

赵建敏,李艳,李琦,等.基于卷积神经网络的马铃薯叶片病害识别系统[J].江苏农业科学,2018,46(24):251-255.
doi:10.15889/j.issn.1002-1302.2018.24.069

基于卷积神经网络的马铃薯叶片病害识别系统

赵建敏¹,李艳¹,李琦¹,芦建文^{1,2}

(1. 内蒙古科技大学信息工程学院,内蒙古包头 014010; 2. 包钢集团公司信息服务中心,内蒙古包头 014010)

摘要:深度学习是图像处理领域的研究热点,为实现马铃薯叶片病害识别,达到及时防治的目的,采用深度学习理论设计病害识别系统,系统包括分层卷积神经网络识别模型、WEB 服务器和手机端 APP。基于 TensorFlow 框架,搭建 8 层 CNN + softmax 分层卷积神经网络模型,自动学习到 256 个病害图像特征,采用 softmax 分类器识别病害,简单背景单一病斑识别准确率达到 87%。在 ubuntu 上搭建 Nginx Web 服务器,应用 Flask 框架开发后台服务,基于 vue.js 开发手机端 APP,实现手机采集、上传病害图像、获取病害结果等功能,为相关应用提供完整全栈式解决方案。

关键词:深度学习;卷积神经网络;马铃薯;病害识别系统

中图分类号: S126; TP391.41 **文献标志码:** A **文章编号:** 1002-1302(2018)24-0251-05

2006 年多伦多大学 Hinton 等采用深层神经网络进行特征降维,开启了深度学习的时代^[1-2]。近年来,基于卷积神经网络的深度学习已经广泛运用于图像识别领域,2012 年, Hinton 等采用深层网络自动提取特征,在 Image Net 图像集分类问题上取得了很好的结果^[3]。深度学习应用于图像识别避免了人工特征提取的算法消耗,并且深度学习对输入样本的平移、缩放、扭曲保持高度不变性,训练时共享权值,降低了训练时间,大大提高了准确性^[3-4]。

在国内农业图像应用领域,杨国国等基于卷积神经网络在复杂自然背景下定位和识别茶园害虫,识别率达到 90% 以上,取得较好的效果^[5]。谭文学等设计了多层卷积神经网络,采用弹性动量的参数学习方法,通过识别果体病理图像,诊断常见病害,识别率高达 95% 以上^[6]。张帅等搭建了 8 层

卷积神经网络模型在 PlantNet 和自扩展叶片库上进行训练和测试,简单背景下叶片识别率均达到 90% 以上^[7]。鲁恒等在无人机影像耕地信息提取中,利用深度神经网络提取耕地特征,达到较高的识别精度^[8]。针对马铃薯早疫病识别,徐明珠等应用高谱成像技术,建立了多种特征波长的 BP 神经网络和支持向量机模型,识别率高达 98% 以上,为早疫病快速识别提供理论依据^[9]。

本研究首次将分层卷积神经网络技术应用于马铃薯病害识别中,基于 TensorFlow 平台设计了马铃薯病害特征提取和病害识别算法,并全栈式开发了病害识别系统的服务器和前端,实现了手机端采集、上传病害图片、服务器处理推送识别结果,经测试可以满足识别和访问要求。

1 材料与方法

1.1 材料

以马铃薯病害图像为识别对象,所有原始数据均采集于内蒙古中部周边马铃薯种植区。共有 4 类病害图像样本 6 000 张,代表性样本见图 1。4 类病害图像经过预处理,得到

收稿日期:2017-08-05

基金项目:内蒙古自治区高等学校科学研究项目(编号:NJZY144)。

作者简介:赵建敏(1982—),男,内蒙古土左旗人,硕士,讲师,主要从事图像处理、人工智能研究。E-mail:zhao_jm@imust.edu.cn。

参考文献:

- [1] 宗全利,刘焕芳,郑铁刚,等.微灌用网式新型自清洗过滤器设计与试验研究[J].灌溉排水学报,2010,29(1):78-82.
- [2] 孟剑,郑传祥.微灌系统全自动反冲洗过滤器的试验与设计[J].农机化研究,2006(7):143-145.
- [3] Hermans J. Auto-line filter: self-cleaning, continuous filtration system[J]. Filtration & Separation, 2002, 39(7): 28-30.
- [4] Amburgey J E. Optimization of the extended terminal sub fluidization wash (ETSW) filter backwash-ing procedure[J]. Water Research, 2005, 39(2/3): 314-330.
- [5] 李亚雄,陈学庚,温浩军,等.自动清洗河(渠)水网式过滤器的研究开发[J].中国农机化学报,2005(4):75-76.
- [6] 李强强,宗全利,刘贞姬,等.卧式自清洗网式过滤器排污时间试验及计算[J].排灌机械工程学报,2014(12):1098-1104.
- [7] 刘建华,贾焕丽,王启伦.水力驱动全自动过滤器的设计分析[J].机械设计与研究,2012,28(2):103-105.
- [8] 刘飞,刘焕芳,郑铁刚,等.微灌用自吸式自动过滤器滤网内外工作压差的设置研究[J].中国农村水利水电,2010(4):50-53.
- [9] 徐昭.移动轮流反冲洗网式过滤器设计应用[J].塑料制造,2016(5):37-39.
- [10] 郑铁刚,刘焕芳,宗全利.微灌用过滤器过滤性能分析及应用选型研究[J].水资源与水工程学报,2008,45(4):36-39.
- [11] 孙新忠.离心筛网一体式微灌式过滤器的试验研究[J].排灌机械,2006,24(3):20-23.
- [12] 董文楚.微灌用滤网过滤器设计原理与方法[J].喷灌技术,1989,62(3):7-14.
- [13] 李光永,张琼.反冲洗网式过滤器:CN1471994[P]. 2004-02-04.
- [14] 刘飞.新型自清洗网式过滤器结构优化研究[J].中国农村水利水电,2010(10):18-21.
- [15] 刘焕芳,刘飞,谷趁趁,等.自清洗网式过滤器水力性能试验[J].排灌机械工程学报,2012,30(2):203-208.



图1 马铃薯4种病害图像样本

4 000 张图片作为训练样本,1 000 张作验证样本,1 000 张作测试样本。其中,验证样本用来检验神经网络中样本是否出现过拟合现象。验证样本与训练样本同时作为神经网络的输入,计算损失函数大小,但验证样本损失函数大小仅用来判断网络是否过拟合,而不参与网络参数调整。

为提高模型的特征学习能力和抗干扰能力,对原始图像进行随机反转、亮度和对比度调节等预处理,并将处理后的图片作为训练集,训练模型网络,预处理函数如下:

tf. image. random_flip_left_right () 随机翻转。

tf. image. random_flip_left_right (distorted_image) 随机亮度变换。

tf. Image. random_contrast (distorted_image) 随机对比度变换。

最后对数据做白化处理,均衡均值与方差,降低图像采样中明暗、光照差异引起的影响。

TensorBoard 是 TensorFlow 可视化平台,模型对输入图像进行变换之后 TensorBoard 显示的结果见图 2,从图 2 可以看出,变换之后的图像比原图像更加光滑并且降低图像明暗、光照差异引起的影响。

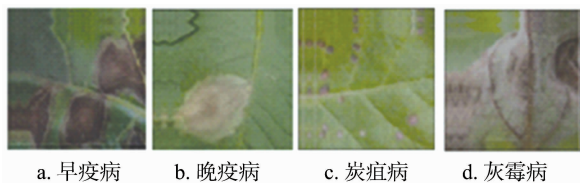


图2 预处理图像

TensorFlow 平台输入数据格式有图片格式、二进制格式、文本格式以及标准的 TensorFlow 格式。本研究采用的是二进制格式的读入。首先初始化读入数据量、类别、标签字节大小、尺寸和通道数,应用 tf. FixedLengthRecordReader () 函数读取固定长度的样本字节 (包括 label);之后用 tf. decode_raw () 函数进行解析。

1.2 方法

马铃薯病害图像识别系统由服务器端和前端组成。服务器端在 ubuntu 操作系统上基于 TensorFlow 计算框架,搭建了多层卷积神经网络病害识别模型;后端搭建了 Mysql 病害数

据库,采用 flask 框架,异步读写数据;手机端基于 vue. js 开发了 Web app,完成图像采集并通过 rest API 接口上传到服务器,并实时获取病害识别结果,总体结构见图 3。

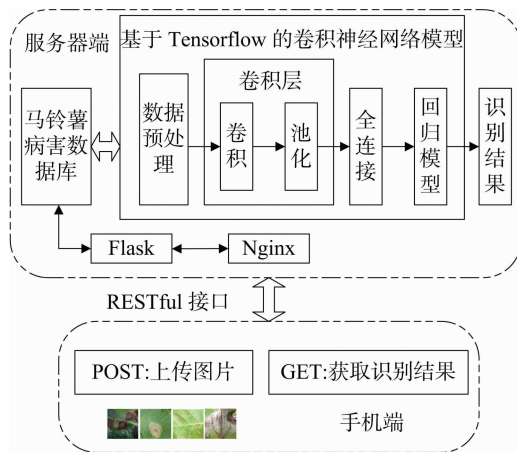


图3 系统结构

1.2.1 卷积神经网络模型设计 深度卷积神经网络模型设计包括样本预处理、神经网络搭建、模型训练及模型评估。本研究初始马铃薯病害图片样本经过预处理,整理得到 6 000 张病害样本图片,设计了 8 层 CNN + Softmax 神经网络模型,通过梯度下降法进行训练,在单病斑样本测试中得到了较好的准确率。TensorFlow 已经广泛用于图像识别等多项机器深度学习领域,先后提出了 AlexNet^[10]、VGG^[11]、GoogLeNet^[12]、PReLU-net^[13] 等代表性网络结构。在 TensorFlow 平台上,参照 Alex Krizhevsky 在 CIFAR-10^[10] 数据集上应用的深度卷积置信网络,搭建了 3 个卷积层、2 个全连接层,网络结构见图 4。在卷积层中分为卷积、池化过程,全连接层包括 ReLu (Rectified Linear Units) 非线性映射和 dropout 2 个过程,网络结构见图 4。

1.2.2 服务器设计 马铃薯病害应用系统分为服务器和前端 2 个部分,识别系统前端-后端结构见图 5。服务器由服务进程、神经网络识别进程、数据库和 Web 服务构成,采用 RESTful API 接口。前端设计了基于 Vue 的 web app,实现了手机端拍照上传,服务器端识别病害、存储信息并推送识别结果的功能。

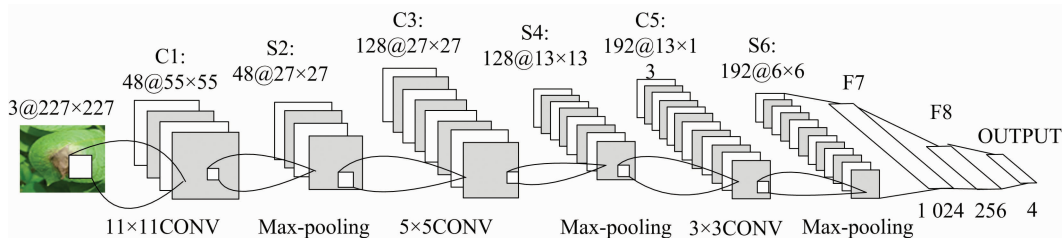


图4 CNN 模型结构

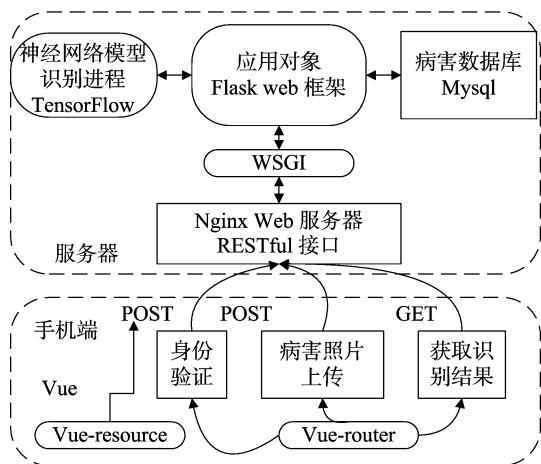


图5 识别系统前端-后端结构

本研究服务器的构建采用 Python Web 框架 Flask 和 Nginx Web 服务器。Flask 是一个轻量级的 Python Web 框架，通过 uWSGI 服务网关接口可实现 Nginx 和 Flask 的交互，执行 Python 应用^[14]。

以上传病害图片和获取识别结果为例，服务器软件设计结构和原理见图 6。用户上传病害照片时，手机端通过网页地址访问服务器，采用标准 RESTful 接口，经过网页地址解析产生 POST /pttdmap 请求。本研究创建了 Flask 实例 fapp，fapp 通过路由 router 操作，fapp.route 修饰器把修饰的函数注册为路由，转入执行 Insert2tab() 业务，将图片存入病害数据库。

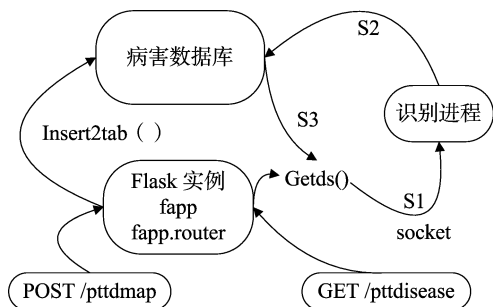


图6 服务器端程序原理和结构

用户获取病害识别结果时，通过网页地址访问服务器，经过解析产生 GET /pttdisease 请求，fapp 通过路由执行 Getds() 业务。本研究设计了病害识别线程，该线程无识别请求时挂起，线程中开启 SOCKET 连接，方法如下：

```
pttdregsver = socket.socket(socket.AF_UNIX, socket.SOCK_STREAM)
```

```
if os.path.exists("/usr/pttdreg.sock"):
```

```
os.unlink("/usr/pttdreg.sock");
pttdregsver.bind("/usr/pttdreg.sock");
pttdregsver.listen(0);
```

Getds() 业务创建 pttdregclient SOCKET，通过 pttdregclient.connect("/usr/pttdreg.sock") 连接 pttdregsver[]，client.send(pttdmap) 将图片送到模型并启动识别模型，完成 Step1。之后 Step2、Step3 完成识别结果存储，并响应手机的请求返回识别结果。

1.2.3 手机 web app 设计 为提高跨平台、可移植性，本研究手机应用 web app 方式，采用 MVVM 框架 Vue.js，它具有数据双向绑定，异步实时更新数据的功能，运行在移动端，方便快捷。客户端识别系统主要采用的技术栈有 vue.js、vue-resource、vue-router，Vue.js 是轻巧的可组件化 MVVM 库；vue-resource 负责与服务器进行数据交互；vue-router 完成界面跳转^[15-16]。各项功能的代码完成后用 webpack 打包部署到 Nginx 服务器，供移动端访问。

2 结果与分析

2.1 特征提取及分类过程分析

在图 4 所示的模型结构中，图像通过输入层、卷积层、全连接层自动学习特征并送入 softmax 分类器，过程如下：

输入层：CNNs 具有对二维图像的特征进行自主学习的特性，故在此将原始图像像素直接作为网络输入，输入大小为 $227 \times 227 \times 3$ ，其中 3 表示颜色通道，表示输入为 RGB 图像。

卷积层 C1 (conv1)，由 48 个 11×11 卷积核，步长为 4，采用 ReLu 非线性映射激励，产生 $55 \times 55 \times 48$ 个特征映射，之后输入池化层 S2 (pool1)，经过 3×3 过滤器、步长为 2 的 max pooling 重叠池化，学习到 $27 \times 27 \times 48$ 个特征。关键代码如下所示：

```
with tf.variable_scope('conv1') as scope:
    kernel = _variable_with_weight_decay('weights', shape =
[11, 11, 4, 48], stddev = 1e-4, wd = 0.0)
    conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding =
'SAME')
    biases = _variable_on_gpu('biases', [48], tf.constant_
initializer(0.0))
    bias = tf.nn.bias_add(conv, biases)
    conv1 = tf.nn.relu(bias, name = scope.name)
    _activation_summary(conv1)
    pool1 = tf.nn.max_pool(conv1, ksize = [1, 3, 3, 1],
strides = [1, 2, 2, 1], padding = 'SAME', name = 'pool1')
    卷积层 C3 过滤器大小为  $5 \times 5$ ，深度为 128，移动步长为
```

1, C3 特征映射大小为 $27 \times 27 \times 128$, 激励函数采用 Relu。池化层 S4 仍然采用 3×3 滤波器、步长为 2, max pooling 重叠池化, 输出特征个数为 $13 \times 13 \times 128$ 。卷积层 C5 过滤器大小为 3×3 , 深度为 192, 移动步长为 1, 本层输出矩阵大小为 $13 \times 13 \times 192$ 。池化层 S6 采用 3×3 滤波器、步长为 2 重叠池化, 输出特征矩阵为 $6 \times 6 \times 192$ 。

S6 的所有特征作为全连接层 F7 的输入, 第 1 个全连接层有 1 024 个神经元, 采用 Relu 激活函数输出进行 dropout 后, 输出 1 024 个特征值; 最后一个全连接层 F8 再次进行 ReLu 和 dropout 后再进行全连接, 输出 256 个特征, 全连接层的偏置都设置为 0.04, 添加 L2 范式强调稀疏化, 最后输出为融合 label 的 softmax loss, 本层节点数是 4, 对应 4 个对象。

卷积核、池化核、步长大小的选取, 整体的预测模型参数 (包括学习率、学习策略) 均是通过试验得到最佳。卷积核尺寸根据图像明显特征所占像素大小确定, 奇数尺寸的滤波器能获取到更好的中心, 故特征卷积核设为奇数。重叠池化能够在一定程度上防止过拟合^[12]降低识别错误率, 故采用重叠池化。全连接层采用的 ReLu 激活函数, 其优点为梯度不饱和、计算速度快; 梯度计算公式为 (1), 在反向传播过程中减轻了梯度弥散问题, 神经网络前几层的参数也可以很快地更新。正向传播过程中 ReLu 函数仅须要设置阈值, 加快了正向传播的计算速度, 可以极大地加快收敛速度, dropout 的设置是为了防止过拟合。

本研究采用多项式逻辑回归 (也称为 Softmax 回归) 方法进行分类, Softmax 回归在网络的输出层上计算归一化的预测值和标签的交叉熵, 模型的目标函数是求交叉熵损失和所有权重衰减项的和。使用 cross_entropy 计算交叉熵, 然后计算一个 batch 运算后的平均值, tf.add_to_collection 计算总损失。本研究模型使用标准的梯度下降算法来训练模型。

梯度下降法的迭代公式为

$$a_{k+1} = a_k + \rho_k \hat{s}^{(k)} \tag{1}$$

式中: $\hat{s}^{(k)}$ 表示的是梯度的负方向; ρ_k 表示在梯度方向上的搜索步长。梯度方向是通过对方函数求导得到的, 步长是由线性搜索算法确定, 即把下一点的坐标 a_{k+1} 看作是 ρ_k 的函数, 然后求满足 $f(a_k + 1)$ 的最小值 ρ_k 即可。梯度下降算法迭代的终止条件是梯度向量的幅值接近 0, 设置非常小的常数阈值即可。采用 withtf.control_dependencies([loss_averages_op]): opt = tf.train.GradientDescentOptimizer(lr) grads = opt.compute_gradients(total_loss) 计算梯度; 采用 apply_gradient_op = opt.apply_gradients(grads, global_step = global_step) 更新梯度。模型训练完成后, 通过 tf.train.Saver 类提供的方法保存, 在识别进程中加载。

2.2 模型特性分析

模型评估使用测试集样本来评估训练模型的预测性能。为了得到可靠稳定的模型及对模型性能进行无偏估计, 使用交叉验证的方法, 也就是对分类器进行训练后利用测试集来测试训练得到的模型, 以此来作为评价分类器的性能指标。

本研究搭建的神经网络学习速率设置为 0.01。训练过程中得到的损失函数随迭代次数变化而变化的曲线、正确率随着迭代次数变化而变化的曲线见图 7、图 8。

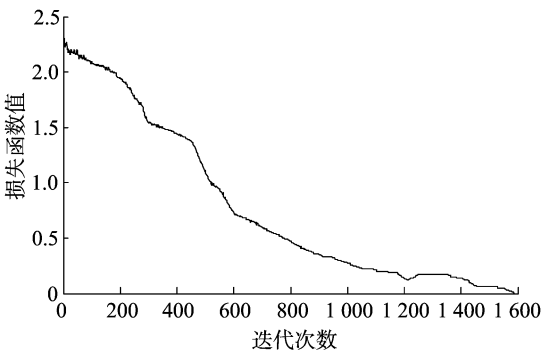


图7 损失函数和迭代次数关系

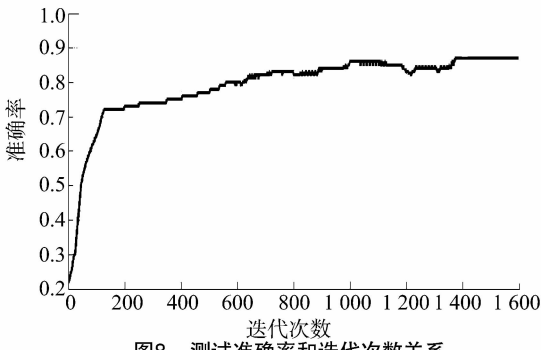


图8 测试准确率和迭代次数关系

从图 7 可以看出, 随着迭代次数增加, 损失函数值越来越小, 几乎接近于 0, 说明函数拟合越来越好。训练样本识别精度到 93% 时基本收敛。训练在服务器上进行, CPU 采用 Intel i7 4 核、单 GPU GTX980、4 G 显存, 训练用时不到 1 h。用测试集样本测试训练好的模型, 测试集样本 1 000 个, 测试时间为 1.97 s, 简单背景、单病斑图像测试准确率达到 86%, 其与迭代次数关系见图 8。

从表 1 可以看出, 炭疽病和早疫病相互误判率较高, 灰霉病和晚疫病相互误判率次之, 炭疽病被误判成早疫病的比率也较高, 表明本研究模型针对上述病斑的特征提取效果有待进一步提高。笔者所在课题组尝试增加 CNN 层数, 由于样本数量受限, 当增加到 6 层 CNN 时出现过拟合现象。通过对比, 在当前条件下设置 3 个卷积层的网络取得的识别效果相对较好。

表 1 病害分类测试准确率统计

病害	病害识别率 (%)				错误率 (%)
	早疫病	晚疫病	灰霉病	炭疽病	
早疫病	85	9	4	2	15
晚疫病	11	83	5	1	17
灰霉病	2	7	91	0	9
炭疽病	8	2	1	89	11

2.3 手机端应用软件测试

手机端 app 应用界面见图 9。身份验证触发 POST 请求, 从后台获取 token, 通过验证后进入系统。病害识别界面有拍摄病害图像、上传图片功能, 执行 POST 函数将手机病害图像上传到后台服务器, 点击获取通过 get 请求将获取识别结果显示在客户端。

3 结论与讨论

本研究设计了 8 层 CNN + softmax 分层卷积神经网络模



图9 手机端应用界面

型,对于简单背景单一病斑识别率达到 86%。但是,由于病害图像的特殊性,局部和全局取样对识别效果影响很大,以 191 农资人上传的呼伦贝尔某地区马铃薯叶片病害图为例,系统识别为炭疽病;若采用局部取样,将病斑单独取样,系统识别为早疫病,差异很大(图 10)。

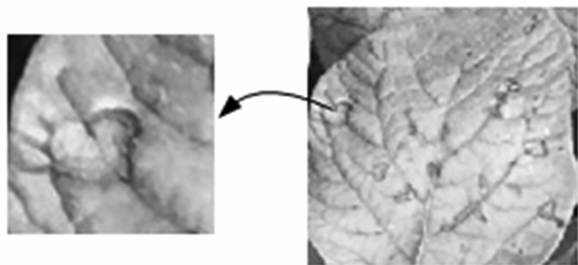


图10 局部采样与全局采样

针对这一问题可采用 CNN 物体检测方法,Gishick 等在基于神经网络的物体检测领域进行了深入研究^[17],笔者所在课题组正在尝试采用 R-CNN^[17]算法和 Fast R-CNN^[18],即 Regions with Convolution Neural Network Features 和快速 R-CNN 算法予以改进。

此外,马铃薯病害种类多、分布分散,叶片、茎、块茎等部位均有病害症状,病害除本研究列出的早疫病、晚疫病、灰霉病、炭疽病,还包括小叶病、青枯病、叶斑病、病毒、块茎部位癌肿、根腐、黑痣、疮痂病等,笔者所在课题组正在不断搜集整理病害图片样本,制作样本库,为模型设计提供数据支撑。

本研究搭建的 Nginx Web 服务器,应用 flask 框架开发后台服务,基于 Vue.js 开发手机端 APP,实现采集、上传、识别功能,满足了系统要求,为相关应用提供了完整全栈式解决方案。

参考文献:

- [1] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. Science, 2006, 313(5786): 504-507.
- [2] 尹宝才, 王文通, 王立春. 深度学习研究综述[J]. 北京工业大学学报, 2015, 41(1): 48-59.
- [3] 余凯, 贾磊, 陈雨强, 等. 深度学习的昨天、今天和明天[J]. 计算机研究与发展, 2013, 50(9): 1799-1804.
- [4] 刘建伟, 刘媛, 罗雄麟. 深度学习研究进展[J]. 计算机应用研究, 2014, 31(7): 1921-1930, 1942.
- [5] 杨国国, 鲍一丹, 刘子毅. 基于图像显著性分析与卷积神经网络的茶园害虫定位与识别[J]. 农业工程学报, 2017, 33(6): 156-162.
- [6] 谭文学, 赵春江, 吴华瑞, 等. 基于弹性动量深度学习神经网络的果体病理图像识别[J]. 农业机械学报, 2015, 46(1): 20-25.
- [7] 张帅, 淮永建. 基于分层卷积深度学习系统的植物叶片识别研究[J]. 北京林业大学学报, 2016, 38(9): 108-115.
- [8] 鲁恒, 付萧, 贺一楠, 等. 基于迁移学习的无人机影像耕地信息提取方法[J]. 农业机械学报, 2015, 46(12): 274-279, 284.
- [9] 徐明珠, 李梅, 白志鹏, 等. 马铃薯叶片早疫病的高光谱识别研究[J]. 农机化研究, 2016, 38(6): 205-209.
- [10] Krizhevsk A. Convolutional deep belief networks on CIFAR-10 [EB/OL]. [2017-07-29]. <http://www.cs.utoronto.ca/~kriz/conv-cifar10-aug2010.pdf>.
- [11] Aimonian K, Zisserman A. Very deep convolutional networks for large-scale image recognition [EB/OL]. [2017-07-29]. <http://www.robots.ox.ac.uk/5000/-vgg/publications/2015/Simonyan15/simonyan15.pdf>.
- [12] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions [C]//Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, 2015: 1-8.
- [13] He K, Zhang X, Ren S, et al. Devling deep into rectifiers: surpassing human-level performance on ImageNet classification [C]//Proceedings of the 2015 IEEE International Conference on Computer Vision, 2015: 1026-1034.
- [14] Grinberg M. Flask Web 开发: 基于 Python 的 Web 应用开发实战 [M]. 安道, 译. 北京: 人民邮电出版社, 2015: 1-3.
- [15] 张耀春, 黄轶, 王静. Vue.js 权威指南 [M]. 北京: 电子工业出版社, 2016: 1-4.
- [16] 周飞燕, 金林鹏, 董军. 卷积神经网络研究综述[J]. 计算机学报, 2017, 40(6): 1229-1251.
- [17] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation [C]//2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014: 580-587.
- [18] Girshick R B. Fast R-CNN [EB/OL]. [2017-07-29]. http://www.cv-foundation.org/openaccess/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf.