UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Mecânica

MIGUEL NAKAJIMA MARQUES

Identificação de HLB em plantações de citros utilizando redes convolucionais profundas

CAMPINAS
2022

MIGUEL NAKAJIMA MARQUES

Identificação de HLB em plantações de citros utilizando redes convolucionais profundas

Dissertação apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Mecânica, na área de Mecatrônica.

Orientador: Prof. Dr. Ely Carneiro de Paiva

Este trabalho corresponde à versão final da Dissertação defendida pelo aluno Miguel Nakajima Marques e orientada pelo Prof. Dr. Ely Carneiro de Paiva.

CAMPINAS
2022

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

M348i

Marques, Miguel Nakajima, 1983-
Identificação de HLB em plantações de citros utilizando redes convolucionais profundas / Miguel Nakajima Marques. – Campinas, SP : [s.n.], 2022.

Orientador: Ely Carneiro de Paiva.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Visão por computador. 2. Agricultura. 3. Python (Linguagem de programação de computador). I. Paiva, Ely Carneiro de, 1965-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: HLB identification in citrus crops using deep convolutional networks
Palavras-chave em inglês:
Computer vision
Agriculture
Python (Computer program language)
Área de concentração: Mecatrônica
Titulação: Mestre em Engenharia Mecânica
Banca examinadora:
Ely Carneiro de Paiva [Orientador]
Denis Gustavo Fantinado
Niederauer Mastelari
Data de defesa: 29-04-2022
Programa de Pós-Graduação: Engenharia Mecânica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-1109-4631
- Currículo Lattes do autor: http://lattes.cnpq.br/0855970136984135

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

DISSERTAÇÃO DE MESTRADO ACADÊMICO

Identificação de HLB em plantações de citros utilizando redes convolucionais profundas

Autor: Miguel Nakajima Marques

Orientador: Ely Carneiro de Paiva

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

Prof. Dr. Ely Carneiro de Paiva, Presidente
Universidade Estadual de Campinas

Prof. Dr. Denis Gustavo Fantinado
Universidade Federal do ABC

Prof. Dr. Niederauer Mastelari
Universidade Estadual de Campinas

A Ata de Defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 29 de abril de 2022

Agradecimentos

Eu gostaria de agradecer ao meu orientador Ely Carneiro de Paiva por me guiar e me ensinar os primeiros passos na minha vida acadêmica e na elaboração dessa dissertação de mestrado.

Além disso gostaria de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq que tornou possível o desenvolvimento deste projeto através do apoio financeiro do Projeto Temático INCT-InSAC (CNPq 465755/2014-3, FAPESP 2014/50851-0), bem como da bolsa de estudos, por este mesmo projeto, CNPq 380703/2020-3.

Ademais eu gostaria de agradecer ao professor Marco Henrique Terra da Escola de Engenharia de São Carlos por liderar o projeto temático acima citado e por abrigar o desenvolvimento deste projeto de mestrado.

Agradeço também a Cristiano Okada Pontelli que coordenou a coleta de dados em campo e aplicação do projeto junto à Jacto Máquinas Agrícolas e seus parceiros de P&D.

Agradeço também à minha esposa Rubia por me apoiar e me encorajar durante todo o processo do desenvolvimento desse mestrado.

Resumo

O método tradicional mais difundido para a identificação de plantas doentes com Huanglongbing é a inspeção visual. O uso de redes convolucionais profundas para tarefas de classificação e segmentação de imagens em diversos campos de aplicação já foi comprovada como eficaz, robusta e possível de ser implementada em sistemas embarcados. O trabalho aqui proposto visa apresentar uma técnica para utilização de redes neurais convolucionais profundas para identificação dessa doença em plantações de citros através de imagens capturadas por uma câmera digital. Foi selecionada a arquitetura padronizada InceptionV3 com as camadas classificadoras finais adaptadas para a aplicação em questão. São utilizados três tipos de imagens: folhas com fundo real (de campo), folhas com fundo homogêneo (estúdio) e imagens de plantas inteiras de duas classes diferentes: saudáveis e doentes. Para acelerar o processo de treinamento é utilizada a técnica de *transfer learning* com o pré carregamento de pesos do desafio ImageNet. São comparados dois procedimentos diferentes para o treinamento a partir dos pesos pré-carregados: congelar os pesos das camadas convolucionais e treinar somente as camadas classificadoras finais e treinar todas as camadas da rede. O sistema para o treinamento da rede foi implementado em linguagem Python. Para melhorar a robustez do treinamento foi utilizada a técnica de *data augmentation* e comparado o resultado com e sem o uso dessa técnica. Para os conjuntos de dados finais foram criados conjuntos de teste com imagens segregadas e não utilizadas no treinamento e no processo de *data augmentation*. Os resultados se mostraram satisfatórios quando treinada a rede toda e utilizando *data augmentation* nos dados de treinamento, chegando a atingir 95% de acurácia em imagens nunca vistas pela rede.

Palavras-chave: visão computacional, transfer learning, HLB, keras, inceptionv3.

Abstract

The traditional method for identifying Huanglongbing infected plants is visual inspection. The usage of convolutional neural networks for image classification and segmentation tasks has been proved to be efficient, robust, and capable of being implemented in embedded solutions. The work presented here proposes the use of a technique that employs deep convolutional neural networks to identify healthy and infected plants through images captured by a digital camera. The standard architecture InceptionV3 was selected to be used with its final classifier layers adapted to the task at hand. It uses three different image types: images with homogeneous background, images with field background and whole plant images of two different classes: healthy and infected. To speed up the training process the transfer learning technique was used to pre-load the weights of the ImageNet challenge. Starting from the pre-loaded weights, two different training procedures are compared: freezing the convolutional layer's weights and training only the final classifier layers and training the whole network. The system used for the training was implemented in the Python language. To enhance the robustness of the training, the data augmentation technique was used, and its results were compared to the case where this technique is not used. For the final datasets, test batches were created using images that were segregated from the training and validation and that were not used in the data augmentation process. The results have shown to be satisfactory when the whole network is trained and data augmentation technique is used in the training data, reaching 95% accuracy in images never seen by the network before.

Key Words: computer vision; transfer learning; HLB; keras; inceptionv3

Sumário

Índice de Figuras

Índice de Tabelas

OBJETIVOS

O objetivo principal da pesquisa será construir uma rede neural artificial capaz de, a partir de imagens de campo de árvores de citros, separar elementos atingidos pela doença Huanglongbing dos elementos saudáveis.

Dividindo o objetivo principal tarefas, a primeira tarefa é definir qual das redes convolucionais profundas já estabelecidas na literatura tem potencial demonstrado para atingir o objetivo principal.

A segunda tarefa é reproduzir a arquitetura escolhida em um ambiente computacional simulado.

A terceira tarefa é selecionar dados disponíveis atualmente (*datasets*) que possam servir de entrada para a rede neural reproduzida.

A quarta tarefa é ajustar os parâmetros da rede e dos dados de entrada para atingir os resultados da literatura.

A quinta tarefa é produzir dados coletados em ambiente real, mas que sejam semelhantes do ponto de vista informacional (enquadramento, ângulo da captura, quantidade de elementos etc.) aos dados utilizados na simulação.

A sexta tarefa é apresentar esses dados coletados em campo para a rede neural e efetuar possíveis ajustes nos parâmetros e na arquitetura para produzir resultados próximos dos obtidos anteriormente (com os dados do conjunto anterior).

A sétima tarefa é produzir dados coletados em campo com características semelhantes às encontradas nos dados obtidos em campo de forma automática (através de câmeras instaladas em máquinas agrícolas)

A oitava tarefa é processar esses dados para que eles possam ser apresentados para a rede neural para que se obtenha resultados semelhantes aos apresentados com os dados anteriores.

1 INTRODUÇÃO

O cultivo de citros tem como um dos principais problemas a infecção por HLB, comumente conhecido como *greening*, uma doença sem cura que diminui a produtividade e obriga os agricultores a destruir as plantas infectadas para evitar o contágio das saudáveis. Assim, a identificação de plantas contaminadas é de suma importância para a manutenção da produtividade e para evitar a disseminação dessa doença (LI, WU, *et al.*, 2020) (RIBEIRO, JORGE e DE PAIVA, 2011). Um exemplo típico de uma planta contaminada apresentando sintomas visíveis pode ser visto na Figura 1.

A prevenção do contágio é feita através do controle do inseto transmissor da bactéria causadora do *greening*. Esse controle é tradicionalmente realizado através da aplicação de inseticidas sistêmicos e foliares, ambos capazes de contaminar o fruto que é consumido e usado na indústria de derivados (FUNDECITRUS, 2015).

Figura 1: Folha de laranjeira com manchas claras (sintomas visíveis de HLB)

O processo mais utilizado para identificação de plantas contaminadas é a inspeção visual, que tem sua eficácia dependente da experiência dos profissionais envolvidos além da necessidade de condições climáticas para presença por tempo prolongado dos profissionais em ambiente externo (não sendo recomendável executar a inspeção da lavoura em condições chuvosas ou de temperatura extremas). De acordo com (JORGE e INAMASU, 2014) esse processo manual tem uma taxa de erro de até 60%. Resumidamente, as principais desvantagens da inspeção visual são listadas a seguir:

- A eficácia da inspeção está sujeita ao nível de experiência dos inspetores;

- Temperaturas muito baixas ou condições climáticas extremas podem impedir a realização da inspeção visando proteger a saúde dos inspetores;
- A precisão dos inspetores no final da inspeção pode não ser a mesma que no início;
- Flutuações no humor dos inspetores podem causar variações na eficácia da inspeção;

As soluções técnicas automatizadas que visam substituir a inspeção visual se dividem em dois tipos de abordagens: utilizando imagens captadas ao nível do solo e imagens captadas a partir de veículos aéreos não tripulados. Os detalhes de cada abordagem e as publicações em que cada uma é apresentada estão descritas no item 2 a seguir.

2 REVISÃO BIBLIOGRÁFICA

Inicialmente foram buscadas no agregador de publicações científicas “Google Scholar” as palavras-chave “HLB”, “Huanglongbing”, “*greening*”, “plant disease detection” e “plant disease recognition”.

Além dessa busca, foram consultadas informações disponíveis nas páginas on-line de associações de produtores de laranja e outros grupos de interesse para entender o panorama geral sobre como o setor lida com o problema aqui apresentado.

Foram encontrados inicialmente 25 publicações que continham os termos pesquisados. Dessa lista foram excluídas as publicações que não continham como foco principal o uso de visão computacional, e foi realizada uma análise preliminar destacando o método utilizado por cada publicação, assim resultando na lista apresentada na Tabela 1:

Título do artigo	Método utilizado
Application of texture analysis for differentiation of the *greening* from others pests (2014)	Método analítico utilizando os pixels da imagem da folha + toolbox WEKA
Comparison of machine learning methods for citrus *greening* detection on UAV multispectral images (2019)	machine learning

Deep Green Diagnostics: Urban Green Space Analysis Using Deep Learning and Drone Images (2019)	machine learning (uso geral, não focado na detecção de doenças) DCNN + MLP
Detecção de *Greening* dos citrus por imagens multiespectrais (2014)	machine learning + API WEKA
Diferenciação da *Greening* do citros de outras doenças foliares a partir de descritores de cor e forma (2011)	machine learning (MLP)
Plant Disease Detection and Classification by deep learning (2019)	machine learning (transfer learning)
Plant Disease Detection Using Machine Learning (2018)	machine learning (random forests classifier)
Recent advances in image processing techniques for automated leaf pest and disease recognition – A review (2020)	machine learning
Systematic review of deep learning techniques in plant disease detection (2019)	machine learning (CNN)
UAV-based multispectral imagery for fast Citrus *Greening* detection (2019)	machine learning (SVM)
Integrating Optical Imaging Tools for Rapid and Non-invasive Characterization of Seed Quality Tomato and Carrot as Study Cases	método analítico com imagens laboratoriais e técnicas matemáticas para potencial de crescimento de sementes

Tabela 1: Publicações revisadas

Após a seleção das publicações de interesse, foram destacados 4 artigos para serem analisados mais detalhadamente, para assim auxiliar na escolha de um método que será reproduzido pela pesquisa. No caso de artigos que são revisões bibliográficas, foram destacados os artigos originais que contêm o detalhamento dos resultados revisados apresentados. Assim foi produzida a Tabela 2:

Review / publicação	*Artigo utilizado pelo review*
Comparison of machine learning methods for citrus *greening* detection on UAV multispectral images	N/A (trata-se de um artigo e não de um review)
Plant Disease Detection and Classification by deep learning (2019)	Plant disease and pest detection using deep learning-based features
	Deep learning for image-based cassava disease detection
	A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition
	Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning
	An in-field automatic wheat disease diagnosis system
	Deep convolutional neural network for classifying Fusarium wilt of radish from unmanned aerial vehicles
	Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case

Recent advances in image processing techniques for automated leaf pest and disease recognition – A review (2020)	Image recognition of plant diseases based on backpropagation networks
	Plant leaf disease analysis using image processing technique with modified SVM-CS classifier
	SVM Classifier Based Grape Leaf Disease Detection
	Detection of unhealthy region of plant leaves using Image Processing and Genetic Algorithm
	Detection of plant leaf diseases using image segmentation and soft computing techniques
	Tomato Plant Disease Classification in Digital Images using Classification Tree
	Recognition and classification of paddy leaf diseases using Optimized Deep Neural network with Jaya algorithm
Systematic review of deep learning techniques in plant disease detection (2019)	Weed detection in soybean crops using ConvNets
	IoT-Based Strawberry Disease Prediction System for Smart Farming
	A segmentation method for greenhouse vegetable foliar disease spots images using color information and region growing
	A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network
	A fuzzy clustering segmentation method based on neighborhood grayscale information for defining cucumber leaf spot disease images

Tabela 2: Lista de artigos e reviews analisados com detalhes

Para cada um dos artigos analisados nessa fase foram destacadas as seguintes informações:

- Nome do artigo: título do artigo analisado pelo review
- Ano do artigo: é dada preferência para artigos que contenham técnicas mais atuais
- Tipo do dataset utilizado: foram identificados datasets utilizando imagens aéreas (tiradas a partir de VANTs) e imagens em que o enquadramento é focado na imagem da folha da planta sob análise. Dessa última categoria foram identificadas fotos com fundo homogêneo e com fundo em campo. Foi dada preferência para datasets de folhas e com fundo da imagem em ambiente de campo (ao invés de imagens com fundo homogêneo, produzidas em estúdio). Separou-se também as imagens feitas a partir de um veículo aéreo (drone) das demais.
- Câmera: especificação da câmera utilizada para coleta das imagens do dataset. É dada preferência para câmeras comuns que têm um custo de manutenção e substituição baixo.
- Framework: tipo de tecnologia utilizada para construção da rede. É dada preferência para tecnologias conhecidas pelo autor

Assim foi produzido o quadro comparativo da Tabela 3:

Nome do artigo	Ano	Tipo de dataset	Câmera	Framework
Comparison of machine learning methods for citrus *greening* detection on UAV multispectral images	2019	aéreo	ADC-lite	Python
Plant disease and pest detection using deep learning-based features	2019	folha	Não informa	Matlab
Deep learning for image-based cassava disease detection	2017	folha	Sony Cybershot 20.2-megapixel	Python (Tensorflow)
A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition	2017	folha	câmera simples (não especifica marca/modelo)	Não informa
Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning (fornece código e dataset)	2017	folha	Canon EOS Rebel + Sony a6000	Python
An in-field automatic wheat disease diagnosis system	2017	folha	Não informa	Python + Theano
Deep convolutional neural network for classifying Fusarium wilt of radish from unmanned aerial vehicles	2017	aéreo	12MP RGB camera	Python (Caffe)
Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case	2017	folha	iPad, iPhone4, Dell-tablet, Galaxy Note, Windows Phone, Samsung S3, iPhone5	Python
Image recognition of plant diseases based on backpropagation networks	2012	não informa	"Câmera digital comum"	Matlab
Plant leaf disease analysis using image processing technique with modified SVM-CS classifier	2017	folha	Não informa	Matlab
SVM Classifier Based Grape Leaf Disease Detection	2016	folha	"Câmera digital"	Não informa
Detection of unhealthy region of plant leaves using Image Processing and Genetic Algorithm	2015	folha	"Câmera digital"	Matlab
Detection of plant leaf diseases using image segmentation and soft computing techniques	2017	folha	"Câmera digital"	Matlab
Tomato Plant Disease Classification in Digital Images using Classification Tree	2016	folha	"Câmera digital comum"	Matlab
Recognition and classification of paddy leaf diseases using Optimized Deep Neural network with Jaya algorithm	2019	folha	"Câmera digital de alta resolução"	Python + Java
Weed detection in soybean crops using ConvNets	2017	aéreo	Sony EXMOR 1/2.3″	Python-weka-wrapper (Java)
IoT-Based Strawberry Disease Prediction System for Smart Farming	2018	N/A	N/A	N/A
A segmentation method for greenhouse vegetable foliar disease spots images using color information and region growing	2017	folha	Nikon Coolpix S3100	Não informa
A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network	2018	folha	Nikon Coolpix S3100	Matlab
A fuzzy clustering segmentation method based on neighborhood grayscale information for defining cucumber leaf spot disease images	2017	folha	Não informa	Não informa

Tabela 3: Detalhes destacados de cada artigo analisado

A partir da análise das publicações foram identificados dois métodos principais usados para classificação de doenças em plantas utilizando redes convolucionais e aprendizado de máquina: imagens de folhas ou plantas individuais capturadas a partir do ângulo do solo (normalmente objetivando o uso em veículos terrestres autônomos) e imagens de grupos de

plantas capturadas a partir de veículos aéreos não tripulados (VANTs) comumente chamados de drones. Tem-se como exemplares de publicações que utilizam imagens aéreas (LAN, HUANG, *et al.*, 2020), (HA, MOON, *et al.*, 2017) e (FERREIRA, FREITAS, *et al.*, 2017) e como exemplos de imagens a partir do solo (RAMCHARAN, BARANOWSKI, *et al.*, 2017), (TÜRKOĞLU e HANBAY, 2019), (FUENTES, YOON, *et al.*, 2017), (DECHANT, WIESNER-HANKS, *et al.*, 2017), (LU, HU, *et al.*, 2017), (JOHANNES, PICON, *et al.*, 2017), (TIWARI e GUPTA, 2017), (PADOL e YADAV, 2016) e (RAMESH e VYDEKI, 2019).

Esses dois métodos apresentados nas publicações aqui destacadas possuem cada um suas vantagens e desvantagens. Para as soluções de imagens terrestres a principal vantagem é a individualização das plantas, que permite a captura de um maior nível de detalhes, dado que cada planta ocupará um número maior de pixels dentro da imagem, enquanto a principal desvantagem é a necessidade de um caminho de acesso terrestre até as plantas, o que dificulta o uso dessa técnica em terrenos não preparados ou em condições que inviabilizem o acesso (como por exemplo se houver uma grande quantidade de lama no caminho até a planta analisada). Já na técnica com o uso de VANTs, as principais vantagens são a captura de múltiplos indivíduos em uma única imagem, diminuindo assim a quantidade de imagens necessárias para cobrir uma mesma área, e a capacidade de acessar locais por via aérea, que torna a solução independente das condições do solo no local analisado. As duas principais desvantagens do método aéreo são a dependência de condições favoráveis ao voo, (principalmente levando em conta que os veículos utilizados nas publicações têm dimensões menores que 1 m^3 e peso menor que 3 kg) e a necessidade de que haja um espaçamento mínimo entre as árvores para que a solução seja capaz de individualizar a análise, inviabilizando assim o uso dessa solução em áreas com alta densidade de vegetação.

Em (LAN, HUANG, *et al.*, 2020), (RAMCHARAN, BARANOWSKI, *et al.*, 2017), (LU, HU, *et al.*, 2017), (HA, MOON, *et al.*, 2017) e (JOHANNES, PICON, *et al.*, 2017) tem-se como ponto forte o uso da linguagem Python, que já é usada em outras pesquisas realizadas pelo grupo apoiador da solução aqui descrita. Já como pontos desinteressantes em (LAN, HUANG, *et al.*, 2020) tem-se o uso de câmera multiespectral e de *drones* para a captura das imagens (também usados em (HA, MOON, *et al.*, 2017) e (FERREIRA, FREITAS, *et al.*, 2017)), que são características não desejadas para a solução final. A primeira por reduzir os tipos possíveis de câmeras a serem usadas e a segunda por não ser do mesmo tipo que a plataforma já desenvolvida pelo grupo apoiador da presente pesquisa. Já (TÜRKOĞLU e HANBAY, 2019), assim como (WANG, LI, *et al.*, 2012), (TIWARI e GUPTA, 2017), (SINGH, VARSHA e

MISRA, 2015), (SINGH e MISRA, 2017), (SABROL e SATISH, 2016) e (MAA, DUA, *et al.*, 2018) apresentam soluções codificadas no software Matlab, o que dificulta a adoção pela presente pesquisa já que esse não é a tecnologia padrão de desenvolvimento no grupo apoiador da pesquisa. Como (FUENTES, YOON, *et al.*, 2017), juntamente com (PADOL e YADAV, 2016), (MA, DU, *et al.*, 2017) e (BAI, LI, *et al.*, 2017) não explicitaram em suas publicações qual a tecnologia em que a solução apresentada foi desenvolvida, essas publicações foram desconsideradas para adoção pela presente pesquisa.

Por último, foram consideradas as medidas dos resultados de cada artigo, para todas as arquiteturas analisadas, sendo elas:

$$Precisão = \frac{Positivo\ verdadeiro}{Positivo\ verdadeiro + Falso\ positivo}$$

$$Recall = \frac{Positivo\ verdadeiro}{Positivo\ verdadeiro + Falso\ negativo}$$

$$Acurácia = \frac{Negativo\ verdadeiro + Positivo\ verdadeiro}{total\ de\ predições}$$

$$F1 = 2 * \frac{Precisão * Recall}{Precisão + Recall}$$

Essas informações estão destacadas na Tabela 4:

Nome do artigo	Arquitetura	Precisão	*Recall*	Acurácia	F1
Comparison of machine learning methods for citrus *greening* detection on UAV multispectral images	Ensemble learning (AdaBoost)	100	100	100	100
	Neural network	97,86	98,91	97,28	98,38
	SVM	88,57	87,63	79,76	88,1
	kNN	85,00	92,24	81,27	88,48
	LR	76,07	89,49	72,2	82,24
	Naïve Bayes	88,57	87,94	80,06	88,26
Plant disease and pest detection using deep learning-based features	ResNet50 + SVM	97,35	Não informa	97,86	93,28
Deep learning for image-based cassava disease detection	Inception v3	Não informa	Não Informa	83 a 98	Não informa
A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition	Faster R-CNN + VGG-16	83,06	Não informa	Não informa	Não informa
Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning (fornece código e dataset)	Conv2D + Maxpooling	Não informa	Não informa	97,80	Não informa
An in-field automatic wheat disease diagnosis system	VGG-FCN-VD16 + Soft-agg	Não informa	Não informa	97,95	Não informa
Deep convolutional neural network for classifying Fusarium wilt of radish from unmanned aerial vehicles	VGG-A network	Não informa	Não informa	93,02	Não informa
Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case	Random-Forest (classificador)	Não informa	Não informa	82,00	Não informa

Image recognition of plant diseases based on backpropagation networks	RNA com 1 camada escondida	Não informa	Não informa	97,50	Não informa
Plant leaf disease analysis using image processing technique with modified SVM-CS classifier	Modified SVM-CS	Não informa	Não informa	96,50	Não informa
SVM Classifier Based Grape Leaf Disease Detection	Linear Support Vector Machine (LSVM)	Não informa	Não informa	88,89	Não informa
Detection of unhealthy region of plant leaves using Image Processing and Genetic Algorithm	Algoritmo Genético	Não informa	Não informa	Não informa	Não informa
Detection of plant leaf diseases using image segmentation and soft computing techniques	Algoritmo Genético + SVM	Não informa	Não informa	95,71	Não informa
Tomato Plant Disease Classification in Digital Images using Classification Tree	Classification Tree	Não informa	Não informa	97,30	Não informa
Recognition and classification of paddy leaf diseases using Optimized Deep Neural network with Jaya algorithm	Deep Neural Network JOA (Java Optimized Algorithm)	92,80	Não informa	98,90	96,86
Weed detection in soybean crops using ConvNets	ConvNet	99,10	Não informa	Não informa	Não informa
IoT-Based Strawberry Disease Prediction System for Smart Farming	N/A	N/A	N/A	N/A	N/A
A segmentation method for greenhouse vegetable foliar disease spots images using color information and region growing	Novel Segmentation Method	97,29	Não informa	Não informa	Não informa
A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network	Deep Convolution Neural Network	98,20	Não informa	93,40	97,00
A fuzzy clustering segmentation method based on neighborhood grayscale information for defining cucumber leaf spot disease images	Fuzzy Clustering Segmentation	N/A	N/A	N/A	N/A

Tabela 4: Resultados apresentados nas publicações analisadas

Considerando-se os melhores resultados de acurácia, para publicações que utilizam imagens do ponto de vista terrestre (ou seja, foram excluídos os artigos que utilizavam datasets de imagens aéreas obtidas por drones) e que utilizam linguagem Python, tem-se (RAMCHARAN, BARANOWSKI, *et al.*, 2017), com uma acurácia que varia entre 83% e 98% dependendo da doença analisada, (DECHANT, WIESNER-HANKS, *et al.*, 2017) com 97,8%, (LU, HU, *et al.*, 2017) com 97,95% e (JOHANNES, PICON, *et al.*, 2017) com 82%.

Com os dados destacados nas tabelas anteriores, foram selecionados os artigos com dataset de imagens terrestres. Dentre esses artigos foram selecionados os que utilizavam a linguagem Python, por se tratar de uma ferramenta conhecida pelo autor. Entre os artigos resultantes dessa seleção com melhor acurácia, foi escolhido “Deep learning for image-based cassava disease detection” para ter a técnica reproduzida e adaptada para o problema aqui apresentado. Resumidamente sua escolha se deve principalmente a:

- Tecnologia de desenvolvimento de domínio do autor e do grupo apoiador da pesquisa (python)
- Resultados satisfatórios (83% a 98%)

- Uso de uma arquitetura padronizada e de fácil acesso (InceptionV3)
- Imagens com características semelhantes às desejadas na solução final (imagens com fundo de campo)

3 METODOLOGIA

Nessa seção será explicada a metodologia utilizada no desenvolvimento da solução apresentada nas seções seguintes.

No primeiro item será apresentada a metodologia utilizada para montar o conjunto de dados que foi usado no treinamento.

No segundo item será apresentada a arquitetura de rede neural artificial profunda que foi utilizada na solução proposta.

No terceiro item será apresentada a diferença técnica entre treinar somente as camadas finais da rede neural e treinar todas as camadas da rede.

No quarto item será apresentado o problema de generalização enfrentado ao expor a rede neural treinada a imagens de árvore inteira. Algumas soluções para esse problema são descritas nesse item.

3.1 O conjunto de dados

Para o conjunto de dados a ser utilizado para o treinamento da rede neural classificadora foram idealizadas as seguintes etapas:

1. Utilizar um conjunto de dados disponível de forma livre na Internet que seja, do ponto de vista informacional, o mais próximo possível dos dados a serem utilizados na aplicação final, ou seja, idealmente seriam imagens digitais de folhas de citros em ambiente de campo. Com esse conjunto pretende-se balizar a reprodução da arquitetura através da comparação entre os resultados obtidos pela reprodução construída pelo autor e os resultados apresentados no artigo que serviu como base.

2. Com a arquitetura base da rede definida, coletar imagens em campo com o equipamento similar ao utilizado na aplicação final (ou seja, câmera digital comercial de alta resolução) e com características informacionais (enquadramento, iluminação, brilho etc.) o mais próximo possível do conjunto de dados utilizado na definição da arquitetura (passo anterior). Assim será possível avaliar se os equipamentos utilizados na aplicação final são capazes de produzir dados com a qualidade necessária para o funcionamento da rede. Pequenos ajustes nos parâmetros da arquitetura são permitidos nessa etapa, dado que pode haver diferenças entre os dados utilizados anteriormente e os coletados em campo. Ex: diferença na resolução das imagens;
3. Coletar dados em campo utilizando o mesmo tipo de equipamento do conjunto de dados anterior e com características próximas às da aplicação final (com ângulo, enquadramento, iluminação etc., parecidos com as imagens que seriam obtidas na aplicação final da solução). É preciso verificar a necessidade de adequação dos parâmetros da rede neural e de utilização de algum tipo de pré-processamento para adequar os dados à entrada da rede neural. A aplicação final utiliza um veículo terrestre não tripulado equipado com uma câmera em cada lateral, apontada para as culturas, que captura imagens das árvores a uma distância de aproximadamente 1,5m.

Com essas etapas em vista, foram utilizados 4 conjuntos de dados que serão descritos a seguir:

- Conjunto Citrus
- Conjunto Jacto + Citrus
- Conjunto Jacto
- Conjunto de árvores inteiras

Para todos os conjuntos de dados, com exceção do conjunto Citrus, a separação entre classes das imagens foi feita a partir da identificação da doença em exame laboratorial realizada pelos funcionários da fazenda. Cada árvore identificada com a doença era marcada com uma fita colorida.

3.1.1 Conjunto de dados Citrus

Inicialmente o conjunto de dados a ser utilizado como base para construção da arquitetura foi definido como o apresentado em (RAMCHARAN, BARANOWSKI, *et al.*, 2017), aqui identificado como *Citrus Dataset*. Esse conjunto de dados possui 611 imagens de folhas e 150 imagens de frutos de citros divididos em 5 categorias, listadas na Tabela 5.

As imagens presentes nesse conjunto foram produzidas em campo e em estúdio. Alguns exemplos de imagens presentes nesse conjunto de dados são mostrados na Figura 2:

Figura 2: Exemplo de imagens do Citrus Dataset

Desse conjunto de dados serão usadas para treinamento da rede inicialmente as imagens de folhas de todas as cinco classes sem o uso da técnica de *data augmentation*, apresentada na seção 3.1.2.

As imagens desse conjunto de dados estão divididas nas classes na seguinte proporção:

Objeto da imagem	**Classe**	**Quantidade de imagens**
Folha	Saudável	60
	Mancha preta (*black spot*)	171
	Cancro cítrico (*canker*)	163
	HLB (*greening*)	204
	Melanose	13
	Total (folhas)	611
Fruto	Saudável	22
	Mancha preta (*black spot*)	19

Cancro cítrico (*canker*)	78
HLB (*greening*)	16
Verrugose (*scab*)	15
Total (frutos)	150

Tabela 5: Divisão das imagens no conjunto de dados

Considerando o uso do conjunto de dados para treinamento de uma rede neural para identificação de imagens de folhas de citros, o conjunto de dados Citrus Dataset apresentado anteriormente é suficiente para obtenção dos primeiros resultados.

3.1.2 Conjunto de dados Jacto + Citrus

A fim de validar se a especificação técnica do equipamento que será usado para captura das imagens coletadas em campo é suficiente para a aplicação pretendida, foram feitas algumas capturas de imagens em campo com características informacionais próximas das presentes nas imagens do conjunto Citrus. Assim foi gerado o conjunto de dados híbrido Citrus + Jacto que contém imagens geradas pelo autor e uma mesma quantidade de imagens provenientes do conjunto Citrus. Como estavam disponíveis somente árvores saudáveis e infectadas pelo HLB, foram excluídas desse conjunto as demais classes de doenças presentes no conjunto Citrus.

Abaixo são mostrados alguns exemplos de imagens presentes nesse conjunto em comparação com imagens presentes no conjunto Citrus:

Figura 3: Imagem de folha proveniente do conjunto Citrus (esquerda), imagem com fundo homogêneo produzida pelo autor (centro) e imagem com fundo real produzida pelo autor (direita)

Assim a divisão das imagens para esse conjunto híbrido foi feita da seguinte forma:

	Saudável	***Greening***
Imagens provenientes do conjunto Citrus	30	30
Imagens capturadas em campo com fundo homogêneo	30	30
Imagens capturadas em campo com fundo real	30	30
Total por classe	**90**	**90**

Tabela 6: Divisão das imagens dentro do conjunto de dados híbrido

A partir desse conjunto foi aplicado o método de *data augmentation* com as transformações de rotação, variação do brilho, zoom e translação e então foi gerado o conjunto de dados nomeado de Citrus + Jacto Augmented, com as seguintes quantidades de imagens:

	Saudável	***Greening***
Imagens provenientes do conjunto Citrus	120	120
Imagens capturadas em campo com fundo homogêneo	120	120

Imagens capturadas em campo com fundo real	120	120
Total por classe	**360**	**360**

Tabela 7: Divisão das imagens dentro do conjunto de dados híbrido após aplicação de data augmentation

3.1.3 Conjunto de dados Jacto

A partir do conjunto de dados Citrus + Jacto foi possível fazer a montagem do conjunto de dados totalmente obtidos pelo autor, com imagens de folhas de citros doentes e saudáveis, com fundo homogêneo e com fundo de campo.

Assim foi gerado o conjunto de dados aqui chamado de Jacto Dataset, contendo a seguinte divisão de imagens:

	Saudável	***Greening***
Imagens de folhas com fundo homogêneo	184	190
Imagens de folhas com fundo de campo	191	186
Total de imagens por classe	375	376

Tabela 8: Divisão das imagens no conjunto de dados Jacto

Figura 4: Amostras de imagens do conjunto de dados Jacto

Assim como no caso anterior, o conjunto de dados original passou pelo processo de *data augmentation* com transformações de rotação, variação do brilho, zoom e translação. Antes da aplicação desse processo de *data augmentation*, foram separadas 50 imagens de cada classe do conjunto original para servir de conjunto de teste, ficando segregadas das demais imagens, não passando assim pelo processo de *data augmentation* e nem sendo utilizadas durante o treinamento e validação, sendo consideradas assim como imagens nunca vistas pela rede neural. Após esse processo, o conjunto de dados ficou com as seguintes quantidades de imagens em cada grupo:

Classe	**Saudável**	***Greening***
Treinamento	1300	1304
Teste	50	50

Tabela 9: Divisão das imagens no conjunto de dados Citrus Augmented

3.1.4 *Data augmentation*

Com o objetivo de melhorar a desempenho do treinamento da rede neural sob estudo, pode-se utilizar a técnica de *data augmentation* para, a partir dos dados originais, produzir mais dados aplicando pequenas transformações, como por exemplo: rotação, espelhamento e deslocamento da imagem. Na Figura 5 são mostradas duas imagens transformadas produzidas a partir da imagem original à esquerda.

Figura 5: Exemplo de uso da técnica de data augmentation

Para o conjunto de dados em questão foram utilizadas as transformações de rotação, alteração do brilho, zoom e deslocamento da imagem. Para tal foi gerado um código Python parametrizado em que é necessário indicar na entrada a quantidade de cada transformação para cada classe, o caminho do dataset a ser lido, o caminho onde serão salvas as imagens do dataset aumentado e as classes que passarão pela transformação.

Para o processo de *data augmentation* é usada a função ImageDataGenerator da biblioteca do tensorflow keras.preprocessing.image. Essa função é configurada de 4 modos diferentes, cada um resultando em uma transformação:

- Com o parâmetro "rotation_range" com valor 90: realiza rotações na imagem de entrada com valores entre -90º e 90º;
- Com o parâmetro "brightness_range" entre 0.2 e 1.2: realiza alterações no brilho da imagem de entrada deixando o brilho final entre 80% e 120% do original;
- Com o parâmetro "zoom_range" com valor 0.25: realiza alterações no zoom da imagem entre afastar e aproximar a imagem até 25%;
- Com os parâmetros "width_shift_range" e "height_shift_range" com valor 0.2: realiza translações na imagem de entrada de até 20% dos pixels no eixo vertical e horizontal.

Os códigos produzidos a fim de aplicar as técnicas de *data augmentation* nos conjuntos de dados utilizados estão disponibilizados no ANEXO A.

3.2 Arquitetura de Rede Neural

Para a arquitetura da rede neural foi seguido o indicado em (RAMCHARAN, BARANOWSKI, *et al.,* 2017) e montada uma rede neural utilizando como base o apresentado em (SZEGEDY, VANHOUCKE, *et al.*, 2016) e aqui nomeado como *InceptionV3* em série com duas camadas convolucionais *fully connected* que atuam como classificador.

As camadas convolucionais profundas que vêm junto com o modelo InceptionV3 são responsáveis por transformar a imagem da entrada em uma série de características relevantes que então são entregues para as camadas *fully connected* finais que usam essas características detectadas como entrada para classificar a imagem em uma das classes disponíveis.

Um diagrama de blocos representando as partes da rede neural é exibido na Figura 6:

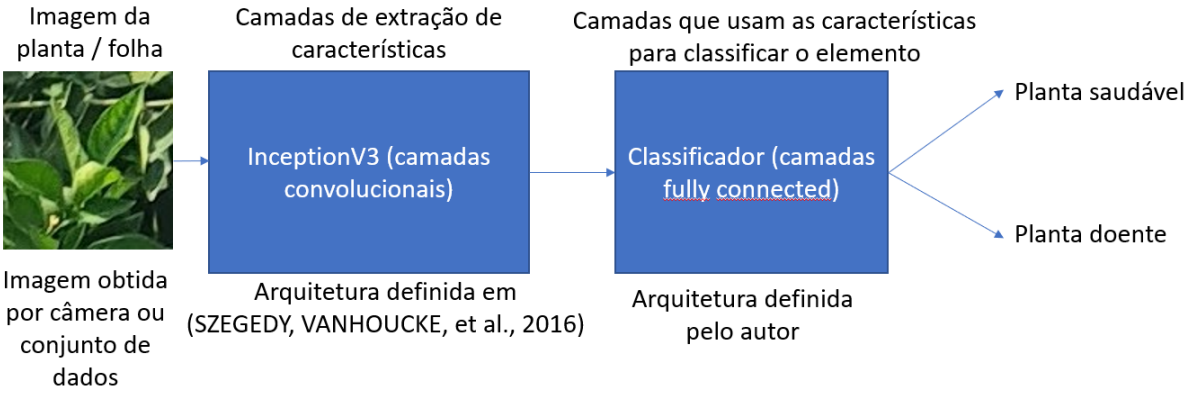

Figura 6: Diagrama de blocos da rede

Para o bloco classificador foi escolhida uma rede neural de camadas *fully connected* devido aos resultados apresentados por essa arquitetura em (RAMCHARAN, BARANOWSKI, *et al.*, 2017). Nessa publicação também foram comparados os resultados utilizando as seguintes arquiteturas para o bloco classificador:

- *Fully connected;*
- KNN;
- SVM;

Para a métrica das perdas é utilizada a função *Categorical Cross Entropy* que pode ser definida matematicamente como:

$$Perda_{CCE} = -\sum_{i=1}^{n} y_i * \log \hat{y}_i$$

Onde $\hat{y}_i$ é o i-ésimo valor escalar de saída da rede, y_i é a saída esperada correspondente e n é a quantidade de amostras consideradas para essa medida. Essa métrica é adequada para resumir o quão distinguíveis são duas distribuições de probabilidade discretas entre si. Assim quando as duas distribuições se aproximam (significando que a saída da rede se aproxima da saída esperada), o valor da perda diminui.

Para a acurácia da rede é utilizada a função *Categorical Accuracy* que pode ser definida como:

$$Acuracia_{Categorical\ Accuracy} = \frac{quantidade\ de\ estimativas\ corretas}{total\ de\ estimativas\ realizadas}$$

Assim a acurácia representa a porcentagem de vezes que a rede obteve uma saída de acordo com o valor esperado para a entrada.

A rede neural apresentada aqui tem resolução de entrada 256x256 pixels, sendo realizado o processo de redução da escala de imagens maiores que essa resolução antes da apresentação para a rede.

3.2.1 Redes Neurais Convolucionais

Redes neurais convolucionais são um sub tipo de rede neural muito usado em aplicações de reconhecimento de imagens, sistemas de recomendação (VAN DEN OORD, DIELEMAN e SCHRAUWEN, 2013), classificação de imagens, segmentação de imagens, análise de imagens médicas, processamento de linguagem natural (COLLOBERT e WESTON, 2008), interface cérebro-computador (AVILOV, RIMBERT, *et al.*, 2020) e séries temporais financeiras (TSANTEKIDIS, PASSALIS, *et al.*, 2017).

Sua principal vantagem para essas aplicações é o uso da técnica de convoluções em janelas deslizantes que são capazes de extrair características dos dados de entrada (que podem ser imagens ou outros tipos de dados) para assim conseguir utilizar essas características (ou *features*) para executar a tarefa em questão.

No âmbito de classificação de imagens (aplicação de interesse da solução aqui apresentada) as redes neurais convolucionais utilizam cada pixel de cada camada da imagem como uma posição na matriz que forma a imagem. O valor dessa posição é proporcional a intensidade daquele pixel naquela camada (um pixel vermelho terá um valor alto na camada vermelha e baixo nas camadas verde e azul). Entre a entrada e saída de cada camada da rede é executada a operação de convolução entre uma matriz de filtragem e cada submatriz representada por uma sub-região da imagem. O resultado dessa operação é então aplicado à função de ativação de cada neurônio da camada e o resultado é apresentado como saída da camada para aquela região. O processo é repetido para cada região até que se aplique a convolução em toda a imagem. Esse processo pode ser repetido após a saída de uma camada convolucional (realizando a convolução de uma convolução).

Nesse processo de convoluções a rede é capaz de identificar certas características da imagem e usar essas características na classificação da imagem ou para realizar a tarefa proposta para a rede.

3.2.2 Camadas *fully connected*

Após as camadas convolucionais extraírem as características da imagem apresentada na entrada, podem ser utilizadas camadas chamadas de "totalmente conectadas" ou "*fully connected"* que são camadas em que cada neurônio de uma camada está conectado com todos os outros neurônios da camada seguinte. Como cada conexão tem um peso sináptico diferente referente àquela conexão, esse tipo de camada aumenta de forma exponencial o número de parâmetros treináveis da rede, mas também é uma ótima ferramenta para a tarefa de regressão, comumente delegada às camadas classificadoras finais de uma rede convolucional como a InceptionV3.

3.2.3 InceptionV3

A arquitetura de rede neural InceptionV3 foi proposta em (SZEGEDY, VANHOUCKE, *et al.*, 2016) e foi uma evolução da rede que era como "a rede neural do Google" (GoogLeNet) por ter como um dos seus autores um engenheiro dessa empresa. Ela foi idealizada para diferenciar imagens de objetos como proposta para uma solução do desafio ImageNet. As categorias de imagens contidas nesse desafio e que formam também um conjunto de dados de acesso público contabilizam mais de 20.000 classes diferentes como "balão" e "carro".

Uma das principais vantagens do uso dessa arquitetura para esse projeto se encontra no fato de ser uma rede relativamente pequena (menos de 25 milhões de pesos neurais) se comparada com outras arquiteturas de capacidade equivalente como a AlexNet (60 milhões de pesos neurais) apresentada em (KRIZHEVSKY, SUTSKEVER e HINTON, 2012).

A arquitetura da Inception é composta de uma série de camadas convolucionais seguidas por um bloco classificador. As camadas convolucionais extraem características da imagem que então são usadas pela camada classificadora para colocar a imagem em uma das categorias da classificação. A lista com a descrição das camadas da InceptionV3 como ela foi idealizada em (SZEGEDY, VANHOUCKE, *et al.*, 2016) pode ser vista na Tabela 10.

Tipo da camada	***Tamanho do filtro / passo***	***Resolução da entrada***
Convolucional	3x3/2	299x299x3
Convolucional	3x3/1	149x149x32
Convolucional padded	3x3/1	147x147x32
Pooling	3x3/2	147x147x64
Convolucional	3x3/1	73x73x64
Convolucional	3x3/2	71x71x80
Convolucional	3x3/1	35x35x192
3xInception	Módulo 1	35x35x288
5xInception	Módulo 2	17x17x768
2xInception	Módulo 3	8x8x1280
Pooling	8x8	8x8x2048
Linear	Logística	1x1x2048
Softmax	Classificador	1x1x1000

Tabela 10: Camadas da rede InceptionV3

A partir da Tabela 10 é possível verificar que a rede InceptionV3 utiliza filtros (*kernels*) de tamanhos diferentes que resultam em resoluções de saída da camada diferentes (que se tornam resoluções de entrada diferentes para a próxima camada).

3.2.4 Processo de seleção do *Dropout*

Redes profundas (como é o caso da InceptionV3) podem facilmente sofrer do mal de *overfitting*, que consiste na rede se adequar demais aos dados de treinamento e com isso perder a capacidade de generalização. No caso de entradas com grande possibilidade de variação (como é o caso com imagens, por exemplo) essa perda da capacidade de generalização pode tornar a rede ineficaz.

Com isso em mente é utilizada uma camada de *dropout* entre as camadas classificadoras que faz com que somente parte dos neurônios da camada classificadora sejam usados a cada amostra, criando assim "subredes" de neurônios dentro da rede neural completa sendo utilizada. Esse processo diminui a chance de ocorrer *overfitting* ao treinarmos a rede.

O valor do *dropout* varia entre 0 e 1 e indica a porcentagem de neurônios que serão desligados de cada vez. Para cada projeto esse valor é influenciado pelas características da rede e dos dados de entrada.

Para o projeto aqui apresentado, a escolha do *dropout* se deu após o seguinte procedimento:

1) Foram definidos 9 *dropouts* candidatos iniciando em 0,1 e aumentando de um décimo de unidade até 0,9. Um desses valores é selecionado como o candidato da vez;

2) A rede é inicializada com os valores dos neurônios das camadas convolucionais carregados do desafio ImageNet;

3) A camada de *dropout* que é colocada entre as camadas *fully connected* da parte classificadora da rede (final) tem seu valor definido para o valor candidato da vez;

4) Realiza-se um treinamento por 10 épocas com o dataset Citrus augmented, armazenando em um arquivo CSV os valores de acurácia e perda, ambos de validação, para cada época de treinamento;

6) Repetir os passos 2 a 4 para cada valor de *dropout* candidato;

7) Após terminar o treinamento com todos os valores de *dropout* candidatos, calcula-se para cada candidato a variação entre a maior e a menor acurácia (delta acurácia) do histórico de treinamento;

8) Gera-se uma lista com os candidatos em ordem crescente de delta acurácia e outra lista em ordem crescente de menor perda no histórico;

9) Iniciando-se com o candidato com o menor delta acurácia (ou seja, que variou menos durante o treinamento e, portanto, demonstrou menor propensão a *overfitting*) verifica-se se aquele candidato também está entre as 3 menores perdas. Caso o candidato não esteja, passa-se para o próximo;

10) Ao se encontrar o candidato com a menor variação de acurácia e que esteja entre as 3 menores perdas, seleciona-se esse candidato como *dropout* escolhido.

3.2.5 Os conjuntos de validação e teste

Durante a realização dos treinamentos é utilizado um grupo de imagens para realizar a validação dele. Essas imagens não são apresentadas durante o treinamento e são usadas somente

no final de cada época para verificar o resultado do treinamento. Esse grupo de validação é composto por 30% das imagens do dataset utilizado para treinamento e o resultado da inferência do grupo de validação a cada época de treinamento é utilizado para determinar qual o melhor conjunto de pesos de todo o histórico de treinamento. Essa técnica de utilizar um grupo de treinamento e outro de validação diminui as chances de ocorrer *overtraining.* Para evitar a possibilidade de que o grupo de validação tenha um viés, esse grupo é refeito a cada época de treinamento. Portanto imagens que foram utilizadas para treinamento em uma época podem ser usadas para validação em outra.

Ao final do treinamento como um todo, os pesos neurais que atingiram o melhor resultado com o grupo de validação são carregados na rede e é então apresentado o conjunto de teste para verificar a capacidade de generalização da rede. Esse grupo de teste é composto por imagens que são segregadas das imagens de treinamento e validação e não são apresentadas durante o treinamento. Como essas imagens são fixas e segregadas manualmente na concepção do conjunto de dados, elas não participam de forma alguma do processo de treinamento.

Para os conjuntos de dados em que é aplicada a técnica de *data augmentation*, o conjunto de teste não passa por esse processo.

3.3 *Transfer Learning*

De acordo com (LIN e JUNG, 2017) transferência de aprendizado ou *transfer learning* em aprendizado de máquina é o uso de capacidades aprendidas através do treinamento para uma tarefa em outra tarefa que pode ou não estar relacionada à tarefa original. Por exemplo, as mesmas capacidades aprendidas para a tarefa de diferenciar modelos de carros entre si podem ser utilizadas para a tarefa de separar caminhões de diferentes modelos.

Formalmente podemos definir: Um domínio D consistindo em um espaço de características X e uma distribuição de probabilidade $P(X)$, onde $X = \{x_1, \dots, x_n\} \epsilon X$. Dado um domínio específico $D = \{X, P(X)\}$, uma tarefa consiste em dois componentes: um espaço de classes Y e uma função preditiva objetivo $f : X \rightarrow Y$. A função f é usada para predizer a classe $f(x)$ de uma nova instância x. Essa tarefa, identificada como $T = \{Y, f(x)\}$, é aprendida a partir dos dados de treinamento que consiste nos pares $\{x_i, y_i\}$ onde $x_i \in X$ e $y_i \in Y$.

Dado um domínio fonte (ou *source*) D_S e uma tarefa de aprendizagem T_S, um domínio alvo (ou *target*) D_T e uma tarefa de aprendizagem T_T, onde $D_S \neq D_T$, ou $T_S \neq T_T$, *transfer learning* tem por objetivo ajudar a melhorar o aprendizado da função preditiva $f_T(\cdot)$ em D_T usando o conhecimento em D_S e T_S.

Assim sendo temos que para a solução apresentada o domínio *source* será o conjunto de imagens do desafio ImageNet e a tarefa *source* será a classificação dessas imagens em aproximadamente 1.000 classes diferentes. Já o domínio *target* será o conjunto de imagens de folhas de citros *Citrus Dataset*, Jacto Dataset e Dataset de árvores inteiras, e a tarefa *target* será a separação dessas imagens em saudáveis e doentes.

Assim todos os resultados mostrados aqui têm como ponto de partida do treinamento a rede utilizada no desafio ImageNet, realizando o treinamento com as características e os conjuntos de dados definidos em cada resultado.

3.3.1 Treinamento da rede toda X treinamento das camadas finais

Uma das facilidades do uso da arquitetura InceptionV3 como base para a solução proposta é a possibilidade de carregá-la no projeto de forma simples. A arquitetura da rede e todos os seus pesos neurais podem ser carregados com valores pré treinados a partir da solução encontrada para o desafio ImageNet (esse processo de carregamento com valores pré-treinados é conhecido como aprendizado por transferência ou *transfer learning*).

Na linguagem Python, basta adicionar ao projeto a biblioteca interna relacionada ao InceptionV3. Uma vez com a biblioteca inclusa no projeto, basta salvar o modelo da rede neural InceptionV3 em uma variável usando o comando InceptionV3 com os parâmetros descritos em sua documentação.

Dentro dos parâmetros da rede neural invocada dessa forma pode-se determinar se os pesos dos neurônios que compõe a rede serão inicializados com valores aleatórios ou se serão atribuídos os valores do treinamento utilizado no desafio ImageNet. A atribuição desses valores pré-fixados simplifica o treinamento, pois pode-se deixar esses pesos fixos em seus valores iniciais e aplicar o treinamento somente nas camadas posteriores da rede, exigindo assim uma menor capacidade computacional dos equipamentos que farão o processamento do ajuste dos pesos durante essa fase.

A desvantagem de utilizar os pesos com valores pré-fixados é que nem sempre essas camadas extratoras de características da imagem de entrada são capazes de entregar como saída características suficientemente diferenciáveis do ponto de vista das camadas posteriores que farão a segmentação das entradas (camadas classificadoras). Em outras palavras, os valores dos pesos neurais suficientes no desafio ImageNet para diferenciar uma cadeira de um balão podem não ser capazes de diferenciar uma folha de laranjeira saudável de uma outra folha da mesma espécie de árvore que esteja apresentando características visuais de alguma doença.

Nesses casos em que os valores pré-treinados das camadas extratoras não são suficientes para atingir o desempenho da tarefa desejada, é possível realizar o treinamento de todas as camadas da rede, utilizando os valores pré-estabelecidos como um ponto inicial para o treinamento e avançando no desempenho a partir de lá.

Nesse caso do treinamento da rede toda, a arquitetura apresentada na InceptionV3 também possui uma vantagem em relação às outras arquiteturas com o mesmo nível de desempenho, pois seu tamanho reduzido permite que o treinamento da rede seja feito em hardware considerado “doméstico” em tempo considerado reduzido. Como exemplo, o computador portátil do autor que possui 8 núcleos e 8 Gb de memória RAM, sem placa de vídeo, precisa de 2 minutos por época para treinar somente as camadas finais da rede e 10 minutos por época para treinar toda a rede. Em um treinamento de 20 épocas (configuração da maioria dos treinamentos apresentados aqui) é necessário em torno de 3 horas e meia para executar todo o treinamento da rede completa.

3.4 O problema de identificação de imagem de árvore inteira

Os primeiros objetivos do projeto apresentado utilizam como entrada de dados imagens de folhas individuais como mostrado no item referente ao conjunto de dados. Porém o uso final do projeto prevê a entrada de imagens capturadas em campo que, por sua natureza, contém imagens com múltiplas folhas e galhos, assemelhando-se mais a vista de uma planta inteira. Assim é necessário adequar o projeto para, nas etapas finais, ser capaz de receber entradas com essas características.

Um dos problemas práticos desse tipo de imagem de planta inteira encontra-se no fato de que a maioria das folhas em uma planta doente não apresenta os sintomas visíveis da doença.

Assim é esperado que se a imagem de uma planta inteira for dada como entrada para a rede neural treinada com imagens de folhas individuais, a rede não seja capaz de diferenciar plantas doentes de saudáveis. A solução mais simples para esse problema é realizar o retreinamento da rede com imagens das plantas inteiras. Essa solução, porém, pode não ser suficiente dado que na entrada da rede é feito um redimensionamento da imagem para a configuração de entrada pré-estabelecida de 256x256 pixels. Esse redimensionamento é necessário ao fazer o treinamento da rede para que o computador que está realizando esse processo não fique sem memória. Assim devido a esse processo, caso a imagem contenha um resumido número de pixels que indique a presença da doença, a rede pode não ser capaz de detectar tal diferença.

Uma das soluções idealizadas é dividir a imagem da planta inteira em fatias iguais de tamanho fixo e usar essas fatias como entrada da rede neural que identifica a presença de HLB.

Outra solução proposta é treinar uma rede neural que seja capaz de identificar e apresentar as coordenadas de uma caixa de contorno em volta de cada ramo da árvore, independente se o ramo em questão contém ou não características dos sintomas da doença. Assim seria possível apresentar cada ramo individualmente para a rede classificadora.

Em ambos os casos dada uma imagem de uma árvore inteira, essa seria dividida em várias partes (ramos ou fatias) que seriam então apresentadas para a rede neural classificadora. A maioria dessas partes provavelmente não terá características visíveis que indiquem a presença de HLB, porém se uma certa quantidade dessas partes for classificada como doente, então a árvore toda poderá ser classificada como tal. Um exemplo de imagem de árvore inteira com destaque para um ramo que apresenta características visíveis da infecção por HLB é mostrado na Figura 7.

Figura 7: Imagem de planta inteira com destaque para ramo afetado por HLB

Nas seções a seguir serão detalhados cada um desses métodos apresentados anteriormente.

3.4.1 Divisão em fatias (*slices*)

A divisão em fatias pode ser feita de forma simples utilizando a linguagem Python que já é usada na montagem e operação da rede neural. As imagens de entrada devem ser divididas em pedaços iguais de tamanhos preestabelecidos e pedaços que sobrarem no final da divisão da imagem e que forem menores que uma área mínima devem ser descartados.

Para a divisão em fatias, o primeiro desafio estabelecido é definir o tamanho ideal para as fatias de tal modo que a rede seja capaz de diferenciar pedaços que contenham características da doença de imagens que contenham somente folhas saudáveis. Para atingir tal objetivo foi proposto testar diversos tamanhos de fatias em imagens de plantas que contenham

características do HLB e verificar em qual dos tamanhos a rede neural classificadora tem melhor desempenho. Para essa etapa foram idealizados os seguintes passos:

1. A partir de imagens de plantas que contenham as características da doença, gerar fatias dos tamanhos preestabelecidos;
2. Analisar as fatias e separá-las entre as que contém folhas com características do HLB e as que apresentam somente folhas saudáveis;
3. Montar para cada tamanho de fatia um conjunto de dados com essas duas classes de imagens;
4. Apresentar cada um desses conjuntos de dados para o classificador treinado e verificar em qual deles obtêm-se o melhor desempenho.

O segundo desafio a ser vencido para essa técnica proposta é evitar que o processo de fatiamento da imagem faça com que as folhas com características da doença sejam particionadas entre fatias diferentes, caso isso aconteça será diminuído também o número de pixels úteis na identificação da doença para cada imagem, além de comprometer a identificação da estrutura da folha doente pela rede neural.

Esse objetivo pode ser atingido usando durante o fatiamento uma janela móvel que sobrepõe parte da fatia anterior na próxima. Assim, caso um grupo de folhas que apresente as características desejadas fique sobre o limite de uma das fatias, na imagem seguinte esse trecho ficará afastado da margem dela.

3.4.2 Divisão em ramos

Outra solução idealizada é utilização de uma rede neural artificial de classificação semântica treinada para identificar a posição de ramos de folhas na imagem da árvore inteira. Essa rede deve ser capaz de, a partir de uma imagem de uma planta inteira, ter como saída as coordenadas das caixas de contorno que cercam cada um dos ramos. Com essa informação é possível fazer uma rotina que recorta os pixels da imagem original referentes a cada caixa de contorno e alimenta essa parte da imagem para a rede neural classificadora treinada. Um exemplo de resultado esperado da saída da solução para divisão de imagens de árvore inteira em ramos é mostrado na Figura 8

A solução selecionada para essa tarefa é a rede neural YoloV5, como apresentada em (REDMON, DIVVALA, *et al.*, 2016). Conhecida como “a rede neural do Facebook”, pelo seu desenvolvimento e uso ter se dado primordialmente dentro dessa empresa, essa rede neural é de fácil aplicação e tem suas configurações apresentadas de modo simplificado. Por não ser o ponto técnico principal da solução para identificação da presença de HLB, essas características simplificadas se apresentam como uma vantagem para o projeto geral aqui apresentado.

Um exemplo de uso gerado a partir do conjunto de dados que vem junto com a instalação do YoloV5 pode ser visto na Figura 9:

Figura 8: Exemplo de saída esperada da rede YoloV5, entrada (acima) e saída esperada (abaixo)

Figura 9: Exemplo de uso do YoloV5 gerado pelo autor a partir do conjunto de dados padrão

4 RESULTADOS

Primeiro serão apresentados os resultados do treinamento da rede neural classificadora utilizando o conjunto de dados Citrus Dataset na sua forma original e com a aplicação de *data augmentation*.

Depois serão apresentados os resultados do treinamento utilizando um conjunto de dados que contém parte das imagens provenientes do Citrus Dataset e parte capturada em ambiente de campo.

4.1 Treinamento com Citrus Dataset

Para o treinamento utilizando o conjunto de dados Citrus primeiramente foram utilizadas 4 das 5 classes de folhas presentes no dataset. A classe “Melanose” foi descartada devido à baixa qualidade e quantidade das imagens no dataset. Nessa fase a rede foi configurada para ter somente os pesos das camadas finais (*fully connected*) modificados, sendo os demais pesos fixados nos valores do treinamento “ImageNet” como descrito anteriormente.

Em um segundo momento essas 4 classes passaram pelo processo de *data augmentation* para aumentar a quantidade e a variabilidade de imagens de cada classe e verificar o impacto na performance do treinamento. Nessa etapa ainda foi utilizada arquitetura com os pesos das camadas convolucionais fixos enquanto os pesos das camadas *fully connected* eram treináveis.

Finalmente foi realizado o treinamento de todos os pesos da rede, inclusive das camadas provenientes do modelo InceptionV3.

Para todos os treinamentos foi utilizado particionamento de 80% das imagens para treinamento e 20% para validação. As camadas finais foram configuradas com 1024 neurônios (penúltima camada) e 4 neurônios (cama final) respectivamente.

Os resultados desses treinamentos e seus respectivos desempenhos são mostrados na Tabela 11:

Conjunto de dados	Quantidade de imagens	Accuracy de treinamento	Loss de treinamento	Accuracy de validação	Loss de validação	Treinamento de todas as camadas?
Citrus	609	0.5731	0.9741	0.6777	0.9419	Não
Citrus Augmented	9240	0.5258	1.1032	0.6107	0.9082	Não
Citrus	609	0.9811	0.0732	0.9580	0.1753	Sim
Citrus Augmented	9240	0.9994	0.0020	0.9933	0.0516	Sim

Tabela 11: Resultado do treinamento utilizando o dataset Citrus

O histórico dos desempenhos de acurácia e perdas para treinamento e validação com os conjuntos de dados apresentados na Tabela 11 são mostrados a seguir:

Conjunto de dados: Citrus

Treinamento de todas as camadas: não

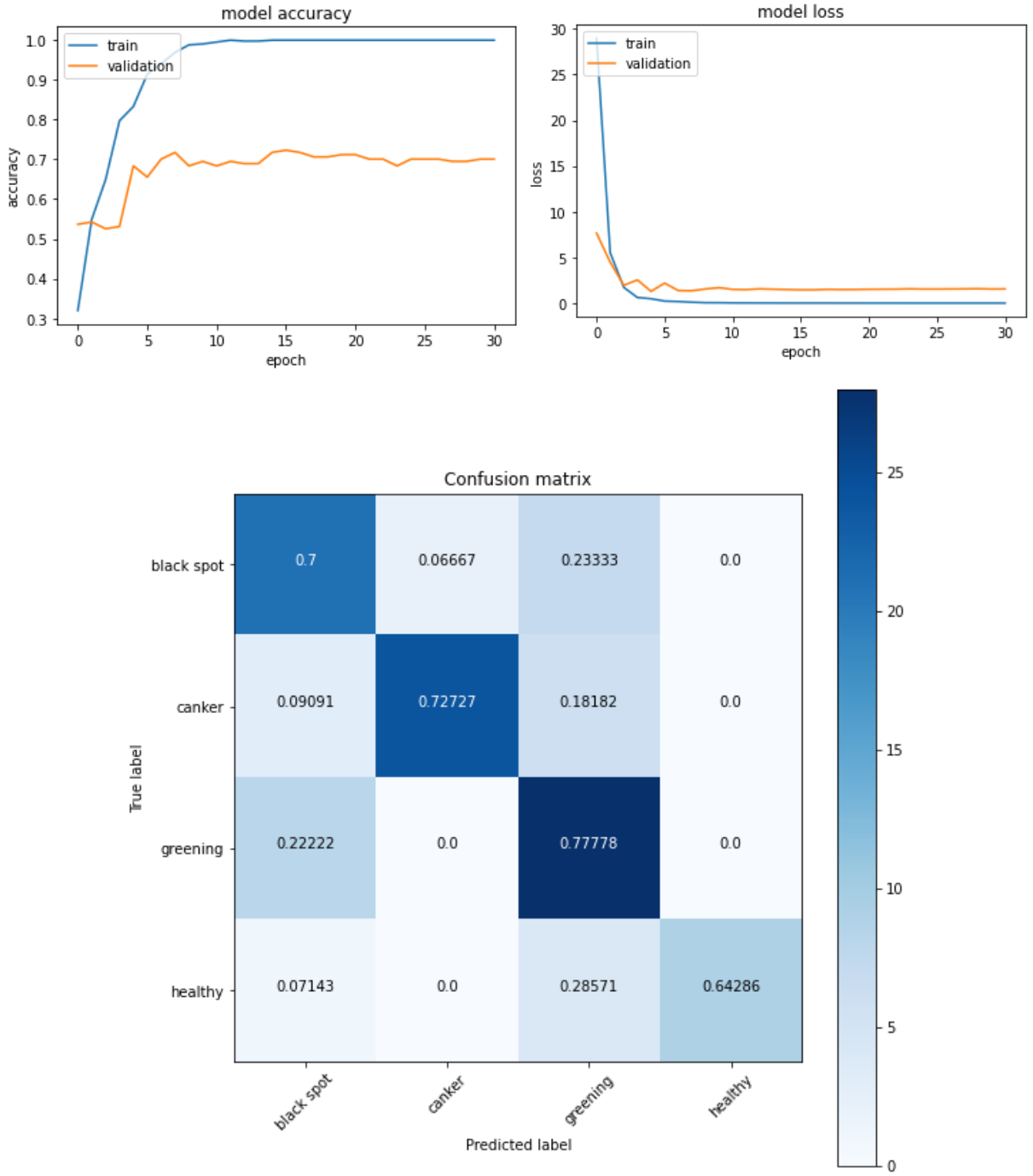

Figura 10: Desempenho durante o treinamento utilizando o conjunto de dados Citrus treinando somente as camadas finais

Conjunto de dados: Citrus augmented

Treinamento de todas as camadas: não

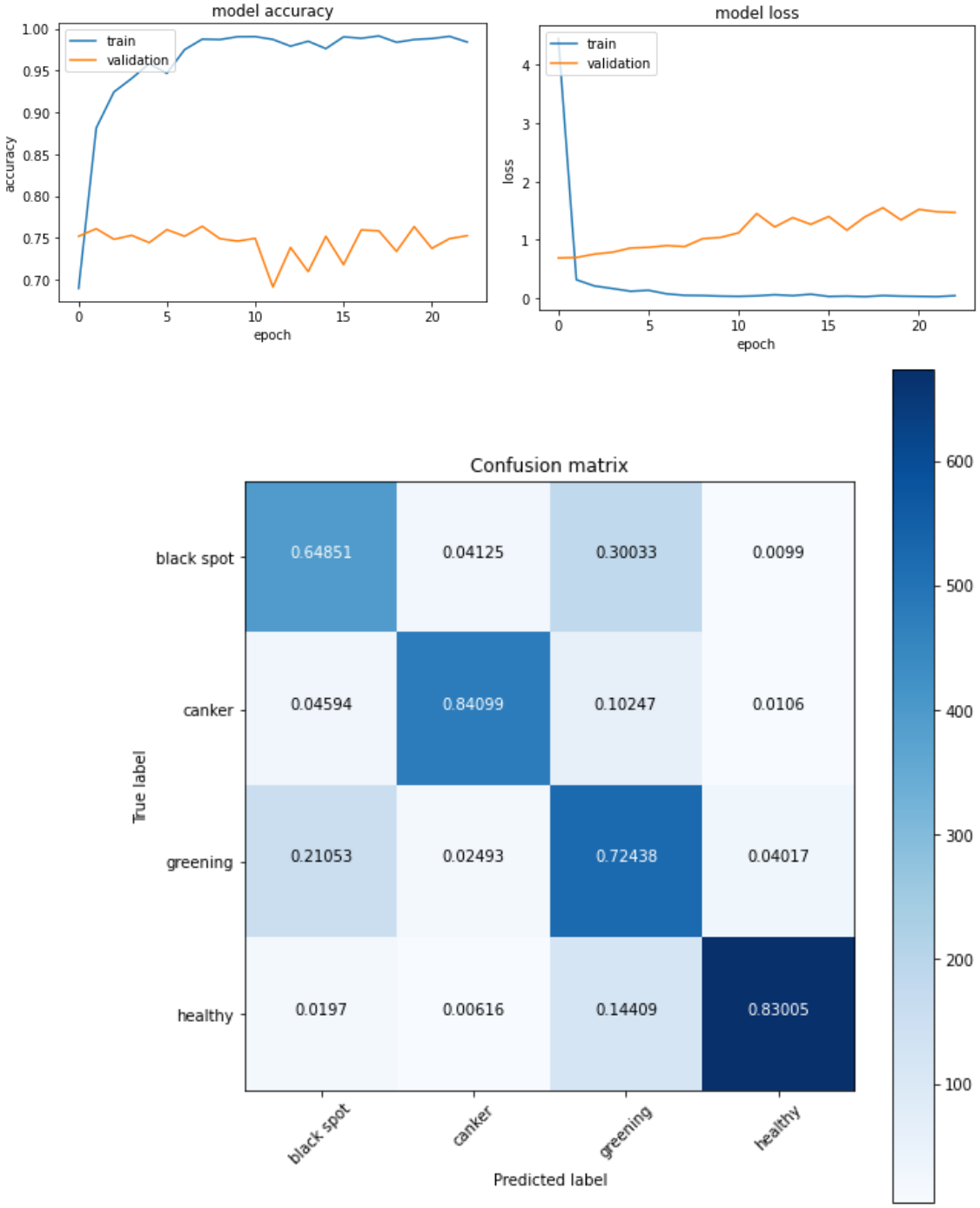

Figura 11: Desempenho durante o treinamento utilizando o conjunto de dados Citrus treinando somente as camadas finais

Conjunto de dados: Citrus

Treinamento de todas as camadas: sim

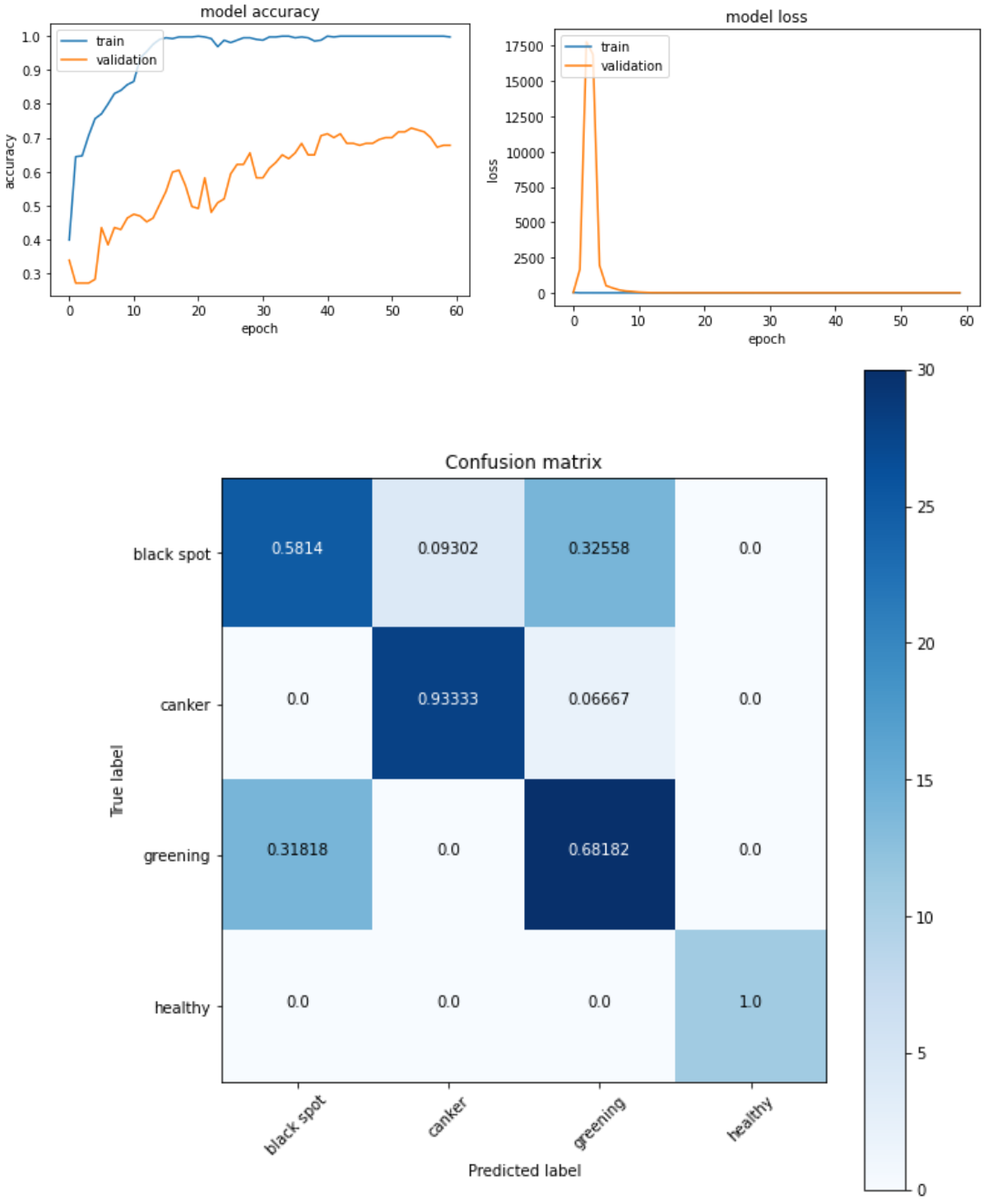

Figura 12: Desempenho durante o treinamento utilizando o conjunto de dados Citrus treinando todas as camadas

Conjunto de dados: Citrus augmented

Treinamento de todas as camadas: sim

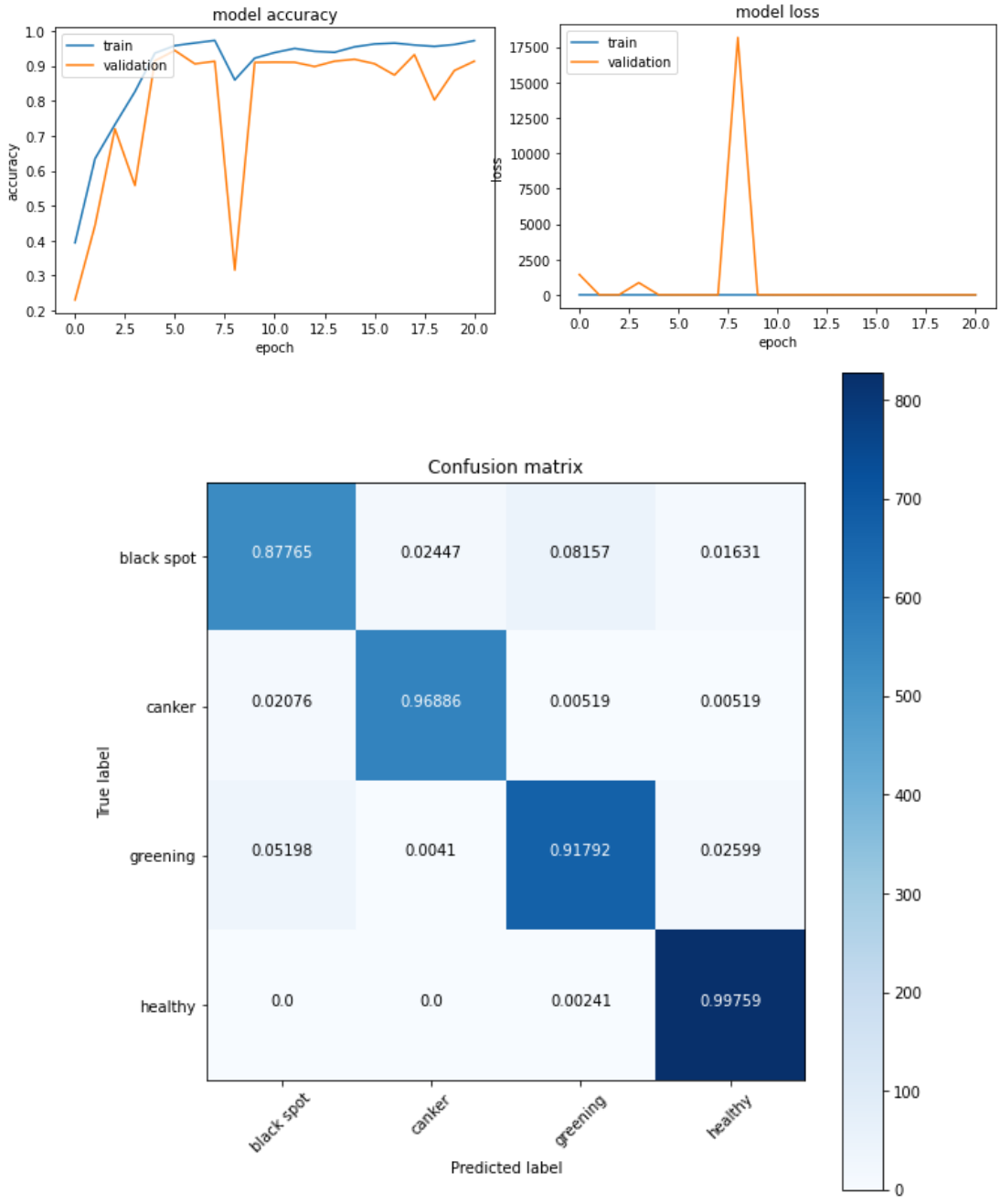

Figura 13: Desempenho durante o treinamento utilizando o conjunto de dados Citrus augmented e treinando todas as camadas

Analisando os resultados dos treinamentos pode-se inferir que o principal fator que impacta as métricas é o treinamento da rede inteira ao invés do treinamento somente das camadas finais. A hipótese levantada é que como a rede treinada para o desafio ImageNet foi idealizada para diferenciar imagens muito diferentes entre si, as camadas convolucionais com os pesos herdados desse treinamento são incapazes de diferenciar dois objetos muito parecidos com pequenas diferenças de tonalidade como é o caso da separação entre folhas saudáveis e doentes.

O aumento da quantidade de imagens através do uso da técnica de *data augmentation* se mostrou capaz de melhorar o resultado do treinamento quando todos os pesos da rede são treináveis. Já quando somente os pesos das camadas classificadoras são treinados, essa técnica não mostrou impacto positivo nos resultados, possivelmente causando *overfitting* o que piorou os resultados obtidos.

4.2 Treinamento com Citrus + Jacto Dataset

A partir dos treinamentos utilizando o conjunto de dados Citrus, foi definida a arquitetura da rede para os demais treinamentos, sendo aplicáveis daqui para frente somente pequenos ajustes.

Para os conjuntos de dados “Citrus + Jacto” e “Citrus + Jacto *augmented*” foram realizados treinamentos com os mesmos parâmetros dos anteriores: um com o conjunto híbrido original e outro com o conjunto híbrido após o uso da técnica de *data augmentation*. Nos dois casos foram treinados todos os pesos da rede. Para esses dois treinamentos foram obtidos os seguintes resultados:

Conjunto de dados	Quantidade de imagens	Accuracy de treinamento	Accuracy de validação	Loss de treinamento	Loss de validação
Citrus + Jacto	180	0.8619	0.9500	0.3323	0.1922
Citrus + Jacto Augmented	720	0.9664	0.9714	0.1302	0.0901

Tabela 12: resultados dos treinamentos utilizando o conjunto híbrido original e aumentado

A partir desses resultados estabeleceu-se que os equipamentos disponíveis para a captura das imagens em campo têm as características técnicas necessárias para montagem do conjunto de dados próprios, com imagens capturadas em campo.

4.3 Treinamento com Jacto Dataset

Para esse conjunto de dados foram realizados treinamentos com os mesmos parâmetros dos anteriores e obtidos os seguintes resultados:

Conjunto de dados	Quantidade de imagens	Accuracy treinamento	Accuracy validação	Loss de treinamento	Loss de validação	Accuracy de teste	Loss teste
Jacto Dataset	751	1.0000	0.9598	0.0023	0.1577	0.9598	0.6907
Jacto Augmented Dataset	1604	1.0000	0.9949	0.0027	0.0132	0.95990	0.2525

Tabela 13: Resultados do treinamento com conjunto de dados Jacto

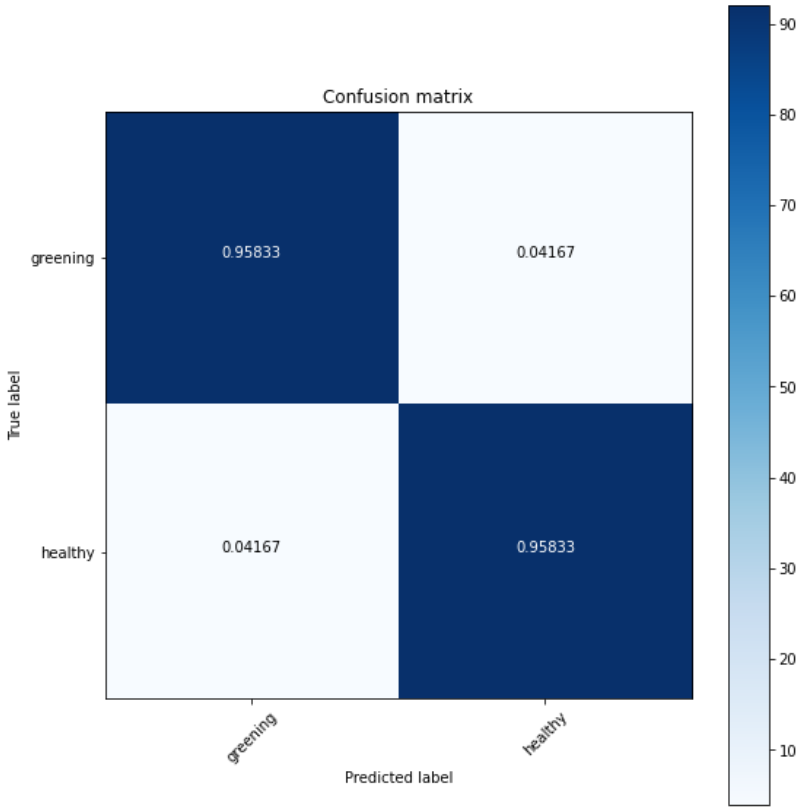

Figura 14: Matriz de confusão do treinamento com o conjunto Jacto

A comparação com os valores apresentados em (RAMCHARAN, BARANOWSKI, *et al.*, 2017) é mostrada na Tabela 14.

Arquitetura	Dataset	Accuracy (dataset de teste 10%) [%]
InceptionV3	**Jacto**	**95.98**
SVM	Cassava Leaflet	93.00
InceptionV3	Cassava Leaflet	92.90
SVM	Cassava Original	90.90
InceptionV3	Cassava Original	89.60
KNN	Cassava Leaflet	83.30
KNN	Cassava Original	78.40

Tabela 14: Comparação com os valores da publicação que serviu como base para a arquitetura

4.4 Treinamento com Dataset de árvores inteiras

Para o dataset de árvores inteiras foi realizado o treinamento utilizando duas técnicas diferentes para dividir as imagens das árvores em pedaços: usando a técnica de separação por fatias (*slices*) e utilizando a técnica de separação por ramos. Abaixo são apresentados os resultados com cada uma dessas técnicas.

4.4.1 Usando a técnica de fatias (*slices*)

Para a divisão por fatias, realizou-se o treinamento separando as imagens de árvores inteiras em fatias quadradas de 300 pixels de lado. Foram separadas 10% das imagens para o conjunto de teste (imagens que não serão apresentadas para a rede durante o treinamento). Os resultados obtidos são mostrados a seguir:

Conjunto de dados	Quantidade de imagens	Accuracy treinamento	Accuracy validação	Loss de treinamento	Loss de validação	Accuracy de teste	Loss teste
Árvores inteiras (fatias)	1152	0.9160	0.7849	0.2263	0.6634	0.75	0.66233

4.4.2 Usando a técnica de ramos

Para a divisão por ramos, realizou-se o treinamento separando as imagens de árvores inteiras em ramos anotados manualmente e posteriormente segregados da imagem principal. Foram separadas 10% das imagens para o conjunto de teste (imagens que não serão apresentadas para a rede durante o treinamento). Os resultados obtidos são mostrados a seguir:

Conjunto de dados	Quantidade de imagens	Accuracy treinamento	Accuracy validação	Loss de treinamento	Loss de validação	Accuracy de teste	Loss teste
Árvores inteiras (ramos)	1432	0.9851	0.6434	0.0583	0.9182	0.77777	0.43383

5 CONCLUSÃO

De acordo com os resultados mostrados em 4.1, a arquitetura proposta em (RAMCHARAN, BARANOWSKI, *et al.*, 2017) é capaz de diferenciar elementos doentes de saudáveis quando treinada a partir de imagens de folhas em fundo homogêneo desde que o treinamento abranja todos os pesos da rede e não somente os pesos das camadas classificadoras finais. Foi possível reproduzir a arquitetura apresentada na publicação, ajustando seus parâmetros para obter resultados compatíveis com os apresentados, utilizando *datasets* de terceiros e criados pelo autor.

A partir dos resultados de 4.2 é possível concluir que uma câmera digital comum é capaz de capturar as imagens com o nível de detalhe necessário para que a rede neural proposta seja capaz de diferenciar entre as classes.

A partir dos resultados de 4.3 é possível concluir que a rede é capaz de identificar as plantas doentes e separá-las das plantas saudáveis mesmo com o fundo da imagem em ambiente de campo (não-homogêneo). Isso demonstra que, para a solução proposta de captura de imagens por uma câmera posicionada em um robô móvel se movimentando pelas árvores de citros de

uma plantação, dado o enquadramento correto, a solução técnica é capaz de atingir os resultados aqui demonstrados.

A partir dos resultados de 4.4 é possível inferir que as soluções propostas em 4.4.1 e 4.4.2 não foram capazes de produzir entradas para a rede neural a partir de imagens de árvores inteiras com informações suficientes para reproduzir os resultados demonstrados com os outros conjunto de dados utilizados.

Pode-se então concluir que o objetivo principal da solução aqui apresentada foi atingido, pois ela é capaz de separar elementos atingidos por HLB de elementos saudáveis a partir de imagens de campo de folhas de árvores de citros.

6 REFERÊNCIAS

AVILOV, O. et al. **Deep Learning Techniques to Improve Intraoperative Awareness Detection from Electroencephalographic Signals**. 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society. [S.l.]: IEEE. 2020.

BAI, X. et al. A fuzzy clustering segmentation method based on neighborhood grayscale information for defining cucumber leaf spot disease images. **Computers and Electronics in Agriculture**, 2017. 157-165.

COLLOBERT, R.; WESTON, J. **A Unified Architecture for Natural Language Processing:** Deep Neural Networks with Multitask Learning. 25th International Conference on Machine Learning. [S.l.]: [s.n.]. 2008. p. 160-167.

DECHANT, C. et al. Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning. **Phytopathology**, jun. 2017.

FERREIRA, A. D. S. et al. Weed detection in soybean crops using ConvNets. **Computers and Electronics in Agriculture**, out. 2017.

FUENTES, A. et al. A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition. **Sensors**, set. 2017.

FUNDECITRUS. Os dez mandamentos do HLB. **Citricultor**, fev. 2015.

HA, J. G. et al. Deep convolutional neural network for classifying Fusarium wilt of radish from unmanned aerial vehicles. **Journal of Applied Remote Sensing**, dez. 2017.

JOHANNES, A. et al. Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. **Computers and Electronics in Agriculture**, abr. 2017.

JORGE, L. A. D. C.; INAMASU, R. Y. Detecção de Greening dos citrus por imagens multiespectrais. **AGRICULTURA DE PRECISÃO: RESULTADOS DE UM NOVO OLHAR**, 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. **NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems**, v. 1, p. 1097-1105, dez. 2012.

LAN, Y. et al. Comparison of machine learning methods for citrus greening detection on UAV multispectral images. **Computers and Electronics in Agriculture**, p. 11, fev. 2020.

LI, S. et al. Citrus Greening: Management Strategies and Their Economic Impact. **HORTSCIENCE**, v. 55, mar. 2020.

LIN, Y.-P.; JUNG, T.-P. Improving EEG-Based Emotion Classification Using Conditional Transfer Learning. **Frontiers in Human Neuroscience**, 2017.

LU, J. et al. An in-field automatic wheat disease diagnosis system. **Computers and Electronics in Agriculture**, set. 2017.

LU, J. et al. An in-field automatic wheat disease diagnosis system. **Computers and Electronics in Agriculture**, n. 142, p. 369-379, set. 2017. ISSN https://doi.org/10.1016/j.compag.2017.09.012.

MA, J. et al. A segmentation method for greenhouse vegetable foliar disease spots images using color information and region growing. **Computers and Electronics in Agriculture**, 2017. 110-117.

MAA, J. et al. A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. **Computers and Electronics in Agriculture**, 2018. 18-24.

PADOL, P. B.; YADAV, A. A. SVM Classifier Based Grape Leaf Disease Detection. **2016 Conference on Advances in Signal Processing (CASP)**, Pune, jun. 2016.

RAMCHARAN, A. et al. Deep Learning for Image-Based Cassava Disease Detection. **Frontiers in Plant Science**, 2017.

RAMESH, S.; VYDEKI, D. Recognition and classification of paddy leaf diseases using Optimized Deep Neural network with Jaya algorithm. **INFORMATION PROCESSING IN AGRICULTURE**, set. 2019.

REDMON, J. et al. You Only Look Once: Unified, Real-Time Object Detection, 2016.

RIBEIRO, P. P. E.; JORGE, L. A. D. C.; DE PAIVA, M. S. V. Diferenciação da Greening do citros de outras doenças foliares a partir de descritores de cor e forma. **VII Workshop de Visão Computacional – WVC 2011**, Curitiba, maio 2011.

SABROL, H.; SATISH, K. **Tomato Plant Disease Classification in Digital Images using Classification Tree**. International Conference on Communication and Signal Processing. [S.l.]: [s.n.]. 2016. p. 1242-1246.

SINGH, V.; MISRA, A. K. Detection of plant leaf diseases using image segmentation and soft computing techniques. **INFORMATION PROCESSING IN AGRICULTURE 4**, 2017. 41-49.

SINGH, V.; VARSHA; MISRA, A. K. **Detection of unhealthy region of plant leaves using Image Processing and Genetic Algorithm**. 2015 International Conference on Advances in Computer Engineering and Applications (ICACEA). [S.l.]: IEEE. 2015.

SZEGEDY, C. et al. Rethinking the Inception Architecture for Computer Vision. **IEEE Xplore**, 2016.

TIWARI, V. M.; GUPTA, T. Plant leaf disease analysis using image processing technique with modified SVM-CS classifier. **IJEMT**, 2017.

TSANTEKIDIS, A. et al. **Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks**. IEEE 19th Conference on Business Informatics. [S.l.]: [s.n.]. 2017. p. 7-12.

TÜRKOĞLU, M.; HANBAY, D. Plant disease and pest detection using deep learning-based features. **Turkish Journal of Electrical Engineering & Computer Sciences**, fev. 2019.

VAN DEN OORD, A.; DIELEMAN, S.; SCHRAUWEN, B. **Deep content-based music recommendation**. [S.l.]: [s.n.]. 2013.

WANG, H. et al. **Image Recognition of Plant Diseases Based on Backpropagation Networks**. 5th International Congress on Image and Signal Processing (CISP 2012). [S.l.]: IEEE. 2012. p. 894-900.

ANEXO A – CÓDIGO PYTHON USADO PARA DATA AUGMENTATION

```
# importar os pacotes necessários
import numpy as np

import tensorflow as tf

from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import ImageDataGenerator

import keras.backend as k

from os import listdir

class Augmentation:
    def __init__(self, rot, bri, zoo, shi, name):
        self.rotation = rot
        self.brightness = bri
        self.zoom = zoo
        self.shift = shi
        self.name = name

#parametros
#DATASET_PATH = "augmented_citrus/"
#OUTPUT_PATH = "augmented_citrus/"
#IMAGE_PATH = "Citrus/Leaves/greening/g (5).png"
#DATASET_PATH = "augmented_jacto_citrus/"
#OUTPUT_PATH = "augmented_jacto_citrus/"
DATASET_PATH = "dataset_jacto/estudio_campo/"
OUTPUT_PATH = "augmented_jacto/"

#TESTE_IN = 'teste_dir/in/'
#TESTE_OUT= 'teste_dir/out/'

#fatores de multiplicacao da augmentacao para cada classe
#black spot = 12x
#canker = 13x
#greening = 10x
#healthy = 35x
#BLACK_SPOT = Augmentation(3,0,0,2, 'Black spot')
#CANKER = Augmentation(3,0,0,2, 'canker')
#GREENING = Augmentation(3,0,0,2, 'greening')
#HEALTHY = Augmentation(3,1,1,2, 'healthy')
GREENING = Augmentation(1,0,0,1, 'greening')
HEALTHY = Augmentation(1,0,0,1, 'healthy')
#AUGMENTATIONS = [BLACK_SPOT, CANKER, GREENING, HEALTHY]
AUGMENTATIONS = [GREENING, HEALTHY]
#quantas vezes vai aplicar cada transformacao:
#(rotation,brightness, zoom, shift)

PREFIXOS = ['rot', 'bri', 'zoo', 'shi']

ROTATION_RANGE = 90
ZOOM_RANGE = 0.25
```

```
BRIGHTNESS_RANGE = [0.2,1.2]
SHIFT_RANGE = [.2]

OUTPUT_FORMAT = 'png'

classes = listdir(DATASET_PATH)
print('Número de classes encontradas: ' + str(len(classes)))
print(classes)

#define os transformadores que serao usados para a augmentation
imgAugRot = ImageDataGenerator(rotation_range = ROTATION_RANGE,
horizontal_flip = True, vertical_flip = True)
imgAugBri = ImageDataGenerator(brightness_range = BRIGHTNESS_RANGE,
horizontal_flip = True, vertical_flip = True)
imgAugZoo = ImageDataGenerator(zoom_range = ZOOM_RANGE,
horizontal_flip = True, vertical_flip = True)
imgAugShi = ImageDataGenerator(width_shift_range=SHIFT_RANGE,
height_shift_range=SHIFT_RANGE, horizontal_flip = True, vertical_flip
= True)
imgAugOri = ImageDataGenerator()


def Transforma(diretorio_origem, diretorio_destino, arquivo,
transformacao, prefixo, n_transformacoes):
    if(n_transformacoes > 0):
        # carregar a imagem original e converter em array
        image = load_img(diretorio_origem + arquivo)
        image = img_to_array(image)

        # adicionar uma dimensão extra no array
        image = np.expand_dims(image, axis=0)

        # criar um gerador (generator) com as imagens do
        # data augmentation
        imgAug = transformacao
        imgGen = imgAug.flow(image, save_to_dir=diretorio_destino,
                             save_format=OUTPUT_FORMAT,
save_prefix=arquivo + '_' + prefixo + '_')
        for (i, newImage) in enumerate(imgGen):
            if(i>= n_transformacoes-1):
                break

def Transforma_dir(diretorio_origem, diretorio_destino, transformacao,
prefixo, n_transformacoes, classe):
    if(n_transformacoes > 0):
        # data augmentation
        diretorio_classe = diretorio_destino + '/' + classe + '/'
        imgAug = transformacao
        imgGen =
imgAug.flow_from_directory(directory=diretorio_origem,

save_to_dir=diretorio_classe,
                                           target_size=(256,256),
                                           save_format=OUTPUT_FORMAT,
                                           save_prefix=prefixo,
                                           classes=[classe],
                                           batch_size=1,
                                           shuffle=False,
                                           )
        quantidade_arquivos = len(imgGen.filenames)
        rodadas = n_transformacoes*quantidade_arquivos
```

```
        for (i,images) in enumerate(imgGen):
            if (i>=rodadas-1):
                break

for (i,classe) in enumerate(classes):
    print(classe)
    #for transformacoes ?
    #lista arquivos de cada classe
    original_path = DATASET_PATH + classe + '/'
    destination_path = OUTPUT_PATH + classe + '/'

    print('Rotação')
    Transforma_dir(DATASET_PATH, OUTPUT_PATH, imgAugRot, 'rot',
AUGMENTATIONS[i].rotation, classe)

    print('Brightness')
    Transforma_dir(DATASET_PATH, OUTPUT_PATH, imgAugBri, 'bri',
AUGMENTATIONS[i].brightness, classe)

    print('Shift')
    Transforma_dir(DATASET_PATH, OUTPUT_PATH, imgAugShi, 'shi',
AUGMENTATIONS[i].shift, classe)

    print('Zoom')
    Transforma_dir(DATASET_PATH, OUTPUT_PATH, imgAugZoo, 'zoo',
AUGMENTATIONS[i].zoom, classe)

    print(' ')
print('finalizado')
```