密级★保密期限：

浙江理工大學

Zhejiang Sci-Tech University

硕 士 专 业 学 位 论 文

Professional Master's Thesis

中文论文题目：基于深度学习的农业害虫图像智能识别系统的研究与实现

英文论文题目：Research and Implementation on Intelligent Recognition System of Agricultural Pest Image Based on Deep Learning

专业学位类别：工程硕士

专业学位领域：计算机技术

作者姓名：姜梦洲

指导教师：姚青　宋滢　唐健

完成日期：2018 年 12 月 14 日

浙江理工大学学位论文独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得浙江理工大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：姜梦洲

签字日期：2019年3月11日

摘　要

实时准确地识别出农业害虫种类是有效开展虫情防治工作的重要前提。目前，我国农作物害虫的诊断方法主要是依靠人工识别，主观因素大，实时性差，而且昆虫分类学家较少，农民专业知识又相对缺乏，在基层植保人员日益减少的情况下亟需便捷的农作物害虫智能识别工具。随着智能手机的普及和深度学习在图像识别中的成功应用，利用手机拍摄害虫图像并进行实时识别已成为许多学者研究的热点。本论文以 34 种常见农作物害虫为研究对象，建立了基于深度学习的农业害虫图像智能识别系统，包括手机客户端，云服务端和深度学习算法，实现了害虫图像高效、实时、准确的识别。主要研究内容与成果如下：

(1) 开发基于 Andriod 平台的客户端应用软件。可通过注册账号、短信验证或 QQ 验证进行登录。当在农业生产活动中遇到未知的害虫时，可通过客户端应用软件拍摄或选取相册中的害虫图像，经预览、裁剪等操作处理后上传至服务器进行识别诊断，同时可以通过信息确认的方式将识别结果同步到地图界面上，与广大用户共同了解身边以及全国范围内的害虫分布信息。用户还可以通过查询的方式进行农作物害虫的认识与了解，学习正确的化学防治与生物防治措施。此外还可在个人中心查看以往的识别记录，对于不确定的识别结果还可以通过远程专家诊断功能进行实时在线问答。

(2) 搭建基于 Windows Server 系统的云服务器。设计并实现 Servlet 业务逻辑，实现登录信息、用户注册信息、农业害虫图像、GPS 信息、专家诊断信息等数据的接收与处理；通过 MySQL 与 JDBC 实现了数据库管理系统的设计与连接；采用 DAO 模式实现了业务代码和数据库逻辑操作之间的解耦；采用连接池优化了数据库连接；在微机上实现了 Caffe 深度学习框架的搭建与重构，并将项目打包成 War 包，部署在云服务端。

(3) 基于深度学习的农业害虫识别算法的研究。针对 34 类主要农作物害虫，1 万多幅图像，在应用较广泛的 CaffeNet、VGGNet 和 ResNet 上对卷积神经网络结构进行微调，以获取农业害虫的识别模型，通过对大量害虫图像进行测试和对比得出，101 层的 ResNet 模型有较好识别效果，平均识别准确率可达到 93.5%，具有良好的鲁棒性以及泛化能力。

(4) 采用 JNI 本地方法接口与 DLL 动态链接库实现了深度学习识别模型的调用。通过在 Native 本地方法中添加模型文件，均值文件、权值文件和标签文件的地址参数实现农业害虫图片的识别。经测试，在 CPU 模式下农业害虫的平均识别响应速度在 1.1 秒左

右，GPU 模式下的平均识别响应时间在 0.67 秒左右，偏差在 200ms 以内，满足实际应用的需求。

本文研究的农业害虫图像智能识别系统，可以在线查阅害虫信息，实时识别用户上传的害虫图片，指导农户正确的开展防治活动，有着良好的应用前景，可广泛应用于田间农作物害虫的识别和诊断。

关键词：农业害虫；Android 平台；云服端；深度学习；卷积神经网络；智能识别

Abstract

Accurate identification of agricultural pest species in real time is an important prerequisite for effective pest control. At present, the diagnosis methods of crop pests in China mainly rely on artificial identification, subjective factors are large, real-time is poor, and there are fewer insect taxonomists, and farmers' professional knowledge is relatively lacking. In the case of the decreasing number of grassroots plant protection personnel, it is urgent to have convenient Crop pest intelligent identification tool. With the popularization of smart phones and the successful application of deep learning in image recognition, it has become a hot spot for many scholars to use mobile phones to capture pest images and identify them. Based on 34 common crop pests, this paper establishes an intelligent recognition system for agricultural pest images based on deep learning, including mobile client, elastic compute service and deep learning algorithm, which realizes efficient, real-time and accurate identification of pest images. The main research contents and results are as follows:

(1) This article developed a client application based on the Andriod platform. Users can log in by registering an account, SMS verification or QQ verification. When the user encounters an unknown pest in the agricultural production, the client application software can be used to capture or select the pest image in the album, and after previewing, cropping, etc., it is uploaded to the server for identification diagnosis. The method synchronizes the recognition result to the map interface to learn about the distribution of pests around and across the country. Users can also learn and understand crop pests by means of inquiry, and learn the chemical control and biological control measures. Users can view identification records in the personal center, and can also conduct real-time online question and answer through remote expert diagnosis function for uncertain results.

(2) This paper builds a cloud server based on Windows Server system. Receive and process data such as login information, user registration information, agricultural pest images, GPS information, and expert diagnostic information by writing Servlet business logic; realize the design and connection of database management system through MySQL and JDBC; Decoupling between business code and database operations through DAO mode; using the connection pool to optimize the database connection; the Caffe deep learning framework is

built and reconstructed on the microcomputer, and the project is packaged into a War package and deployed on the cloud server.

(3) Research on agricultural pest identification algorithm based on deep learning. Based on 34 major crop pests and more than 10,000 images, the identification model of agricultural pests was obtained by fine-tuning the convolutional neural network structure on the widely used CaffeNet, VGGNet and ResNet. By testing and comparing a large number of pest images, the 101-layer ResNet model has great recognition effect, and the average recognition accuracy can reach 93.5%, which has good robustness and generalization ability.

(4) The call of the deep learning model is implemented using the JNI native method interface and the DLL dynamic link library. The identification of agricultural pest images is achieved by adding model files, address files of the mean files, weight files and tag files in the Native native method. After testing, the average recognition response speed of agricultural pests in CPU mode is about 1.1 seconds, and the average recognition response time in GPU mode is about 0.67 seconds, and the deviation is within 200ms, which satisfies the needs of practical applications.

The agricultural pest image intelligent identification system studied in this paper can view pest information online, identify the pictures of pests uploaded by users in real time, and guide farmers to carry out corrective activities correctly. It has a good application prospect and can be widely used in the identification and diagnosis of crop pests in the field.

Key words: Agricultural Pests; Android Platform; Elastic Compute Service; Deep Learning; Convolutional Neural Networks; Intelligent Recognition

目 录

第一章　绪论

1.1 研究背景、目的与意义

1.1.1 研究背景

中国自古以来就是农业大国，水稻、小麦、玉米、大豆等农作物的生产状况直接影响着国民经济的稳定发展。近年来，我国政府持续加大农业科技化的投入，粮食生产水平稳步提升，但农业仍然是最易受环境影响的脆弱行业，气候变化、环境污染、农业灾害以及管理不当等因素都会造成粮食减产，其中病虫害是导致作物产量减少的主要危害之一。2017年我国农作物病虫害对小麦的侵害总面积达9.6亿亩次，玉米的侵害面积达11亿亩次，水稻的侵害面积达13.9亿亩次，严重影响了农产品的产量和品质，害虫防治迫在眉睫[1]。

不同农作物的防治方法不同，目前主要的害虫防治措施是化学防治，喷洒农药是常用的方法。但是由于害虫种类繁多，种间相似度高，容易混淆，导致无法对症下药，最终适得其反，造成环境污染、害虫抗药性增加、生态环境破坏等问题。在这种情况下，亟需对当前的防治措施进行改进，而准确迅速地识别出害虫的类别是有效开展防治工作的前提[0]。传统的害虫诊断方法主要是依赖人工识别[3]，主观因素大，实时性差，且我国农民科技文化素质普遍偏低，专业知识相对缺乏，在昆虫分类学家较少、基层植保人员日益减少等情况下，开发面向广大农户的害虫智能识别系统是非常有必要的。随着计算机视觉的迅速发展，基于图像的农业害虫识别研究[3]已有很多文献报道，然而识别方法虽然越来越多，但相关研究在实际中的应用却非常有限，主要原因就是农业害虫的体积小、辨识度低、移动等问题导致在日常拍摄的照片中难以捕捉到清晰的目标，同时由于农业害虫种类繁多，背景多样化，通过传统的模式识别方法获取的识别模型泛化能力差，难以投入实际应用。

随着深度学习理论的应用[3]，移动设备相机配置的不断提高，市场上面出现了许多针对商标、动植物、菜品、车型等目标的图像智能识别软件，如“花伴侣”、“形色”、“汽车之家”等，但还没有关于农业害虫智能识别的客户端应用软件。根据上述情况，本文提出了基于深度学习的农业害虫智能识别系统，通过将手机客户端，服务端以及深度学习模型相结合，实现了高效，实时，准确的害虫识别。基层植保人员和农民可以通过安卓平台应用软件在农田随时随地拍摄上传图片，得到实时反馈的识别结果与防治措施，实现与用户的点对点服务，及时地为农民提供最有效的害虫防治措施。

1.1.2 目的与意义

农作物田间害虫种类繁多，容易混淆，农民盲目过量使用农药，导致害虫抗性增加、农药残留和环境污染等现象日趋严重。而对农作物害虫进行准确实时地预测预报能有效防治害虫，降低经济损失和减少环境污染。

本文鉴于深度学习方法中的卷积神经网络在大规模图像识别任务中的出色表现[12-16]，研究并实现了基于深度学习的农业害虫智能识别系统，当在农业生产活动中遇到未知的害虫时，可以通过手机端拍照上传，经云服务器调用深度学习算法后给出害虫识别结果、基本信息以及相应的防治措施。与此同时还可以在识别反馈界面通过确认的方式将识别结果同步到地图上，与广大用户共同了解附近田间乃至全国范围内的害虫分布信息。用户还可以通过查询的方式了解我国主要农作物害虫的生物学信息、形态和防治措施。此外还可以在个人中心中翻看过往的识别记录，对于不确定的识别结果还可以通过专家远程诊断功能进行在线问答。

该系统通过对用户上传识别的害虫图像进行分类收集，不断扩充训练样本优化识别模型来降低实际应用的泛化误差，提高农业害虫的识别率。无论在任何地点任何时间都可以为用户提供高频次的害虫识别服务，通过拍摄自然状态下的害虫图像进行快速的识别匹配，获取目标信息、防治措施以及置信度等综合信息，切合用户实用性需求，做到真正意义上的害虫识别智能化。

1.2 国内外研究现状

1.2.1 基于经典模式识别的农业害虫识别方法研究

目前，国内已知的害虫种类约 1000 余种，密度高，危害大[17]。为了实现农业害虫智能化识别，国内外许多学者开始着重研究基于图像的农作物害虫自动识别方法，但由于害虫种类繁多，种间相似度高，田间背景复杂，导致识别方法呈多样化发展，主要是针对不同农作物的具体害虫进行实现。

李文斌[错误!未找到引用源。]等采用基于 RGB 色彩单像素背景分割的方式对水稻害虫图像进分割处理，对获取到的二化螟和水稻飞蛾图像提取 HSV 直方图颜色特征以及纹理特征，通过 PAC 对特征数据进行降为后利用 SVM 支持向量机识别分类，与传统的识别方式相比识别率有较好的提高，但该识别方法中害虫图片样本数量较少，且目标只限于水稻二化螟和飞蛾，实际的泛化能力还有待考证。陈晶[错误!未找到引用源。]等采集茶小绿叶蝉图像，通过提取均值

和标准差特征，构建了最小二乘支持向量机识别模型，利用 SMOTE 算法和 KS 算法来解决分类超平面偏移的问题，经测试，该识别方法对茶小绿叶蝉的查准率可达 91.7%。刘国程[错误!未找到引用源。]等以叶螨为研究对象，使用 K-means 聚类算法对农作物叶片上的叶螨图像进行分割和识别，通过 MATLAB 编写程序进行测试，平均识别时间可达到 3.6 秒，平均识别率可达 93%。Pan Chunhua [错误!未找到引用源。]等以蔬菜害虫为研究对象，首先通过网格搜索算法提高图像目标害虫区域的搜索效率，然后利用 SVM 支持向量机训练多个分类器对 4 种主要蔬菜害虫进行识别，平均识别率可达到 93%。ZHAO JUAN [错误!未找到引用源。]等以田间农业害虫为研究对象，采用 Gabor 小波变换提取害虫图像特征，通过 AdaBoost 选取后，结合 SVM 支持向量机对害虫进行分类识别，结果表明，该方法能够对基于纹理的害虫和背景实现高效的识别检测。王德发[错误!未找到引用源。]等在获取标准化特征（几何形状特征、Hu 不变矩特征）的基础上，结合颜色特征，通过采用多分类 SVM 分类器，实现了对 9 种储粮害虫的分类识别检测，平均识别率可达到 94.7%。胡永强[错误!未找到引用源。]等，针对农业害虫颜色特征、形状特征、纹理特征的差异性问题，提出了一种基于稀疏表示的多特征融合害虫图像识别方法，以青海油菜田中 5 种常见害虫为研究对象，通过与支持向量机方法、神经网络方法、稀疏表示方法进行了识别对比，结果表明，该方法在实际的识别应用中有更好的效果。Yaakob [错误!未找到引用源。]等采用六种不同类型的不变矩来提取昆虫图像的形状特征，针对 20 种常见农业害虫，采用 Fuzzy Associative Memory（FAM）神经网络对多种不变矩获取的特征有效性进行分类检测，结果表明该方法较其它识别方法准确率有明显提高。杨国国[错误!未找到引用源。]等以中华稻蝗为研究对象，首先利用 HSV 颜色模型的 S 通道对害虫图像进行增强，然后采用 Otsu 算法自适应寻找阈值完成图像分割，提取目标害虫的颜色矩，形态学特征，不变矩和纹理特征，最后将提取的 23 个全局特征进行归一化处理，由 SVM 分类器进行分类识别，准确率可达到 88.3%。杨文翰[错误!未找到引用源。]等利用 Canny 算子与 Otsu 阈值分割法对 15 种棉花害虫进行分割处理，并提取各个部位的数学特征参数，包括周长、面积、形状参数等，通过二叉树分类法与支持向量机进行分类识别，结果表明，该方法在棉花害虫识别过程中效果良好。

1.2.2 基于深度学习的农业害虫识别方法研究

深度学习尤其是卷积神经网络作为近几年来模式识别中研究的重点，在大规模图像和视频识别领域获得了巨大的成功，越来越受到人们的关注。梁万杰[错误!未找到引用源。]等以水稻二化螟为研究对象，选用预处理后的 3300 余张样本训练测试数据集，通过 Torch 机器学习算

法框架，设计并实现一个 10 层的卷积神经网络模型，采用 Holdout 交叉验证方法对识别模型进行验证，平均识别率可达到 86%，该识别方法可较好的提取图像特征，针对二化螟的识别具有良好抗干扰性和鲁棒性。桂便[29]等以 5 种储粮害虫作为训练样本，利用 CNN 的 Alexnet 模进行训练以及测试，平均识别率可达到 97%，具有较高的准确率和可靠性。张苗辉[错误!未找到引用源。]等利用 Caffe 深度学习框架构建网络模型用于提取农作物害虫特征，在获取到训练样本的特征向量后，采用稀疏表示算法对 10 中常见的害虫进行分类识别测试，平均识别准确率可达到 85.3%。马梦园等[错误!未找到引用源。]基于端到端深度卷积神经网络和 DCNN 特征提取器实现了对鳞翅目昆虫图像的识别，对 70 类鳞翅目昆虫图像集进行测试，识别率可达到 99%，效果优于基于颜色、纹理等传统特征的分类识别手段。Liu 等[32]通过显著性分割方法定位害虫图像来构建训练集，利用 DCNN 模型对 12 类自然环境中的水稻害虫进行局部图像特征的学习以及分类测试，平均识别率可达到 95%。程曦[33]等以储粮害虫为研究对象，通过深度学习卷积神经网络实现了特征的自动化提取与害虫分类，对 7 种储粮害虫测试后，平均识别率可达到 98%，具有很高的实用价值。

1.2.3 农业害虫智能识别应用现状

随着互联网应用的飞速发展，手机平板电脑等便携式移动设备的普及，针对在自然环境下的农业害虫，国内外许多学者不断尝试搭建完整的识别应用系统，用于实现作物病虫害的管理、检测、预报以及防治，这对推动农业智能化、现代化具有重要意义。

孙鹏[错误!未找到引用源。]等通过移动端与服务端相结合，设计并实现了一个基于 Android 平台的储粮害虫识别系统，采用 Web Service 轻量级通信框架解决了移动终端和服务器之间数据交换的难题，为储粮害虫的及时诊断提供了途径，但该系统图像传输效率较差，且伴随信息丢失等问题，实际应用过程中未能有较好的表现。刘蒙蒙[错误!未找到引用源。]等为了监测温室黄瓜的害虫种类和数量变化情况，通过拍照盒与安卓客户端实现了害虫图像的定时采集，利用移动网络将图像上传到远端物联网服务平台进行识别监控。杨和平[错误!未找到引用源。]等构建了一个网络版的昆虫图文查询识别系统，应用 PHP 进行编程和页面设计，采用 MySQL 数据库管理系统存储数据。该系统能够随着样本量的增加和分类研究的发展，不断改进完善，但由于昆虫样本量较少，目前只能对少量害虫实现自动识别。杨慧勇[错误!未找到引用源。]等以农业害虫为研究对象，通过客户端、服务器端和数据库 3 层结构建立了农业害虫远程自动识别诊断系统，利用 CCD 高清摄像头获取图像视频，经远程传送到服务端后进行图像分析与处理，实现了对农业害虫的识别诊断。YueQi Han [错误!未找到引用源。]等研究了基于 Android 平台的

蝗虫数据采集与诊断系统，经 Android 客户端采集蝗虫图像、GPS 等数据信息后，通过移动通信网络传送至服务端，获取反馈结果，目前该系统只是为了方便收集图片和查询蝗虫信息，并没有实现识别诊断功能。张谷丰[错误!未找到引用源。]等研究并实现了基于 Android 平台的的水稻害虫识别诊断系统，首先用户需要在 Android 客户端上根据水稻害虫的体型大小、危害部位、危害症状、以及危害特性进行初步分类，然后将分类数据上传至服务端，后台依靠描述的特征信息和害虫图片进行识别诊断，该系统的诊断方式主要依靠人工推理，实际应用能力较差。高华[错误!未找到引用源。]等以储粮害虫为研究对象，分析设计了基于云平台的害虫图像检测识别系统，利用 Hadoop 分布式系统框架实现了云平台的搭建，同时将并行 BP 神经网络害虫识别检测系统布置在云平台上，用户通过 Web 浏览器登录系统即可上传害虫图片进行识别。贾瑞[错误!未找到引用源。]通过对微信公众号进行开发设计，在移动设备上实现了农业害虫图像识别技术的应用，只需要手机等移动设备和网络，就可以实时拍摄害虫图片进行识别，经后台处理后相关信息再以“消息”的方式发送给用户。

虽然国内外相关方面的研究已经取得了一定进展，但大多数害虫识别系统还是无法投入实际应用。主要原因就是农作物害虫存在体积小、种类多、背景复杂、形态特征多样化等问题，需要针对具体的场景进行识别算法的设计。其次就是现有的识别系统普遍存在实时性差、可应用范围窄、识别准确率低以及泛化能力弱等问题。通过对研究现状的分析，本文提出了基于深度学习的农业害虫图像智能识别系统，通过将手机客户端，服务端以及深度学习模型相结合，实现大田环境下农业害虫高效，实时，准确的识别检测。

1.3 论文研究内容

本论文以 34 种常见农作物（水稻、小麦、玉米、大豆等）害虫为研究对象，通过开发安卓客户端应用软件、建立云服务器、设计数据库、训练深度学习识别模型来实现农业害虫图像智能识别系统的搭建。具体研究内容如下：

(1) 基于 Andriod 平台的客户端应用软件的开发。设计安卓平台客户端的程序架构；实现客户端与服务器之间的网络通信；实现用户登录功能，包括专家登录与普通用户登录两种不同模式；创建客户端 UI 界面；实现农业害虫信息的查询功能，包括害虫的形态特征、防治措施以及图像浏览等；设计自定义相机和图库用于选取图片，实现农业害虫图像的采集、上传和识别结果的接收；开发地图功能为用户提供识别害虫在地图上的位置分布；开发远程专家诊断功能，实现实时在线问答。

(2) 云服务平台的设计与实现。在 Windows Server 系统下搭建轻量级应用服务器；创

建 Java Web 项目，编写 Servlet 业务逻辑以实现农业害虫图像信息、GPS 信息、用户信息等数据的存储；实现二进制数据流的接收与图像压缩；在微机上搭建 Caffe 深度学习框架并重构至项目中；通过 JNI 接口实现农业害虫识别模型的调用；将项目部署在云服务端。

(3) 农业害虫数据库的建立。创建服务端数据库管理系统；实现数据库的访问；创建用户注册信息数据表、第三方登录信息数据表、害虫图像信息数据表、诊断信息数据表；对数据库进行优化。

(4) 农业害虫识别模型的获取。在 Caffe 深度学习框架中训练农业害虫识别模型，并进行优化、对比以及测试。

1.4 技术路线

基于深度学习的农业害虫图像智能识别系统是基于客户端（Client）/服务器（Server）模式的应用开发，主要包括手机客户端，远端服务器和深度学习算法。利用卷积神经网络（基于 caffe 框架）训练深度学习识别模型，并在服务器中添加相应的动态链接库依赖，通过本地方法接口实现模型调用。用户可通过客户端 APP 拍照或者图库选取照片上传实现害虫图像实时高效的识别检测，此外客户端部分提供了农业害虫的信息查询、地图分布、识别历史等功能，对于不确定的识别结果用户还可以通过远程专家诊断功能进行在线问答。本研究的技术路线如图 1.1 下所示。

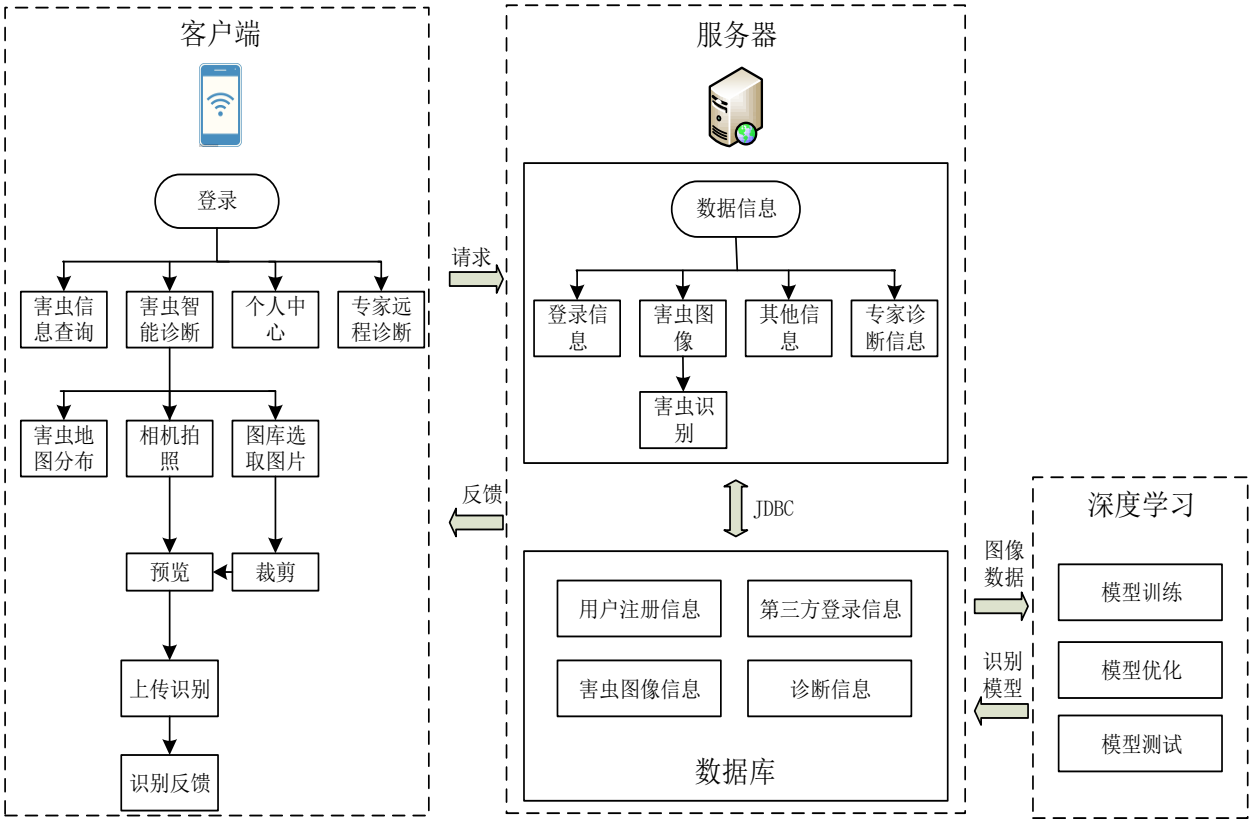


图 1.1 系统技术路线图

1.5 论文安排

本文主要安排如下：

第一章 绪论。主要阐述了基于深度学习的农业害虫智能识别系统的研究背景、研究目的和意义，详细介绍了基于经典模式识别的农业害虫识别方法的研究、基于深度学习的农业害虫识别方法的研究、农业害虫智能识别应用现状，同时也对本文的研究内容和技术路线做了简要说明。

第二章 基于 Andriod 平台客户端的设计与实现。本章给出了客户端核心功能的设计图，从软件开发的角度出发详细介绍了客户端程序的架构设计和项目结构；利用 Okhttp 开源库实现网络通信，使用 Glide 库实现网络图片加载；开发了客户端用户登录模块、害虫信息查询模块、害虫地图分布模块、害虫智能诊断模块以及专家远程诊断模块。

第三章 云服务平台的设计与实现。本章阐述了服务器的开发与数据库的搭建，包括 Tomcat 轻量级应用服务器的搭建、二进制数据流的接收与解析、图片压缩、JNI 本地方法接口对深度学习算法的调用、云服务器部署和 MySQL 数据库的访问与优化。

第四章 基于深度学习的农业害虫图像识别算法的研究。本章阐述了 Caffe 深度学习框架在 Windows 系统下的搭建以及应用，利用 CaffeNet、VGGNet 和 ResNet 训练农业害虫识别模型并进行测试对比，同时生成学习曲线进行分析，最后利用分类程序测试每类样本的识别率。

第五章 基于深度学习的农业害虫智能识别系统的测试与分析。本章主要阐述了该系统各部分功能的操作与测试，在确保运行良好的情况下，检测了客户端程序的 CPU 的使用率、内存消耗、农业害虫识别的平均响应时间以及系统依赖库。

第六章 总结与展望。总结主要研究成果和创新点，并提出下一步需要完善的方向。

第二章 基于 Andriod 平台的农业害虫智能识别客户端的设计与实现

客户端 APP 是基于安卓平台的应用开发，在安卓集成开发环境 Android Studio 下进行设计开发。客户端核心功能有：用户登录、害虫信息查询、害虫地图分布，害虫智能诊断、专家远程诊断。用户可以在手机端通过拍照或者图片上传的方式进行害虫识别诊断，了解害虫的基本信息以及相应的防治措施，同时可以通过信息确认的方式将识别结果同步到地图界面上，与广大用户共同了解身边以及全国范围内的害虫分布信息，用户还可以通过查询的方式进行农作物害虫的认识与了解，学习正确的化学防治与生物防治措施。对于不确定的识别结果通过远程专家诊断功能进行实时在线问答。核心功能如图 2.1 所示：

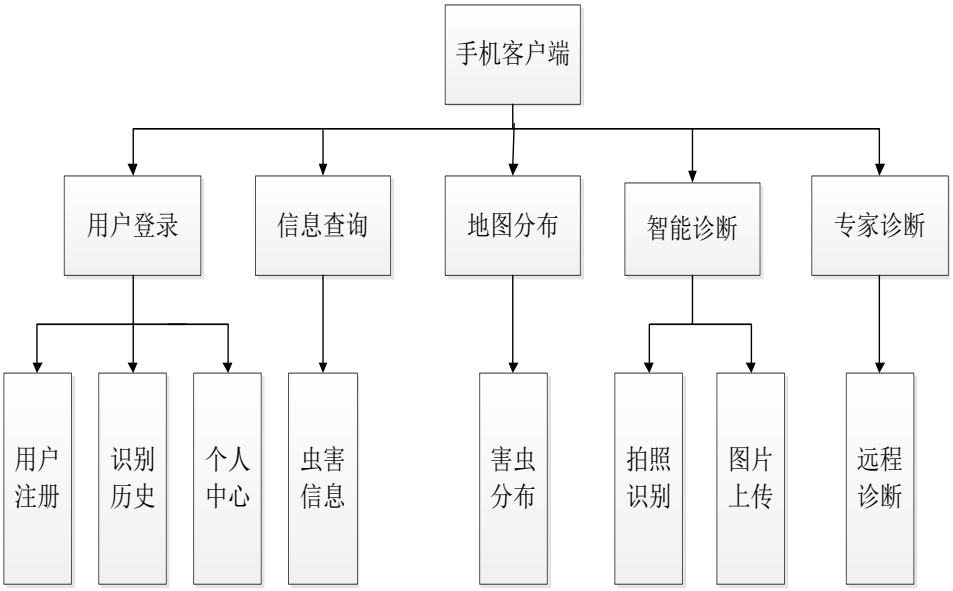


图 2.1 农业害虫图像智能识别系统手机客户端核心功能

2.1 程序架构设计

架构设计的目的是使程序模块化，做到模块内部的高聚合和模块之间的低耦合，使得在程序的开发过程中只需要专注提高开发的效率即可，同时还使后续的模块测试以及定位变地更加方便。

2.1.1 MVC 与 MVP 架构模式

MVC（Model View Controller）模式[42]是将业务逻辑、数据、界面显示相分离的形式

来组织代码的，通过将业务逻辑聚集到一个部件里面，使得在定制个性化界面和改进用户交互的时候不需要重新编写逻辑功能。其中 M 层用于处理数据，业务逻辑等；V 层处理界面的显示结果；C 层起到桥梁的作用，通过控制 V 层和 M 层通信，实现视图显示层和业务逻辑层的分离。MVC 的目的是将 M 和 V 的实现代码分离，从而使同一个程序可以拥有不同的表现形式。Android 的四大组件：活动（Activity）、服务（Service）、广播接收器（Broadcast Receive）、内容提供器（Content Provider）已经规范了应用程序的代码结构，随着引入一系列的实体类，工具类和第三方类库再遵循基本的 Android 原生开发即可实现 MVC 设计模式的客户端开发，但由于 MVC 模式下视图与控制器间的连接过于紧密使得各部分代码的重用性降低，随着需求的不断增加，功能的不断完善，会产生大量的 Activity 与 layout 布局，最终导致程序的臃肿，在后期的开发、测试、维护中需要耗费大量的精力。MVC 架构如图 2.2 所示：

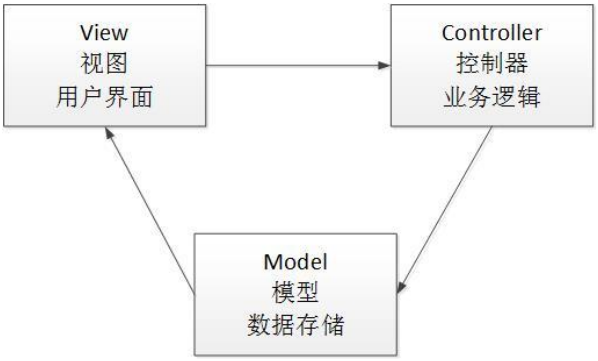


图 2.2 MVC 架构

MVP（Model-View-Presenter）[42]是一种从 MVC 框架演变过来的使用广泛的基础架构模式，是基于事件驱动的应用框架，与 MVC 有一定的相似性。MVP 框架由 3 部分组成：View 负责显示，Presenter 负责逻辑处理，Model 提供数据。MVP 模式与 MVC 模式之间最主要的区别在控制层上，通过将 Activity 中负责业务逻辑的代码移到 Presenter 中使其成为框架的控制者，Model 与 View 通过 Presenter 间接进行交互，使得业务逻辑与视图相分离，提高了代码的扩展性与组件之间的复用能力，实现了低耦合的设计理念，提高了程序的的开发和测试效率。MVP 架构如图 2.3 所示：

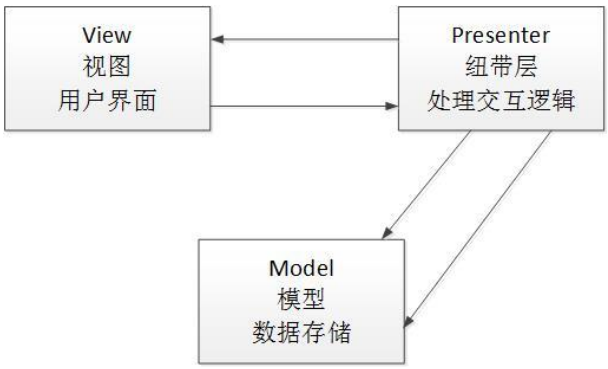


图 2.3 MVP 架构

2.1.2 程序项目结构分析

客户端应用程序架构模式设计完成之后开始创建工程项目，首先需要对整个工程项目目录结构进行分析，开发工作主要是在 app 目录下进行的，包括项目代码等资源内容：AndroidManifest.xml 是项目的配置文件包括四大组件的注册及权限声明；java 目录中保存所有的 java 逻辑代码；res 目录下存放项目中的图片（drawable 目录）、布局（layout 目录）、菜单（menu 目录）、图标（mipmap 目录）、xml 文件（xml 目录）等，values 目录下存放数组，颜色，字符串和主题风格等资源。项目结构如图 2.4 所示：

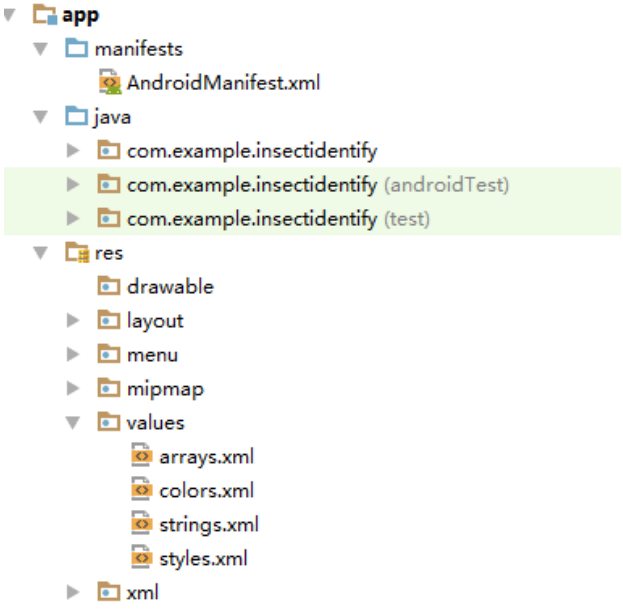


图 2.4 项目结构视图

常用的还有 libs 目录，用于存放项目中使用到的第三方 jar 包，androidTest 和 test 目录用于编写测试用例对项目进行自动化测试。项目中 build.gradle 文件是至关重要的，客户端所有的配置以及打包解析都需要通过 build.gralde 文件来完成，其内部常用配置有：compileSdkVersion 用来设置编译时的 Android 版本；buildToolsVersion 用来设置编译时的

构建工具版本；defaultConfig 闭包用来指定项目包名、兼容版本、项目指定目标版本（表示在该目标版本上已经做过充分测试）、版本号及名称；buildTypes 闭包用来指定生成安装文件的主要配置；productFlavors 闭包用于多个渠道的配置；dependencies 闭包用于定义项目的依赖关系。build.gradle 文件配置如图 2.5 所示：

```
android {
    compileSdkVersion 26
    buildToolsVersion "26.0.3"
    defaultConfig {
        applicationId "com.example.insectidentify"
        minSdkVersion 15
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled true  // Zipalign优化
            zipAlignEnabled true  // 移除无用的resource
            shrinkResources true  // 混淆配置
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    productFlavors {}
}
dependencies {
```

图 2.5 build.gradle 文件配置

2.2 Android 开源库

Android 是一种基于 Linux 的自由及开放源代码的操作系统，通过合理应用开源的类库，可以节约开发周期，减少出错的可能，将时间都集中在处理特有的业务逻辑上，在本课题的客户端开发中需要通过 OkHttp 类库实现网络通信，Glide 类库实现各种渠道的图片加载。

2.2.1 OkHttp 网络通信框架

客户端与服务器之间的通信实现需要依赖 HTTP 请求，而 Android 上发送 HTTP 请求的方式一般有两种：HttpURLConnection 和 HttpClient，但由于 HttpClient 存在 API 数量过多、扩展困难等缺点，在 Android6.0 系统中已被移除，而 OkHttp[44]作为 Square 公司开源的轻量级网络通信框架，相比于原生的 HttpURLConnection 不仅在接口封装上简单易用，而且底层更是支持 SPDY、连接池、Gzip 和 Http 缓存。OkHttp 允许所有同一个主机地址

的请求共享同一个 socket 连接，并通过连接池来降低响应延迟问题；透明的 Gzip 压缩可以减少响应数据的大小；Http 响应缓存可以避免重复的网络请求；当网络出现问题或请求失败时 OkHttp 还可以自动交替尝试配置主机的其他 ip，具有良好的稳定性。在 app 目录下的 build.gradle 文件中添加封装好的 OkHttp 开源库依赖，项目同步后会自动下载对应依赖版本的 OkHttp 库以及负责基础通信的 Okio 库，用于开发设计。

实现 get 请求。首先要创建 OkHttpClient 和 Request 对象实例，通过 Request 对象的 url()方法设置目标网络地址。之后调用 OkHttpClient 对象的 newCall()方法创建 Call 对象并调用其 excute()方法来发送请求，最后处理获取到的服务器返回数据，完成操作。实现 post 请求的方式和 get 请求类似，只需要创建 RequestBody 对象来存放待提交数据，并将其传入 Request 对象的 post()方法中即可。通过对需求进行封装，还可以实现表单文件上传、文件下载、图片加载以及进度回调。OkHttp 请求流程如图 2.6 所示：

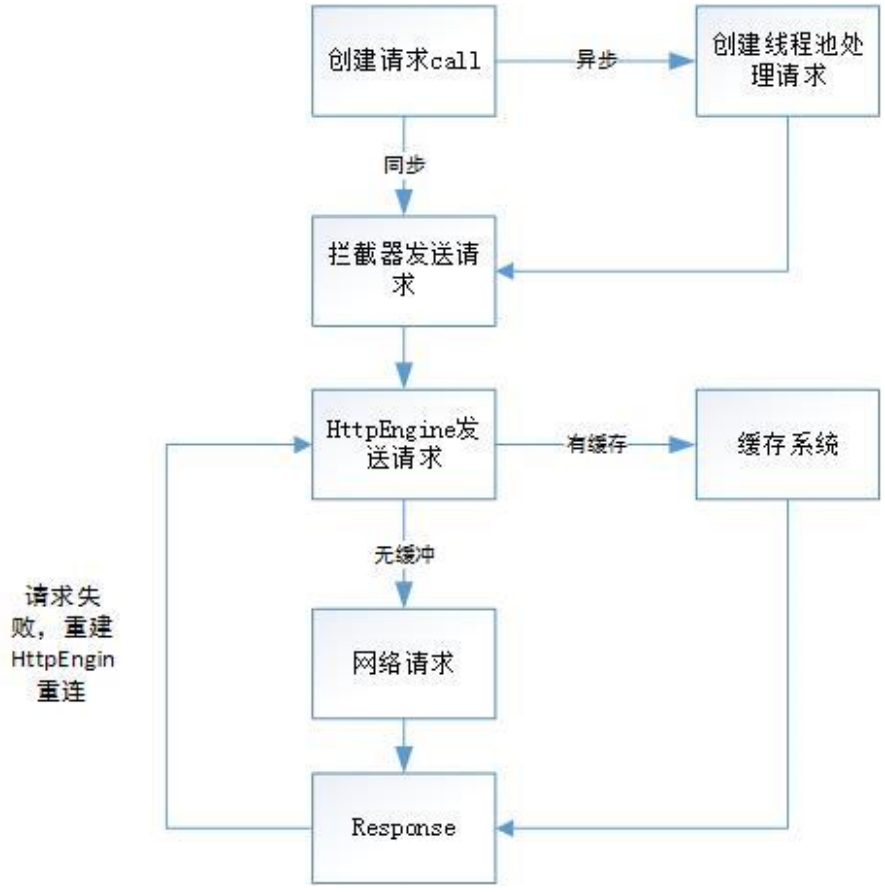


图 2.6 OkHttp 请求流程图

2.2.2 Glide 图片加载框架

Android 客户端开发中会经常遇到与图片处理相关的问题，如图片加载到内存中产生

OOM（OutOfMemory）异常问题，图片多源加载问题以及大量图片的缓存处理问题等。Glide 作为 Google 推荐的图片加载库在其内部封装了非常好的缓存机制，在处理图片的时候能自动减少图片尺寸，使内存保持较低的消耗，并且为了加快加载速度，提高用户体验，会优先加载预览图。更重要的是 Glide 可以从多个源进行加载，如：本地图片、网络图片、GIF 图片、甚至是本地视频。Glide 图片加载库还拥有良好的生命周期管理，图片的加载任务会与 activity 或者 Fragment 的生命周期绑定，当界面执行 onStop 的时候会自动暂定使用，而当执行 onStart 的时候又会自动重新开启，此外 Glide 支持 OkHttp，会对网络状态做监听，当网络状态发生改变时，会重启失败的任务，以减少任务因网络连接问题而失败的概率。

Glide 的核心原理是为 bitmap 维护一个对象池，对象池的主要目的是通过减少大对象的内存分配，以重用来提高性能。编辑配置 app/build.gradle 文件，在 dependencies 闭包中添加依赖后进行同步，涉及到网络加载图片时需要增加网络权限。配置完成后，可通过 Glide 的 with()方法提供上下文、load()方法提供图片资源、into()方法绑定对应的 View 控件实现图片加载，对于加载失败的情况还可以使用 error()设置失败后显示的图片，最重要的是可以通过 diskCacheStrategy()方法来设置缓存机制，总之利用 Glide 库可以方便快捷地解决许多由图片加载所带来的问题。

2.3 程序界面与功能开发

2.3.1 登录界面设计

客户端部分实现了普通用户和专家两种登录方式，对于普通用户而言登录后可以在个人中心中查开以往的识别历史，对于专家而言，登录后可进入专家远程诊断模块，对用户上传的不确定的识别结果进行分析诊断。登录模块使用 post 请求携带用户名，密码等信息参数访问服务器，服务器接受请求后在数据库中进行查询并返回回调信息，此外还提供了 SMS 短信验证登录与 QQ 登录两种第三方登录方式。

首先创建 LoginActivtiy 活动类，在 AndroidManifest.xml 中进行注册，配置为主活动，在 res/layout 目录下找到对应的 activity_login.xml 布局文件，选用 RelativeLayout 的相对布局，并在其中添加两个 EditText 控件用于输入用户名和密码，通过 hint 属性设置提示信息。然后添加两个 CheckBox 控件用于勾选“记住密码”与“自动登录”功能，添加两个 TextView 文本框控件，通过 text 属性设置文本内容，分别为“帐号注册”以及“第三方登录”功能。最后添加 ImageButton 控件，用于提供登录按钮，通过 background 属性设置背景。登录界面如图 2.8(a)。

登录部分的逻辑实现：在活动中通过 getSharedPreferences()创建两个 SharedPreferences 实例，分别为 userInformation（用于存储用户名和密码）以及 checkBoxState（用于记录“记住密码”和“自动登录”多选框的状态）。获取 ImageButton 实例，通过调用 setOnClickListener()方法注册监听器，并复写监听器中的 onClick()方法，在点击登录后会判断文本框中用户名和密码的状态，如果为空，则通过弹出 Toast 提示用户输入账号或密码，否则通过 OKhttp 网络框架 post 上传表单文件至服务器查询数据库，在 onResponse()方法中获取反馈结果 response，得到验证参数，参数值对应信息：0（用户名不存在）、1（密码错误）、2（专家登录）、3（普通用户登录）。如勾选“记住密码”框则，会在登录成功后将用户名与密码写入 userInformation 中保存，再次开启程序后通过 setText()方法自动输入文本框中，如果同时勾选“自动登录”框则会在下次开启程序后自动验证登录。

“帐号注册”功能逻辑实现：添加三个 EditText 控件分别对应用户名，密码与确认密码，添加两个 Spinner 控件分别用于“身份选择”与实现“省市区三级联动”。Spinner 是列表选择框，点击后会展示一个列表供用户进行选择，首先需要在/res/xml 目录下新建 XML 格式的省市区（县）资源文件数据。然后通过 XmlResourceParse 类创建解析器，动态解析 XML 文件，最后通过 ArrayAdapter 对象来设置适配器获取解析到的数据，并通过 setAdapter()方法将其与 Spinner 列表视图相关联。XML 资源文件如图 2.7 所示：

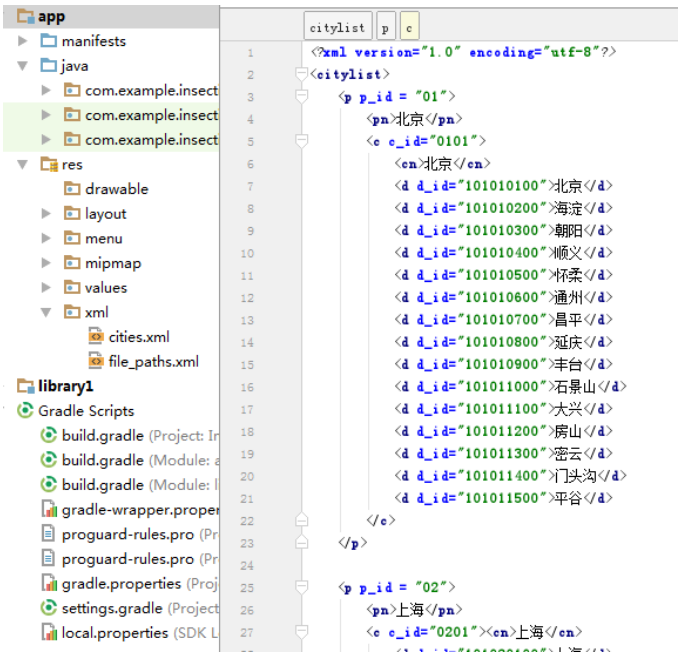


图 2.7 省市区 XML 资源文件

点击注册后，客户端 post 上传表单文件至服务器，并将注册信息插入数据库，返回 response 参数，参数值对应信息：0（注册失败）、1（注册成功）、2（用户名已存在）。

注册界面如图 2.8(b)。

第三方登录主要包括 SMS 短信验证登录与 QQ 登录两种方式，都是通过开发平台提供的 SDK 软件开发工具包中的 API 接口进行实现的，以 SMS 短信验证为类，首先在 MOB 移动开发者平台进行注册，申请并下载短信验证码 SDK，添加相应的 jar 文件到项目 app/libs 目录下，然后在 build.gradle 文件中添加依赖，配置相应的权限。最后通过 APP Key 和 APP Secret 初始化 SDK，实现可视化界面接口，即可进行短信的发送和验证。第三方登录界面如图 2.8(c)。

(a) 用户登录界面　　**(b) 注册界面**　　**(c) 第三方登录界面**

图 2.8 登录模块

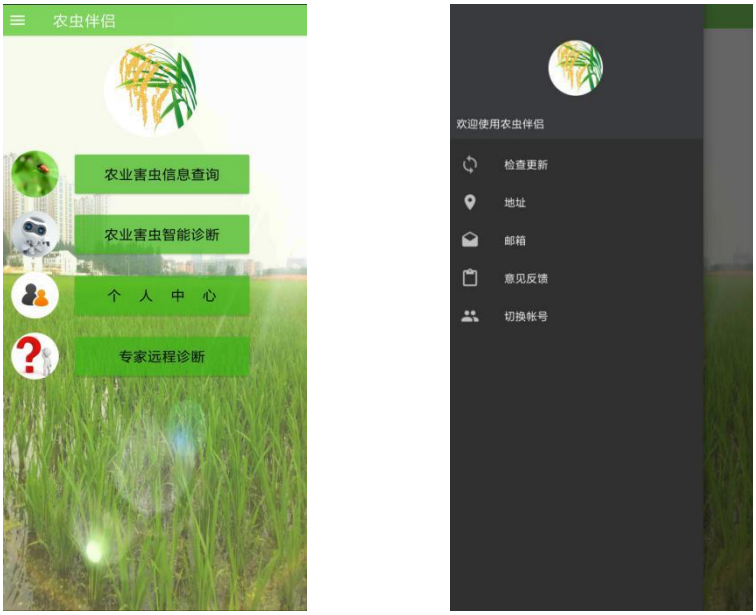
2.3.2 主界面设计

手机客户端所针对的使用人群主要是农作物种植户、植保人员以及对农业害虫感兴趣的用户，这就要求应用软件应具有简单方便的操作性和清晰明了的逻辑性，而通过 Material Design 控件设计 Android 客户端 UI 界面可以很好地满足这些需求。Material Design 是 Google 在 2014 年的开发者大会上首次提出的设计语言，相比于较早的 Android Designs 设计语言，Material Design 包含了视觉、运动、互动等特效并且解决了 Android 平台界面风格不统一的问题。

通过 Google 提供的 Design Support 库来进行安卓客户端 UI 界面 Material 化，使用 Toolbar 代替 ActionBar 实现 Material Designs 风格的导航控件。首先在 res/values 目录下找到 styles.xml 样式文件，修改 AppTheme 主题为 Light.NoActionBar 淡色主题，然后在 Activity

对应的 layout 布局文件中添加 Toolbar 控件，设置属性，最后通过调用 setSuppportActionBar 方法传入 Toolbar 实例。

使用 DrawerLayout 与 NavigationView 相结合实现滑动菜单功能，滑动菜单最主要的作用就是将菜单选项隐藏起来，节省屏幕空间并实现绚丽的动画效果。首先在 app 目录下的 build.gradle 文件中添加两行依赖，分别是 Design Support 库与开源项目 CircleImageView。创建主布局文件添加 DrawerLayout 控件作为整合界面的根布局，在其中放置两个直接子控件分别是 FrameLayout（主屏幕中显示的内容）与 NavigationView（滑动菜单中显示的内容）。然后在 res/menu 目录下创建 nav_menu.xml 文件用于设置具体的菜单项包括：检查更新功能、地址、邮箱、意见反馈、以及帐号切换功能。创建 nav_header.xml 布局文件，用 CircleImageView 实现客户端 logo 图片圆形化功能。通过 app:menu 和 app:HeaderLayout 属性将 nav_menu 与 nav_header 设置进 NavigationView 中，实现布局文件。最后通过在活动中获取 NavigationView 实例，并在 onNavigarionItemSelected()方法中添加菜单项对应的逻辑处理，实现整个 Material 风格的 UI 界面。用户界面如图 2.9 所示：



(a) 主界面　　**(b) 滑动菜单**

图 2.9 用户界面

2.3.3 农业害虫信息查询模块

信息查询模块中已录入的主要农作物害虫种类有 220 多种，可以通过下滑浏览或者关键字查找的方式快速地找到想要了解的害虫，包括学名、形态特征、发生分布地区、危害特性、生物防治以及化学防治等内容，亦可通过该模块浏览数据库中保存的害虫图片，帮

助用户快速地了解想要防治的害虫，提供相应的防治建议和措施。

信息查询部分主要使用 SearchView 控件结合 ListView 控件实现，列表视图 ListView 是用于展示大量数据的，通过上下滑动的方式可将屏幕外的数据滚动到屏幕内，同时屏幕内原有的数据则会滚出屏幕。但是数组数据是无法直接传递给 ListView 的，需要借住适配器来完成。首先自定义 ArrayAdapter 适配器，重写 getView()方法并通过 convertView 参数来缓存布局，提高运行效率，添加内部类 ViewHolder 用于提供实例。然后设计子项布局文件，获取 ListView 实例后借助自定义的 ArrayAdapter 适配器传递当前上下文、子项布局以及所需的适配数据。最后利用 setAdaper()方法将构建好的适配器传入 ListView，实现列表和自定义适配数据之间的关联。

SearchView 是 google 提供的搜索框视图控件，可结合 Filterable 接口来实现 ListView 的过滤功能。自定义 Filterable 接口中的 Filter 类，通过复写 Filter 类中的 performFiltering()方法实现过滤规则的修改，使用 TextUtils.isEmpty()方法判断搜索框中的内容，如为空就显示全部数据，否则过滤出符合规则的数据，复写 SearchView 控件中的 setOnQueryTextListener 方法为搜索框设置文本监听事件，当搜索内容改变时会自动触发 Filterable 接口中的 getFileter 方法来获取 Fileter 实例。最后搜索框中输入的文字会作为参数传到 Fileter 实例构造函数中，进行筛选后通过适配器的 notifyDataSetChanged()方法来通知 ListView 刷新数据，显示过滤后的农业害虫信息界面。信息查询界面如图 2.10(a)，2.10(b)：

害虫图片浏览部分主要是使用 CoordinatorLayout、AppBarLayout 结合 RecyclerView 实现嵌套滑动机制；使用 PhotoView 实现图片预览查看。RecyclerView 相比于 ListView，以一种耦合度更低的方式来复用 ViewHolder，并可以轻松实现网格式布局，非常适用于图片浏览，首先需要在项目中添加依赖，继承并复写 RecyclerView.Adapter 适配器。然后在 onBindViewHolder()方法中通过 Glide 加载网络害虫图片实现子项数据的处理。最后在 activity 中获取列表控件实例，利用 GridLayoutManager 实现自定义的网格效果。

CoordinatorLayout 是加强版的 FrameLayoutbu 布局，可以监听子控件的各种事件，协调元素之间的依赖关系，配合 AppBarLayout 可以解决 Toolbar 被遮挡的问题，实现下拉显示与上划隐藏功能。PhotoView 是一款开源的图片查看库继承自 ImageView，支持单点或多点触摸，添加依赖同步后直接实现其对象实例，通过 setImageBitmap()方法添加位图即可进行害虫图片放大缩小预览。图片浏览界面如图 2.10(c)：

(a) 信息查询界面　　**(b) 害虫详情介绍界面**　　**(c) 图片浏览界面**

图 2.10 信息查询模块

2.3.4 农业害虫地图分布模块

地图分布模块主要是为用户提供识别害虫在地图上的位置分布，所有经过上传识别过的图片都会在数据库中保存其相应的地理位置信息、GPS 信息、类别信息和图片存储位置信息等，用户可以在该模块中了解附近和全国范围内的害虫分布，并且可以通过点击害虫图片覆盖物来获取当前图片物种与用户位置的距离，点击详细按钮还可以获取该害虫的具体信息。实现步骤包括：使用百度地图定位、位置范围内数据的获取与处理、批量地图覆盖物实现。

该模块中最基础的功能便是是百度地图的显示和定位功能的实现，具体流程如下：

(1) 利用百度提供的 Android 地图 SDK 接口实现定位功能，在百度地图开放平台下创建应用，并且在 Android Studio 项目中获取 SHA1 指纹信息，注册生成 API Key。

(2) 下载基础地图和定位功能两个 SDK 并解压，在 app/libs 目录下添加需要的.jar 文件，将压缩包中剩余文件添加至 src/main/jniLibs 目录，添加本地依赖同步后便可进行开发。

(3) 在 AndroidManifest.xml 文件中正确填写 API Key，添加网络定位、GPS 定位、Wifi 网络访问等权限，需要注意的是 Google 在 Android 6.0 版本中引入了动态权限获取机制，要在调用涉及相关权限的代码时，使用系统接口来动态申请相应权限。

(4) 在布局文件中添加 MapView 控件用于显示百度地图界面，在 activity 中通过调用 SDKInitializer 工具类的 initialize()静态方法进行初始化操作，复写当前活动的 onResume()、

onPause()和 onDestroy()方法用于管理 MapView，保证资源能够及时释放。

(5) 实现定位，创建 LocationClient 实例，注册监听器用于获取回调的位置信息，通过 LocationClientOption 类设置百度地图的定位模式、坐标系、地址信息以及位置更新等信息，调用 LocationClient 实例的 Start()方法开启定位服务。定位的结果会回调到监听器中，通过 BDLocation 参数提供的方法可获取当前位置的纬度、经度、以及具体的地址信息等，在默认模式下会优先使用 GPS 定位，在无法接收 GPS 信号时会使用网络定位。

(6) 在默认情况下地图显示的是北京市中心的位置，为了显示当前位置的光标，需要进一步设置。百度地图 SDK 的 API 中提供了一个 BaiduMap 类作为地图的总控制器，首先调用 MapView 的 getMap()方法获取 BaiduMap 实例，然后通过 MapStatusUpdateFactory 工厂类的 zoomTo()静态方法设置百度地图的缩放级别以达到需要的视图效果；newLatlng()静态方法设置经纬度。最后将经纬度信息以及工厂类的返回参数传入 BaiduMap 即可实现当前位置的光标显示。

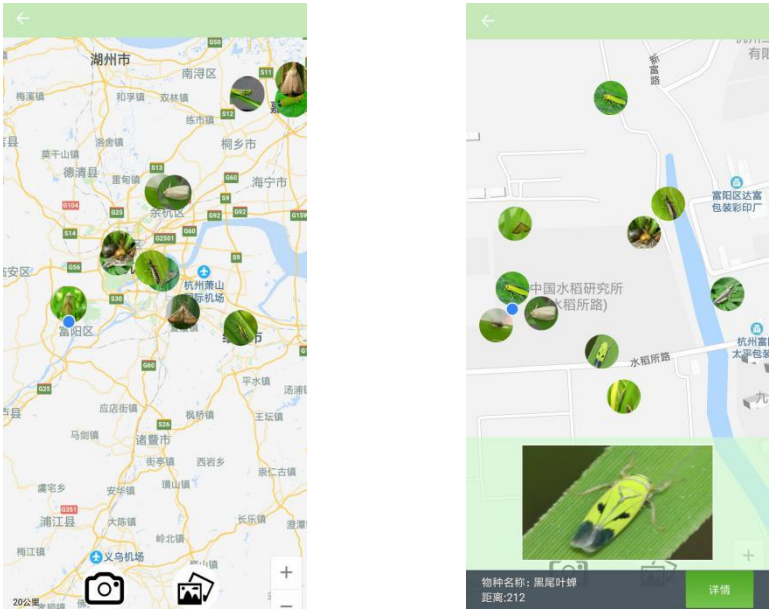
用户上传识别过的图片都会在数据库中保存其相关信息，要在客户端实现农业害虫在地图上的位置分布，就需要获取这些信息数据并进行相应的处理。

数据请求：要获取害虫图片的数据信息，首先需要确定当前手机屏幕内的经纬度范围，地图 API 中的 MapStatus 类可提供当前的地图状态，通过 BaiduMap 的 getMapstatus()方法获取其实例，在实例的 bound 字段中可以获取到当前屏幕可见显示范围的最大外接矩形数据，包括东北角和西南角的经纬度信息，进而得到经纬度上下限值。通过 OkHttp post 携带范围参数值上传服务器进行数据请求。随着用户的触摸和点击等手势操作，地图的状态会不断地发生改变，而显示的数据也需要随着地图的状态不断更新，通过创建 OnMapStatusChangeListener 监听器，用于监听地图状态，复写监听器中的 onMapStatusChangeFinish()方法，在地图状态改变后即可实时向服务器进行请求更新数据。

数据处理：服务器与客户端的交互使用的是 Json 数据格式，通过 Gson 开源库进行解析，将获取的 Json 字符串数据解析为 java 对象（已序列化），首先配置依赖，新建 Gson 实例对象，借助 TypeToken 将期望解析的数据类型传入 fromJson()方法中即可实现数据处理。

数据处理完成后通过将对应害虫的图片覆盖在地图上，以最直观的方式展示农业害虫的分布信息。地图 API 中提供了 OverlayOptions 类用于添加覆盖物，通过实现其子类 MarkerOptions 添加对图片覆盖物的操作，包括经纬度信息、位图、标题等。从获取的数据中得到害虫图片的存储信息，通过 Glide 向服务器请求进行网络图片加载。

由于要实现批量的图片覆盖，为防止网络延迟和图片缓存引起的线程阻塞，需要应用异步消息处理机制加载图片覆盖物。使用 AsycnTask 工具可以方便地在子线程中实现延时处理，并在 UI 主线程中进行更新操作，重写 AsycnTask 类中的 doInBackground()方法，在其内部实现农业害虫图片的缓存和数据处理，重写 onPoseExecute()方法，在其内部实现覆盖物的更新操作。创建 AsycnTask 实例通过执行 excute()方法启动任务即可进行地图的批量覆盖。害虫地图分布部分如图 2.11 所示：



(a) 害虫地图分布　　**(b) 害虫详情**

图 2.11 害虫地图分布模块

2.3.5 农业害虫智能诊断模块

智能诊断模块中提供了自定义相机拍照与图库选取两种农业害虫图片上传方式，图片经服务端调用深度学习算法后输出置信度最高的 5 类害虫，在客户端反馈界面中顺序显示，包括农业害虫名称与其对应的相似度信息。点击详细按钮还可以获取该害虫的基本信息和防治措施。

Android 系统提供了两种使用手机相机资源实现拍照功能的方法，一种是直接通过 Intent 调用系统相机组件，这种方法快速方便，但局限性较强，只适用于直接获得照片的场景，另一种是使用相机 API 来定制自定义相机，适用于需要定制相机界面或者开发特殊相机功能的场景。农业害虫的体积较小，在拍摄的照片中难以捕捉到具体目标，为了提高识别的准确度，需要将目标害虫放置在相机预览界面的固定识别框中进行拍摄，自定义相机无疑是最好的选择，使用 Camera API 接口实现自定义相机，步骤如下：

(1) 为防止调用相机时出现崩溃，需要检查设备是否存在相机资源。首先添加 Camera 相关权限，然后创建 PackageManager 实例，该类是 Android 系统提供的服务管理类，可用于获取应用程序信息，最后通过 hasSystemFeature()方法进行判断，测试该设备是否有相机资源。

(2) 创建预览类，设置拍照监听器，继承 SurfaceView 视图类用于显示相机的实时预览画面；实现 SurfaceHolder.CallBack 状态监听接口，首先重载 surfaceCreated()，在该方法中调用 Camera.open()、Camera.setPreviewDisplay()，实现相机资源的获取和预览视图的连接。然后覆写 surfaceChanged()，在该方法中调用相机的 startPreview()用于开启相机预览，并通过 getParameters()来获取 Camera 的参数，用于设置预览尺寸、拍摄尺寸、预览帧、拍摄帧、曝光、聚焦、颜色等特效。最后覆写 surfaceDestroyed()，在该方法中调用相机的 stopPreview()、release()来停止预览，释放资源。

(3) 建立预览界面的固定识别框，继承 AppCompatImageView 类，绘制半透明相框，创建布局文件，将预览类、半透明相框与设计好的用户界面控件融合在一起。

(4) 拍照并保存文件，通过 Camera.takePicture()方法进行拍照，并实现 Camera 类的 PictureCallback 接口，用于接收和保存 JPEG 格式的图片。

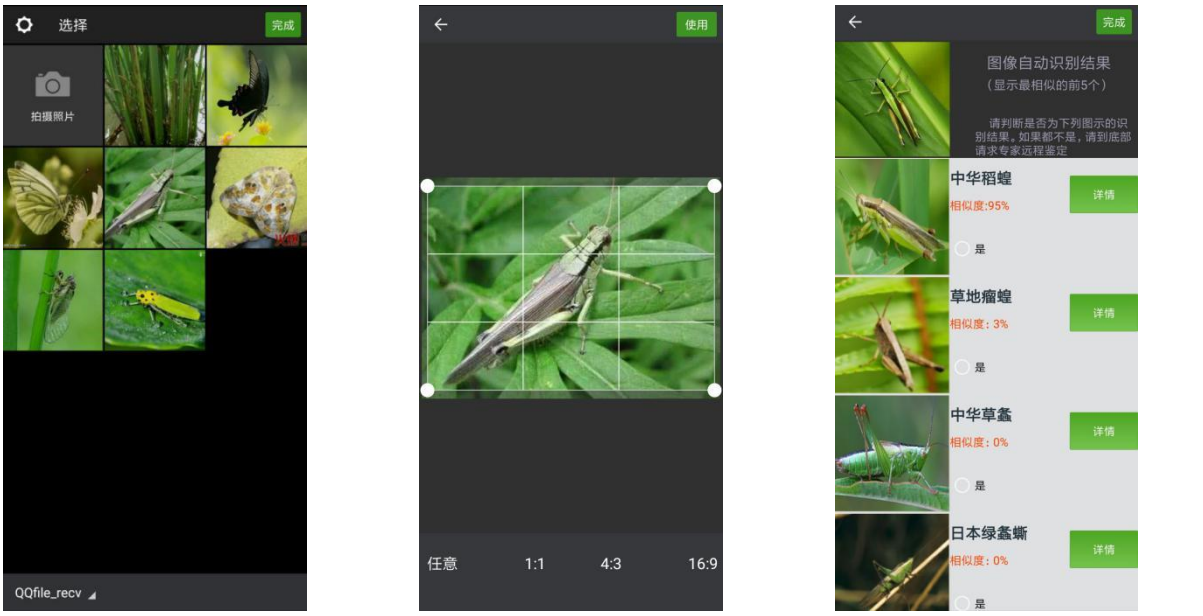
(5) 释放相机资源，相机是一个共享资源，必须对其生命周期进行细心的管理。当相机使用完毕后，应用程序须通过 Camera.release()方法将其正确释放，以免其它程序使用时，发生冲突。自定义相机实现效果如图 2.12 所示：

图 2.12 自定义相机

图库选取界面通过 RcyclerView 控件和 Glide 库来实现图片列表展示，良好的缓存机制

可以在下滑的瞬间更新图片。使用 PopupWindow 控件用于弹出所有包含图片的文件夹，并显示每个文件夹中的图片数，如图 2.13(a)。添加依赖后通过 SimpleCropView 图片裁剪库实现图片的裁剪功能，startLoad()方法用于异步加载 Uri 获取图片路径，startCrop()方法用于裁剪和保存。为了满足应用需求，分别提供了自由比例、1 比 1、4 比 3、16 比 9，4 种比例的裁剪框选项，通过裁剪使得样本可以达到使用的标准，如图 2.13(b)。

点击“使用”按钮，以表单文件形式 post 上传害虫图片、GPS、地理位置等信息，服务端解析后，调用深度学习算法进行识别检测，客户端可在 onResponse()方法中获取回调参数，得到识别信息并跳转至反馈界面，如图 2.13(c)。反馈界面通过 ListView 进行实现，顺序显示识别的农业害虫名称与其相似度，点击详细按钮可获取该害虫的基本信息以及相应的防治措施。对于相似度较低且不确定的识别结果还可以下滑到界面底部请求专家远程诊断，用户可在个人中心查看专家的诊断反馈信息。



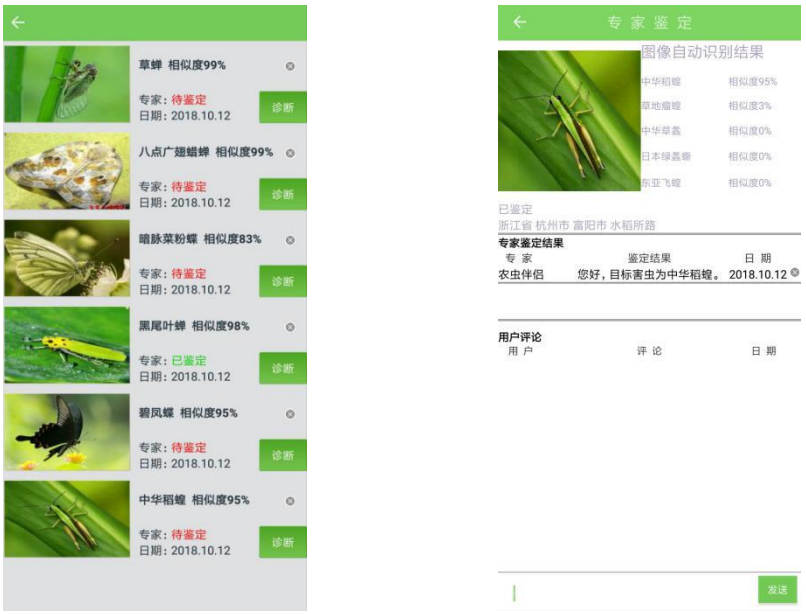
(a) 图库选取界面　　**(b) 图片裁剪界面**　　**(c) 识别反馈界面**

图 2.13 智能诊断模块

2.3.6 专家远程诊断模块

该模块主要是对专家开放，用于实时反馈用户有疑惑的农业害虫识别结果，进入该模块后会自动向服务器发送请求，返回结果通过 Gson 解析为 list 列表数据，借助自定义的 ArrayAdapter 适配器，将数据适再适配到 ListView 中，通过列表的形式显示。在列表的子项布局中包含有害虫图片、深度学习算法识别的相似度、专家诊断的状态、以及用户请求的具体日期。如图 2.14(a)。

具体的诊断细节通过点击“诊断”按钮后进行，用户发送的每份专家诊断请求都会在数据库中保存其单独的 id，开启“诊断”后会通过 id 向服务器请求诊断详情类集，类中含有主键 id、诊断主题 id、诊断专家名、诊断状态、诊断内容、以及诊断时间等字段，在该界面中会结合 listView 进行显示，2.14(b)。用户的回复内容以同样的方式进行。为了让专家更加准确地判断目标害虫，还添加了地理位置信息和图片预览功能。



(a) 用户诊断请求界面　　**(b) 诊断界面**

图 2.14 专家远程诊断模块

2.4 本章小结

本章设计了一个基于 Android 平台的农业害虫识别客户端 App，从程序架构设计方面分析了开发模式和项目结构，通过使用 MVP 模式实现了程序模块内部的高聚合和模块之间的低耦合。利用 Okhttp 开源库实现了客户端与服务器之间的网络通信，使用 Glide 库实现了客户端网络图片的缓存与加载。在客户端开发方面分别介绍了用户登录界面、主界面、害虫信息查询、害虫地图分布、害虫智能诊断以及专家远程诊断的具体实现方式和步骤，同时还展示了各部分的功能和实现的效果。

第三章 云服务平台的设计与实现

该系统的服务端是基于 Java Web 的后台开发，在企业级集成开发环境 MyEclipse 下进行搭建设计，通过第三方服务器软件 Tomcat，结合 Caffe 深度学习框架进行部署[45-49]，实现了农业害虫图片的识别诊断、图片文件存储、数据存储、以及客户端版本更新等功能。首先在微机上进行逻辑开发和部署，然后通过“动态域名解析软件”实现内网穿透，进行功能的调整和测试，最后将 web 程序打包成 War 包部署至云服务端投入使用。

3.1 服务端开发设计

3.1.1 应用程序服务器搭建

应用程序服务器主要是为客户端程序提供后台服务，通过在 Windows Server 2012 服务器系统下配置 Tomcat 进行实现。Tomcat 作为轻量级的开源应用服务器，性能稳定且免费，被广泛地应用在中小规模的 Java Web 项目中，可以支持运行 Serlvet 与 JSP，同时 Tomcat 也扩展了一些 App Server 的功能，如 JNDI，数据库连接池，用户事务处理等[50-51]。

在 Windows Server 系统下 Tomcat 有解压版和安装版两种配置方式，解压版主要应用于后台开发设计，在配置好 JDK(Java 开发工具包)后直接解压到任意文件夹下配置环境变量即可，通过在 MyEclipise 中添加 Tomcat 后还可以开启自动部署功能。安装版的 Tomcat 主要应用于项目部署，在云服务器中安装后可在后台自动运行。Tomcat 中默认的运行端口号为 8080，在 conf 目录下的 server.xml 文件中通过修改可开启多个应用服务，本机设置的端口号为 8081。

访问 Tomcat 服务器的方式有很多，可通过域名 localhost 添加端口号进行访问，但仅限于在本机上进行；通过 IP 地址添加端口号可实现局域网内访问，但无法满足客户端与服务器的测试需求；通过花生壳等第三方端口可实现内网穿透（NAT 穿透），用于后台功能测试；在云服务器上可通过域名直接进行访问。

Servlet 是用 java 编写的服务端程序，作为 java Web 的核心，实现了 javax.servlet.Servlet 接口，可用于实现数据库读取，权限检查等业务逻辑处理。Web 项目中通过继承 HttpServlet 类并覆写 doGet()与 doPost()方法即可实现客户端 app 的请求响应，具体响应步骤如图 3.1 所示，首先 Tomcat 容器启动后会将 Servlet 类加载到内存中，通过调用 init()方法进行初始化，客户端的所有的请求会通过 Tomcat 接收并解析，封装成 HttpServletRequest 对象，作为参数供 Servlet 中的 doPost()与 doGet()方法调用。接着实现业务逻辑处理，Tomcat 会将

输出流封装为 HttpServletResponse 对象，通过设置属性控制输出响应内容。最后在服务器端停止运行且卸载 Servlet 后，调用 destroy()释放占用的资源。

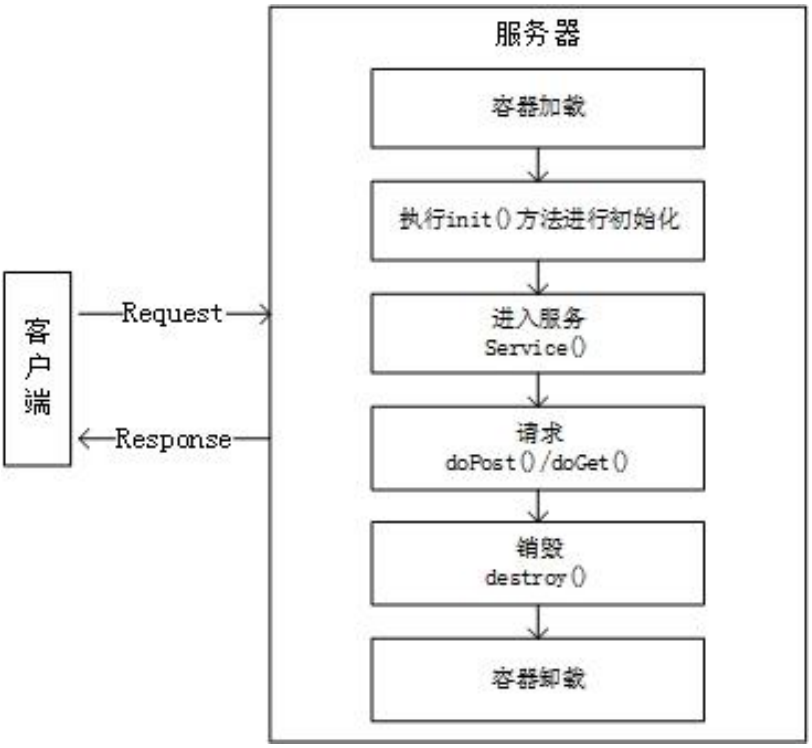


图 3.1 Servlet 响应步骤

3.1.2 二进制流解析与图片压缩

安卓客户端是以二进制流的方式上传图片与表单数据的，Servlet 中想要获取其内容必须依据 HTTP 协议规定的格式解析客户端提交的请求，而利用 Apache Commons Fileupload 开源类库可以快速实现这些工作。Commons 是 Apache 开放源代码组织的一个 Java 子项目，其中的 FileUpload 是用来处理 HTTP 文件上传的类库。将 commons-fileupload.jar 包与 common-io.jar 包导入项目中的 WEB-INF/lib 目录下，添加基础依赖，即可进行使用。

首先需要在 Servlet 中 import 导入工具包，并覆写 doPost()方法，创建磁盘文件工厂 DiskFileItemFactory 类，实现处理逻辑，通过工厂类的 setSizeThreshold()方法设置上传文件的缓存大小，setRepository()方法用于设定磁盘文件的缓存路径。然后创建 ServletFileUpload 文件处理类，解析客户端上传的 request 请求，并将返回结果放置在 List<FileItem>文件列表中。最后遍历文件列表，通过 isFromField()方法判断是否为文本域，如果为“true”则为表单文件，获取对应的表单属性名和 value 值，如果为“false”则是图片文件，新建 io 流，将其保存到指定目录中。

随着手机相机拍照水平的提高，图片质量以及大小也在不断增长，为了保证农业害虫

的识别率以及客户端缓存图片的速率，需要在识别后将害虫图片进行压缩，以保证客户端地图模块的害虫分布显示效果。在本论文中主要通过 BufferedImage 类进行实现，BufferedImage 类加载的图片可以在内存中生成一个图像缓冲区，利用这个缓冲区，可以实现图片大小变换、灰度图、透明度设置等操作。首先使用 createGraphics()方法创建画布，然后通过画布的 drawImage()方法进行图片的压缩，最后保存害虫图片用于输出。

3.1.3 基于深度学习的农业害虫识别算法的调用

手机客户端上传的农业害虫图片解析完成之后，需要调用相应的深度学习算法进行识别诊断。本课题中算法的研究是在 Caffe 深度学习框架下开展的，（具体的深度学习算法研究见第四章），但 Caffe 作为纯粹的 C++/CUDA 架构是无法直接与 Java Web 进行交互的，为了解决算法模型的调用问题，论文中采用了 JNI 本地方法接口[52-55]，通过生成 DLL 动态链接库来进行实现。

JNI（Java Native Interface）作为 java 平台的一部分，在设计之初主要就是为了实现 Java 与 C/C++编程语言之间的交互，区别于一般 C/C++代码的动态链接库构建流程，Caffe 作为一个深度学习框架，首先需要将 caffe-windows 源代码重新编译至服务端项目的 WebRoot 目录下，并添第三方库，实现 Caffe 框架在服务端的重构（重构的前提是在微机上实现深度学习框架的部署，具体实现方式见第四章）。然后在 Visual Studio 中对 caffe-windows 提供的识别文件进行修改并生成 DLL 动态链接库，最后在 Java Web 程序中调用 Native 本地方法接口，添加模型文件，均值文件、权值文件和标签文件的地址参数即可实现农业害虫图片的识别。JNI 接口的具体实现步骤如下：

(1) 在 MyEclipise 项目中创建 CaffeInsectIdentification.java 类，用于提供静态的 Native 本地方法接口，接口方法中需要声明农业害虫图片以及识别文件的路径参数。在该类中还需要添加静态代码块，用来调用 System.loadLibrary()方法，使得 JVM 虚拟机在第一次加载该类的同时，实现农业害虫识别算法动态链接库的装载。

(2) 在 MyEclipise 中使用 javah 命令来编译 CaffeInsectIdentification 类，生成.h 头文件。

(3) 在 Visual Studio 中创建 DLL 动态链接库项目，添加步骤 2 中生成的头文件、JDK 中的 jni.h 和 jni_md.h 头文件、caffe-windows 源代码中的测试文件。

(4) 修改测试文件，修改的内容包括：头文件中 Java 地址参数的命名、农业害虫图片的接收、异常处理、Caffe 深度学习模型的添加、输出结果的格式转换。最后编译生成动态链接库。

(5) 将动态链接库配置在环境变量中，由于 Java JDK 在安装时已将 bin 目录配在了环境变量中，所以优先添加在该目录下，方便在云服务器中的实现。

(6) 在 Servlet 中直接调用步骤 1 中的静态方法接口，传入对应的方法参数即可实现农业害虫图片的识别诊断。

3.1.4 云服务器部署

云服务器的优势在于支持资源的弹性扩充，部署起来方便快捷，在实际应用中不需要配置网络以及安装系统，无需考虑硬件维护的问题，运维流程和成本相对较低。实现云服务器的部署首先要将 java web 项目生成 war 包，然后添加到 Tomcat 的工程目录下，最后启动 Tomcat 实现项目的解压与发布。只需要在客户端中添加云服务器的域名，就可以直接进行外网访问，实现实时的农业害虫识别检测。云服务器部署的具体流程如下：

(1) 创建实例，本论文中选用的是 Windows Server 2012 版服务器操作系统。

(2) 项目配置，在微机上通过 Mstsc 实现与云服务器的远程桌面连接，安装并配置 JDK、Opencv 与 Tomcat，在微机上将项目导出成 war 包，放置在 tomcat webapp 目录下，并设置端口号为 8081。

(3) Caffe 框架配置，由于 Caffe 深度学习框架已重构至项目中，所以不需要在云服务器上再次搭建，只需将农业害虫识别算法 DLL 动态链接库添加至 JDK bin 目录下即可。

(4) 内存设置，Tomcat 默认的堆内存为 128M，在项目应用中容易造成溢出问题，需通过-Xms 与-Xmx 调整应用程序的初始内存和最大内存。在服务器中通过 JNI 本地方法接口实现了 DLL 动态链接库的调用，但这部分内存并非属于堆内存，而是由直接内存进行管理，如果不进行合理分配同样会导致内存溢出，需通过-XX：MaxDirectMemorySize 进行调整。

(5) 启动安装版的 Tomcat，即可自动解压并发布项目，实现云服务端的部署。

3.2 数据库搭建

3.2.1 数据库的访问与优化

服务端的数据库管理系统采用的是 Oracle 旗下 MySQL[56]，后台需要通过相应的数据库驱动程序才能使用该系统，在 Java Web 中访问数据库主要是通过 JDBC[57]实现。作为 java 语言访问数据库的标准 API，JDBC 将 SQL 语句的执行、查询遍历、以及数据库的连接等操作定义为接口，封装在 java.sql 包内，通过在项目的 WEB-INF/lib 目录下添加驱动 jar 包即可进行使用，具体实现步骤如下：

(1) 注册数据库驱动。数据库厂家会提供 Driver 驱动程序接口用于实现访问，在使用数据库前必须先装载该接口，生成对应的 DriverManager 驱动管理器类。在 Servlet 中可通过 registerDriver()静态方法进行注册，或者通过创建驱动对象生成该类。

(2) 获得连接对象。要连接数据库，须在驱动管理器的 getConnection()静态方法中传入数据库的用户名、密码、数据表路径、来获得 Connection 连接对象。

(3) 创建 Statement 对象。通过 Connnection 连接对象的 creatStatement()方法可创建 Statement 对象实例，用于执行静态 SQL 语句并返回结果。

(4) 遍历 ResultSet。执行结束后返回的数据会被封装在 ResultSet 中，通过 next()方法可以逐行遍历输出结果。

(5) 释放资源。先关闭 ResultSet，然后是 Statement，最后是 Connetion，顺序释放占有资源。

(6) 开发过程中通过 DAO（Database Access Object）数据库操作对象实现对 JDBC 的优化，将业务逻辑与数据库访问相分离，降低程序的耦合性。DAO 模式的关键在于设计数据库操作类，实现对数据库的注册、连接、增删改查、释放等操作，并提供相应的接口方法供 Servlet 调用。对每个数据表封装一个 DAO 类，避免在业务代码中混杂 JDBC 调用语句，使得业务逻辑处理更加清晰。

除了 DAO 设计模式外还可以通过 DBCP（Database Connexction Pool）连接池实现对数据库连接的优化，客户端用户的每次数据请求都需要数库创建连接，而在大量并发访问的情况下，创建与断开 Connection 会消耗巨大的 IO 资源以及时间，非常容易造成服务器的内存溢出，通过连接池可以实现 Connection 的复用，以减少频繁的连接创建，提高数据库操作的性能。

3.2.2 数据库表设计

数据库关系表如图 3.2 所示：

图 3.2 数据库表关系图

用户注册信息数据表（USERINFO）用于存储用户的注册信息，包括用户 id（由数据库编号，自动增长）、用户名、密码、身份、地址、创建时间，其中身份是用来在登录时区分用户类型，为专家开启客户端远程诊断功能。表的结构如下所示。

表 3.1 用户注册信息表结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	INTEGER	NO	YES	AUTO_INCREMENT	主键
user_name	VARCHAR(45)	NO	NO	NULL	用户名
password	CHAR(64)	NO	NO	NULL	密码
user_identity	VARCHAR(45)	NO	NO	NULL	身份
address	VARCHAR(255)	NO	NO	NULL	地址

create_time	TIMESTAMP	NO	NO	CURRENT_TIMESTAMP	创建时间

第三方登录信息数据表（3PARTYINFO）用于实现第三方登录功能，包括第三方登录 id、用户 id、第三方登录类型、第三方登录标识。客户端 SMS 手机短信登录验证或登录 QQ 验证后会提供唯一的登录标识，通过和已注册的账号信息绑定后即可实现快速登录。表的结构如下所示。

表 3.2 第三方登录信息表结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	INTEGER	NO	YES	AUTO_INCREMENT	主键
user_id	INTEGER	NO	NO	NULL	用户 id
login_type	VARCHAR(45)	NO	NO	NULL	三方登录类型
identifying	VARCHAR(45)	NO	NO	NULL	三方登录标识

害虫图像信息数据表（IMAGEINFO）用于记录用户上传图片的识别信息，包括害虫图片 id、用户 id、识别结果、地址、经纬度信息、图片存储位置、请求与诊断状态、创建时间。其中识别结果中包含 5 类置信度最高的农业害虫名与相似度。表的结构如下所示。

表 3.3 害虫图像信息表结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	INTEGER	NO	YES	AUTO_INCREMENT	主键
user_id	INTEGER	NO	NO	NULL	用户 id
address	VARCHAR(255)	NO	NO	NULL	地址
result	VARCHAR(255)	NO	NO	NULL	识别结果
longitude	DOUBLE(10,6)	NO	NO	NULL	经度
latitude	DOUBLE(10,6)	NO	NO	NULL	纬度
image_path	VARCHAR(255)	NO	NO	NULL	图片存储位置
request_state	BOOLEAN	NO	NO	FALSE	是否请求诊断
diagnose_state	BOOLEAN	NO	NO	FALSE	诊断状态
create_time	TIMESTAMP	NO	NO	CURRENT_TIMESTAMP	创建时间

诊断信息数据表（DIAGNOSEINFO）用于记录远程专家诊断的结果，包括诊断信息 id、请求诊断的害虫图片 id 和用户名、诊断专家用户名、诊断结果、身份和诊断的时间信息。

表的结构如下所示。

表 3.4 诊断信息表结构

字段名	数据类型	是否为空	是否主键	默认值	描述
id	INTEGER	NO	YES	AUTO_INCREMENT	主键
image_id	INTEGER	NO	NO	NULL	图片 id
from_uname	VARCHAR(45)	NO	NO	NULL	评论用户名
to_uname	VARCHAR(45)	NO	NO	NULL	目标用户名
diagnose_result	VARCHAR(255)	NO	NO	NULL	诊断结果
user_identity	VARCHAR(45)	NO	NO	NULL	身份
create_time	TIMESTAMP	NO	NO	CURRENT_TIMESTAMP	创建时间

3.3 本章小结

本章实现了云服务平台的搭建，通过 Tomcat 在 Windows Server 系统下搭建轻量级应用服务器，同时编写 Servlet 业务逻辑以实现二进制数据流的接收与解析，通过压缩识别后的农业害虫图片，提高了客户端缓存图片的速率以及地图模块害虫分布的显示效果。利用 JNI 本地方法接口实现了服务端对深度学习算法模型的调用以及农业害虫的识别诊断，详细介绍了 Web 项目在云服务端的部署与内存调优。利用 JDBC 实现了 Web 项目对 MySQL 数据库的访问，并通过使用 DAO 设计模式与连接池实现了数据库的优化。

第四章 基于深度学习的农业害虫图像识别算法的研究

通过使用深度学习框架可以免去复杂的神经网络代码编写，依据需求，使用已有的深度学习模型加以改进和优化，获取最佳的算法模型。目前主流的深度学习框架有Tensorflow[58-59]、Caffe、MXNet[60]、Torch[61]和 PyTorch[62]等，被广泛应用于图像处理，语言翻译，语音识别，自动驾驶等方面。本论文中农业害虫识别算法是基于 Caffe 框架进行研究的，Caffe 作为纯粹的 C++/CUDA 架构，专精于图像处理，其模块化的构建理念可以实现在项目工程中的快速应用，文本形式的模型定义可以方便地进行优化设置。本章将针对 caffe 框架　　的搭建与算法研究进行展开。

4.1 Caffe 深度学习框架的搭建

Caffe 深度学习框架版本较多，由伯克利大学 BVLC（Berkeley Vision And Learning Center）进行维护的主版本，可以编译配置到 Linux 和 Windows 系统，支持英伟达的 CUDA 加速，但是对 CPU 的优化效果较差。微软也维护了一个 Caffe 版本，主要是为了简化 Windows 下的安装配置与操作，支持基于 CPU 和 CUDA 的算法实现。英特尔维护了两个版本的 Caffe 框架，第一个版本优化了 Intel CPU 的实现算法，主要是为了在 Linux 服务器平台上做训练工作。第二个版本优化了基于 Intel 显卡的加速算法，支持 Windows 和 Linux 平台。

由于本论文中涉及的服务器与深度学习研究都是在 Windows 系统下进行，所以优先选择微软维护的版本在台式机上进行搭建，处理硬件 CPU 为 Intel（R）Core（TM）i5-4590@3.30GHz，GPU 为 NVIDIA GeForce GTX 1080，主机内存为 16.00GB，在 Caffe 提供的模型基础下进行训练以及测试。具体框架的搭建过程如下：

(1) 在官方 Github 提供的链接中下载 caffe-windows.zip 源码压缩包进行解压，在 windows 目录下找到 CommonSettings 文件修改其后缀名为 props。由于需要在 GPU 下进行训练测试，还需要安装 CUDA 与 CuDNN 进行协助，通过 CUDA 架构可以使用 GPU 处理大量并行运算，弥补 CPU 在深度学习算法运算量上的不足，而 CuDNN 是英伟达专门针对深度神经网络设计的 GPU 加速库。

(2) 进行 GPU 模式配置，在 CommonSetting 文件中修改 CudaVersion 版本号为当前 CUDA 版本（本机为 8.0 版），修改 UseCuDNN 字段为 true，表示使用 CuDNN 进行运算加速。GPU 模式配置如图 4.1 所示。

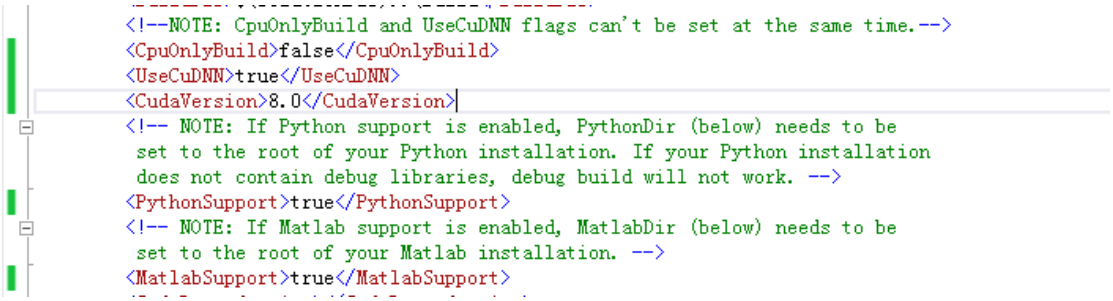


图 4.1 GPU 模式配置

(3) 使用 Visual Studio 打开 windows 目录下的 Caffe.sln 工程文件，在项目中先找到 libcaffe.lib 文件，进行编译并重新生成，目的就是下载包括 boost，opencv2.4.10，gflags，glog，hdf5，lmdb，LevelDB，OpenBLAS，protobuf 等在内的预编译依赖包，下载结束后会在在同级目录下生成 NugetPackages 文件夹。

(4) 重新编译整个解决方案，运行项目，会在 Build 目录中生成发布版本，将该目录下的 release 与 debug 配置在环境变量中，即可实现 Caffe 深度学习框架在 Windows 系统下的搭建。

4.2 基于深度卷积神经网络的农业害虫识别模型的训练与测试

深度学习的实践最早始于卷积神经网络在图像识别方面的应用，在传统的模式识别方法[63-66]（KNN、SVM、NN 等）中，需要人为设计算法实现图像的特征提取[67-70]，并没有发挥出计算机提取特征的能力，同时算法的设计更是要考虑到图像特征的旋转不变性、尺度不变性以及光照不变性等问题，最后还要选择合适的机器学习方法，输入提取的特征数据，实现分类与识别，对于人为判断的依赖性较强。而卷积神经网络则是将图像特征提取交给计算机处理，通过不同 filter 的卷积解决各种不变性问题，实现了识别分类的最优化求解。本论文中主要通过 Caffe 框架中提供的卷积神经网络建立模型。

4.2.1 农业害虫识别模型的训练

在目前已收集的农业害虫样本图片中，水稻、小麦、玉米、大豆等农作物的主要害虫种类就有 34 种，共 10846 幅原图，其中 9043 幅作为训练样本，通过添加噪声、变化亮度、对比度、翻转等方法来扩充数据集，剩余的 1803 幅作为测试样本，在应用较广泛的 CaffeNet 模型[71-72]、VGGNet 模型[73]与 ResNet 模型[74-75]基础上，利用迁移学习方法[76]，对其网络参数进行微调，通过训练获得害虫自动识别模型。

CaffeNet 由 8 层网络结构组成，包括 5 个卷积层和 3 个全连接层[77-79]。前两个卷积层

包含卷积（Convolution）、激活（Activation）、池化（Pooling）和局部响应归一化（Local Response Normalization）操作，第 3、4 个卷积层只包含卷积和激活操作，最后一个卷积层包含卷积、激活和池化操作。第 6、7 全连接层包含激活和 dropout 操作，第 8 全连接层的输出是 34 类的 Softmax 层。使用的 VGGNet 模型有 16 层网络，其核心思想是增加网络深度，减小卷积核尺寸，包括 5 组卷积和 3 个全连接层，每组卷积由 2~3 个卷积层与一个最大池化层组成，均使用统一大小的卷积核（3×3），同时卷积核的数量也随层数的增加而增多，3 个全连接层包含激活和 dropout 操作，最终输出 34 类的 Softmax 层。ResNet 借鉴了高速网络（Highway Network）的跨层链接思想，主要用于解决网络深度增加导致的准确率下降问题，本研究中使用的是 101 层的残差网络，相比于 VGGNet 网络深度提高了 6 倍，由 1 个输入卷积层、33 个残差学习单元和 1 个全连接层构成，其中残差学习单元包括 3 个卷积层，通过首端 1×1 的卷积核来降低维度，并在末端进行升维，从而达到减少参数，降低计算量的目的。最后通过全连接层输出分类结果。

Caffe 框架中提供了一套完整的工具集，可用于处理数据、训练模型以及测试等操作，具体训练步骤如下：

(1) 样本处理，将已分类的农业害虫图片数据按 5:1 的比例分为训练集和测试集两个部分，由于害虫图片数量较多，为了方便错误查找以及标签创建，图片名称应有规则地命名。

(2) 创建标签，在分类过程中需要通过标签文件读取图片，通过创建脚本文件快速生成农业害虫的图片标签编号，需要注意的是训练集和测试集中的每张图片都必须在标签文件有唯一的映射。

(3) 图片资源转为数据库文件，在 Caffe 框架中用来训练神经网络的图片格式必须为 lmdb 或 leveldb 格式（数据库文件），可以在 Caffe 编译好的 release 目录下通过 convert_imageset.exe 可执行文件实现格式转换，默认为 lmdb 格式。

(4) 均值计算，图片资源减去均值后会提高训练速度和精度，在编译好的 release 目录下通过批处理文件输入 lmdb 数据并执行 compute_image_mean.exe 文件，即可获取后缀名为.binaryproto 的均值文件。

(5) 模型训练，在 models/caffenet 目录下修改 train_val.prototxt 文件配置网络结构，修改 solver.prototxt 文件设置超参数，通过批处理文件训练模型即可。

4.2.2 农业害虫识别模型的测试

以 1803 幅农业害虫图像样本为测试对象，对训练输出的 CaffeNet 模型、VGGNet 模型

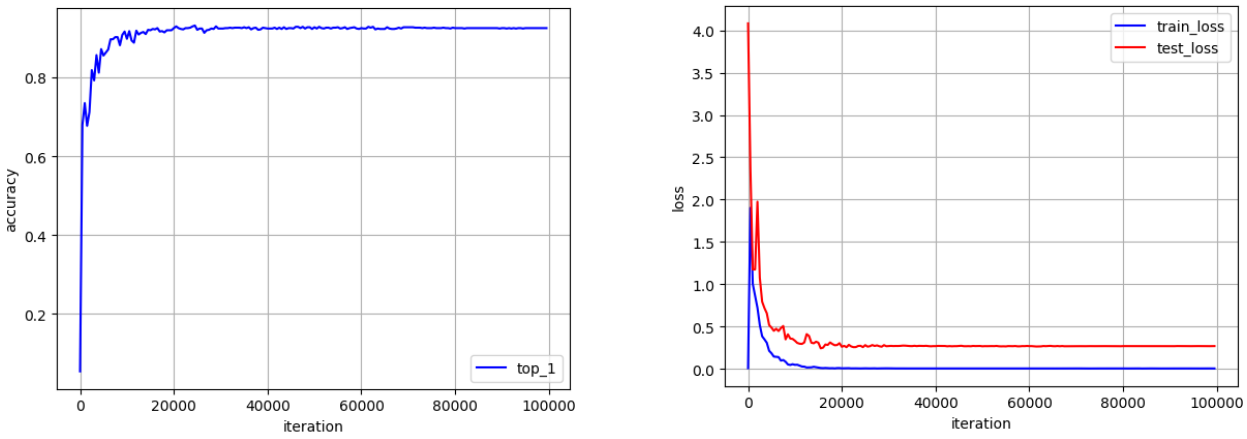
与 ResNet 模型进行测试，获得的结果见表 4.1：

表 4.1 训练模型比较

网络模型	测试样本量	平均识别率(%)	损失值
CaffeNet	1803	87.8	0.397
VGGNet	1803	90.4	0.351
ResNet	1803	93.5	0.334

从表中可以看出，相对于 CaffeNet 与 VGGNet，101 层的 ResNet 网络在平均识别率上有较大的提升，且模型的损失函数值较小，鲁棒性以及泛化能力相对较高。

在模型训练过程中会输出每次迭代的损失值和准确率，通过在训练指令后加入导出命令即可将迭代的结果输入日志中，利用 Caffe 框架中提供的 Python 接口，执行脚本文件绘制相应的损失值曲线和准确率曲线，用来判断当前卷积神经网络的学习状态。以 ResNet 训练的农业害虫识别模型为例，每迭代 500 次输出一次测试结果，绘制的结果如图 4.2 所示。模型在训练过程中准确率曲线保持平稳，损失值曲线缓步下降且没有出现二次上升的情况，最终输出的准确率达到了 93.5%，测试集的损失值保持在 0.334 左右，网络学习情况良好。



(a) 准确率曲线　　**(b) 损失曲线**

图 4.2 ResNet 学习曲线

学习曲线可以宏观地判断网络结构设计的合理性，但无法获取每类害虫的测试准确率。在根目录的 examples/cpp-classification 文件夹中修改 Caffe 框架提供的分类文件，可以进行分类识别，通过结合害虫的训练测试样本量可以获取更加精确的模型信息。首先在 Visual Studio 中新建工程项目并添加分类文件，配置包含目录、库目录以及链接输入等属性，然后添加训练好的模型文件、均值文件、权值文件和标签文件，并采用列表的形式读取害虫图片进行测试，最后将输出结果保存在文本文档中，统计后的识别结果见表 4.2。

表 4.2 基于 ResNet 的农业害虫图像识别

农业害虫名称	训练样本数	测试样本数	识别率
大螟	239	47	93.8%
二化螟	355	70	94.3%
玉米螟	223	44	90.7%
中华稻蝗	319	63	95.5%
稻纵卷叶螟	241	48	91.2%
蝼蛄	268	53	94.6%
小麦叶蜂	194	36	84.8%
花蓟马	239	46	86.4%
黑尾叶蝉	315	62	97.6%
黄腹灯蛾	255	51	92.3%
玉米蓟马	208	41	88.7%
.....	.....	.....	.....

由于篇幅问题，表 4.2 中只给出了 11 种害虫的测试结果，可以看出，在训练样本较多的情况下，获得的测试识别率会有所增长，且在实际应用中发现，二化螟、中华稻蝗、蝼蛄等体积偏大的害虫种类泛化能力相对较高。在目前已有数据集的情况下，基于深度学习的农业害虫智能识别系统会对用户上传识别的害虫图像进行分类收集，通过不断扩充训练样本来优化识别模型，降低实际应用的泛化误差，提高农业害虫的识别率。

4.3 本章小结

本章主要介绍了 Caffe 深度学习框架在 Windows 系统下的搭建以及应用，首先在应用较广泛的 CaffeNet、VGGNet 与 ResNet 基础上对网络结构进行微调、训练，并通过测试比较获取最优的农业害虫的识别模型，然后通过生成学习曲线对模型进行分析，最后利用分类程序得到每类测试样本的识别率。需要一提的是，本论文中对深度学习的研究只是为了在服务器中实现 Caffe 框架的重构以及害虫识别模型的应用。在实际的算法研究中，还需要根据农业害虫样本的数量和特征选择更多的网络模型进行训练对比，通过可视化结果调整网络参数，以获取最优的识别率。总之，作为农业害虫识别系统的核心，关于深度学习算法的优化也是本课题下一步研究的重要部分。

第五章 基于深度学习的农业害虫智能识别系统的测试与分析

通过安卓智能手机对基于深度学习的农业害虫智能识别系统进行测试，在确保系统各部分功能运行正常的情况下，检测客户端 CPU 的使用率、内存消耗、农业害虫识别的平均响应的时间、系统依赖库等。

5.1 测试环境

用于测试的安卓智能手机型号有小米 8、三星 SM-G9350、魅族 MX、华为 P10，在 Android studio 2.3.3 平台下进行性能评估。服务端分别在装有 Windows7 系统的微机与 Windows server 2012 系统的阿里云服务器上进行搭建，农业害虫识别的平均响应时间在 MyEclipise 平台下进行测试，由于识别接口是通过 JNI 调用动态链接库实现的，还需要 depends22_x64 反编译工具测试 DLL 所需的系统依赖库。

5.2 系统测试与分析

5.2.1 功能测试

功能测试主要针对用户登录、害虫信息查询、害虫地图分布，害虫智能诊断、专家远程诊断这五个核心模块进行。首先在安卓智能手机上安装“农虫伴侣”农业害虫图像智能识别系统客户端应用软件，第一次启动后会进行手机的权限请求，如图 5.1(a)，分别允许获取定位、手机信息、多媒体等相关权限后即可开启服务。新用户需要进行账号注册登录，同时可以选择“记住密码”与“自动登录”方便再次使用，除此之外还可以在第三方登录中通过手机短信验证或 QQ 进行快捷登录，如图 5.1(b)。主界面中的内容主要有农业害虫信息查询、智能诊断、个人中心和专家远程诊断，通过点击左上角的图标或者手指拖动界面向右滑动可打开菜单栏，进行客户端检查更新和账号切换等功能，主界面如图 5.1(c)。

点击“农业害虫信息查询”按钮，可对常见 200 余种农业害虫进行浏览，如图 5.1(d)，通过关键字可以进行快速精确地查找，点击“详情”还可以获取害虫相应的基本信息和防治措施。点击“农业害虫智能诊断”按钮，可浏览附近以及全国范围内用户分享的农业害虫识别信息，见图 5.1(e)，点击左下角的相机图标可开启拍照功能，将待识别的农业害虫置于识别框中拍照上传识别，点击右下角的照片图标可开启图库功能，选取手机中已有的农业害虫图片，经过裁剪后上传识别，见图 5.1(f)。害虫图像经服务端识别后将置信度最高的 5 类结果反馈回手机客户端，如图 5.1(g)，用户可以通过下滑进行浏览并对识别度较

低且不确定的识别结果请求专家诊断，专家反馈的信息在“个人中心”中进行查看。

专家登录后可点击“专家远程诊断”按钮查看用户请求的诊断内容，如图 5.1(g)，对于未鉴定的请求，可通过点击“查看”按钮可进入具体的诊断界面，如图 5.1(i)，包括用户上传的害虫图片，拍摄或上传图像的地理位置以及智能识别系统给出的诊断信息，同时还可以点击害虫图片进行放大预览。

(a) 权限请求

(b) 登录界面

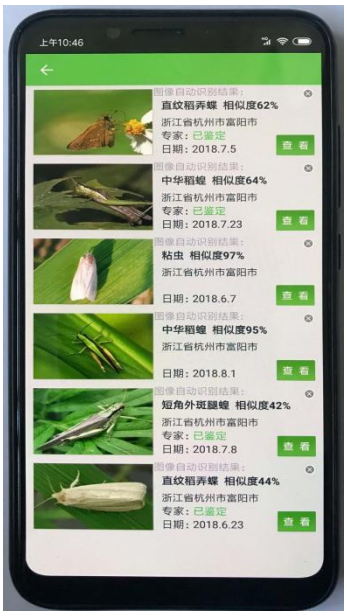
(c) 主界面

(d) 信息查询界面

(e) 害虫地图分布

(f) 图像上传

(g) 识别反馈界面



(h) 用户诊断请求界面

(i) 诊断界面

图 5.1 系统功能测试

5.2.2 性能测试

利用 Android studio 集成开发环境，测试客户端软件的性能，如图 5.2 所示。Free 代表空闲内存，Allocated 代表已分配使用的内存，User 代表用户使用的 CPU 占比，Kernel 代表内核使用的 CPU 占比，在前台运行时客户端内存消耗和 CPU 使用率保持稳定，后台运行时基本不占用内存，在反复的测试中没有出现飙升溢出的问题。

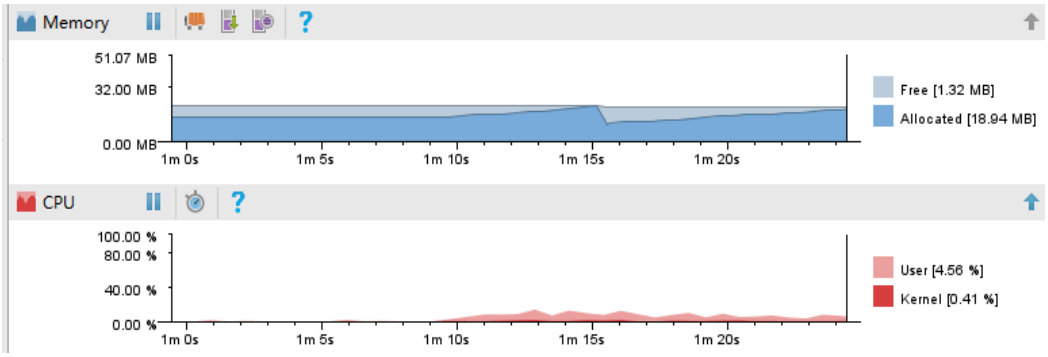


图 5.2 客户端性能测试

5.2.3 识别响应时间

为排除网络带宽的影响，识别响应的时间应按服务器解析完二进制信息开始计时，即 JNI 调用深度学习算法进行农业害虫识别诊断的耗时。利用 MyEclipse 集成开发环境进行测试，服务器识别打印信息如图 5.3 所示，首先解析上传的农业害虫图片，并将存储地址

以及相关信息保存在数据库中，id 为保存数据的主键返回值。然后在 JNI 接口中传入图片存储的位置参数进行识别，输出的识别结果为置信度最高的 5 类害虫编号。识别结束后通过主键将对应害虫图片的识别信息保存在数据库中。经过大量测试，CPU 模式下农业害虫的平均识别响应时间在 1.1 秒左右，GPU 模式下的平均识别响应时间在 0.67 秒左右，偏差在 200ms 以内。

```
I1123 22:33:30.605381 351352 net.cpp:774] Copying source layer relu5_16/x2
I1123 22:33:30.605381 351352 net.cpp:774] Copying source layer conv5_16/
id:148
图片存储位置：E:\apache-tomcat-7.0.70\webapps\DetectionServer\insectImage\2018-11-23
开始识别
识别结果：80 79 116 72 183 置信度：99 0 0 0 0
识别耗时：1081ms
数据存储完成
```

图 5.3 识别响应打印信息

5.2.4 系统依赖库测试

在系统搭建过程中会进行服务器的移植，通过在 Tomcat 中添加 war 包，可以实现 Java Web 项目的快速解压发布，因为本论文中的农业害虫识别接口是通过 JNI 调用深度学习动态链接库实现的，而该动态链接库的运行需要一些系统运行库支持，所以要进行反编译测试。通过 depends22_x64 工具对 DLL 文件进行测试的结果如图 5.4 所示，除了 Opencv 以外还需要 MSVCR120、MSVCP120 等依赖库，只要在项目移植过程中将系统缺少的运行库添加到 system32 目录下即可实现依赖。

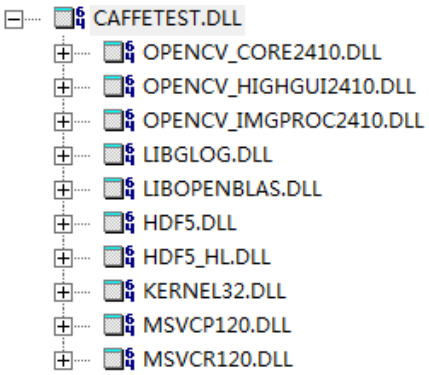


图 5.4 系统依赖库测试

5.3 本章小结

本章主要介绍了基于深度学习的农业害虫智能识别系统的功能和性能测试，测试结果

表明该系统各个模块的功能运行良好，客户端 CPU 的使用率和内存消耗稳定。在不考虑网络带宽的影响下，服务端识别响应的时间较短，可以为用户提供实时快速的害虫识别诊断，此外通过在系统下添加所需的依赖库还可以实现服务端的快速移植。

第六章 总结与展望

6.1 总结

中国自古以来就是农业大国，而昆虫对农作物的侵害严重影响了农产品的产量和品质。农作物田间害虫种类繁多，种间相似度高，容易混淆，且不同植物的防治方法不同，导致害虫无法及时根治。随着深度学习理论的应用，移动设备的普及，本文针对水稻、小麦、玉米、大豆等农作物的主要害虫，实现了基于深度学习的农业害虫智能识别系统，基层植保人员和农民可以通过安卓平台应用软件在农田随时随地拍摄上传害虫图片，得到实时反馈的识别结果与防治措施，实现与用户的点对点服务，及时地为农民提供最有效的害虫防治措施，为害虫诊断与防治的难题提供了有效解决途径。本文的主要研究内容及创新点如下：

(1) 研究并实现了基于深度学习的农业害虫的智能识别系统，用户可以在手机端通过应用软件拍摄或选取相册中的害虫图像，经预览、裁剪等处理操作后上传至服务器进行识别和诊断，同时可以通过信息确认的方式将识别结果同步到地图界面上与广大用户共同了解身边以及全国范围内的害虫分布信息，用户还可以通过查询的方式进行农作物害虫的认识与了解，学习正确的化学防治与生物防治措施。用户可在个人中心中查看以往的识别记录，对于不确定的识别结果通过远程专家诊断功能进行实时在线问答。

(2) 实现了基于深度学习的农业害虫图像识别算法在服务端的应用，首先搭建 Caffe 深度学习框架并重构至 Java Web 项目中。针对 34 类主要农作物害虫，1 万多张图片数据，在应用较广泛的 CaffeNet、VGGNet 与 ResNet 基础上，利用迁移学习方法，对网络结构进行微调，训练网络权值参数，获得害虫自动识别模型，最后通过比较得出 101 层的 ResNet 网络模型效果最优，平均识别准确率可达到 93.5%。

(3) 利用 JNI 本地方法接口与 DLL 动态链接库实现了农业害虫识别模型的调用，通过在 Native 方法中添加模型文件，均值文件、权值文件和标签文件的地址参数即可实现农业害虫图片的识别。在 CPU 模式下农业害虫的平均识别响应速度在 1.1 秒左右，GPU 模式下的平均识别响应时间在 0.67 秒左右，偏差在 200ms 以内。

(4) 基于深度学习的农业害虫智能识别系统对用户上传识别的害虫图像进行分类收集，通过不断扩充训练样本优化识别模型来降低实际应用的泛化误差，提高农业害虫的识别率。无论在任何地点任何时间都可以为用户提供高频次的害虫识别服务，通过拍摄特征图快速匹配识别害虫，获取目标信息、防治措施以及置信度等综合信息，切合用户实用性

需求，做到真正意思上的害虫识别智能化。

(5) 相比于传统的虫害预测预报站，该系统可以很好地记录识别害虫的发生位置，通过对大量数据进行挖掘和分析后得出各地主要的农作物害虫分布情况，这对虫害的防治以及预测有着极为重大的意义。

6.2 展望

论文完成了基于深度学习的农业害虫智能识别系统的研究，实现了手机客户端开发、云服务端搭建、深度学习算法应用三部分内容，在识别准确率和识别响应效率上取得了较好效果，能够满足实际的应用需求，但同时还需要对以下几个方面进一步完善。

(1) 本论文中对深度学习的研究仅限于在服务器中实现Caffe框架的重构以及害虫识别模型的应用，在实际的算法研究中，需要根据农业害虫样本的数量和特征选择更多的网络模型训练对比，通过可视化结果调整网络参数，提高识别模型的鲁棒性与范化能力，获取最优的识别率。

(2) 目前搭建的云服务端只能适应一定数量的用户，随着后期用户量的增加，服务器的压力会越来越大，在提高硬件配置的同时，可考虑采用分布式部署来提高后台的响应能力。通过将大量请求分摊给多台服务器实现均衡负载，满足用户需求。

参考文献

[1] 央广网.农业部预警:今年我国农作物病虫害呈重发趋势[EB/OL].2017-02-13.http://country.cnr.cn/focus/20170213/t20170213_523591684.shtml.

[2] 韩瑞珍. 基于机器视觉的农田害虫快速检测与识别研究[D]. 浙江大学, 2014.

[3] 耿英. 基于图像识别的作物病害诊断研究[D]. 中国科学技术大学, 2009.

[4] Wen C, Guyer D. Image-based orchard insect automated identification and classification method[J]. Computers and Electronics in Agriculture, 2012, 89: 110-115.

[5] Venugoban K, Ramanan A. Image classification of paddy field insect pests using gradient-based features[J]. International Journal of Machine Learning and Computing, 2014, 4(1): 1-5.

[6] Boniecki P, Koszela K, Piekarska-Boniecka H, et al. Neural identification of selected apple pests[J]. Computers and Electronics in Agriculture, 2015, 110: 9-16.

[7] Miranda J L, Gerardo B D, Tanguilig III B T. Pest detection and extraction using image processing techniques[J]. International Journal of Computer and Communication Engineering, 2014, 3(3): 189.

[8] Cai Q, He D J. Identification of vegetable leaf-eating pests based on image analysis[J]. Jisuanji Yingyong/Journal of Computer Applications, 2010, 30(7): 1870-1872.

[9] 吴翔. 基于机器视觉的害虫识别方法研究[D]. 浙江大学, 2016.

[10] Ding W, Taylor G. Automatic moth detection from trap images for pest management. Computers & Electronics in Agriculture, 2016, 123(C):17-28.

[11] Wen C, Wu D, Hu H, et al. Pose estimation-dependent identification method for field moth images using deep learning architecture[J]. Biosystems Engineering, 2015, 136:117-128.

[12] 卢宏涛, 张秦川. 深度卷积神经网络在计算机视觉中的应用研究综述[J]. 数据采集与处理, 2016, 31(1):1-17.

[13] 楚敏南. 基于卷积神经网络的图像分类技术研究[D]. 湘潭大学, 2015.

[14] Mittelman R. Time-series modeling with undecimated fully convolutional neural networks[J]. arXiv preprint arXiv:1508.00317, 2015.

[15] 李明威. 图像分类中的卷积神经网络方法研究[D]. 南京邮电大学, 2016.

[16] Lécun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 2001, 86(11):2278-2324.

[17] 高素红,余金咏. 植物虫害与防治[M]. 北京:科学出版社,2017,1-2.

[18] 李文斌. 基于支持向量机 SVM 的水稻害虫图像识别技术研究[D]. 杭州电子科技大学,2015.

[19] 陈晶,朱启兵,黄敏,郑阳. 基于机器视觉的茶小绿叶蝉识别方法研究[J]. 激光与光电子学进展,2018,55(01):348-355.

[20] 刘国成,张杨,黄建华,汤文亮. 基于 K-means 聚类算法的叶螨图像分割与识别[J]. 昆虫学报,2015,58(12):1338-1343.

[21] 潘春华,肖德琴,林探宇,王春桃. 基于 SVM 和区域生长结合算法的南方主要蔬菜害虫分类识别(英文)[J]. 农业工程学报,2018,34(08):192-199.

[22] Juan Z, Xiao‐Ping C. Field pest identification by an improved Gabor texture segmentation scheme[J]. New Zealand Journal of Agricultural Research, 2007, 50(5): 719-723.

[23] 王德发. 基于图像识别的储粮害虫检测[D]. 北京邮电大学,2017.

[24] 胡永强,宋良图,张洁,谢成军,李瑞. 基于稀疏表示的多特征融合害虫图像识别[J]. 模式识别与人工智能,2014,27(11):985-992.

[25] Yaakob S N. An insect classification analysis based on shape features using quality threshold ARTMAP and moment invariant[J]. Applied Intelligence, 2012, 37(1):12-30.

[26] 杨国国. 基于机器视觉的中华稻蝗早期蝗蝻的识别和检测研究[D]. 浙江大学,2017.

[27] 杨文翰. 基于数字图像处理的棉花害虫识别体系研究[D]. 四川农业大学,2015.

[28] 梁万杰,曹宏鑫. 基于卷积神经网络的水稻虫害识别[J]. 江苏农业科学,2017,45(20):241-243+253.

[29] 桂便,祝玉华,甄彤. 卷积神经网络在储粮害虫图像识别中的应用研究[J]. 粮油食品科技,2018(06):73-76.

[30] 张苗辉,李俊辉,李佩琛. 基于深度学习和稀疏表示的害虫识别算法[J]. 河南大学学报(自然科学版),2018,48(02):207-213.

[31] 马梦园. 基于深度学习的鳞翅目昆虫图像处理研究[D]. 浙江工商大学,2018.

[32] Liu Z, Gao J, Yang G, et al. Localization and classification of paddy field pests using a saliency map and deep convolutional neural network[J]. Scientific Reports, 2016, 6, 20410.

[33] 程曦,吴云志,张友华,乐毅. 基于深度卷积神经网络的储粮害虫图像识别[J]. 中国农学通报,2018,34(01):154-158.

[34] 孙鹏. 基于 Android 平台害虫识别系统的设计[D]. 西南交通大学,2014.

[35] 刘蒙蒙. 基于机器视觉的温室害虫自动监测设备研究[A]. 中国植物保护学会.绿色生态可持续发展与植物保护——中国植物保护学会第十二次全国会员代表大会暨学术年会论文集[C].中国植物保护学会:中国植物保护学会,2017:1.

[36] 杨和平. 昆虫自动识别系统及网络版昆虫图文检索查询系统的研究[D]. 中国农业大学,2015.

[37] 杨慧勇,高灵旺,牛国飞,乔建,刘伟,邢鲲,王鑫,李捷. 农业害虫远程自动识别诊断系统应用技术研究[J]. 山西农业科学,2010,38(06):40-42+65.

[38] Han Y Q, Shan X, Zhu D, et al. Design and implementation of locust data collecting system based on android[M].Advances in Computer Science and Education Applications. Springer, Berlin, Heidelberg, 2011: 328-337.

[39] 张谷丰, 罗岗, 孙雪梅,等. 基于 Android 的水稻害虫诊断系统[J]. 应用昆虫学报,2015,52(4):837-843.

[40] 高华. 基于云平台的储粮害虫检测识别技术研究[D]. 河南工业大学,2017.

[41] 贾瑞. 基于微信公众号的农业害虫识别系统[J]. 数字技术与应用,2018,36(01):76-78.

[42] Lucassen J M, Maes S H. MVC (Model-View-Controller) based multi-modal authoring tool and development environment: U.S. Patent 7,900,186[P]. 2011-3-1.

[43] Zhang Y, Luo Y. An architecture and implement model for Model-View-Presenter pattern[C].Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on. IEEE, 2010, 8: 532-536.

[44] 李群. 基于 OkHttp 的文件传输设计与实现[J]. 电子技术与软件工程, 2018(13):180-181.

[45] Bahrampour S, Ramakrishnan N, Schott L, et al. Comparative study of caffe, neon, theano, and torch for deep learning[J]. 2016.

[46] 王茜, 张海仙. 深度学习框架 Caffe 在图像分类中的应用[J]. 现代计算机, 2016(05):72-75.

[47] Jia Y, Shelhamer E, Donahue J, et al. Caffe: Convolutional architecture for fast feature embedding[C].Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014: 675-678.

[48] Tokui S, Oono K, Hido S, et al. Chainer: a next-generation open source framework for deep learning[C].Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS). 2015, 5: 1-6.

[49] 杨楠. 基于 Caffe 深度学习框架的卷积神经网络研究[D]. 河北师范大学,2016.

[50] Agrawal S, Gupta R D. Development and comparison of open source based Web GIS Frameworks on WAMP and Apache Tomcat Web Servers[J]. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 2014, 40(4): 1.

[51] 胡莉萍. Tomcat+JSP+MySQL 整合配置初探[J]. 中国科技信息, 2010(5):102-103.

[52] Lee Y H, Chandrian P, Li B. Efficient java native interface for android based mobile devices[C].Trust,

Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on. IEEE, 2011: 1202-1209.

[53] Li S, Tan G. JET: exception checking in the java native interface[J]. ACM SIGPLAN Notices, 2011, 46(10): 345-358.

[54] 王军弟, 赵恺. JNI 技术在软件开发中的应用研究[J]. 兰州工业学院学报, 2009, 16(5):15-17.

[55] 安百俊. 通过 Java 调用本地方法[J]. 微处理机, 2011, 32(2):41-44.

[56] 吴沧舟, 兰逸正, 张辉. 基于 MySQL 数据库的优化[J]. 电子科技, 2013,26(9).

[57] 林元元. JDBC 连接 MySQL 数据库的方法浅析[J]. 湖南邮电职业技术学院学报, 2009, 8(1):42-45.

[58] Abadi M, Barham P, Chen J, et al. Tensorflow: A system for large-scale machine learning[C].Osdi. 2016, 16: 265-283.

[59] Girija S S. Tensorflow: Large-scale machine learning on heterogeneous distributed systems[J]. 2016.

[60] Chen T, Li M, Li Y, et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems[J]. arXiv preprint arXiv:1512.01274, 2015.

[61] Collobert R, Kavukcuoglu K, Farabet C. Torch7: A matlab-like environment for machine learning[C].BigLearn, NIPS workshop. 2011 (EPFL-CONF-192376).

[62] Kostrikov I. Pytorch implementations of reinforcement learning algorithms[J]. GitHub repository. GitHub, 2018.

[63] 范会敏, 王浩. 模式识别方法概述[J]. 电子设计工程, 2012, 20(19):48-51.

[64] 唐银凤, 黄志明, 黄荣娟, et al. 基于多特征提取和 SVM 分类器的纹理图像分类[J]. 计算机应用与软件, 2011, 28(6):22-25.

[65] 梁燕. SVM 分类器的扩展及其应用研究[D]. 湖南大学, 2008.

[66] 李蓉, 叶世伟, 史忠植. SVM-KNN 分类器——一种提高 SVM 分类精度的新方法[J]. 电子学报, 2002, 30(5):745-748.

[67] Abe S. Feature selection and extraction[M].Support Vector Machines for Pattern Classification. Springer, London, 2010: 331-341.

[68] Kim S K, Park Y J, Toh K A, et al. SVM-based feature extraction for face recognition[J]. Pattern Recognition, 2010, 43(8): 2871-2881.

[69] 曾庆鹏, 吴水秀, 王明文. 模式识别中的特征提取研究[J]. 微计算机信息, 2008, 24(1):226-227.

[70] Lin Y, Lv F, Zhu S, et al. Large-scale image classification: fast feature extraction and svm training[C].Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011:

1689-1696.

[71] 周爱明,马鹏鹏,席天宇,王江宁,冯晋,邵泽中,陶玉磊,姚青. 基于深度学习的蝴蝶科级标本图像自动识别[J]. 昆虫学报,2017,60(11):1339-1348.

[72]]郑弘晖. 基于深度学习的图像分类和人脸识别算法研究[D]. 北京邮电大学,2018.

[73] Wang L, Guo S, Huang W, et al. Places205-vggnet models for scene recognition[J]. arXiv preprint arXiv:1508.01667, 2015.

[74] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, inception-resnet and the impact of residual connections on learning[C].AAAI. 2017, 4: 12.

[75] Wu Z, Shen C, Hengel A. Wider or deeper: Revisiting the resnet model for visual recognition[J]. arXiv preprint arXiv:1611.10080, 2016.

[76] 杨士准. 基于样本和特征的迁移学习方法及应用[D]. 国防科学技术大学, 2013.

[77] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C].Advances in neural information processing systems. 2012: 1097-1105.

[78] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.

[79] Huang G, Liu Z, Van Der Maaten L, et al. Densely connected convolutional networks[C].CVPR. 2017, 1(2): 3.

致谢

时光荏苒，转眼间两年半的研究生生活即将结束，站在毕业的门槛上，回首往昔，努力和奋斗成为丝丝的记忆，甜美与欢笑也都尘埃落定。在读研期间，很多老师、同学、朋友在学习和生活上面给予了我很多的帮助和关怀，让我感受到了温暖。在此，我要向他们表达我最诚挚的谢意！

首先，我要由衷地感谢我的导师姚青教授，她是我的授业恩师，两年半来，恩师渊博的专业知识，严谨的治学态度，精益求精的工作作风，朴实无华、平易近人的人格魅力一直影响着我。导师不仅授我以文，而且教我做人，在学习上，当我遇到困难的时候，恩师会开导我，让我保持着一颗积极乐观向上的心；在科研过程中遇到难题时，恩师会为我答疑解惑指明前进的道路，并培养我独立解决问题的能力，让我懂得了独立思考和团队合作的重要性，虽历时两载，却赋予我终生受益无穷之道。在整个读研期间，恩师不仅在学习和科研工作上给予了诸多帮助和指导，同时在生活上还给予了无微不至的关怀。在此谨向恩师致以最崇高的敬意和最诚挚的感谢！感谢第二导师宋滢老师的指导和关心，感谢在我的课题研究中投入的大量关注和指点，在我实验操作过程中提供的指导和帮助，在我整理和分析数据、论文撰写和修改过程中给予的建议和支持。

感谢中国水稻研究所昆虫组组长唐健研究员、杨保军老师、罗举老师以及刘淑华老师在实习期间对我的关怀和帮助。特别需要感谢的是唐健老师，唐老师以渊博的知识和深邃的洞察力，对我研究生期间的科研工作给予了理论和实践指导，并给我提供了舒适的实验室环境和丰富的科研素材，在唐老师的帮助下我客服了许多困难，找到了正确的研究路线。再次向研究所的各位老师道上最衷心的感谢！

感谢姚之队的师兄、师姐和师弟——吕军、俞佩仕、周爱明、陈国特、马鹏鹏、张永玲、冯晋、邵泽中、郭龙军。怀念与你们一起学习、拼搏、奋斗的日子，因为你们，实验室充满了欢乐，因为你们，研究生生活充满了笑语，谢谢你们。

最后要感谢我的父母与哥哥，谢谢你们一直以来对我的支持和信任，正是因为有你们无私的爱，我才能有今天的成就。感谢你们一直以来都把最好的留给我，你们就是我一路前行与奋斗的的动力。

这里向参加本次论文评阅、答辩的评委老师和专家们，表示真诚的感谢和崇高的敬意，谢谢你们！

攻读硕士学位期间的科研成果

荣获得第四届中国研究生移动终端应用设计创新大赛全国三等奖。