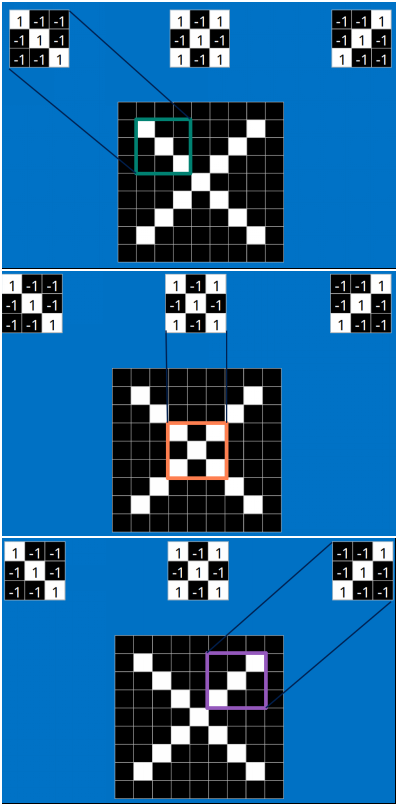


我们的目标是对于各种形态变化的X和O，都能通过CNN准确地识别出来，这就涉及到应该如何有效地提取特征，作为识别的关键因子。

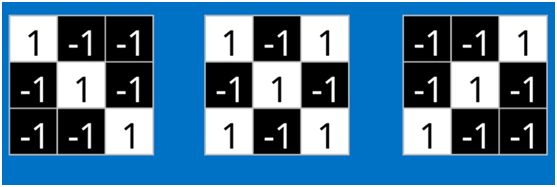
回想前面讲到的“局部感受野”模式，对于CNN来说，它是一小块一小块地来进行比对，在两幅图像中大致相同的位置找到一些粗糙的特征（小块图像）进行匹配，相比起传统的整幅图逐一比对的方式，CNN的这种小块匹配方式能够更好的比较两幅图像之间的相似性。如下图：



以字母X为例，可以提取出三个重要特征（两个交叉线、一个对角线），如下图所示：



假如以像素值"1"代表白色，像素值"-1"代表黑色，则字母X的三个重要特征如下：



那么这些特征又是怎么进行匹配计算呢？

2.1.4 卷积（convolution）

这时就要请出重要嘉宾：卷积。那什么是卷积呢？

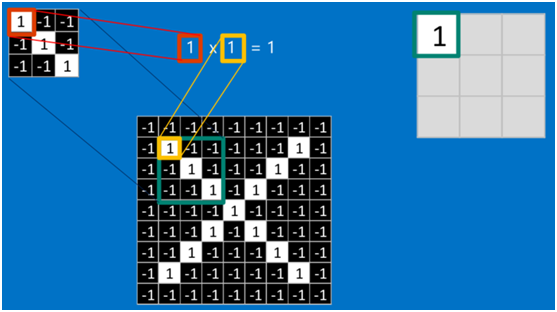
当给定一张新图时，CNN并不能准确地知道这些特征到底要匹配原图的哪些部分，所以它会在原图中把每一个可能的位置都进行尝试，相当于把这个feature（特征）变成了一个过滤器。这个用来匹配的过程就被称为卷积操作，这也是卷积神经网络名字的由来。

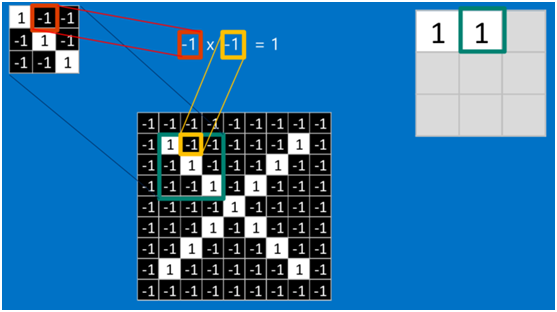
卷积的操作如下图所示：

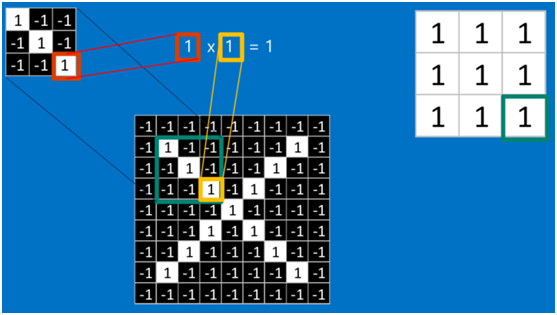


在本案例中，要计算一个feature（特征）和其在原图上对应的某一小块的结果，只需将两个小块内对应位置的像素值进行乘法运算，然后将整个小块内乘法运算的结果累加起来，最后再除以小块内像素点总个数即可（注：也可不除以总个数的）。

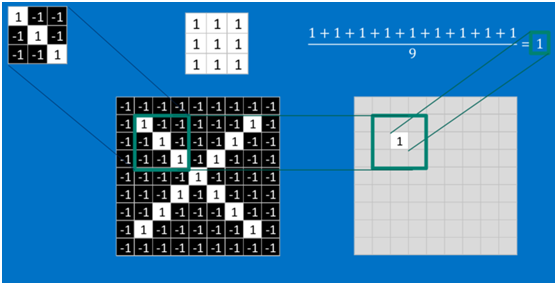
如果两个像素点都是白色（值均为1），那么1\*1 = 1，如果均为黑色，那么(-1)\*(-1) = 1，也就是说，每一对能够匹配上的像素，其相乘结果为1。类似地，任何不匹配的像素相乘结果为-1。具体过程如下（第一个、第二个……、最后一个像素的匹配结果）：



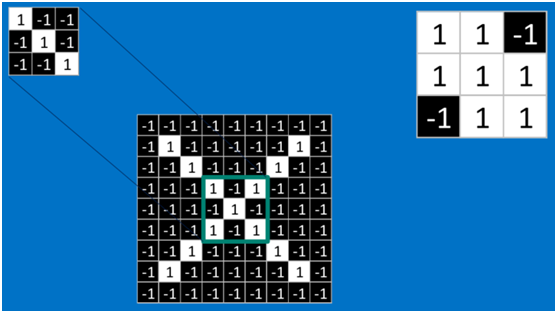




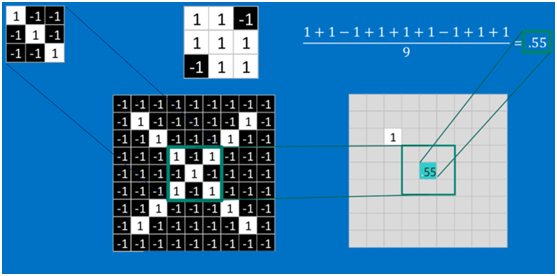
根据卷积的计算方式，第一块特征匹配后的卷积计算如下，结果为1



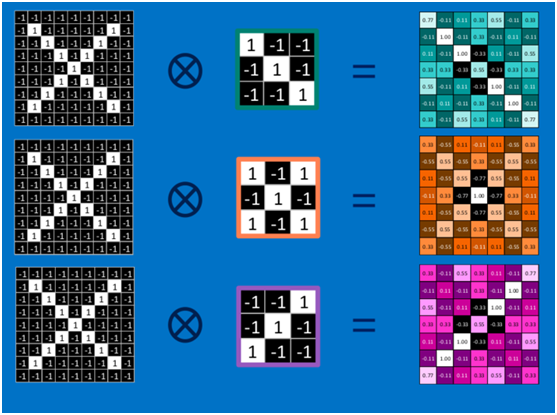
对于其它位置的匹配，也是类似（例如中间部分的匹配）



计算之后的卷积如下



以此类推，对三个特征图像不断地重复着上述过程，通过每一个feature（特征）的卷积操作，会得到一个新的二维数组，称之为feature map。其中的值，越接近1表示对应位置和feature的匹配越完整，越是接近-1，表示对应位置和feature的反面匹配越完整，而值接近0的表示对应位置没有任何匹配或者说没有什么关联。如下图所示：



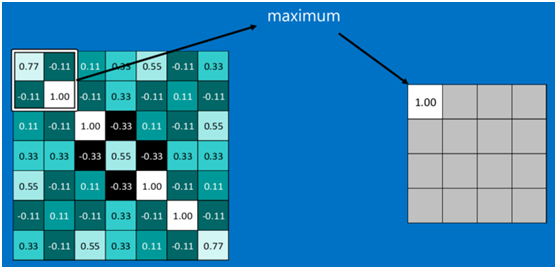
可以看出，当图像尺寸增大时，其内部的加法、乘法和除法操作的次数会增加得很快，每一个filter的大小和filter的数目呈线性增长。由于有这么多因素的影响，很容易使得计算量变得相当庞大。

2.1.5 池化（Pooling）

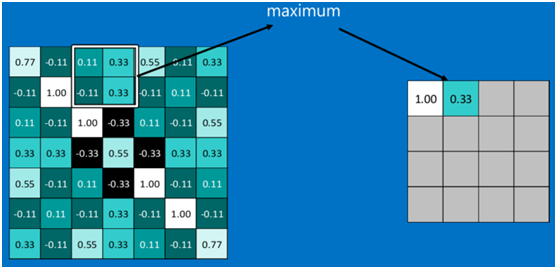
为了有效地减少计算量，CNN使用的另一个有效的工具被称为“池化(Pooling)”。池化就是将输入图像进行缩小，减少像素信息，只保留重要信息。

池化的操作也很简单，通常情况下，池化区域是2\*2大小，然后按一定规则转换成相应的值，例如取这个池化区域内的最大值（max-pooling）、平均值（mean-pooling）等，以这个值作为结果的像素值。

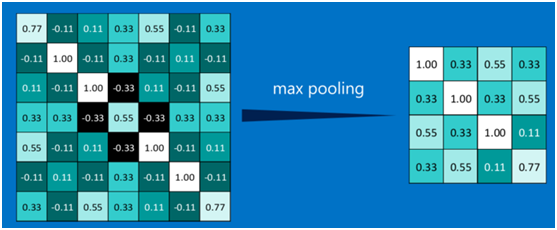
下图显示了左上角2\*2池化区域的max-pooling结果，取该区域的最大值max(0.77,-0.11,-0.11,1.00)，作为池化后的结果，如下图：



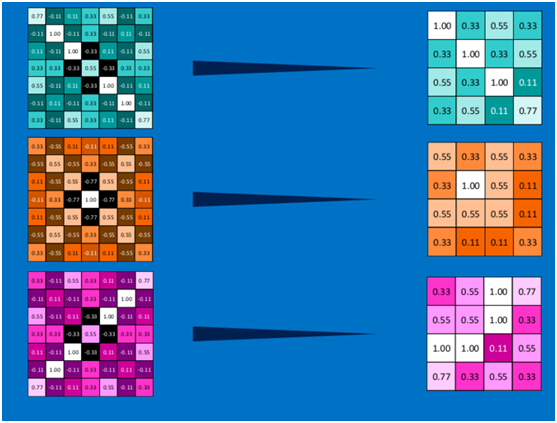
池化区域往左，第二小块取大值max(0.11,0.33,-0.11,0.33)，作为池化后的结果，如下图：



其它区域也是类似，取区域内的最大值作为池化后的结果，最后经过池化后，结果如下：



对所有的feature map执行同样的操作，结果如下：



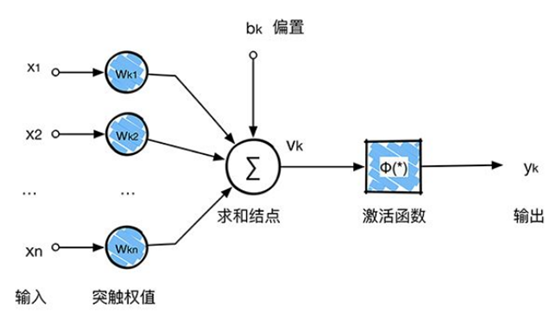
最大池化（max-pooling）保留了每一小块内的最大值，也就是相当于保留了这一块最佳的匹配结果（因为值越接近1表示匹配越好）。也就是说，它不会具体关注窗口内到底是哪一个地方匹配了，而只关注是不是有某个地方匹配上了。

通过加入池化层，图像缩小了，能很大程度上减少计算量，降低机器负载。

2.1.6 激活函数RelU (Rectified Linear Units)

常用的激活函数有sigmoid、tanh、relu等等，前两者sigmoid/tanh比较常见于全连接层，后者ReLU常见于卷积层。

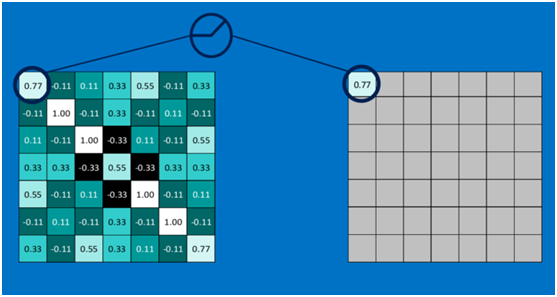
回顾一下前面的感知机，感知机在接收到各个输入，然后进行求和，再经过激活函数后输出。激活函数的作用是用来加入非线性因素，把卷积层输出结果做非线性映射。



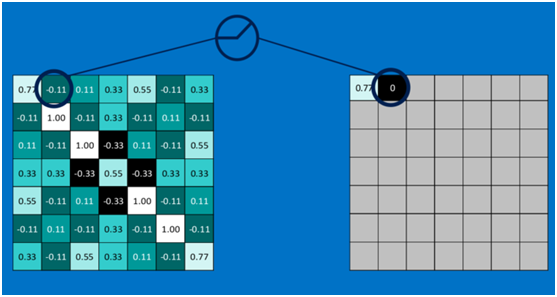
在卷积神经网络中，激活函数一般使用ReLU(The Rectified Linear Unit，修正线性单元)，它的特点是收敛快，求梯度简单。计算公式也很简单，max(0,T)，即对于输入的负值，输出全为0，对于正值，则原样输出。

下面看一下本案例的ReLU激活函数操作过程：

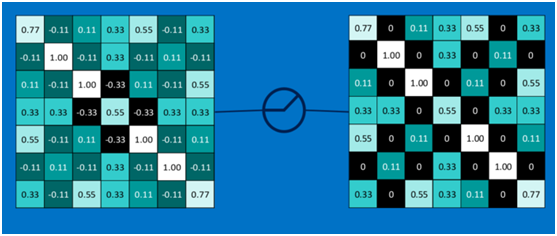
第一个值，取max(0,0.77)，结果为0.77，如下图



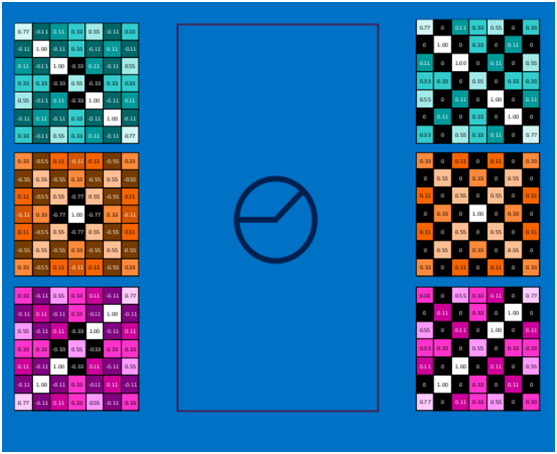
第二个值，取max(0,-0.11)，结果为0，如下图



以此类推，经过ReLU激活函数后，结果如下：

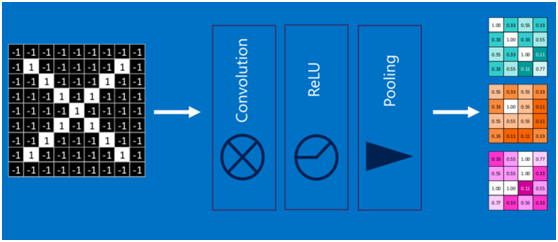


对所有的feature map执行ReLU激活函数操作，结果如下：

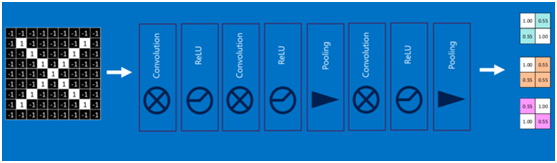


2.1.7 深度神经网络

通过将上面所提到的卷积、激活函数、池化组合在一起，就变成下图：

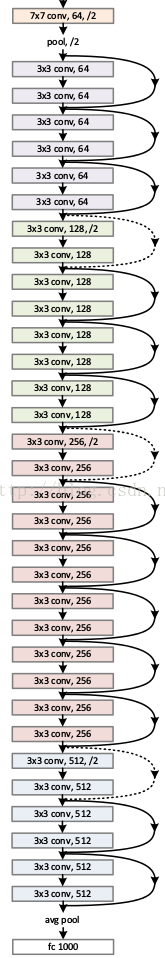


通过加大网络的深度，增加更多的层，就得到了深度神经网络，如下图：

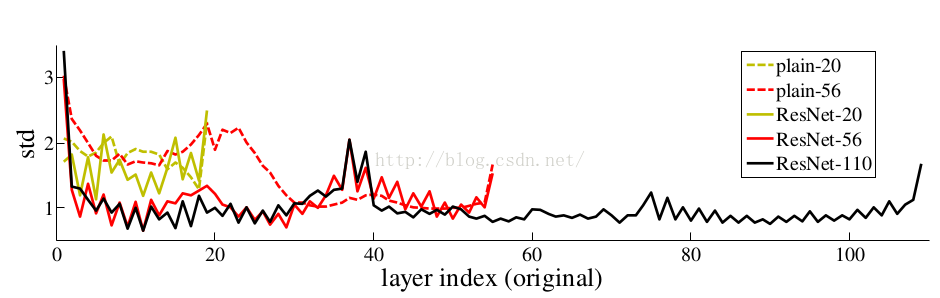


**2.2 Resnet**

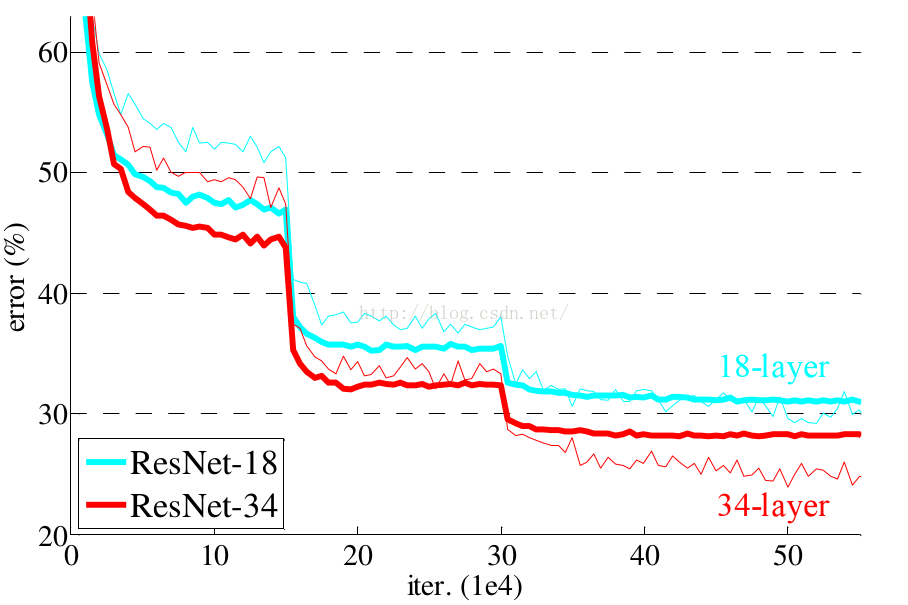
ResNet网络是参考了VGG19网络，在其基础上进行了修改，并通过短路机制加入了残差单元。变化主要体现在ResNet直接使用stride=2的卷积做下采样，并且用global average pool层替换了全连接层。ResNet的一个重要设计原则是：当feature map大小降低一半时，feature map的数量增加一倍，这保持了网络层的复杂度。ResNet相比普通网络每两层间增加了短路机制，这就形成了残差学习，其中feature map数量发生了改变。对于18-layer和34-layer的ResNet，其进行的两层间的残差学习，当网络更深时，其进行的是三层间的残差学习，三层卷积核分别是1x1，3x3和1x1，一个值得注意的是隐含层的feature map数量是比较小的，并且是输出feature map数量的1/4。下图是resnet的网络结构。



RESNET学习的是残差函数F(X) = H(X) - X, 这里如果F(X) = 0, 那么就是恒等映射。事实上，RESNET是“SHORTCUT CONNECTIONS”的在CONNECTIONS是在恒等映射下的特殊情况，它没有引入额外的参数和计算复杂度。 假如优化目标函数是逼近一个恒等映射, 而不是0映射， 那么学习找到对恒等映射的扰动会比重新学习一个映射函数要容易。从下图可以看出，残差函数一般会有较小的响应波动，表明恒等映射是一个合理的预处理。



残差网络解决了退化的问题，在训练集和校验集上，都证明了的更深的网络错误率越小，如下图。



**2.3 Flask**

Flask是一个基于Python开发并且依赖jinja2模板和Werkzeug WSGI服务的一个微型框架，对于Werkzeug本质是Socket服务端，其用于接收http请求并对请求进行预处理，然后触发Flask框架，开发人员基于Flask框架提供的功能对请求进行相应的处理，并返回给用户，如果要返回给用户复杂的内容时，需要借助jinja2模板来实现对模板的处理，即：将模板和数据进行渲染，将渲染后的字符串返回给用户浏览器

“微”(micro) 并不表示你需要把整个 Web 应用塞进单个 Python 文件（虽然确实可以 ），也不意味着 Flask 在功能上有所欠缺。微框架中的“微”意味着 Flask 旨在保持核心简单而易于扩展。Flask 不会替你做出太多决策——比如使用何种数据库。而那些 Flask 所选择的——比如使用何种模板引擎——则很容易替换。除此之外的一切都由可由你掌握。如此，Flask 可以与您珠联璧合。

默认情况下，Flask 不包含数据库抽象层、表单验证，或是其它任何已有多种库可以胜任的功能。然而，Flask 支持用扩展来给应用添加这些功能，如同是 Flask 本身实现的一样。众多的扩展提供了数据库集成、表单验证、上传处理、各种各样的开放认证技术等功能。Flask 也许是“微小”的，但它已准备好在需求繁杂的生产环境中投入使用。

3设计工作和进展情况

3.1项目开发计划及个人阶段性完成情况

3.1.1工作内容

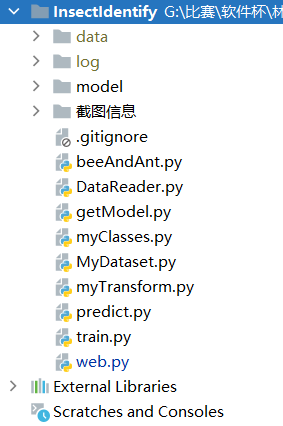
本人工作内容主要分为三个方面，首先是参与讨论需求分析和系统设计，以及参与完成需求分析，开发计划文档，系统设计文档，和主要负责详细设计文档工作；其次是完成数据库建库建表，完成昆虫信息的采集整理；最后是完成深度学习的目标检测部分。

3.1.2工作进度安排

|  |  |  |
| --- | --- | --- |
| 时间 | 工作内容 | 备注 |
| 1-3周 | 完成建组，确定团队成员和项目题目，开会初步讨论项目需求等 | 确定项目题目为《林业害虫识别》，初步确定使用PyTorch为深度学习框架，Flask为后端框架 |
| 4-5周 | 讨论确定项目具体需求，完成项目需求分析文档 | 完成需求分析文档 |
| 6-7周 | 学习目标检测及PyTorch框架相关知识 | 整理了部分笔记 |
| 8-9周 | 完成代码编写，在经典任务蜜蜂和蚂蚁上实现图像分类 | ResNet18在二分类任务上可达90%正确率 |
| 10周 | 迁移代码以满足需求，查阅资料更换为ResNet152提高正确率 | ResNet152在害虫分类任务上可达90%正确率 |
| 11周 | 使用Flask框架完成后端编写  参与完成详细设计文档； | 无 |
| 12-13周 | 进一步完善后端代码部分；  完善项目详细设计文档；  完善个人项目报告； | 无 |

3.1.3具体工作内容

* 图像分类的实现

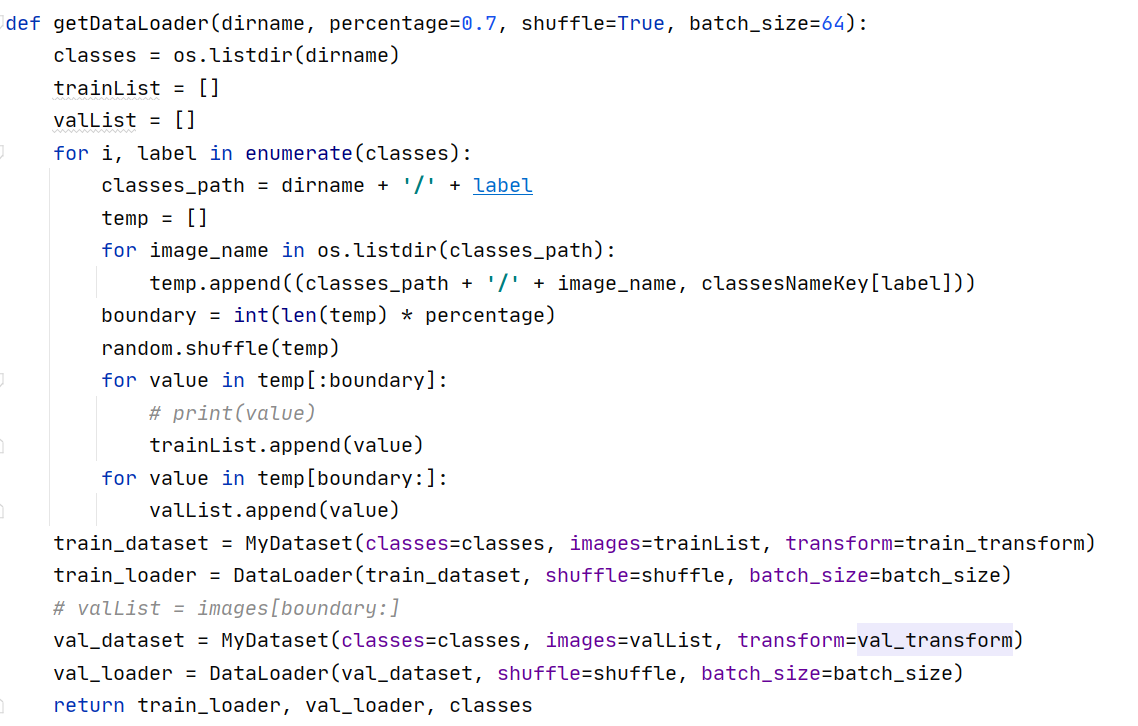


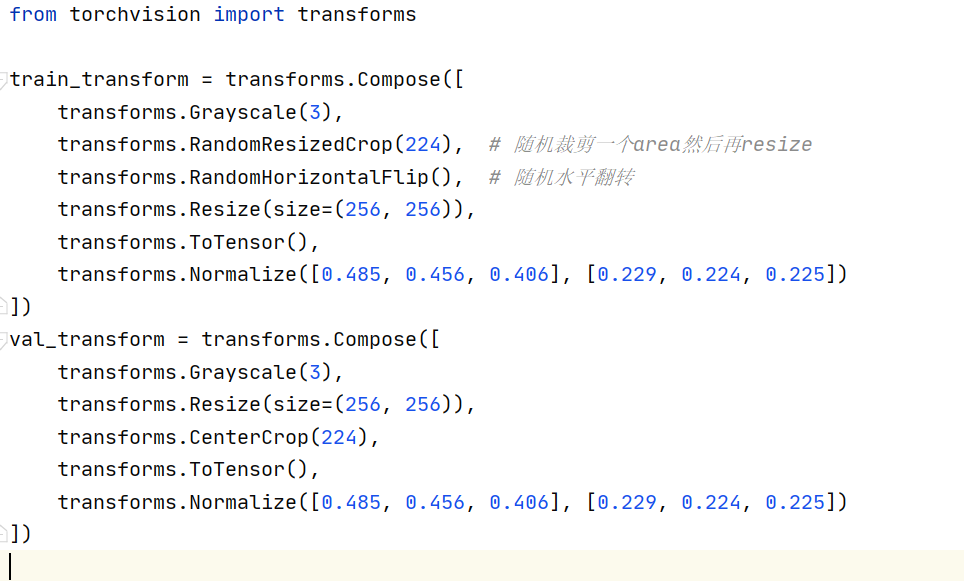
文件及作用

1. myClasses.py：由于pytorch训练部分与数据库分离，故使用myClasses保存分类及id互相对应的信息，便于在训练时将分类转化为di，预测时将获得的id转化为名称。
2. getModel.py：用于在训练时加载预训练模型，预测时加载已经训练好的模型。
3. MyDataset.py：用于存储数据集合并将其作为参数生成Dataloader，便于训练。
4. DataReader.py：用于读取数据集对应的文件夹，生成数据集存入Dataset，并将其转化为Dataloader，以方便后续训练。
5. MyTransform.py：对图像数据进行的一些操作，如缩放，翻转等。
6. train.py：调用其他模块，进行训练。
7. predict.py：使用传入的模型进行图片分类。

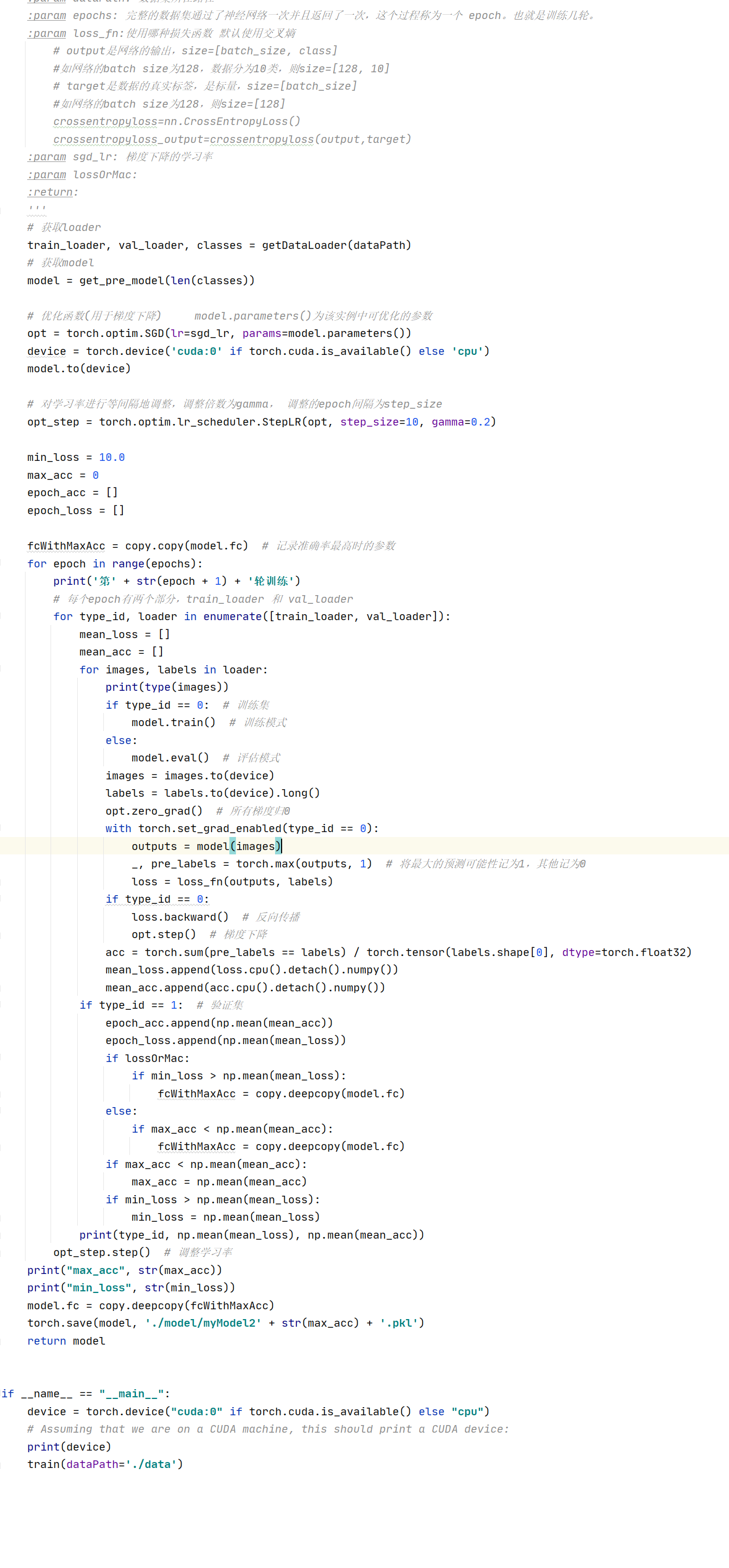
* 构建web服务

1. web.py：使用flask框架，加载模型，等待http请求，调用predict进行分类并返回结果。
2. 首先，对图片读取数据，并对数据进行预处理，随机缩放，裁剪等

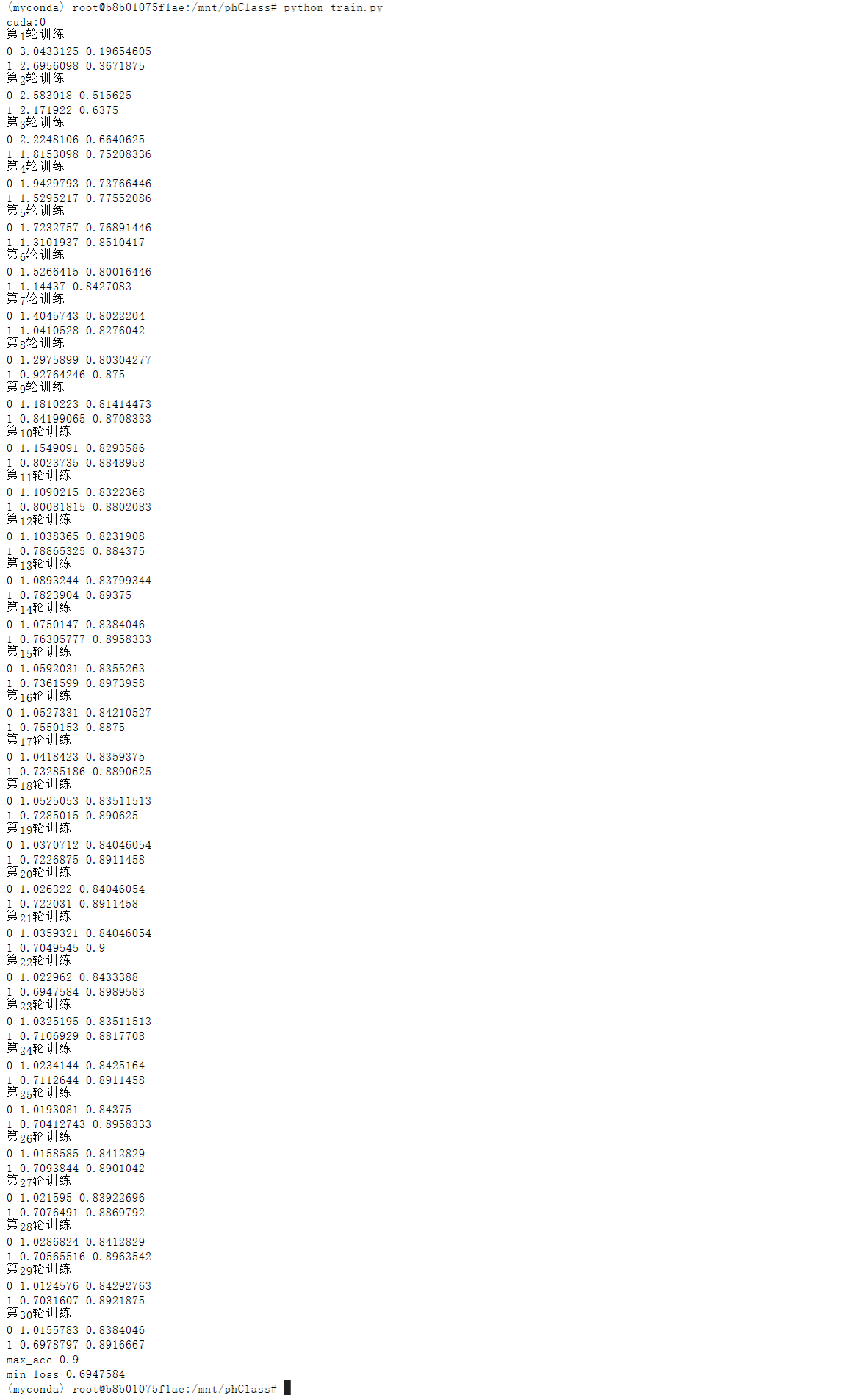




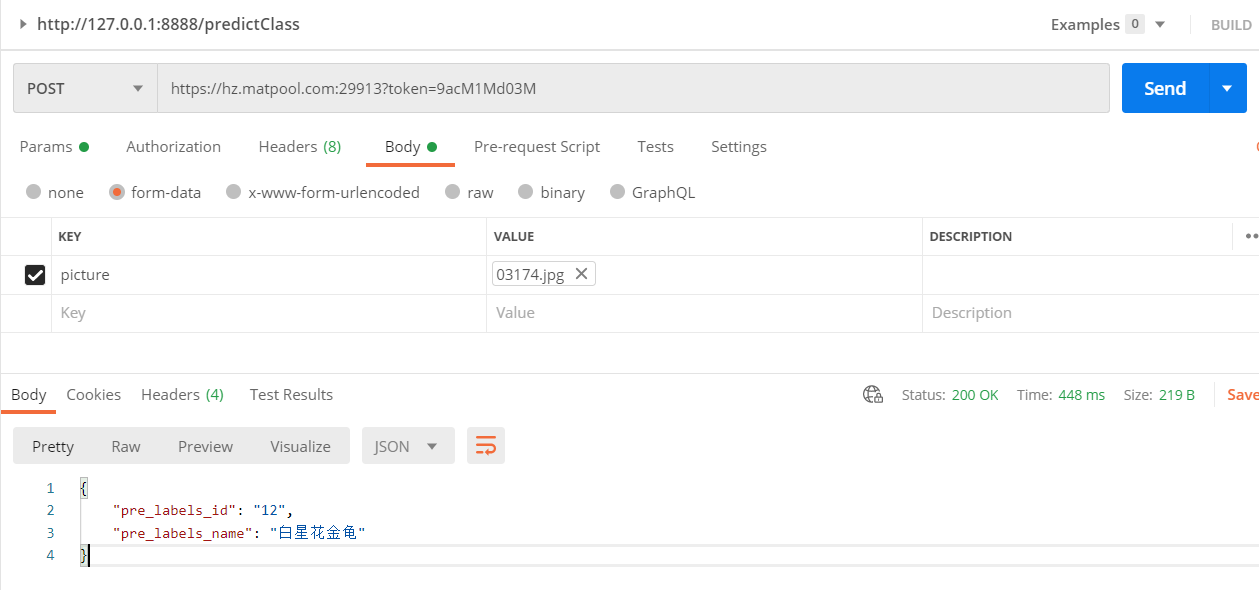
1. 使用迁移学习，重构全连接层构建图像分类模型。
2. 建立损失函数并进行模型的训练



1. 总结：在矩池云训练如下，用1h\*10左右训练了300轮，构建了10个模型，选取准确率最高的模型（90%）作为最终模型。



1. 使用postman测试接口访问



3.2项目的开发内容和结果

3.2.1系统需求分析

1. 功能需求

SRS-0010上传害虫图片

系统应提供用户上传害虫图片功能，包括从图库中上传害虫图片和直接用手机相机拍摄害虫图片两种方式。

SRS-0020反馈害虫信息

系统应提供反馈图片中害虫详细信息功能，包括图片中害虫的数量、种类，以及每种害虫的名称、描述、危害等级、分布和消除办法等详细信息。

SRS-0030查看历史识别记录

系统应提供用户查看害虫识别历史记录的功能，包括每次上传图片的时间以及上传的图片。

SRS-0040历史图片再次识别

系统应提供用户对历史记录中的害虫图片进行再次识别的功能，即用户点击历史图片旁的识别按钮，系统能够对用户反馈图片中害虫的详细信息。

SRS-0050害虫库概览

系统应提供用户查看数据库中所有害虫详细信息的功能，即用户可以浏览每种害虫的名称、描述、危害等级、分布和消除办法等详细信息。

SRS-0060搜索害虫

系统应提供用户搜索害虫库中特定害虫的功能，用户输入害虫名称，系统反馈害虫的详细信息。

1. 性能需求

SRS-009时间要求

用户上传图片后，等待系统反馈害虫信息的时间不超过3s。

SRS-0100准确率要求

在20种害虫的情况下，害虫种类识别结果top3正确率达到95%。

1. 其他需求

SRS-0110安装要求

程序通过微信小程序即可访问，用户通过使用微信小程序就可进行害虫智能识别。

3.2.2系统设计

* 系统结构

采用B/S架构，分为微信小程序前端和识别系统后端。其中前端直接连接数据库，数据库的CRUD在前端实现。后端仅负责识别害虫图片。

* 硬件平台

CPU：Intel i7

CPU：Intel i7

CPU：Intel i7

GPU：RTX2080Ti

内存：16G

硬盘空间：200G

* 软件平台

前端微信小程序开发者工具开发，开发后运行于用户手机上。

* 接口设计

1. 外部接口设计

|  |  |  |
| --- | --- | --- |
| 接口类型 | 接口 | 功能说明 |
| 调用本地接口 | 系统拍照和图库功能 | 用于用户上传图片至云端服务器进行分析 |
| 调用云端接口 | 小程序云端数据库接口 | 用于存储用户数据，用户识别历史数据等 |

1. 内部接口设计

|  |  |  |
| --- | --- | --- |
| 接口说明 | 操作 | |
| 本地小程序端 | 服务器端 |
| 上传图片 | 将图片信息传送至服务器 | 接收图片信息，进行目标检测和识别 |
| 接收识别结果 | 接收识别结果并呈现 | 将识别结果返回至小程序端 |
| 发送查询识别历史信息请求 | 将查询请求发送至服务器端 | 根据请求进行查询 |
| 接收识别历史信息数据 | 接收历史识别信息的数据 | 将查询结果返回至小程序端 |

1. 人机接口设计

|  |  |  |
| --- | --- | --- |
| 接口描述 | 用户请求 | 小程序应答 |
| 上传图片 | 请求系统调用图库或者相机 | 弹出请求窗口，经过允许后打开相机或是图库 |
| 请求历史识别记录 | 请求查看历史记录 | 切换历史记录页面，进行历史识别记录的显示 |

4讨论与体会

**4.1 对项目过程的体会**

**开发历程：**本次项目中，我所负责是主要的深度学习模块和后端服务实现。深度学习模块的核心是残差实现对用于上传的害虫图片图片的识别。我从最开始的小组讨论调研决定选题，到功能质量需求分析，再到系统概要详细设计，然后是代码构建测试，最后到部署与维护，都认真参与用心负责，竭尽全力做好自己所分配的工作，完成好自己所负责的任务。每当做到一个里程碑的时候，心里所获得的成就感是巨大的，欣慰与愉悦是说不出的。中途有一段时间因为数据集稀缺和精度低下，曾迷茫过也困扰过，但经过和小组成员们的头脑风暴和技术帮助，我获得到了很多很好的解决问题的方法和点子，也学到了很多新知识新观点，最后很顺利的实现了害虫识别模块。

**开发体会：**经过此次项目开发过程，我体会最深刻的是我们团队的凝聚力和行动力。我明白了，一个完整的软件项目的开发与实现，来源于每个人的心血与汗水，脱不开每个人的努力和付出。想要做成一个精美的项目，它的开发历程是艰辛也是漫长的，不可能一个人单枪匹马就能全部实现，也不可能最后几日用力一蹴而就。它是很需要时间慢慢地去打磨，需要精力小心地去雕琢。这样，出来的才是用心的项目，才是好的项目。

**4.2 对项目的评价**

**需求评价：**我认为，本次我们的项目是开发一款面向林业管理者所开发的一款害虫识别系统，为林业管理者提供害虫识别分类以及信息获取，也提供了害虫知识库功能，具有很高的实用性和可拓展性。尤其是在当今“绿水青山就是金山银山”规划纲要的倡导和指引下，林业健康也成为十分重要的模块，而且大多林业管理者都只能通过经验分析来防治害虫，这就让我们的项目显得非常有必要且前景市场广阔。

**实现评价：**在实现过程中，我们分工明确，目的准确，每个人都认真地负责了自己的模块，基本实现了项目设计当初拟定的所有需求，同时还进行了适当地追加和改进。然而，个人认为在细节方面，本项目有较大的改善空间，例如提升正确率、界面优化、支撑强化学习等等，如果能够有充分的时间，我相信能够更加完善本项目，让它成为一个真正的可以投入市场投入实际高校使用的系统。总之，项目好，开发顺，可扩展余地大。

5 小结

通过本学期杨老师的软件项目管理课程的学习，以及此次林业害虫识别小程序开发实践的经历，我对软件项目管理和软件项目生命周期有了更深刻的认识和掌握。我明白了项目开发历程不单单只是代码实现，更重要的还有前期需求分析、系统设计跟后期的系统测试等等。同时，我也知道了想要做成一个精美的项目，它的开发历程是艰辛也是漫长的，不可能一个人单枪匹马就能全部实现，也不可能最后几日用力一蹴而就，只有用心雕琢的才是好项目。本次项目我们分工明确，每个人都认真地负责了自己的模块，基本实现了项目设计当初拟定的所有需求。不过在细节方面还有可以改善的空间，如安全管理等。总之，本次项目管理课程实践让我收获颇多，我相信所学到的知识能够在我未来的学习和工作起到重要作用，同时我也将在软件管理这条路上勇往直前，争取学得更精，做得更多。谢谢！