Evaluation of deep learning architectures applied to identification of diseases in grape leaves

Flávio R. S. Oliveira[1], Felipe C. Farias[1], Bernardo João de Barros Caldas[2]

[1]Instituto Federal de Pernambuco – Campus Paulista
Paulista – PE – Brazil

[2]Business Consulting
Recife – PE - Brazil

{flavio,felipe}@paulista.ifpe.edu.br, bcaldas@businessconsulting-ti.com.br

Abstract. *Vale do São Francisco in Pernambuco is one of the most economically important poles in the state and among its cultivars, it is worth mentioning the grape culture. This sector faces challenges related to the response time between identifying a field infestation and taking corrective actions, in order to minimize losses. This work comprises a comparative analysis between deep learning architectures, applied to identification of diseases in grape cultivars. Results suggest that the use of these technologies is plausible to differentiate healthy grape leaves from leaves presenting one of three different types of diseases, obtaining near 100% accuracy in studied database using an architecture that can be employed in embedded devices.*

Resumo. *O Vale do São Francisco em Pernambuco é um dos pólos de maior relevância econômica no estado e dentre seus cultivares, destaca-se a vitivinicultura. Tal setor enfrenta desafios relacionados ao tempo de resposta entre identificar uma infestação no campo e a tomada das devidas ações de manejo, visando minimizar prejuízos. Este trabalho contém análise comparativa entre arquiteturas de aprendizagem profunda, aplicadas à identificação de doenças em cultivares de uva. Os resultados sugerem que o emprego destas tecnologias é plausível para a diferenciar folhas de uva saudáveis de folhas contendo um de três tipos diferentes de doenças, obtendo acurácia próxima de 100% na base de dados estudada.*

1. Introdução

A vitivinicultura no Vale do São Francisco - Petrolina-PE, iniciada na década de 1960, é hoje um dos pólos de maior relevância na produção de uva de mesa no Brasil. O projeto propiciou inclusão social, retorno econômico e visibilidade internacional, em face da qualidade desses produtos e das melhorias constantes dos manejos culturais.

Segundo dados do Relatório de Gestão da Codevasf em 2016 [Codevasf 2016], a área cultivada nos diversos projetos de culturas temporárias e permanentes foi de 54.000 hectares. A parte dessa área dedicada à produção de uva no Vale do São Francisco é destaque, em virtude da possibilidade de se conseguir ter duas safras por ano, ao

contrário de outras localidades.

Face às condições de clima e solo favoráveis à produção, mais áreas são agregadas ao processo produtivo a cada ano, gerando assim, mais riqueza para a região. Para acompanhar o incremento dessa produção mantendo a qualidade e produtividade, torna-se necessário a implantação de novas tecnologias ao manejo desses cultivares.

Dentre as técnicas de manejo, destaca-se a análise do tecido vegetativo e entre as alternativas disponíveis para tal, a análise foliar [Silva e Faria 1999] é uma técnica que permite identificar os micronutrientes e possíveis doenças fúngicas e bacterianas nas folhas dos cultivares. A realização dessa análise em laboratórios pode demandar até 30 dias para a obtenção dos resultados, motivo pelo qual, é plausível que no tempo citado, toda uma área de cultivo possa ser perdida.

Com o propósito de reduzir o tempo entre a identificação de uma infestação e a tomada das devidas ações de manejo, é possível ponderar que essa identificação fosse realizada mediante emprego de tecnologias a serem disponibilizadas aos agricultores ou cooperativas, para uso através de dispositivos de computação móvel.

As tecnologias de aprendizagem profunda [Krizhevsky et al. 2012] têm se mostrado como opções viáveis para solucionar problemas de classificação de imagens em diversos segmentos de aplicação e na Agricultura sua presença também é realidade [Kamilaris e Prenafeta-Boldú 2018].

Este artigo contém análise comparativa de arquiteturas de aprendizagem profunda, aplicadas à classificação de uma base de dados contendo imagens não segmentadas de folhas de uva. Esta base de dados contém padrões de quatro classes: imagens de folhas saudáveis e imagens de folhas contendo um dentre três tipos distintos de doenças.

Considerando que a possível aplicação destas tecnologias seria através de sistemas embarcados, foi dada especial ênfase a arquiteturas que dependem de menor poder computacional para execução, tendo sido avaliadas arquiteturas disponíveis na literatura e algumas geradas a partir de otimização de parâmetros.

O restante deste artigo está organizado conforme a seguir. Na seção 2, pode-se ler o referencial teórico e trabalhos relacionados; na seção 3, são apresentados detalhes acerca da metodologia; na seção 4, são apresentados detalhes da configuração experimental e resultados e por fim; na seção 5, a discussão e conclusão são apresentadas.

2. Background

2.1. Aprendizagem Profunda

A área de Aprendizagem Profunda vem chamando bastante atenção dos pesquisadores nas mais diferentes áreas de aplicação [Schmidhuber 2015], uma dessas áreas que mais se beneficiou ao longo dos últimos anos é a Visão Computacional, na qual o objetivo é fazer a máquina compreender semanticamente as imagens, ou vídeos, apresentadas à mesma. Para isso, são utilizados filtros convolucionais criados a partir da altura, largura e profundidade de imagens. Estes filtros são a base das *Convolutional Neural Networks* (CNNs) [Krizhevsky et al. 2012] e terão pesos aleatórios que serão treinados a partir do

deslizamento dessas janelas sobre a imagem, extraindo então características. A ideia principal é que sejam extraídas características básicas nas primeiras camadas do modelo e, nas camadas seguintes, os filtros vão gerando características com cada vez mais alto nível, facilitando a decisão dos classificadores ao final do modelo. Além dos filtros convolucionais, também é comum fazer parte do modelo as camadas de *pooling*, que reduzem a dimensionalidade da camada seguinte e as camadas densas ou totalmente conectadas, que são as camadas padrão encontradas nas redes rasas *Multi-Layer Perceptron*.

Transfer Learning [Pan et al. 2010] é uma abordagem utilizada para treinar modelos profundos mais rapidamente. A ideia principal é utilizar os pesos das arquiteturas treinadas previamente em um outro domínio de dados para desempenhar tarefas de classificação num domínio possivelmente diferente. Por exemplo, pode-se obter um modelo treinado com os dados da base Imagenet e utilizar esses pesos como pesos iniciais para o problema deste artigo. Uma vez importados estes pesos, pode-se treinar todo o modelo ou apenas parte dele no problema em questão. Esta abordagem também permite que problemas com poucos dados de treinamento se beneficiem, aproveitando filtros que foram treinados para encontrar características mais básicas nas imagens de outros bancos de dados.

A SqueezeNet [Iandola et al. 2016.] é uma arquitetura projetada para usar poucos parâmetros e ainda manter bom desempenho. Como vantagens disso têm-se: (1) menos comunicação entre servidores nos casos de treinamento distribuído; (2) menos banda necessária para transportar o modelo para dispositivos mais modestos; (3) possibilidade de utilizar em hardwares com limitações de memória, como FPGAs [Gschwend 2016.]. Esta rede conseguiu ter um desempenho próximo da AlexNet, mas com 50x menos parâmetros que o modelo original.

A Inception-V3 [Szegedy et al. 2016], é a evolução de uma arquitetura proposta pela Google Brains onde o modelo teve algumas convoluções fatoradas, treinado através do RMSProp e regularização via *Label Smoothing* e *Batch Normalization*, além de usar classificadores auxiliares durante o treinamento para facilitar a passagem do gradiente para camadas iniciais. O módulo de Inception faz o uso de diferentes tamanhos de filtros convolucionais e *pooling*, concatenando o resultado ao final do módulo. Com isso, o modelo tem acesso à avaliação dos filtros mais comumente utilizados (1x1, 3x3 e 5x5), além do *pooling*. Ao usar vários módulos interligados, a ideia é que o modelo escolha a melhor configuração de filtro para aquele determinado problema.

2.2. Trabalhos Relacionados

O uso de aprendizagem profunda em Agricultura possui trabalhos anteriores que mostraram sua viabilidade. O trabalho de Lee et. al [Lee et al. 2015] utilizou CNNs para, mediante aprendizado não-supervisionado, determinar as características mais relevantes de 44 espécies de plantas catalogadas pelo Royal Botanic Gardens. Neste trabalho, o emprego das CNNs produziu classificadores com melhor desempenho quando comparados a classificadores criados a partir de características extraídas manualmente.

No trabalho de Too et al. [Too et al. 2018] foi empregada a técnica *Transfer Learning*

para aprimorar CNNs previamente treinadas na base Imagenet, usando-as para a classificação de 38 classes ligadas a folhas saudáveis ou doentes de 14 tipos de plantas disponíveis na base PlantVillage [Hughes e Salathé 2015]. O resultado do estudo sugeriu ser viável empregar *Transfer Learning* neste domínio, obtendo acurácias superiores a 99% no conjunto de testes.

No Brasil também há trabalhos aplicando CNNs a Agricultura. No trabalho de Sousa et. al [Sousa et al. 2017], foi gerada uma base própria visando diferenciar, mediante o uso de CNNs, duas variedades de guaranazeiro. As CNNs foram utilizadas diretamente para modelar a base de dados estudada e foram obtidos resultados superiores a 95% de precisão na avaliação do conjunto de testes.

No trabalho de Lins e Rieder [Lins e Rieder 2017], foram utilizadas técnicas de aprendizagem profunda para contar automaticamente a quantidade de pulgões em imagens de amostras de trigo, criadas pela equipe que desenvolveu o projeto. Os resultados obtiveram acurácia acima de 98% e se mostraram mais rápidos do que o método tradicional de contagem manual.

Este trabalho se diferencia dos demais: (i) pelo foco exclusivo na classificação de doenças da uva; (ii) pela não realização de segmentação ou *data-augmentation* das imagens, que quando utilizados, tendem a propiciar classificadores com maior poder de generalização e (iii) pelo estudo exclusivo de arquiteturas que requerem menor poder computacional para serem executadas, visando seu possível uso futuro em tecnologias embarcadas.

3. Metodologia

3.1. Caracterização da base de dados

A base de dados utilizada neste artigo foi extraída da base PlantVillage [Hughes e Salathé 2015] que contém originalmente mais de 50 mil imagens de 14 espécies vegetais diferentes. Foi feita a opção de utilizar imagens não segmentadas e sem tratamento adicional, de forma a propiciar a criação de modelos mais robustos e assim mais adequados para a utilização em campo. Na Tabela 1, podem ser vistos os tipos de imagem, o rótulo de classe e a quantidade de imagens disponível em cada classe.

Tabela 1. Caracterização da base de dados

Classe	Tipo de Imagem	Quantidade de imagens
c0	Folha saudável	423
c1	Mancha das folhas (*Isariopsis Leaf Spot*)	1076
c2	Esca (*Black Measles*)	1383
c3	Podridão Negra (*Black Rot*)	1180

Na Figura 1, podem ser vistas imagens que exemplificam as quatro classes trabalhadas.

Pode-se observar que a diferença entre as classes c1, c2 e c3 não parecem tão características quando analisadas a olho nu. Espera-se que as CNNs neste caso sejam capazes de extrair as características mais representativas de cada classe, de forma que possam ser classificadas corretamente.

(c0) **(c1)** **(c2)** **(c3)**

Figura 1. Exemplos de imagens nas classes c0, c1, c2 e c3

3.2. Estudo de arquiteturas definidas empiricamente

Visando estabelecer uma linha de base para comparação com arquiteturas disponíveis na literatura (seção 3.3), algumas CNNs foram definidas empiricamente. Neste estudo, foram consideradas:

- Quantidades diferentes de camadas convolucionais
- Uso ou não de *batch normalization*
- Quantidade de camadas *Max-Pooling*

A arquitetura geral das camadas totalmente conectadas contemplou apenas uma camada de ligação, unindo a saída da última camada convolucional com a camada totalmente conectada de saída, conforme pode ser visto na Figura 2. Detalhes adicionais e a configuração experimental específica deste estudo podem ser vistos na seção 4.2.

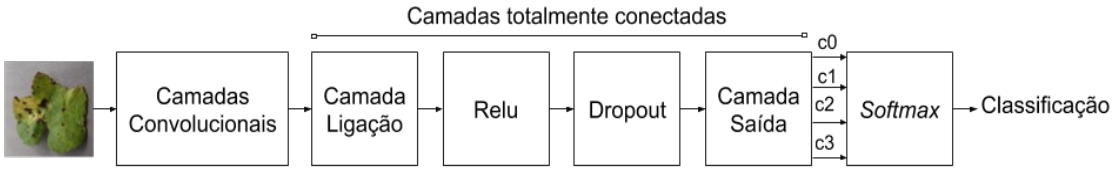

Figura 2. Esquema simplificado das arquiteturas definidas empiricamente

3.3. Estudo de *transfer-learning* utilizando arquiteturas disponíveis na literatura

Considerando que a potencial aplicação destes resultados de pesquisa seria embarcar estas tecnologias para utilização em campo, foram estudadas as arquiteturas Inception v3 [Szegedy et al. 2016] e Squeezenet [Iandola et al. 2016.], ambas disponíveis na API Python Pytorch [Paszke et al. 2017]. Ambas arquiteturas foram pré-treinadas utilizando a base Imagenet e em seguida foi aplicada a técnica *transfer learning,* na qual aspectos

da arquitetura, como por exemplo quantidade de neurônios de saída, são adequados à nova base de dados e em seguida, ocorrem novas épocas de treinamento visando aprimorar o modelo para uso na base objeto deste estudo. Um esquema desses passos pode ser visto na Figura 3.

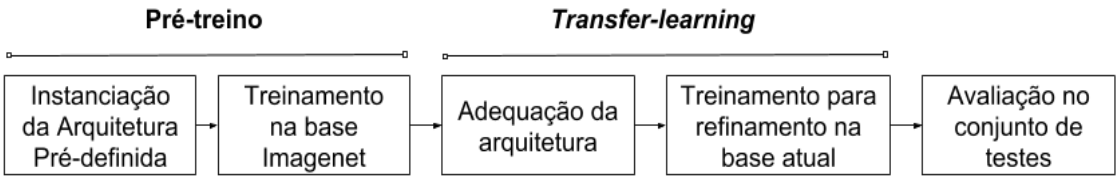

Figura 3. Fluxograma dos passos realizados para *transfer-learning*

Espera-se com este experimento, verificar a viabilidade de utilizar tais arquiteturas mais complexas do que as definidas manualmente na seção 3.2, verificando inclusive se é razoável utilizar este volume de imagens, não necessariamente considerado adequado para emprego em tarefas de aprendizagem profunda e também comparando seus resultados com os obtidos nas arquiteturas definidas anteriormente.

4. Experimentos e Resultados

Para os experimentos apresentados nesta seção, foi utilizada uma instância Amazon AWS *p2xlarge*, cujas configurações são: 4 vCPUs, 61 GB de memória RAM e uma GPU Tesla K80, contendo 4096 *cuda cores* e 11 GB de memória RAM. Todo o código executável foi implementado na linguagem Python, utilizando a API Pytorch [Paszke et al. 2017]. Para validação dos experimentos e aferição de métricas de performance, foi utilizado o método *hold-out* estratificado, de forma que a base de dados foi separada em dois conjuntos disjuntos, com aproximadamente 80% dos padrões para o conjunto de treinamento e 20% para o conjunto de testes. Na tabela 2, pode-se observar a distribuição de imagens por classe em cada conjunto.

Tabela 2. Distribuição das imagens entre os conjuntos de treinamento e testes, por classe

Classe	Imagens Usadas no Conjunto de Treinamento	Imagens Usadas no Conjunto de Testes
c0	339	84
c1	861	215
c2	1107	276
c3	944	236

4.1. Experimentos com arquiteturas definidas empiricamente

Considerando que nos artigos de referência da Squeezenet foram utilizadas no máximo 13 blocos de processamento e na arquitetura Inception foram utilizadas no máximo 11 blocos, nesta seção foram investigadas arquiteturas com no máximo 9 blocos de processamento. Para fins de esclarecimento, as camadas de *batch-normalization* não foram contabilizadas nos blocos de processamento nem nas arquiteturas Squeezenet, Inception e nem nas estudadas nesta seção. A estrutura das arquiteturas tipo 1, 2, 3 e 4 podem ser vistas na Tabela 3.

Tabela 3. Estrutura das arquiteturas utilizadas. "M" representa uma camada *max-pooling*, "B" representa uma camada de *batch normalization*, os demais números representam a quantidade de filtros convolucionais em cada camada.

Arquitetura Testada	Sem *Batch-normalization*	Com Batch-normalization
Tipo 1	64,M	64,B,M
Tipo 2	64,32,M	64,B,32,B,M
Tipo 3	128,64,M,64,32,M	128,B,64,B,M,64,B,32,B,M
Tipo 4	128,128,M,64,64,M,32,32,M	128,B,128,B,M,64,B,64,B,M,32,B,32,B,M

Simulações prévias foram realizadas levando em consideração as configurações mais utilizadas nos artigos de referência, de forma que neste estudo, foram utilizadas as melhores configurações, a saber:

- Quantidade épocas de treinamento: 10
- Taxa de aprendizagem inicial: 0,001
- Decaimento: taxa de aprendizagem reduzida para 1/10 a cada 5 épocas
- Algoritmo de treinamento: Gradiente Descendente Estocástico
- Momentum: 0,9
- *Batch-size*: 16
- Camadas *Max-pooling*: filtros 2x2 com deslocamento 2
- Camadas convolucionais: filtros 3x3 com deslocamento 1

Conforme pode ser visto na Figura 2, a primeira camada totalmente conectada tem a mesma quantidade de entradas que a quantidade de saídas do último filtro convolucional e 100 neurônios com função de ativação Relu. As saídas desta camada são submetidas a uma camada com *dropout* que em seguida são usadas como entradas para a última camada totalmente conectada que possui 100 neurônios e 4 saídas, submetidas a uma camada *softmax* que define a classificação. Os resultados das simulações podem ser vistos na Tabela 4.

Pode-se observar que as acurácias de treinamento e testes para as arquiteturas Tipo 1 e Tipo 2 foram próximas, com ou sem o uso de *batch-normalization*. No caso das arquitetura Tipo 3 houve uma melhora em torno de 5% com o uso de *batch-normalization,* enquanto que na Tipo 4, a melhora foi da ordem de 17%, sugerindo que o uso de *batch-normalization* auxiliou no aprendizado.

Tabela 4. Resultados das simulações para os quatro tipos de arquiteturas estudadas

Arquitetura	**Sem uso de *Batch Normalization***		**Com uso de *Batch Normalization***	
	Acurácia Treinamento (%)	**Acurácia Teste (%)**	**Acurácia Treinamento (%)**	**Acurácia Teste (%)**
Tipo 1	93,33	92,23	90,37	91,86
Tipo 2	96,95	93,46	95,60	94,57
Tipo 3	95,13	91,73	98,64	96,17
Tipo 4	79,26	81,62	99,20	98,27

Por outro lado, analisando os tempos de treinamento, que podem ser vistos na Tabela 5, nota-se que o uso de *batch normalization* tornou as arquiteturas mais lentas para treinar do que sem seu uso. Embora o tempo de treinamento só possa ser considerado como uma medida indireta do tempo necessário para processar uma imagem, é possível inferir que nos casos em que se desejasse uma arquitetura que demandasse menor poder computacional para execução, seria viável utilizar a arquitetura Tipo 2, empregando apenas 2 camadas convolucionais e uma de *max-pooling*. Sua acurácia estaria apenas 4% abaixo da arquitetura tipo 4 que emprega 9 blocos de processamento.

Tabela 5. Tempos de treinamento para as arquiteturas Tipo 1 a 4 com e sem *batch normalization*

Arquiteturas	**Tempo de treinamento (min)**	
	Sem *Batch Normalization*	**Com *Batch Normalization***
Tipo 1	1m41s	1m53s
Tipo 2	5m 57s	6m 40s
Tipo 3	16m 39s	18m 15s
Tipo 4	26m 35s	28m 51s

Os modelos resultantes Tipo 1, 2, 3 e 4 tiveram respectivamente tamanhos 11MB,

39MB**,** 154MB e 307MB, confirmando a hipótese que a arquitetura Tipo 2 poderia ser adequada para execução em dispositivos com menor poder computacional, apresentando equilíbrio entre o tamanho da arquitetura, seu tempo de treinamento e sua acurácia. Na Figura 4 podem ser vistas as matrizes de confusão para as arquiteturas Tipo 2 e Tipo 4. É possível observar que as duas arquiteturas foram capazes de acertar todos os padrões de teste contendo folhas saudáveis (c0), apresentaram erros relativamente baixos na classificação Mancha das folhas (c1), mas apresentaram graus de erro mais elevados na classificação da Esca (c2) e da Podridão Negra (c3).

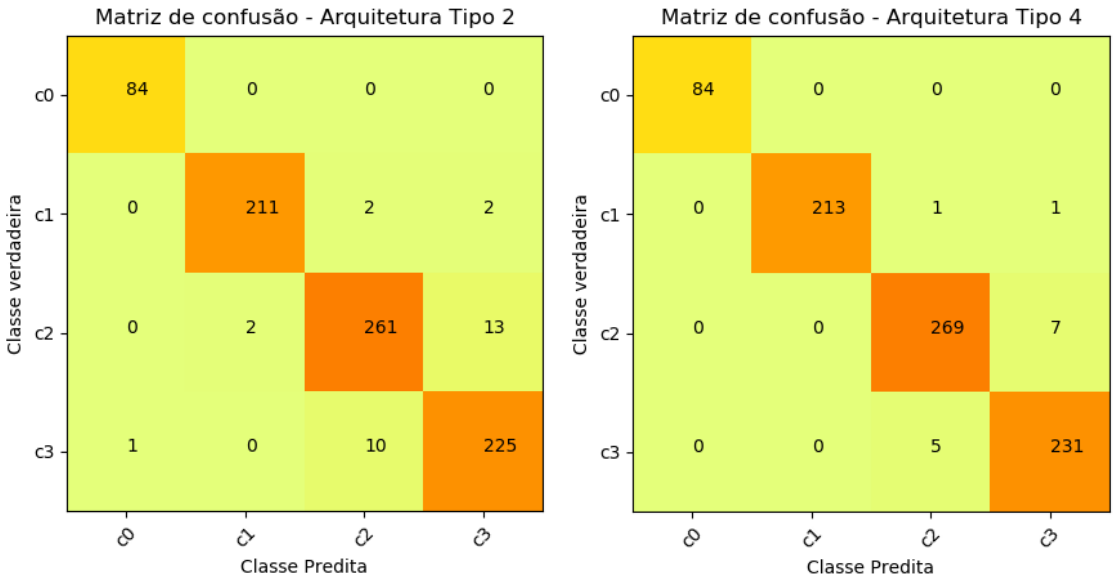

Figura 4. Matrizes de confusão arquitetura tipo 2 (esquerda) e tipo 4 (direita)

4.2. Experimentos com *transfer-learning* utilizando arquiteturas disponíveis na literatura

Conforme detalhes explicados na seção 3.3, após importar as arquiteturas Inception e Squeezenet pré-treinadas com imagens contidas na base Imagenet, apenas a camada de neurônios final foi ajustada para que tivesse 4 saídas, adequando-as ao problema em questão. Em seguida, foi utilizada a mesma configuração experimental da seção 4.1 com a qual as arquiteturas foram refinadas por 10 épocas. Os resultados de acurácia e tempos de treinamento podem ser vistos na Tabela 6.

Tabela 6. Acurácias de treinamento, teste e tempo de treinamento para as arquiteturas Inception e SqueezeNet

Arquitetura	Acurácia Treinamento (%)	Acurácia Teste (%)	Tempo de Treinamento (min)
SqueezeNet	100,00	99,75	2m 26s

Inception v3	99,85	100,00	24m 37s

Pode-se observar que as arquiteturas apresentaram resultados muito próximos em termos de acurácia mas a SqueezeNet concluiu as 10 épocas de treinamento em apenas 2m26s enquanto a Inception necessitou de aproximadamente 12x mais tempo. As arquiteturas resultantes possuem 2,8MB e 97MB para Squeezenet e Inception respectivamente. Entre estas, considerando a necessidade de executar em dispositivos com baixo poder computacional, a acurácia, tempo de treinamento e tamanho da arquitetura resultante, seria indicado utilizar a Squeezenet.
Na Figura 5, podem ser vistas as matrizes de confusão para as duas arquiteturas. Com acurácias muito altas, apenas houve erro da Squeezenet ao classificar 2 padrões Esca (c2) como Podridão Negra (c3).

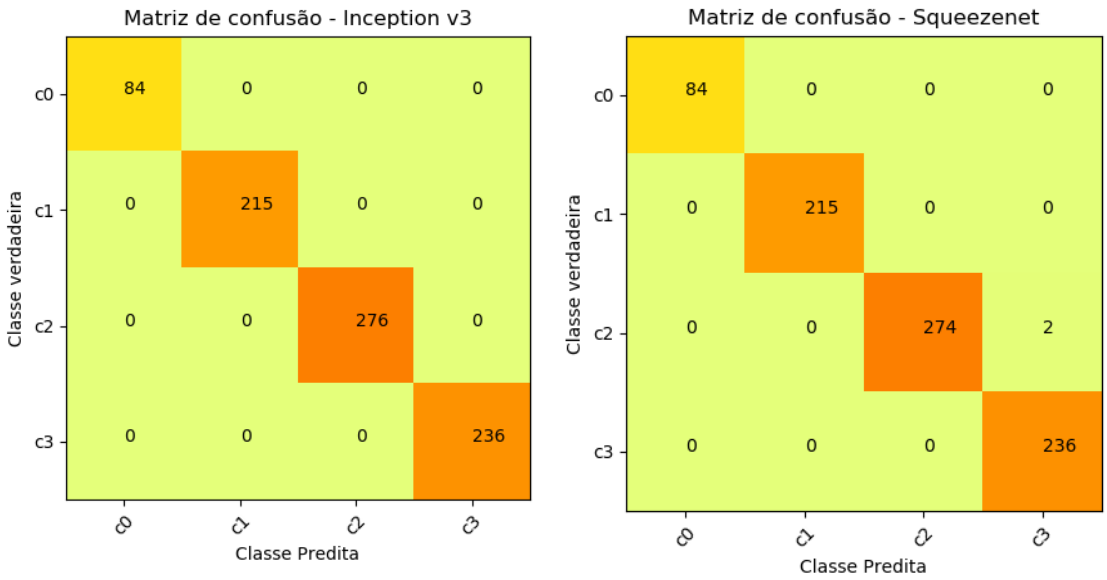

Figura 5. Matriz de confusão da Inception v3 (esquerda) e da Squeezenet (direita).

5. Conclusão

Neste trabalho foram comparadas arquiteturas de aprendizagem profunda, aplicadas à identificação de doenças a partir de imagens de folhas de uva. Considerando que a potencial aplicação para os resultados desta pesquisa seria sua utilização enquanto sistema embarcado, foram enfocadas arquiteturas: (i) desenvolvidas a partir de otimização de parâmetros, utilizando quantidade máxima de 9 camadas e (ii) disponíveis na literatura e voltadas para performance - Inception v3 e SqueezeNet.
No primeiro conjunto de experimentos, cujos resultados podem ser vistos na seção 4.1, os valores de acurácia no conjunto de testes foi sempre superior a 90%, em apenas 10 épocas de treinamento realizado sem que as arquiteturas possuíssem treinamento anterior. Isto sugere que as características necessárias para separar as 4 classes

trabalhadas foram aprendidas de forma satisfatória.
O uso de *batch normalization* aprimorou os resultados em quase todas as arquiteturas, principalmente na Tipo 4 com a maior quantidade de camadas, chegando a aproximadamente 98% de acurácia. Por outro lado, o uso de *batch normalization* aumentou o tempo de treinamento embora este aumento não tenha sido tão significativo. As redes mais complexas, com maior quantidade de camadas, apresentaram melhor resultado em termos de acurácia mas o tempo de treinamento foi progressivamente maior, chegando a aproximadamente 29 min na arquitetura Tipo 4.
No segundo conjunto de experimentos, cujos resultados podem ser vistos na seção 4.2, as duas arquiteturas testadas obtiveram 99,75 e 100% de acurácia no conjunto de testes. É válido observar que embora estes resultados possam ser considerados equivalentes, o tempo de treinamento e o tamanho do modelo apresentaram diferenças significativas. A Inception v3 tem um tamanho maior, por volta de 97MB e levou cerca de 24 min para ser treinada nas 10 épocas permitidas, enquanto a SqueezeNet produziu um modelo inferior a 3MB e foi treinada em apenas 2m26s.

Considerando que os resultados foram promissores, os trabalhos futuros apontam na direção de: (i) utilizar a arquitetura SqueezeNet para o desenvolvimento de um sistema embarcado para análise foliar da uva, já que foi a arquitetura que apresentou melhores balanceamento em termos de tempo de treinamento, tamanho do modelo resultante e acurácia no conjunto de testes e; (ii) ampliar a base de dados com imagens coletadas nas fazendas do Vale do São Francisco e verificar a possibilidade de estender a contribuição deste trabalho para outros tipos de doenças existentes no local.

References

Codevasf - Ministério da Integração Nacional (2016) “Relatório de Gestão do Exercício de 2016”, http://www2.codevasf.gov.br/empresa/auditoria-interna/processos-de-contas-anuais/ relatorio_de_gestao_2016.pdf . Acessado em 20 de Junho de 2018.

Gschwend, D. (2016) “Zynqnet: An fpga-accelerated embedded convolutional neural network”. Master’s thesis, Swiss Federal Institute of Technology Zurich (ETH-Zurich), 2016.

Hughes, D. e Salathé, M. (2015) “An open access repository of images on plant health to enable the development of mobile disease diagnostics”, arXiv:1511.08060 .

Iandola, F. N. et al. (2016) “Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size”, arXiv preprint arXiv:1602.07360, 2016.

Kamilaris, A. e Prenafeta-Boldú, F. (2018) "Deep learning in agriculture: A survey", In: Computers and Electronics in Agriculture 147, p. 70–90

Krizhevsky A., Sutskever, I. e Hinton, G. (2012) “Imagenet classification with deep convolutional neural networks”, In: Advances in neural information processing systems, p. 1097-1105

Lee, S., Chan, C., Wilkin, P., Remagnino, P. (2015) “Deep-plant: Plant identification

with convolutional neural networks", In: Proceedings of 2015 IEEE International Conference on Image Processing (ICIP)

Lins, E. A. e Rieder, R., (2017) "Uma metodologia de contagem e classificação de afídeos utilizando visão computacional", In: Proceedings of 2017 SIBGRAPI - Conference on Graphics, Patterns and Images.

Pan, S. J. et al. (2010) "A survey on transfer learning", IEEE Transactions on knowledge and data engineering, v. 22, n. 10, p. 1345-1359, 2010.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. e Lerer, A. (2017) "Automatic differentiation in PyTorch", In: NIPS 2017 Workshop.

Schmidhuber, J. (2015) "Deep learning in neural networks: An overview.", Neural networks, v. 61, p. 85-117, 2015.

Silva, D. J. e Faria, C. M. B. (1999) "Amostragem para análise foliar da videira", Embrapa Semi-árido,1999 https://www.embrapa.br/busca-de-publicacoes/-/publicacao/154374/amostragem-para-analise-foliar-de-videira . Acessado em 20 de Junho de 2018.

Sousa, A. L., Salame, M. F. A., Nascimento Filho, F. J. e Atroch, A. L. (2017) "Redes Neurais Convolucionais Aplicadas ao Processo de Classificação de Cultivares de Guaranazeiros". In: Proceedings of XIV Encontro Nacional de Inteligência Artificial e Computacional, p. 855-864.

Szegedy, C. et al. (2016), "Rethinking the inception architecture for computer vision", In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 2818-2826.

Too, E. C.,Yujian, L., Njuki, S. e Yingchun, L. (2018) "A comparative study of fine-tuning deep learning models for plant disease identification", In: Computers and Electronics in Agriculture.