Identificação de Huanglongbing (HLB) em plantações de citros utilizando redes convolucionais profundas

Miguel N. Marques*, Cristiano O. Pontelli, Ely C. de Paiva***

**Faculdade de Engenharia Mecânica (FEM) - UNICAMP, Campinas, SP 13083-860 Brasil (e-mail: miguelnakajima@ gmail.com, elypaiva@ fem.unicamp.br)*
***Máquinas Agrícolas Jacto, Pompeia, SP 17580-000 e Universidade de Marília (UNIMAR), Avenida Hygino Muzzy Filho, 1001, Marília-SP, Cep 17.525–902, Brasil (e-mail: pontelli@jacto.com.br)*

Abstract: The most common method for the identification of HLB infected plants is visual inspection. The present work aims to present a technique using deep convolutional neural networks to identify the presence of the disease in citrus plants using digital camera images. Two types of images are used: leaves with real background (field) and homogeneous background (studio) from two different classes: healthy and sick**.**

Resumo: O método tradicional mais difundido para a identificação de plantas doentes com HLB é a inspeção visual. 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. São utilizados dois tipos de imagens: folhas com fundo real (de campo) e com fundo homogêneo (estúdio) de duas classes diferentes: saudáveis e doentes.

Keywords: computer vision; DCNN; HLB; deep learning; citrus; inceptionv3.

Palavras-chaves: visão computacional; RNCP; HLB; aprendizagem profunda; citros; inceptionv3.

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, et al., 2020). O principal meio de contaminação é através do inseto psilídeo. (Ribeiro, et al., 2011)

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)

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 & Inamasu, 2014) esse processo manual tem uma taxa de erro entre 30 e 60%.

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 elas são apresentadas estão descritas a seguir.

1.1. Publicações Relacionadas

Em (Lan, et al., 2020), (Ferreira, et al., 2017) e (Ha, et al., 2017) são apresentados métodos de identificação de pragas a partir de imagens aéreas utilizando veículos aéreos não tripulados (VANTs). Ambos usam redes neurais para atingir os resultados apresentados.

Tem-se como exemplares de publicações que utilizam imagens a partir do solo (Weng, et al., 2021), (Zeng, et al., 2020), (Ramcharan, et al., 2017), (Türkoglu, 2019), (Fuentes, et al., 2017), (Dechant, et al., 2017), (Lu, et al., 2017), (Johannes et al., 2017), (Tiwari, 2017), (Padol, 2016) e (Ramesh, 2019)

Após análise das soluções apresentadas, tomou-se como ponto de partida a rede apresentada em (Ramcharan, et al., 2017) para ser reproduzida nessa publicação e adaptada para imagens de citros infectados com HLB. Essa adaptação foi feita utilizando a mesma arquitetura, trocando as imagens de entrada de folhas de mandioca por folhas de laranjeiras e realizando o ajuste dos parâmetros de quantidade de neurônios e dropout nas camadas finais.

2. METODOLOGIA

Para os conjuntos de dados a serem utilizados no treinamento da rede neural classificadora foram idealizados os conjuntos a seguir:

2.1 Conjunto Citrus

Inicialmente o conjunto de dados a ser utilizado foi definido como o apresentado em (Rauf, et al., 2019), 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 1.

Tabela 1. Divisão das imagens no conjunto Citrus

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

Exemplos das imagens contidas nesse conjunto de dados podem ser vistas na Fig.1:

Fig.1: Exemplos de imagens do conjunto Citrus

Para o treinamento da rede neural objetivado aqui serão utilizadas somente as imagens de folhas desse conjunto de dados.

2.2 Conjunto Jacto

O segundo conjunto de dados utilizado será chamado de Jacto (em referência à empresa que forneceu os meios para a coleta dos dados) e tem imagens de folhas em campo com características informacionais (enquadramento, iluminação, brilho etc) próximas do conjunto de dados Citrus. As imagens foram capturadas usando o equipamento similar ao que estará presente na aplicação final. Assim é possível avaliar se os equipamentos utilizados são capazes de produzir dados com a qualidade necessária para o funcionamento eficaz da rede. Esse conjunto também possui imagens de folhas com fundo de campo (fundo real) para avaliar se a rede é capaz de identificar as folhas doentes mesmo que a imagem contenha elementos como solo, outras folhas, galhos etc. A divisão das imagens desse conjunto de dados é mostrada na Tabela 2

Tabela 2. Divisão das imagens do conjunto de dados Jacto

Tipo de imagem	Classe	
	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

Exemplos das imagens contidas nesse conjunto de dados posem ser vistas na Fig.2:

Fig.2: Exemplo de imagens do conjunto Jacto: classe doente em fundo homogêneo (esquerda extremo), em fundo de campo

(meio-esquerda) e classe saudável em fundo homogêneo (meio-direita) e fundo de campo (direita estremo)

3. METODOLOGIA

3.1 Arquitetura da rede neural

Para a arquitetura da rede neural foi seguido o indicado em (Ramcharan, et al., 2017) que utiliza a rede InceptionV3 de (Szegedy, et al., 2016) em série com duas camadas convolucionais *fully connected* que atuam como classificador.

A arquitetura de rede neural InceptionV3 foi proposta em (Szegedy, et al., 2016) e ficou conhecida como "a rede neural do Google" 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".

As camadas convolucionais profundas que vêm junto com o modelo InceptionV3 de (Szegedy, et al., 2016) 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.

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, et al., 2012)

A arquitetura da rede utilizada pode ser definida como sendo:

1. O bloco de camadas InceptionV3 (com 48 camadas ocultas que vêm no bloco pré-definido);
2. Uma camada de *MaxPooling2D*;

   Uma camada *Flatten*;
3. Uma camada *Dropout* com índice de 0,3 (30%);
4. Uma camada *Dense* (*fully connected*) de 1024 neurônios e função de ativação *relu*;
5. Uma camada *Dense* (*fully connected*) de dois neurônios com função de ativação *softmax*.

Para o treinamento o otimizador utilizado foi o Adam com taxa de aprendizagem decrescente iniciada 0,001 (utilizando a função *InverseTimeDecay* com taxa de decaimento 0,5). Foram configuradas 60 épocas com interrupção precoce caso não consiga melhorar a performance em até 15 épocas seguidas.

Nessa configuração a rede como um todo tem 40.645.795 parâmetros treináveis e 34.432 parâmetros fixos.

Para a métrica de acurácia será utilizada a função *Categorical Accuracy* da biblioteca Keras do Python que calcula a porcentagem de saídas corretas fornecidas pela rede, normalizando-a para estar entre 0(nenhuma resposta correta) e 1(todas as respostas corretas).

Para a métrica de perdas será utilizada a função *Categorical Cross Entropy* da biblioteca Keras do Python que calcula a diferença entre duas distribuições de probabilidades: as respostas esperadas e as respostas dadas pela rede.

3.2 Treinamento de todos os pesos 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 compõem 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 se 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 a performance 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 na performance 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 performance, 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 utilizado nesse trabalho 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.

3.3 Data Augmentation

Para melhorar os resultados obtidos utilizando os conjuntos de dados apresentados, será aplicada a técnica de *data augmentation* nas imagens e o resultado será comparado com

o obtido antes do uso da técnica.

Essa técnica consiste em submeter as imagens a uma série de transformações para aumentar a quantidade e a variabilidade das imagens de um dado conjunto de dados. São utilizadas as seguintes transformações:

1. Translação;

2. Rotação;

3. Variação de *zoom*;

4. Variação do brilho da imagem;

O resultado do uso de aplicação dessas transformações em uma das figuras do conjunto de dados é apresentado na Fig.3:

Fig.3: Exemplo de aplicação de *data augmentation*. Imagem original(em cima), variação de brilho (esquerda), translação (centro esquerda), rotação (centro direita), variação de *zoom* (direita)

4. RESULTADOS

Para os resultados apresentados a seguir foram utilizadas as seguintes bibliotecas do Python: Tensorflow, Keras, numpy, sklearn.

4.1 Treinamento com conjunto de dados Citrus

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.

Primeiro são apresentados na Tabela 3 os resultados treinando somente as camadas finais (para as camadas convolucionais foram adotados os pesos do desafio ImageNet). O histórico de evolução da métrica *accuracy* para esse treinamento é mostrado nas Fig.4 e Fig.6, já a evolução da métrica de *loss* para o treinamento com o conjunto Citrus *Augmented* é mostrada na Fig.7. As matrizes de confusão ao final desse treinamento são mostradas nas Fig.5 e Fig.8.

Tabela 3. Resultados do treinamento com conjunto Citrus treinando somente as camadas finais

	Conjunto de dados	
	Citrus	Citrus *Augmented*
Quantidade de imagens	609	9240
Accuracy de treinamento	0.5731	0.5258
Loss de treinamento	0.9741	1.1032
Accuracy de validação	0.6777	0.6107
Loss de validação	0.9419	0.9082

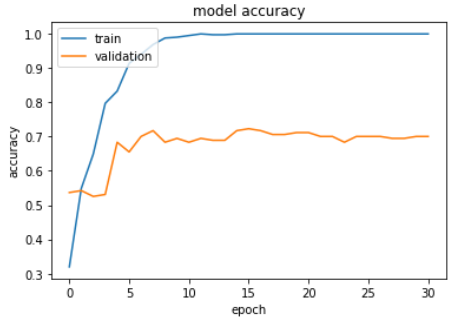

Fig.4: *Accuracy* de treinamento e validação para treinamento com conjunto Citrus somente das camadas finais da rede

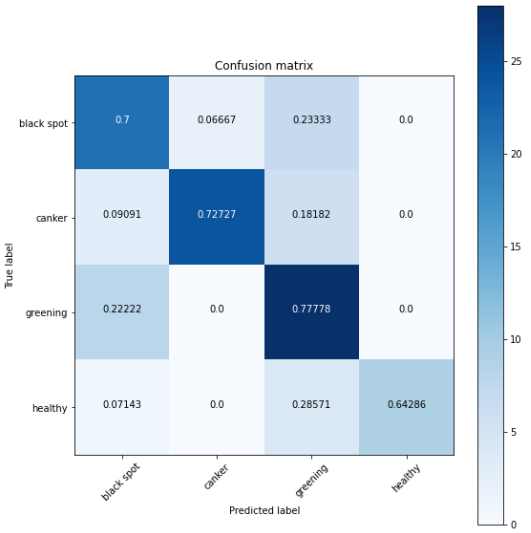

Fig.5: Matriz de confusão do treinamento com o conjunto Citrus treinando somente as camadas finais

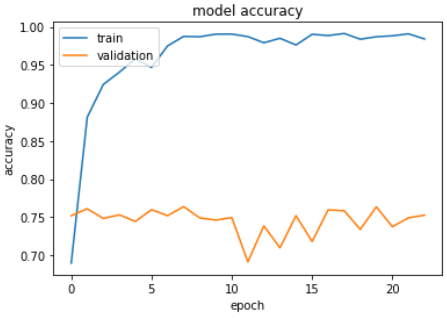

Fig.6: *Accuracy* de treinamento e validação usando o conjunto Citrus *Augmented* treinando somente as camadas finais

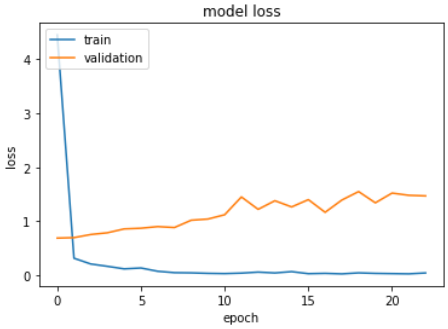

Fig.7: *Loss* de treinamento e validação usando o conjunto Citrus *Augmented* treinando somente as camadas finais

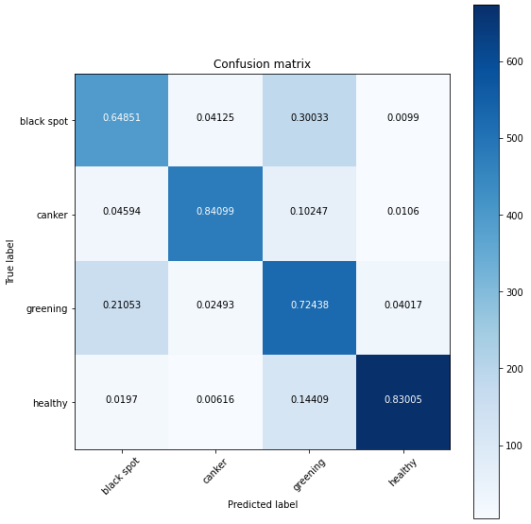

Fig.8: Matriz de confusão do treinamento com o conjunto Citrus *Augmented* treinando somente as camadas finais

São agora apresentados na Tabela 4 os resultados treinando todas as camadas, utilizando os conjuntos Citrus e Citrus *Augmented*. Os históricos de evolução da métrica de *accuracy* são mostrados nas Fig.9 e Fig.11. As matrizes de confusão obtidas ao final do treinamento para cada um dos conjuntos são mostradas nas Fig.10 e Fig.12.

Tabela 4. *Accuracy* e *loss* dos conjuntos Citrus e Citrus *Augmented* treinando todas as camadas

	Conjunto de dados	
	Citrus	Citrus *Augmented*
Quantidade de imagens	609	9240
Accuracy de treinamento	0.9811	0.9994

Loss de treinamento	0.0732	0.0020
Accuracy de validação	0.9580	0.9933
Loss de validação	0.1753	0.0516

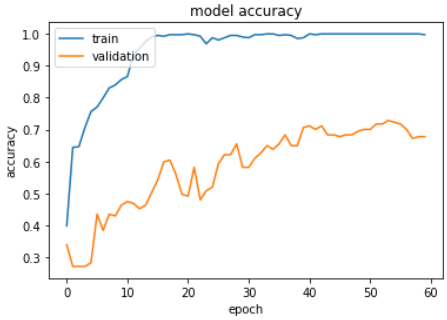

Fig.9: *Accuracy* de treinamento usando o conjunto Citrus e treinando todas as camadas

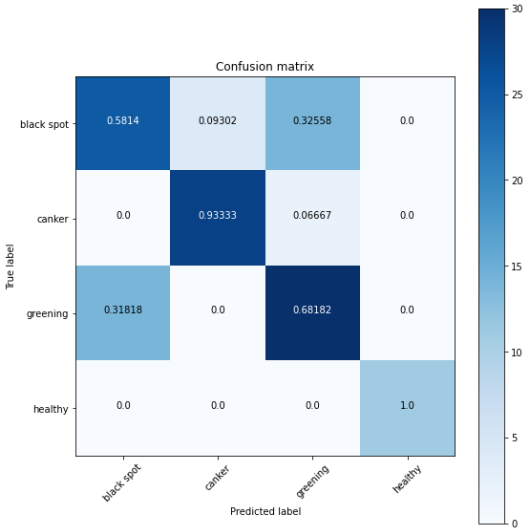

Fig.10: Matriz de confusão do treinamento usando o conjunto Citrus *Augmented* e treinando todas as camadas

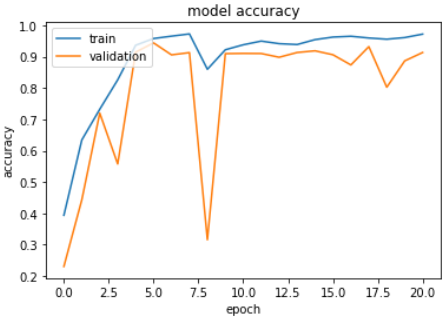

Fig.11: *Accuracy* de treinamento e validação usando o conjunto Citrus *Augmented* e treinando todas as camadas

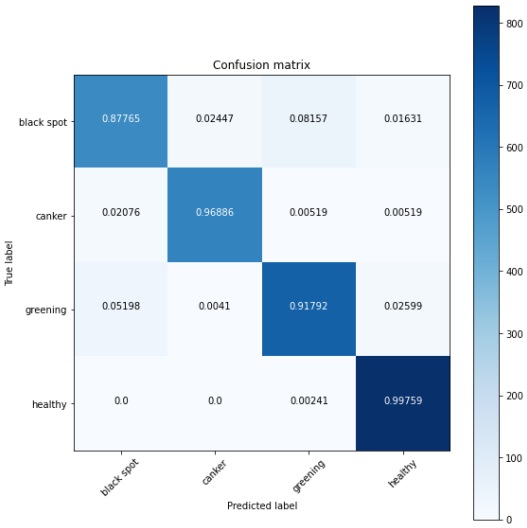

Fig.12: Matriz de confusão após o treinamento com o conjunto Citrus *Augmented* e treinando todas as camadas

Nota-se que a acurácia da rede passou de 67% (Tabela 3) treinando somente as camadas finais para 95% (Tabela 4) treinando todas as camadas, como pode ser visto nas Fig.4, Fig.6, Fig.9 e Fig.11.

5.2 Treinamento usando o conjunto Jacto

Tabela 5. Resultados usando os conjuntos Jacto e Jacto *Augmented* e treinando a rede toda

	Jacto Dataset	Jacto *Augmented* Dataset
Quantidade de Imagens	751	1604
Accuracy de treinamento	1.0000	1.0000
Accuracy validação	0.9598	0.9949

Loss de treinamento	0.0023	0.0027
Loss de validação	0.1577	0.0132
Accuracy de teste	0.9598	0.95990
Loss teste	0.6907	0.2525

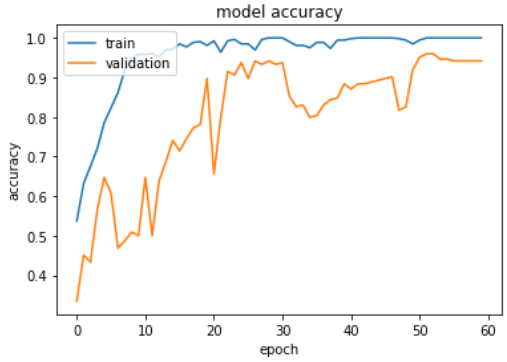

Fig.13: *Accuracy* de treinamento e validação usando o conjunto Jacto

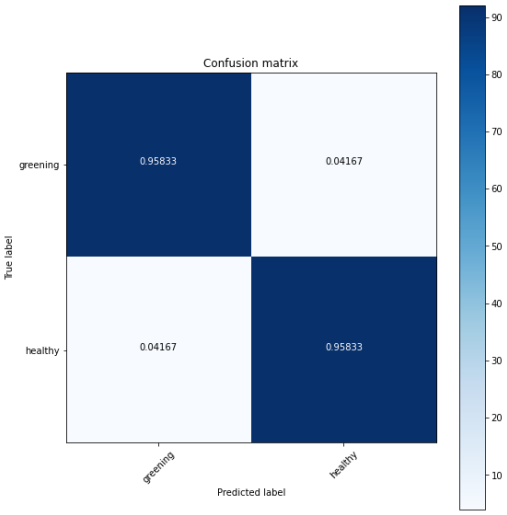

Fig.14: Matriz de confusão após o treinamento usando o conjunto Jacto

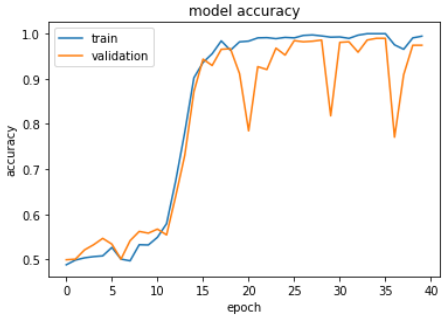

Fig.15: *Accuracy* de treinamento e validação usando o conjunto Jacto *Augmented*

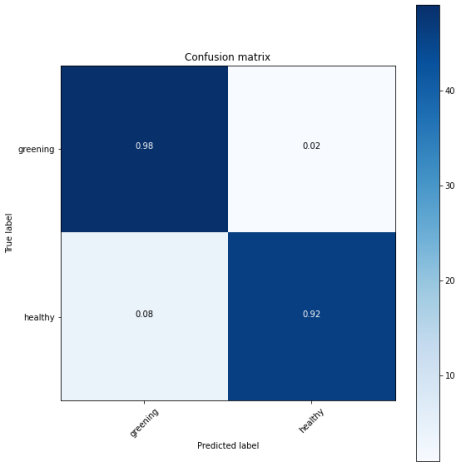

Fig.16: Matriz de confusão após treinamento usando o conjunto Jacto *Augmented*

5. CONCLUSÃO

De acordo com os resultados mostrados em 4.1, a arquitetura proposta em (Ramcharan, 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.

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 (ao comparar os resultados de Tabela 4 e Tabela 5). Ainda em 4.2 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) ao comparar Fig.10 e Fig.12 com Fig.14e Fig.16 nas colunas e linhas aplicáveis. 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.

AGRADECIMENTOS

Este trabalho recebeu o 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.

Agradecimentos à Máquinas Agrícolas Jacto por dar suporte técnico e material à pesquisa e por intermediar a coleta de imagens em campo com seus parceiros, possibilitando assim a montagem dos conjuntos de dados.

REFERÊNCIAS

DeChant, C. et al., 2017. Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning. *Phytopathology*, 06.

Ferreira, A. d. S. et al., 2017. Weed detection in soybean crops using ConvNets. *Computers and Electronics in Agriculture*, 10.

Fuentes, A., Yoon, S., Kim, S. C. & Park, D. S., 2017. A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition. *Sensors*, 09.

Fundecitrus, 2015. Os dez mandamentos do HLB. *Citricultor*, 02.

Fundecitrus, F. d. d. d. c., 2015. Os dez mandamentos do HLB. *Citricultor*, 02, p. 14.

Ha, J. G. et al., 2017. Deep convolutional neural network for classifying Fusarium wilt of radish from unmanned aerial vehicles. *Journal of Applied Remote Sensing*, 12.

Johannes, A. et al., 2017. Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. *Computers and Electronics in Agriculture*, 04.

Jorge, L. A. d. C. & Inamasu, R. Y., 2014. Detecção de Greening dos citrus por imagens multiespectrais. *AGRICULTURA DE PRECISÃO: RESULTADOS DE UM NOVO OLHAR.*

Krizhevsky, A., Sutskever, I. & Hinton, G. E., 2012. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems*, 12, Volume 1, pp. 1097-1105.

Lan, Y. et al., 2020. Comparison of machine learning methods for citrus greening detection on UAV multispectral images. *Computers and Electronics in Agriculture*, 02.p. 11.

Li, S. et al., 2020. Citrus Greening: Management Strategies and Their Economic Impact. *HORTSCIENCE*, 03.Volume 55.

Li, S. et al., 2020. Citrus Greening: Management Strategies and Their Economic Impact. *HortScience*.

Lu, J. et al., 2017. An in-field automatic wheat disease diagnosis system. *Computers and Electronics in Agriculture*, 09.

Padol, P. B. & Yadav, A. A., 2016. SVM Classifier Based Grape Leaf Disease Detection. *2016 Conference on Advances in Signal Processing (CASP)*, 06.

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

Ramesh, S. & Vydeki, D., 2019. Recognition and classification of paddy leaf diseases using Optimized Deep Neural network with Jaya algorithm. *INFORMATION PROCESSING IN AGRICULTURE*, 09.

Rauf, H. T. et al., 2019. A Citrus Fruits and Leaves Dataset for Detection and Classification of Citrus Diseases through Machine Learning. *Mendeley Data.*

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection.

Ribeiro, P. P. E., Jorge, L. A. d. C. & de Paiva, M. S. V., 2011. Diferenciação da Greening do citros de outras doenças foliares a partir de descritores de cor e forma. *VII Workshop de Visão Computacional – WVC 2011*, 05.

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

Tiwari, V. M. & Gupta, T., 2017. Plant leaf disease analysis using image processing technique with modified SVM-CS classifier. *IJEMT.*

TÜRKOĞLU, M. & HANBAY, D., 2019. Plant disease and pest detection using deep learning-based features. *Turkish Journal of Electrical Engineering & Computer Sciences*, 02.

Weng, H. et al., 2021. Citrus Huanglongbing detection based on polyphasic chlorophyll a fluorescence coupled with machine learning and model transfer in two citrus cultivars. *Computers and Electronics in Agriculture.*

Zeng, Q. et al., 2020. Gans-based data augmentation for citrus disease severity detection using deep learning. *IEEE Access*, pp. 172882--172891.