PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
DEPARTAMENTO DE COMPUTAÇÃO

DETECÇÃO DE FERRUGEM DE FOLHAS VEGETAIS POR MEIO DE IMAGENS APLICANDO REDES NEURAIS CONVOLUCIONAIS

WESLEY DA CUNHA JÚNIOR

GOIÂNIA
2020

WESLEY DA CUNHA JÚNIOR

DETECÇÃO DE FERRUGEM DE FOLHAS VEGETAIS POR MEIO DE IMAGENS APLICANDO REDES NEURAIS CONVOLUCIONAIS

Trabalho para conclusão de curso apresentado na faculdade de Ciências Exatas e da Computação da Pontifícia Universidade Católica de Goiás como requisito básico para a conclusão do curso de Engenharia da Computação.

Orientador: Prof. Dr. Clarimar José Coelho

GOIÂNIA
2020

WESLEY DA CUNHA JÚNIOR

DETECÇÃO DE FERRUGEM DE FOLHAS VEGETAIS POR MEIO DE IMAGENS APLICANDO REDES NEURAIS CONVOLUCIONAIS

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Engenharia da Computação, e aprovado em sua forma final pela escola de Ciências Exatas e da Computação, da Pontifícia Universidade Católica de Goiás em _____/_____/_____.

_____

Prof. Dr. Clarimar José Coelho

Banca examinadora:

_____

Prof. Dr. Arlindo Rodrigues Galvão Filho

_____

Prof. Msc. Isaac Yves Lopes de Macêdo

_____

Prof. Dr. Clarimar José Coelho

GOIÂNIA
2020

Dedico este trabalho a minha família, ao meu pai Wesley da Cunha e minha mãe Ednúzia Oliveira de Aguiar Cunha, meus maiores exemplos de garra e perseverança, e dedico também a minha querida esposa Sandylla Patrícia Cunha e Lima.

AGRADECIMENTOS

A Deus que sempre me iluminou na minha caminhada.

Ao meu orientador acadêmico, professor Dr. ClarimarJosé Coelho, pelo apoio e confiança no desenvolvimento deste trabalho.

A minha família e aos meus amigos, por terem me proporcionado os melhores momentos da minha vida. E, com toda certeza, ainda melhores estão por vir.

Sou grato a todo o corpo docente, à direção e administração desta universidade.

A todos que direta ou indiretamente colaboraram para materialização deste trabalho.

“O insucesso é apenas uma oportunidade para recomeçar com mais inteligência.”

Henry Ford

Resumo

A ferrugem da folha é uma patologia fúngica que se define pela apresentação de pústulas com esporos de cor amarelo escuro a marrom na face das folhas, a partir do incidente até o estágio de maturação, causando perdas na produção de grãos que podem superar 50%, reduzindo assim a área fotossintética e aumentando então a respiração. A classificação de folhas vegetais e a identificação de patologia fúngica como a ferrugem é um processo que até então é feito manualmente por taxonomistas e isso acaba empregando muito tempo para a sua realização, é necessário então que o produtor acompanhe o desenvolvimento da infecção para determinar o nível da infecção e aplicar a medicação correta e mesmo assim pode trazer uma taxa de sucesso baixa. Este trabalho propõe um algoritmo para solucionar a classificação de folhas com ferrugem utilizando Redes Neurais Convolucionais (algoritmos de *Deep Learning*) em uma base de 440 imagens de 12 espécies distintas, com o objetivo de classificá-las em portadora ou não de ferrugem. Foi utilizada transferência de aprendizado das arquiteturas de redes neurais convolucionais: AlexNet, GoogleNet, Resnet-18 e VGG-19. Os melhores resultados obtidos foram utilizados a Redes Neurais Convolucionais Resnet-18 e a VGG-19, ambas com treinamento de 20 épocas. A Resnet-18 atingiu uma acurácia de 96,6%, sensibilidade de 97,7% e especificidade de 95,5%, enquanto a VGG-19 atingiu uma acurácia de 96,6%, sensibilidade de 95,5% e especificidade de 97,7%. Deste modo, a utilização de redes neurais convolucionais tem grande potencial para utilização na detecção de patologia fúngica no caso ferrugem.

Palavra-chave: Redes Neurais, Redes Neurais Convolucionais, Classificação de Folhas, *Deep learning*, Transferência de Aprendizado.

Abstract

Leaf rust is a fungal pathology that is defined by the presentation of pustules with dark yellow to brown spores on the face of the leaves, from the incident to the stage of maturation, causing losses in grain production that can exceed 50%, so cool the photosynthetic area and then increase the breath. The classification of plant leaves and the identification of fungal pathology as a rust is a process that until then has been done manually by taxonomists and this ends up taking a lot of time to be carried out, it is then necessary for the producer to monitor the development of the infection to determine the level infection and apply the correct medication and even then it can bring a low success rate. This work proposes an algorithm to solve the classification of leaves with rust using Convolutional Neural Networks (Deep Learning algorithms) based on 440 images of 12 different species, with the objective of classifying them into rust carrier or not. It was transferred according to the convolutional neural network architectures: AlexNet, GoogleNet, Resnet-18 and VGG-19. The best results obtained were used with Resnet-18 Convolutional Neural Networks and VGG-19, both with 20-season training. Resnet-18 achieved an accuracy of 96.6%, sensitivity of 97.7% and specificity of 95.5%, while VGG-19 reached an accuracy of 96.6%, sensitivity of 95.5% and specificity of 97.7%. Thus, the use of convolutional neural networks has great potential for use in the detection of fungal pathology in the case of rust.

Keyword: Neural Networks, Convolutional Neural Networks, Sheet Classification, Deep learning, Transfer of Learning.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

BN = Batch Normalization

CNN = Convolutional Neural Network

DL = Deep Learning

FP = Falso positivo

FN = Falso negativo

IA = Inteligência Artificial

RNA = Rede Neural Artificial

RNAs = Redes Neurais Artificiais

RNC = Redes Neurais Convolucionais

VP = Verdadeiro positivo

VN = Verdadeiro negativo

LISTA DE SÍMBOLOS

∑ - Somatória

Sumário

1. INTRODUÇÃO

A Ferrugem causada pelo fungo *Hemileia vastatrix Berk. et Br*., é a principal doença em plantas em todo o mundo. Esta doença causa a queda precoce das folhas e a consequente seca dos ramos produtivos, antes da época de florescimento, refletindo negativamente sobre o desenvolvimento dos botões florais, agravo da florada, desenvolvimento dos frutos e redução da produtividade do ano agrícola. (SOUZA; ANTONIO FERNANDO,2007).

Com a proliferação da doença as manchas começam a aumentar de tamanho tornando-se alongadas no sentido das nervuras, formando pústulas de coloração pardo-ferruginosa (marrom) rompendo a epiderme e expondo os uredósporos do fungo.

Com baixa intensidade de luz na superfície das folhas e temperaturas elevadas a infecção começa a se proliferar e a doença se desenvolve rapidamente, causando a seca das folhas.

As ferrugens são assim denominadas devido à lesão com massa de esporos pulverulenta de coloração amarela a avermelhada. Os esporos são estruturas de dispersão dos fungos, semelhantes às sementes das plantas. Seu tamanho é diminuto e cada lesão pode conter milhões de esporos sendo que, para haver nova infecção, basta que um único esporo germine em condições ideais de temperatura e umidade. No entanto a viabilidade germinativa dos esporos é restrita e nem todos os produzidos acabam por gerar novas infecções. O principal mecanismo de dispersão dos esporos é o vento, que pode carregá-los por milhares de quilômetros. (GOULART; IVES,2013).

Os danos causados às plantas são irreparáveis partindo do ponto de que os tecidos vegetais afetados não têm capacidade regenerativa.

A identificação e quantificação da severidade de doenças é um dos principais pontos para definir qual a melhor estratégia de controle. A identificação pode ser realiza por um produtor através de uma análise visual simples. Há anos ferramentas têm sido estudadas para essa função, mas algumas com praticabilidade baixa e outras por serem tecnologias hoje pouco utilizadas (CUNHA; PERES, 2010). GODOY et al. (2006).

Deep Learning (DL) é uma subárea da Inteligência Artificial e do Aprendizado de Máquina, que utiliza algoritmos inspirados na estrutura e funcionamento das células do cérebro (neurônios), chamados de Redes Neurais Artificiais (RNAs), com o intuito de ensinar ao software a reconhecer padrões em representações digitais de imagens, sons e outros tipos de dados (JAIN; MAO; MOHIUDDIN, 1996).

Seu primeiro conceito foi introduzido em 1943, mas ganhou popularidade algumas décadas depois com a introdução de algoritmos de treinamento como o *backpropagation*, que permite a realização de um treinamento posterior para aperfeiçoar os resultados do modelo.

Em 1986, David Rumelhart e seus colegas introduziram o algoritmo *backpropagation*, possibilitando o treinamento das redes neurais com diversas camadas através da retropropagação.

Quando uma rede neural artificial é inicializada, os pesos sinápticos recebem valores aleatórios que, quando multiplicados pelos valores recebidos, algumas vezes da camada de entrada e outras de neurônios da camada anterior, não atingem os valores desejados no momento do treinamento. Para corrigir os pesos sinápticos, uma das técnicas populares é a retropropagação, conhecido também como *backpropagation*, que corrige os valores dos pesos pela diferença entre o valor obtido e o esperado (recebido no treinamento) pelo algoritmo, propagando para todas as células nervosas.

Redes Neurais Convolucionais (RNC) são redes que tratam especificamente da análise de imagens. Esses algoritmos têm contribuído em diversas áreas, por apresentarem uma ótima performance e flexibilidade em ambientes não muito bem restritos (JAIN; MAO; MOHIUDDIN, 1996).

O pré-processamento exigido em uma RNC é muito menor em comparação a outros algoritmos de classificação. Enquanto nos métodos primitivos os filtros são feitos à mão, as RNC têm a capacidade de aprender sozinhos esses filtros ou características.

A arquitetura de uma RNC é análoga àquela do padrão de conectividade de neurônios no cérebro humano e foi inspirada na organização do córtex visual.

As arquiteturas abordadas neste trabalho englobam algumas etapas. A primeira etapa envolve a aquisição de imagens, onde as imagens são obtidas do data set do PlantCLEF e através de dispositivos eletrônicos. A segunda etapa é o pré-processamento, em que comumente é realizado ajustes das imagens. A terceira é a segmentação, que engloba o reconhecimento dos objetivos contidos na imagem. E finalmente, a classificação informando a detecção do fungo na planta.

1.1 Justificativa

A detecção precoce da ferrugem na folha aumenta significativamente as chances de cura e evita as complicações e sequelas causadas pelos estágios mais avançados da doença, o que diminui a taxa de mortalidade em decorrência desse tipo de patologia fúngica. A proposta e desenvolvimento de técnicas mais efetivas e precisas, para o diagnóstico da doença, principalmente em seu estágio inicial, é de extrema importância.

Pensando nisso, o trabalho propõe um sistema inteligente de detecção de ferrugem nas folhas, baseado na análise de imagens e algoritmos de DL (RNC), que podem ser uma alternativa ao diagnóstico da doença. Ao final, os resultados indicam se essa abordagem é ou não uma candidata à análise e identificação da doença. Os resultados do trabalho interessam ao conjunto de pessoas que possui lavouras, plantações que são afetadas pela doença (produtores), aos profissionais especialistas em biologia, além de pesquisadores da área de processamento de imagem e DL.

1.2 Objetivo Geral

Com isso o objetivo deste trabalho visa contribuir com as técnicas de taxonomia identificando as folhas por meio de imagens, que apresentam a doença, ou seja, o sistema deverá ser capaz de classificar as folhas em saudáveis (sem fungos) e com ferrugem (fungos). Para isso, RNC são empregados em um banco de dados de imagens de folhas que apresentavam a doença bem como folhas sem a doença.

1.2.1 Objetivos Específicos

- Análise da base de dados disponibilizadas no PlantCLEF e um acervo pessoal;
- Analisar os algoritmos de *Deep Learning* já existentes, comparando-os e

entendendo suas diferenças;

- Construção de um banco de dados de imagens de folhas portadoras e não portadoras de ferrugem (fungos).
- Definir e implementar o algoritmo que será utilizado para a classificação das imagens;
- Utilização de RNC, para a classificação das imagens na base utilizada.
- Utilização de métricas (Ex: *Accuracy*), para análise de desempenho na classificação da arquitetura.
- Foi realizado experimentos em diferentes arquiteturas de CNN consideradas pela literatura e comprovadas como altamente poderosas na tarefa de classificação de imagens.

1.3 Resultados Esperados

Determinar a arquitetura que possui a melhor acurácia, sensibilidade e especificidade para a implementação de uma ferramenta automática baseado em RNC para a detecção da ferrugem em folhas de plantas através de imagens.

1.4 Estrutura do Trabalho

Este trabalho está organizado como segue. No Capítulo 2 será apresentado os materiais e métodos utilizamos como redes neurais convolucionais, transferência de aprendizagem, classificadores, base de dados os métodos e experimentos. O Capítulo 3 descreve os resultados obtidos através das análises dos métodos apresentados no Capítulo 2. Por fim, no Capítulo 4 as considerações finais do trabalho são apresentadas. Estre trabalho também apresenta um Apêndice. O Apêndice A apresenta o código fonte dos métodos desenvolvidos em MATLAB que realiza o processamento das imagens digitais.

2. Materiais e Métodos

2.1 Deep Learning

Deep Learning é a técnica de aprendizado de máquina a partir de Redes Neurais Artificiais. A tecnologia de *Deep Learning* vai ter grandes evoluções nos próximos anos e se destacará muito no seu paradigma de “habilitar o computador a aprender a partir da observação dos dados”.

Uma das vantagens dos algoritmos de *Deep Learning* é sua a capacidade de aprendizagem em grandes quantidades de dados de uma forma não-supervisionada, sendo assim uma ferramenta valiosa para *Big Data Analytics* onde a maioria dos dados são desta natureza, também designados por dados não-estruturados. *Deep Learning* vem sendo largamente utilizado para construção de aplicações de Inteligência Artificial.

A partir do conceito de *Deep Learning* novas propostas foram desenvolvidas, originando as chamadas redes neurais profundas, como Rede Neural Convolucional (RNC).

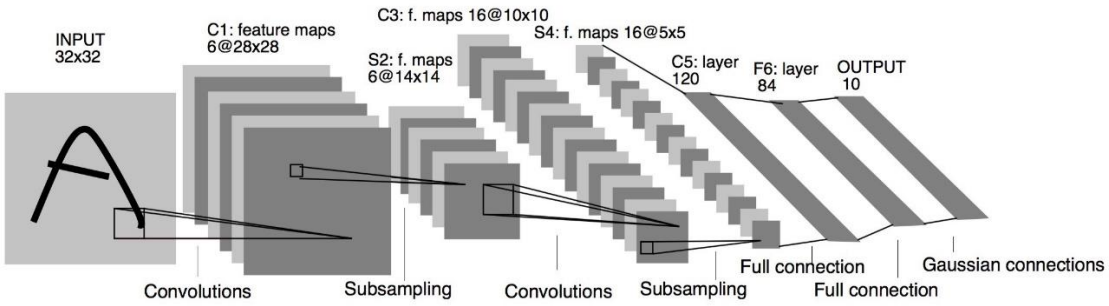
2.1.1 Redes Neurais Convolucionais

Uma Rede Neural Convolucional (RNC) é uma arquitetura de rede neural artificial, variante do *perceptron* de múltiplas camadas (multicamadas) que, assim como as redes neurais são baseadas em um processo biológico. As RNC são usadas em aplicações de detecção, classificação e reconhecimento de imagens e vídeos.

A RNC é dividida em alguns estágios principais. Os primeiros estágios são divididos em dois tipos principais: convolução e *pooling*, enquanto os outros são compostos pela camada completamente conectadas conforme mostra a Figura 1

.

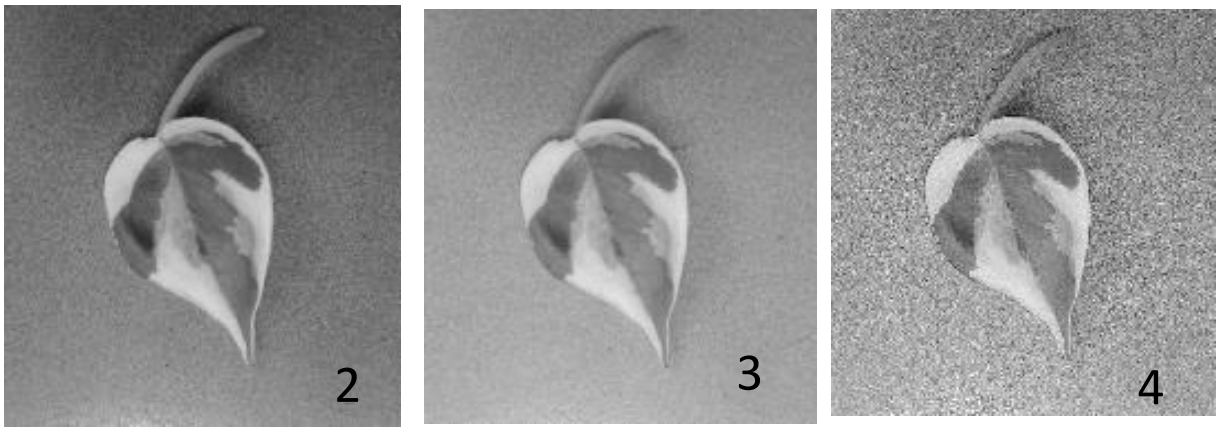
Figura 1 – Rede Neural Convolucional e as diferentes camadas.



Fonte: HAYKIN, S. Bookman. (2001).

As primeiras camadas que são as de convolução são responsáveis por extrair características das imagens de entrada, conhecido como *feature map*. Os neurônios dessas camadas são responsáveis por aplicar filtros (VARGAS; PAES; VASCONCELOS,2016). O resultado da aplicação desses filtros pode ser visto nas Figuras 2, 3 e 4.

Figura 2 – Filtro 1 (nitidez) / Figura 3 –Filtro 2 (borrão) / Figura 4 –Filtro 3 (detecção de borda).



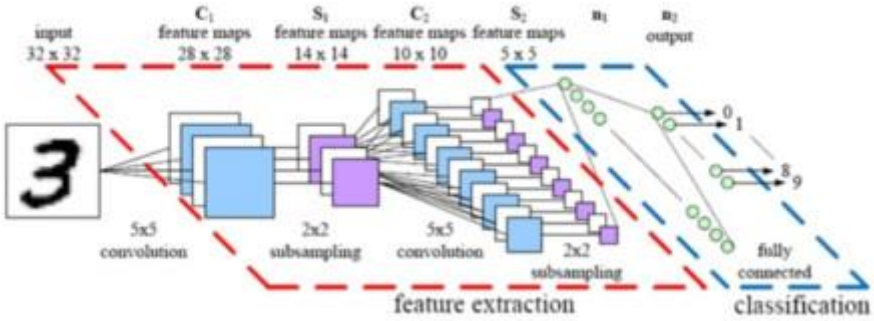
Fonte: Elaborado pelo autor. (2020).

O neurônio olha para uma janela de pixels da imagem da camada anterior, através das conexões sinápticas e seus pesos, que representam o filtro (também chamados de kernel ou máscara). Os neurônios de uma mesma camada aplicam o mesmo filtro na sua janela de pixels, o que faz necessário que os pesos dos neurônios de uma mesma camada sejam compartilhados durante a fase de treinamento (USHIZIMA, 2017).

Existe mais de um neurônio responsável por conectar uma mesma região da imagem de entrada conforme mostra a Figura 5. Um conjunto de *feature maps* quer dizer a utilização demais de um filtro.

Os mapas gerados pelos diversos filtros da camada convolucional são empilhados, formando um tensor cuja profundidade é igual ao número de filtros. Esse tensor será oferecido como entrada para a próxima camada (PONTI; COSTA, 2018).

Figura 5 – Rede Neural Convolucional.



Fonte: RAVINDRA, SAVARAM. (2017).

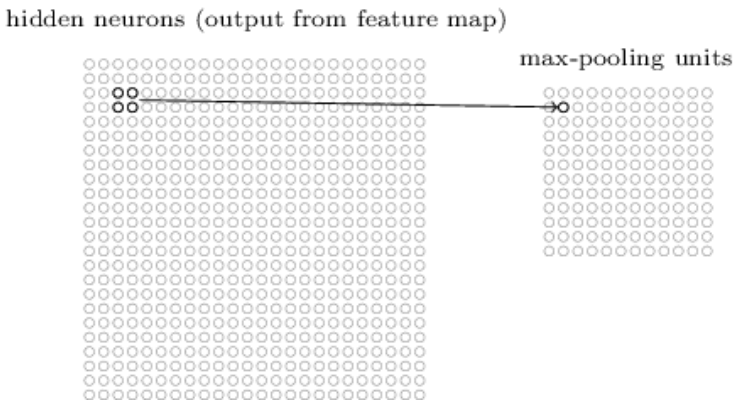
O *batch normalization* (BN) ocorre depois da camada de convolução. O objetivo principal é normalizar os mapas de características, mantendo a ativação média próxima à 0 e o desvio padrão próxima à 1. Isso faz com que o treinamento de rede seja mais rápido reduzindo a sensibilidade de inicialização da rede.

Após as convoluções, assim como nos *perceptron* de múltiplas camadas, uma função de ativação é aplicada, geralmente a função *ReLU* (NWANKPA et al., 2018).

Após as camadas de convolução, existem uma camada de *pooling*, que tem como objetivo diminuir a dimensão espacial dos dados de entrada, possibilitando que camadas mais profundas possam ter mais filtros sem que o custo computacional aumente por camada, além de evitar o *overfitting* (KARPATHY, 2017). No *pooling*, uma região do mapa de atributos é substituída por alguma métrica dessa região, por exemplo, o valor máximo (USHIZIMA, 2017)

Temos um exemplo na Figura 6 a aplicação da camada de *pooling* que utiliza a métrica do máximo valor, *max pooling.*

Figura 6 – Camada de pooling



Fonte: HAYKIN, S. Bookman. (2001).

A camada completamente conectada é a última camada, onde a partir das características extraídas pelas camadas de convolução e *pooling*, classificamos em alguma classe pré-definida.

2.1.2 Treinamento da rede neural convolucional

No início do treinamento todos os valores dos filtros das camadas convolucionais e os pesos das camadas totalmente conectadas são inicializadas de maneira aleatório ou seguindo o método.

Os valores são ajustados otimizando a acurácia da classificação.

A forma de treinamento da CNN mais utilizada é por meio do *backpropagation* (HAYKIN, 2007) conforme explicado anteriormente.

O processo de treinamento ocorre da seguinte forma:

1. Todos os filtros da rede e os pesos das camadas totalmente conectadas são inicializados. Existem diversas maneiras de inicializar os pesos (GLOROT; BENGIO, 2010).

2. A rede realiza o processo de propagação (convolução, *ReLU* e *pooling*, além da propagação nas camadas totalmente conectadas) sobre uma imagem de treinamento de entrada.

3. Uma probabilidade para cada classe de saída é calculada, assim como o erro total obtido na camada de saída conforme a Equação 1.

$$\text{Erro} = \frac{1}{n}\sum_{i=1}^{n}(Yi - \acute{Y}i)^2 \quad (1)$$

Onde Yi = Probabilidade real e Ýi = Probabilidade obtida

4. Utilizamos o *backpropagation* para os valores dos filtros e pesos. O ajuste é realizado na proporção que o parâmetro a ser ajustado contribui para o erro.

5. Os pasos 2 ao 4 são repetidos para todas as imagens do conjunto de treinamento.

A época é o conjunto total de imagens de treinamento que passam pelos passos 2 a 4 citados acima. Nos passos 3 e 4 temos classificações isso depende de quando o cálculo do erro será realizado e quando o algoritmo de aprendizagem irá atualizar os parâmetros da rede.

Definimos então o conceito de *batch* e iteração, o *batch* é um grupo de amostras do conjunto de treinamento com tamanho fixo ao final da passagem o erro é calculado e os ajustes são realizados, já a iteração é a quantidade de batches necessários para que seja completo uma época.

Por exemplo, se tivermos um *dataset* de 4000 amostras, podemos dividi-los em um *batch* de tamanho de 1000, o que será necessário 4 iterações para completar uma época. Podemos então classificar o algoritmo que abrange os passos 3 e 4 em classes distintas:

- Gradiente descendente batch (*Batch Gradient Descent*) – tamanho do *batch* é igual ao tamanho do conjunto de treinamento. Os pesos só são atualizados após todos os dados passarem pela rede.

- Gradiente descente estocástico (*Stochastic Gradient Descent*) – tamanho do *batch* é igual a 1. Os pesos são atualizados para todos das amostras do conjunto de dados. Utilizamos quando temos grandes quantidade de dados.

- Gradiente descente *mini-batch* – tamanho do *batch* é maior que 1 e menor que o tamanho do conjunto de dados. Os tamanhos de *mini-batch* utilizados na literatura são 32, 64 e 128.

Para finalizarmos então o treinamento devemos verificar algumas condições se estão satisfeitas:

- A média do erro obtida pela rede na época atual precisa ser inferior à um limiar ou atingir o número de épocas.
- O término do treinamento irá indicar que a rede está pronta para classificar as imagens de um conjunto de treinamento de maneira ótima.
- O treinamento altera somente os valores dos filtros e pesoas das camadas totalmente conectadas.
- O tamanho do filtro, hiperparâmetros, o *padding* e o número de camadas não são ajustados, eles são definidos antes do passo 1 do fluxo.

2.2 Transferencia de Aprendizagem

Transferência de aprendizagem define-se como sendo a habilidade de um sistema em reconhecer e aplicar o conhecimento aprendido em tarefas anteriores para a solução de novos problemas relacionados. O objetivo dessa abordagem consiste em extrair o conhecimento obtido por um modelo a partir de tarefas de origem aplicando em uma tarefa destino.

A utilização da transferência de aprendizagem é muito popular em *deep learning* visto que é necessária uma grande quantidade de recursos computacionais e uma quantidade de dados enorme para treinar esse tipo de modelo. Redes neurais convolucionais modernas podem levar semanas para treinar em múltiplas GPUs.

Uma das premissas para a utilização da técnica de transferência de aprendizagem é que os domínios de origem e de destino devem ser relacionados. Tan et al. (2015) afiram que essa relação pode ser concretizada na forma de instancias (BICKEL; SAWADE; SCHEFFER, 2009) ou características (SATPAL; SARAWAGI, 2007). Se nenhuma relação direta for encontrada, a transferência forçada não irá

funcionar, resultando em nenhuma melhora e até piorando o desempenho no domínio de destino (FITZGERALD; THOMAZ, 2015).

Uma rede pré-treinada em um banco de dados com milhares de imagens é utilizada, remove-se a última camada completamente conectada, substitui-se por uma camada que classifique nas classes desejadas para o novo conjunto de dados e é realizado o treinamento. A rede neural convolucional funcionará como o extrator de características, uma vez que essas características são adquiridas, basta classificá-las na rede completamente conectada que então classifica nas classes desejadas por conta da substituição da última camada (TRANSFER. . ., 2015).

2.3 Classificadores

Classificação é o processo de tomar algum tipo de entrada e atribuir um rótulo a ela. Sistemas de classificação são usados geralmente quando as previsões são de natureza distinta.

Existem várias métricas utilizadas para a avaliação dos classificadores, as mais comuns e as que iremos utilizar nessa pesquisa serão: acurácia, sensibilidade e especificidade. Para entender sobre essas métricas, é necessário definir os conceitos por trás dos seus cálculos, sendo eles: verdadeiro positivo (VP), falso positivo (FP), verdadeiro negativo (VN) e falso negativo (FN).

No contexto das imagens, as seguintes definições são utilizadas:

- Verdadeiro positivo: Número de amostras rotuladas como doentes que são identificadas como doentes.
- Falso positivo: Número de amostras rotuladas como saudáveis que são identificadas como doentes.
- Verdadeiro negativo: Número de amostras rotuladas como saudáveis que são identificadas como saudáveis.
- Falso negativo: Número de amostras rotuladas como doentes que são identificadas como saudáveis. (BARATLOO et al., 2015)

A Matriz de confusão (tradução livre) é uma matriz de valores reais e valores preditos pelo seu classificador. Imprimir a matriz de confusão é uma forma intuitiva de saber como seu classificador está se comportando.

Imagine que você tem um classificador cuja a tarefa é classificar plantas saudáveis ou plantas doentes a partir de imagens. Só visualizando a acurácia do modelo não é possível ver se o seu classificador está classificando bem as instancias que realmente são plantas saudáveis. O modelo pode ter uma acurácia alta, mas está acertando muito só uma classe, por isso utilizamos a matriz de confusão para analisarmos isso.

Na Tabela 1, tem-se um exemplo de matriz de confusão na qual uma base de testes contém um total de 13 amostras, das quais 8 são plantas doentes, 5 plantas saudáveis.

Tabela 1 – Matriz de confusão para 2 classes

Classes Previstas		Classes	
		Plantas Doentes	Plantas Saudáveis
	Plantas Doentes	5	2
	Plantas Saudáveis	3	3

Fonte: Autoria própria 2020.

Nessa matriz de confusão, das 8 fotos de plantas doentes, o sistema avaliou que 3 eram plantas saudáveis e, das 5 fotos de plantas saudáveis, previu que 2 eram plantas doentes. Todas as previsões corretas estão localizadas na diagonal da tabela, portanto, é fácil inspecionar visualmente a tabela em busca de erros de previsão, pois eles serão representados por valores fora da diagonal.

Em termos abstratos, a matriz de confusão é conforme a Tabela 2.

Tabela 2 – Tabela de Confusão

Classes Previstas		Classes	
		P	N
	P	TP	FP
	N	FN	TN

Fonte: Autoria própria 2020.

onde: P = Positivo; N = negativo; TP = Verdadeiro Positivo; FP = falso positivo; TN = Verdadeiro Negativo; FN = falso negativo.

Para a área de análises preditivas, a tabela de confusão é uma tabela 2x2 a qual apresenta o número de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos, permitindo uma análise mais detalhada, do que a simples proporção de classificações corretas também chamada de *accuracy*.

- Acurácia: É a habilidade do modelo diferenciar a imagem nas classes plantas saudáveis e plantas doentes corretamente, é calculada a partir das proporções de verdadeiros positivos e verdadeiros negativos sobre todos os casos avaliados.

$$\text{Acurácia} = \frac{VP+VN}{VP+VN+FP+FN}$$

- Sensibilidade: É a habilidade de determinar as imagens rotuladas como plantas doente de forma correta

$$\text{Sensibilidade} = \frac{VP}{VP+FN}$$

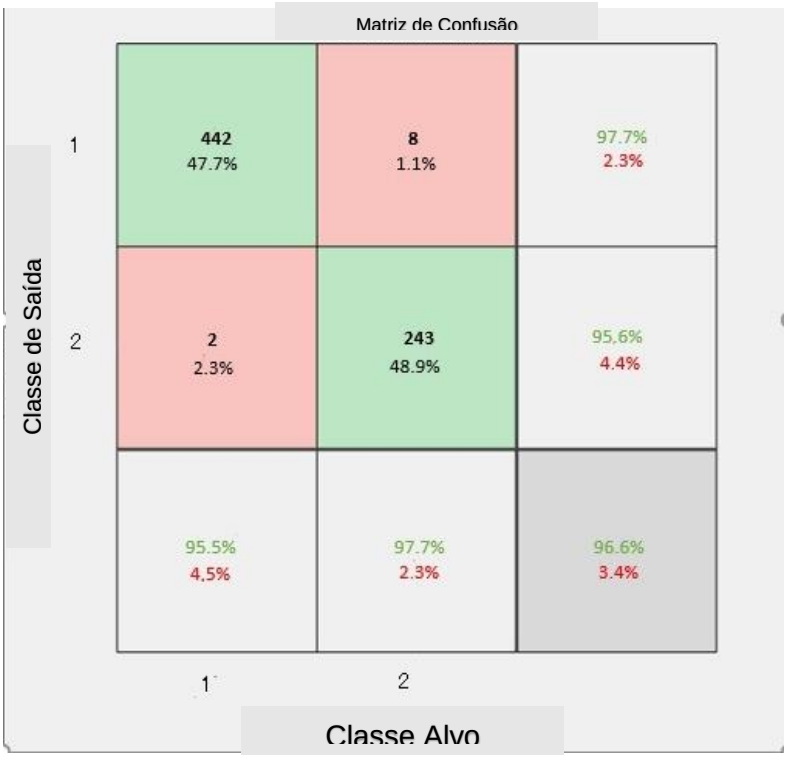
- Especificidade: É a habilidade de determinar as imagens rotuladas como plantas saudáveis de forma correta

$$\text{Especificidade} = \frac{VN}{VN+FP}$$

No exemplo da Figura 7, temos como exemplo 450 classificações como sendo classe 1, em 97,7% das vezes a predição estava correta (442 vezes) e em 2,3% estavam erradas. Para a classe 2, das 245 amostras classificadas como pertencentes da classe 2, em 95,6% ele acerta e 4,4% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 96,6%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os

valores de 95,5% e 97,7%. Podemos também extrair os valores dos VP, FP, FN, VN da matriz, são eles: 442, 8, 2 e 243, respectivamente.

Figura 7 - Exemplo de matriz de confusão



Fonte: Autoria própria 2020.

2.4 Base de dados

A base de dados utilizada possui imagens de 12 tipos de plantas saudáveis e portadores da doença, adquiridos no banco de dados do PlantCLEF (2020) e um acervo pessoal de plantas conforme mostra a Tabela 3.

Os dados do PlantCLEF (2020) a serem analisados podem ser encontrados no site para download. São amostras de imagens submetidas por usuários da aplicação mobile PlantNet a qual está disponível para iPhone e Android e conta com a participação de centenas de usuários ativos, sendo submetidas milhares de imagens por dia.

A coleta do acervo pessoal foi realizada com a câmera do celular Xiaomi redmi note 9 AI Quad Camera com dimensão de 2990x4000 pixels. As imagens das plantas totalizam 440 imagens distintas.

Tabela 3 – Tabela de quantidade de imagens por tipo de plantas

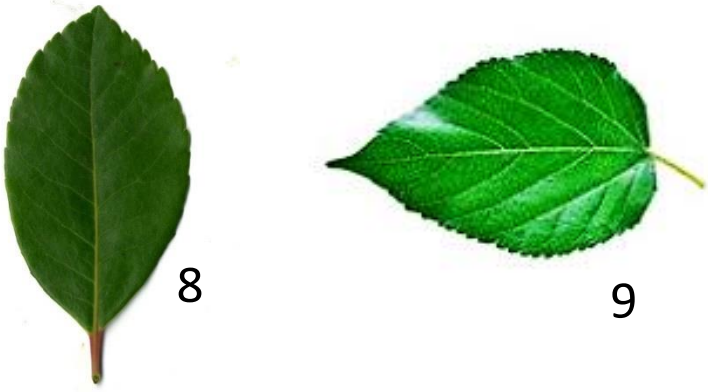
Nome Planta	**Quantidade de Imagens Saudáveis**	**Quantidade de Imagens Doentes**
Uva	12	11
Amora	12	11
Pêra	22	25
Manga	15	15
Café	35	32
Acerola	15	12
Soja	32	35
Abóbora	10	8
Morango	12	15
Mamão	10	12
Medronheiro	10	12
Maracujá	15	12
Total	**200**	**200**

Fonte: Autoria própria 2020.

As imagens rotuladas de acordo com sua classificação normais (saudáveis) e com a patologia fúngica. As imagens são rotuladas por biólogos e taxonomistas especialistas do PlantCLEF (2020). A divisão das pacientes e suas imagens nas duas classes (normais e com patologia) foram feitas de maneira igual, portanto existem 40 imagens de folhas e 200 imagens rotuladas como normais e o mesmo com plantas com patologia.

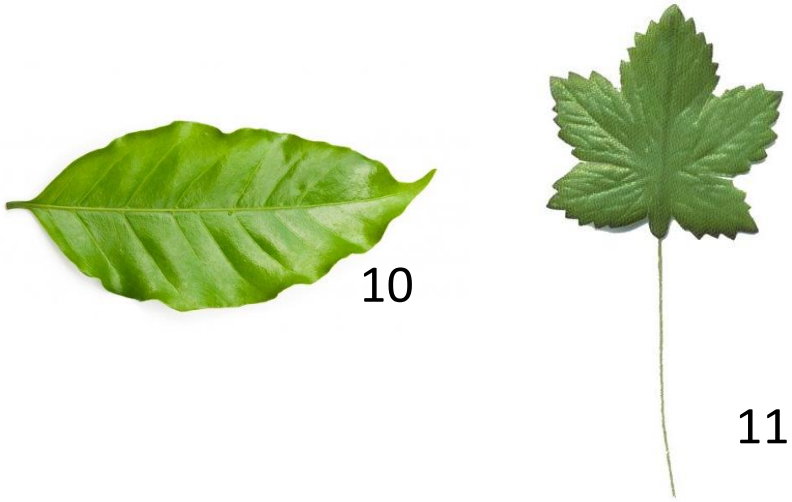
As Figuras 8, 9, 10 e 11, retiradas da base de dados, ilustram as amostras de uma folha normal, enquanto as Figuras 12, 13, 14 e 15, retiradas da base de dados, ilustram as amostras de uma folha doente.

Figura 8 – Folha saudável – (Medronheiro) / Figura 9 – Folha saudável – (Amora)



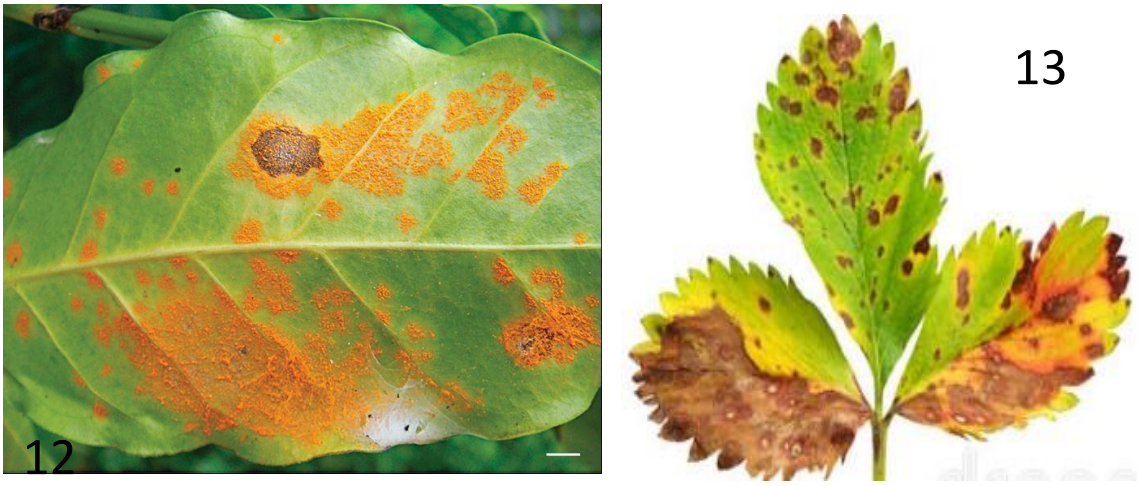
Fonte: Base de Dados PlantCLEF (2020).

Figura 10 – Folha saudável – (Café) / Figura 11 – Folha saudável – (Uva)



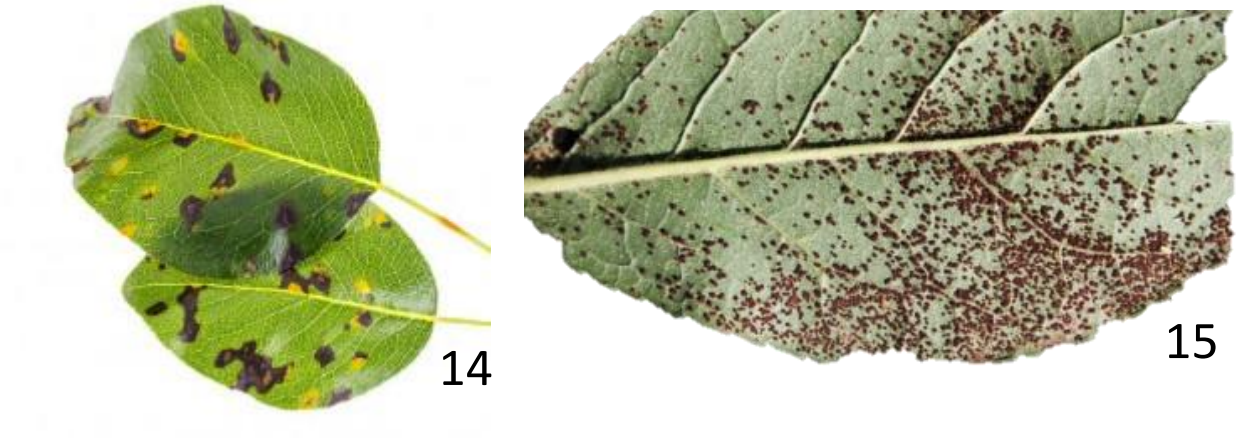
Fonte: Base de Dados PlantCLEF (2020).

Figura 12 – Folha doente – (Café) / Figura 13 – Folha doente – (Morango)



Fonte: Acervo pessoal (2020) / Base de Dados PlantCLEF (2020)

Figura 14 – Folha doente – (Pêra) / Figura 15 – Folha doente – (Amora)



Fonte: Acervo pessoal (2020).

As imagens foram separadas em dois conjuntos distintos, o conjunto utilizado no processo de treinamento e validação e o conjunto utilizado nos testes posteriores, as proporções da divisão foram:

- Conjunto de treinamento e validação: 200 imagens de folhas saudáveis e 200 de folhas doentes (divididos em 80% para treinamento e 20% para validação).
- Conjunto de teste: 20 imagens de folhas saudáveis e 20 de folhas doentes.

O objetivo desse conjunto de teste é validar o modelo CNN treinado com os dados que ainda não foram utilizados no treinamento, ou seja, que não foi visto pela rede, obtendo um desempenho real das classificações da rede, já o conjunto de treinamento é utilizado no processo de aprendizado do modelo.

2.5 Métodos e experimentos

A implementação foi desenvolvida no ambiente e linguagem do Matlab, utilizando a visão computacional e as *ToolBoxes* de *Deep Leraning* (Apendice A). Utilizamos nesse trabalho arquiteturas de redes neurais convolucionais com desempenho comprovado para realizar a classificação das nossas imagens, sendo elas: *AlexNet* (25 camadas, ilustrado na figura 16), *GoogleNet* (144 camadas), *ResNet-18* (72 camadas), *VGG-19* (47 camadas).

Figura 16 - Camadas AlexNet

1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Grouped Convolution	2 groups of 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU
12	'conv4'	Grouped Convolution	2 groups of 192 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Grouped Convolution	2 groups of 128 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench' and 999 other classes

Fonte: AlexNet (2017).

É necessária uma grande quantidade de dados no treinamento em modelos de *deep learning*, por isso foi selecionado por utilizar modelos pré-treinados que oferece uma rica gama de representação de características. O processo de treinamento é reduzido à substituição e ao treinamento das camadas completamente conectadas da rede fornecida para que ela consiga classificar as imagens da base de dados. Treinamos as camadas convolucionais no processo com uma taxa de aprendizado muito menor que as camadas conectadas para não perder os filtros kernel da rede treinada.

Foram definidas algumas opções de treinamento de rede, sendo elas:

- Taxa de aprendizado das camadas convolucionais de 1e-4
- Tamanho do mini-batch de 10
- Taxa de aprendizado das camadas completamente conectadas de 10
- Embaralhamento dos dados ao final de cada época
- Aplicado a técnica de *data augmentation* nas imagens, como: espalhar as imagens aleatoriamento em relação ao eixo vertical, movimentar aleatoriamente as imagens horizantal ou vertical em até 30 pixels e aumentar a imagem nos eixos horizontal e vertical aleatoriamente no intervalo de 0.9 e 1.1. As técnicas de *augmentation* utilizadas tem o objetivo de evitar o overfitting, evitando que o modelo memorize detalhes específicos das imagens de treinamento. O tamanho das imagens foi reajustado ao tamanho de entrada da rede.

3. Resultados

Os experimentos citados na seção 3.2 foram utilizados variando o número de épocas de treinamento. As épocas utilizadas foram 3, 5, 10 e 20, números de épocas entre 20 a 30 e depois de 30 começou a apresentar indicadores de overfitting da rede, por isso não utilizamos essas épocas.

Os resultados apresentados abaixo mostram a acurácia, especificidade e sensibilidade obtidas pelos dados de validação e os dados de teste pós-treinamento. Para melhores resultados das redes utilizado nos dados de teste utilizamos as matrizes de confusão. Na conclusão de resultados da pesquisa consideramos os experimentos nos dados de teste, pois possuem nenhuma relação com o treinamento e expressam melhor o quão genérica a rede treinada está.

3.1 AlexNet

Na Tabela 4 é apresentado o resultado dos dados de teste da arquitetura AlexNet. As épocas utilizadas foram 3, 5, 10 e 20.

Tabela 4 – Resultado AlexNet – dados de teste

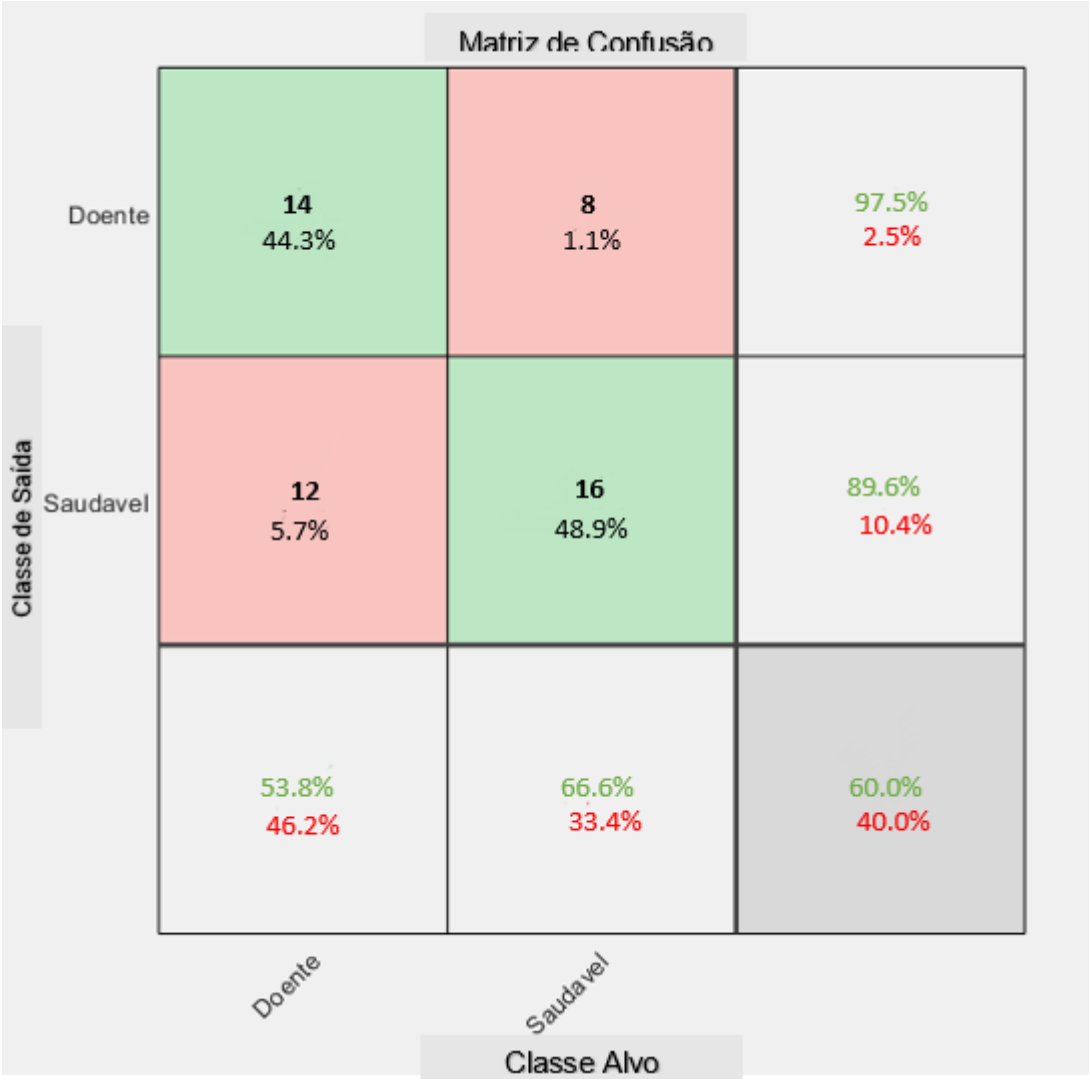
Número de épocas	Acurácia	Sensibilidade	Especificidade
3	60,0%	53,8%	66,6%
5	73,7%	62,5%	81,8%
10	88,2%	85,1%	92,1%
20	93,2%	90,9%	95,5%

Fonte: Elaborado pelo Autor. (2020).

Na Figura 17 temos a matriz de confusão com 3 épocas com 22 classificações como sendo classe 1, em 97,5% das vezes a predição estava correta (14 vezes) e em 2,5% estavam erradas. Para a classe 2, das 28 amostras classificadas como pertencentes da classe 2, em 89,6% ele acerta e 10,4% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 60,0%. A

sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 53,8% e 66,6%.

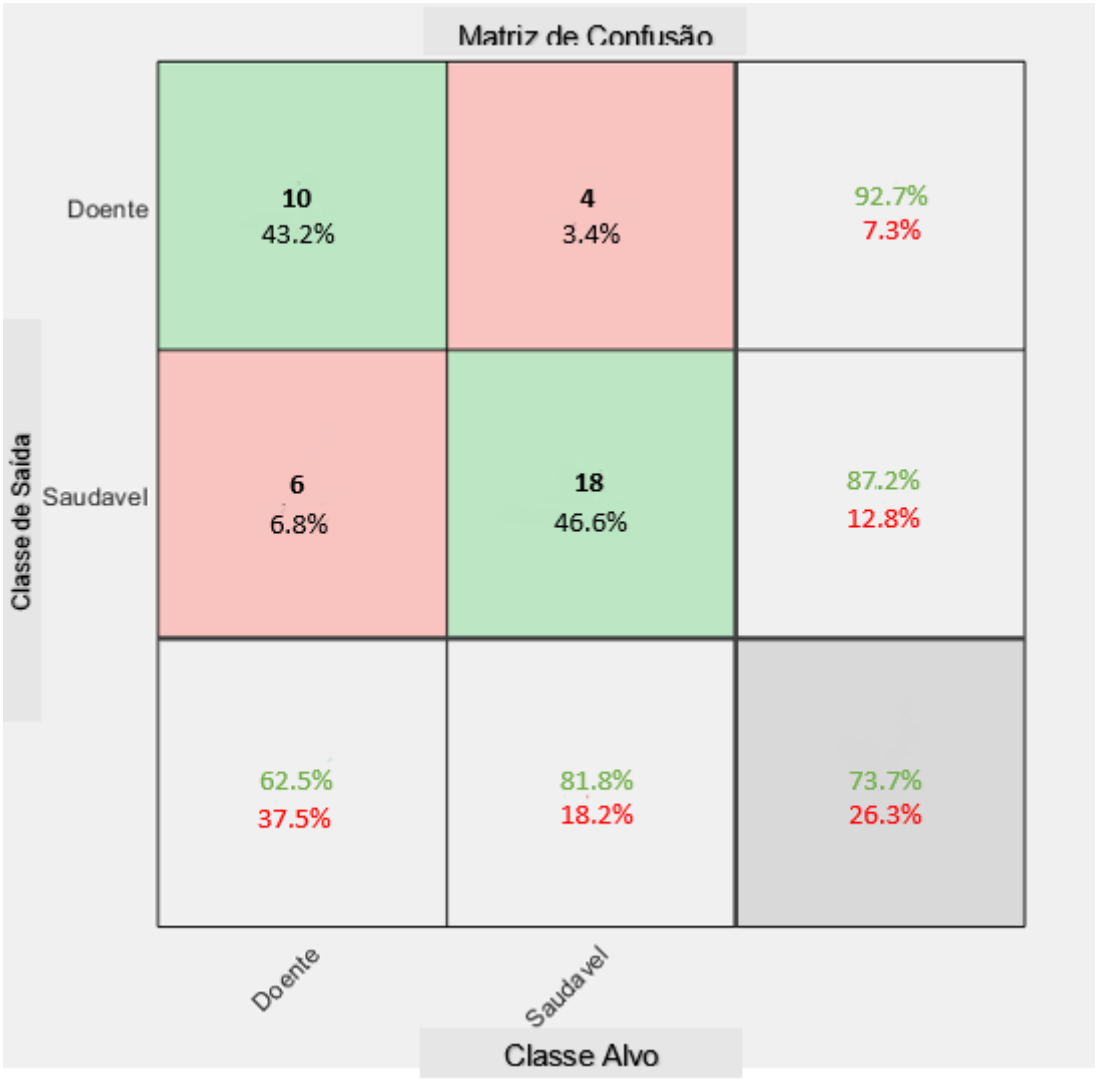
Figura 17 – Matriz de Confusão AlexNet – 3 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 18 temos a matriz de confusão com 5 épocas com 14 classificações como sendo classe 1, em 92,7% das vezes a predição estava correta (10 vezes) e em 7,3% estavam erradas. Para a classe 2, das 24 amostras classificadas como pertencentes da classe 2, em 87,2% ele acerta e 12,8% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 73,7%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 62,5% e 81,8%.

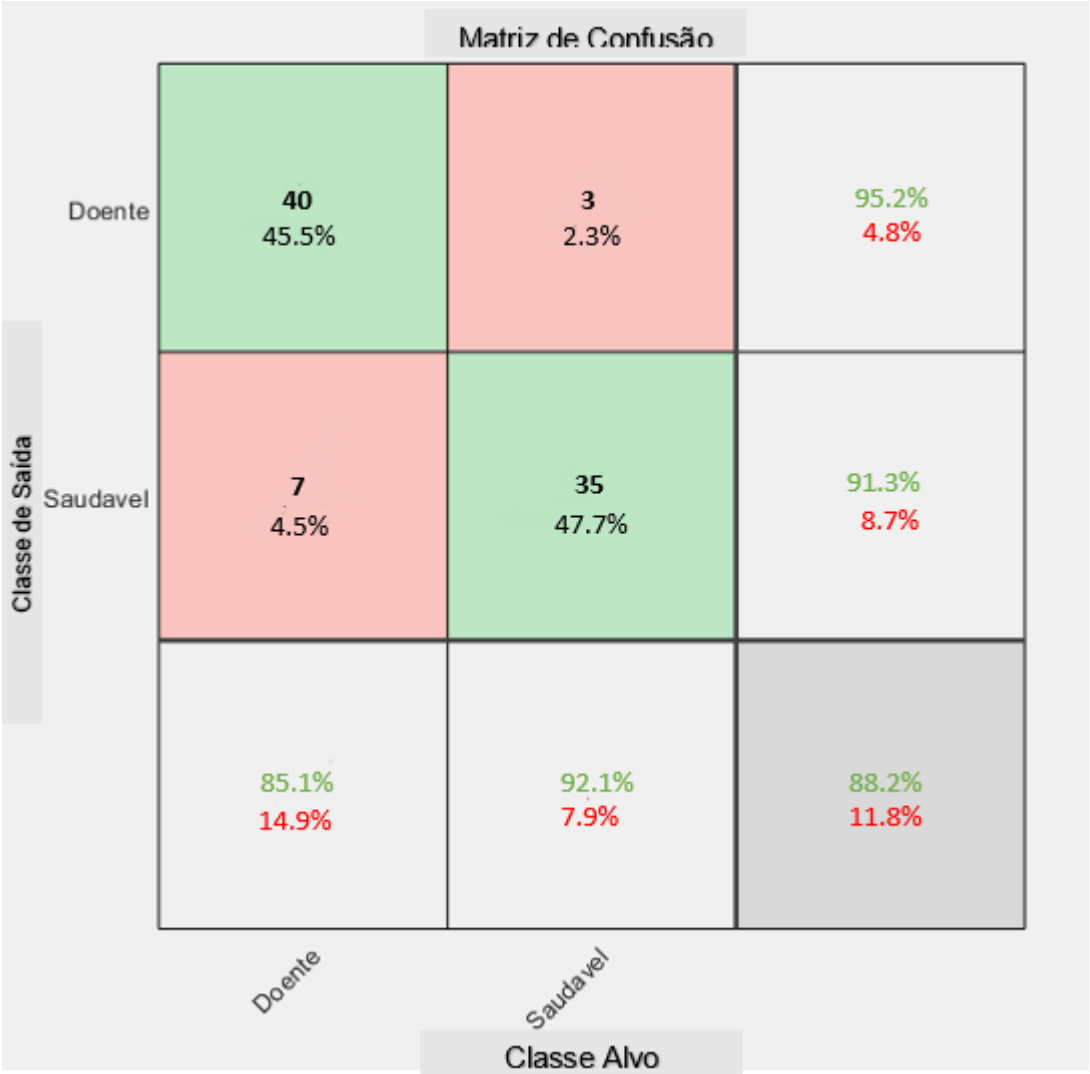
Figura 18 – Matriz de Confusão AlexNet – 5 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 19 temos a matriz de confusão com 10 épocas com 43 classificações como sendo classe 1, em 95,2% das vezes a predição estava correta (40 vezes) e em 4,8% estavam erradas. Para a classe 2, das 42 amostras classificadas como pertencentes da classe 2, em 91,3% ele acerta e 8,7% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 88,2%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 85,1% e 92,1%.

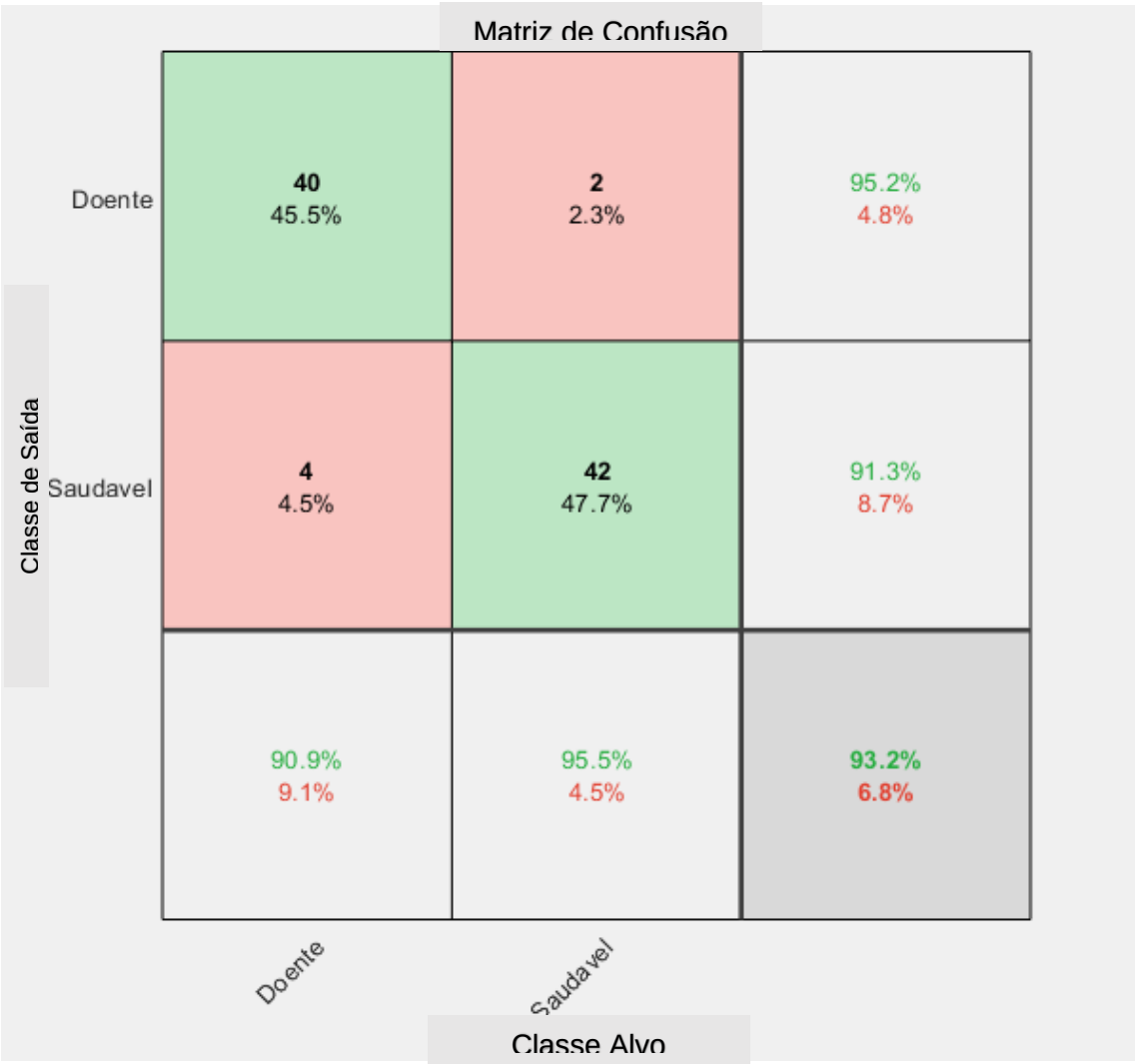
Figura 19 – Matriz de Confusão AlexNet – 10 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 20 temos a matriz de confusão com 20 épocas com 42 classificações como sendo classe 1, em 95,2% das vezes a predição estava correta (40 vezes) e em 4,8% estavam erradas. Para a classe 2, das 46 amostras classificadas como pertencentes da classe 2, em 91,3% ele acerta e 8,7% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 93,2%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 90,9% e 95,5%.

Figura 20 – Matriz de Confusão AlexNet – 20 épocas



Fonte: Elaborado pelo Autor. (2020).

3.2 GoogleNet

Na Tabela 5 é apresentado o resultado dos dados de teste da arquitetura GoogleNet. As épocas utilizadas foram 3, 5, 10 e 20.

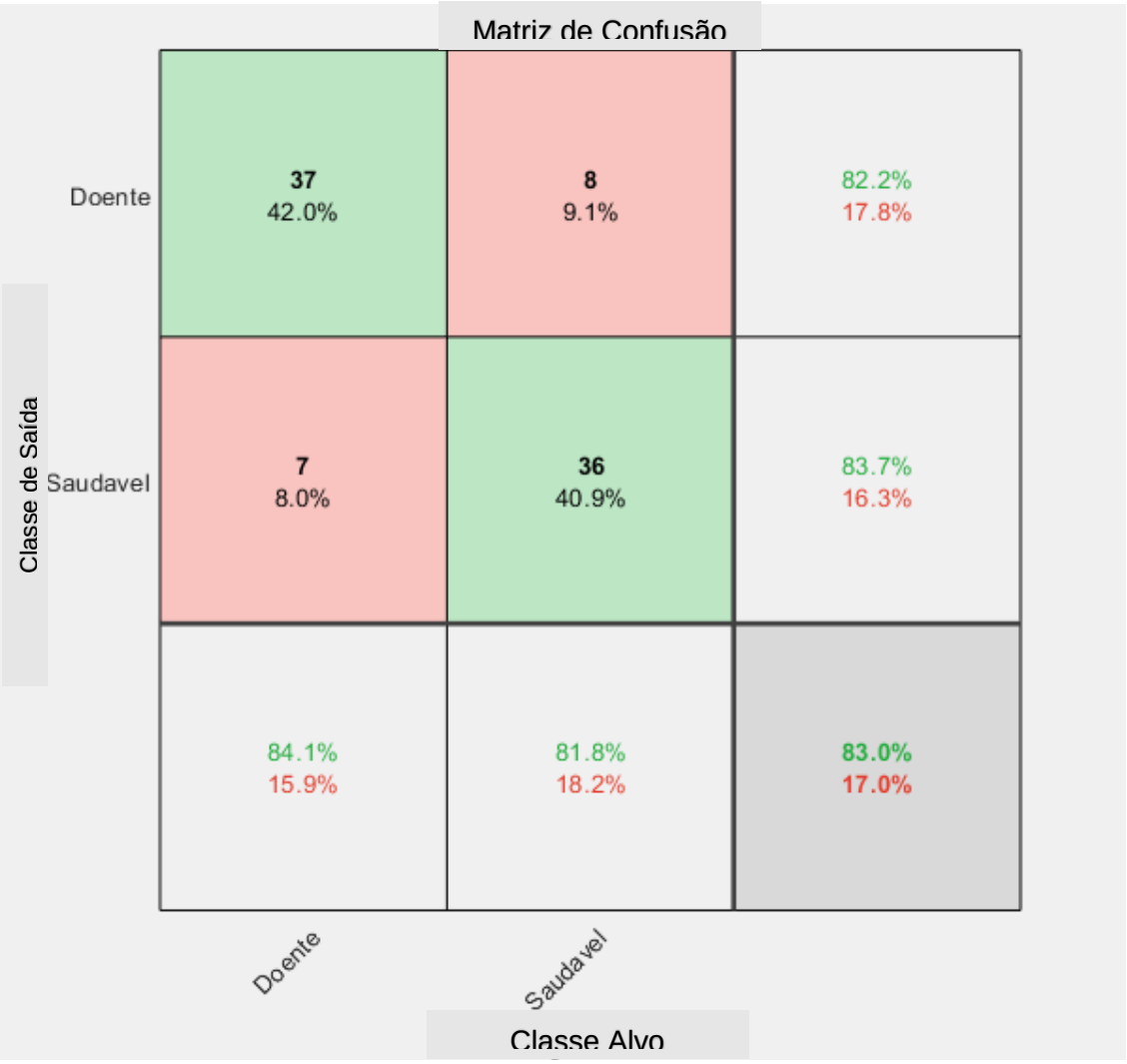
Tabela 5 – Resultado GoogleNet – dados de teste

Número de épocas	Acurácia	Sensibilidade	Especificidade
3	83,0%	84,1%	81,8%
5	93,2%	88,6%	97,7%
10	88,6%	84,1%	93,2%
20	92,1%	86,3%	97,7%

Fonte: Elaborado pelo Autor. (2020).

Na Figura 21 temos a matriz de confusão com 3 épocas com 45 classificações como sendo classe 1, em 82,2% das vezes a predição estava correta (37 vezes) e em 17,8% estavam erradas. Para a classe 2, das 43 amostras classificadas como pertencentes da classe 2, em 83,7% ele acerta e 16,3% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 83,0%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 84,1% e 81,8%.

Figura 21 – Matriz de Confusão GoogleNet – 3 épocas

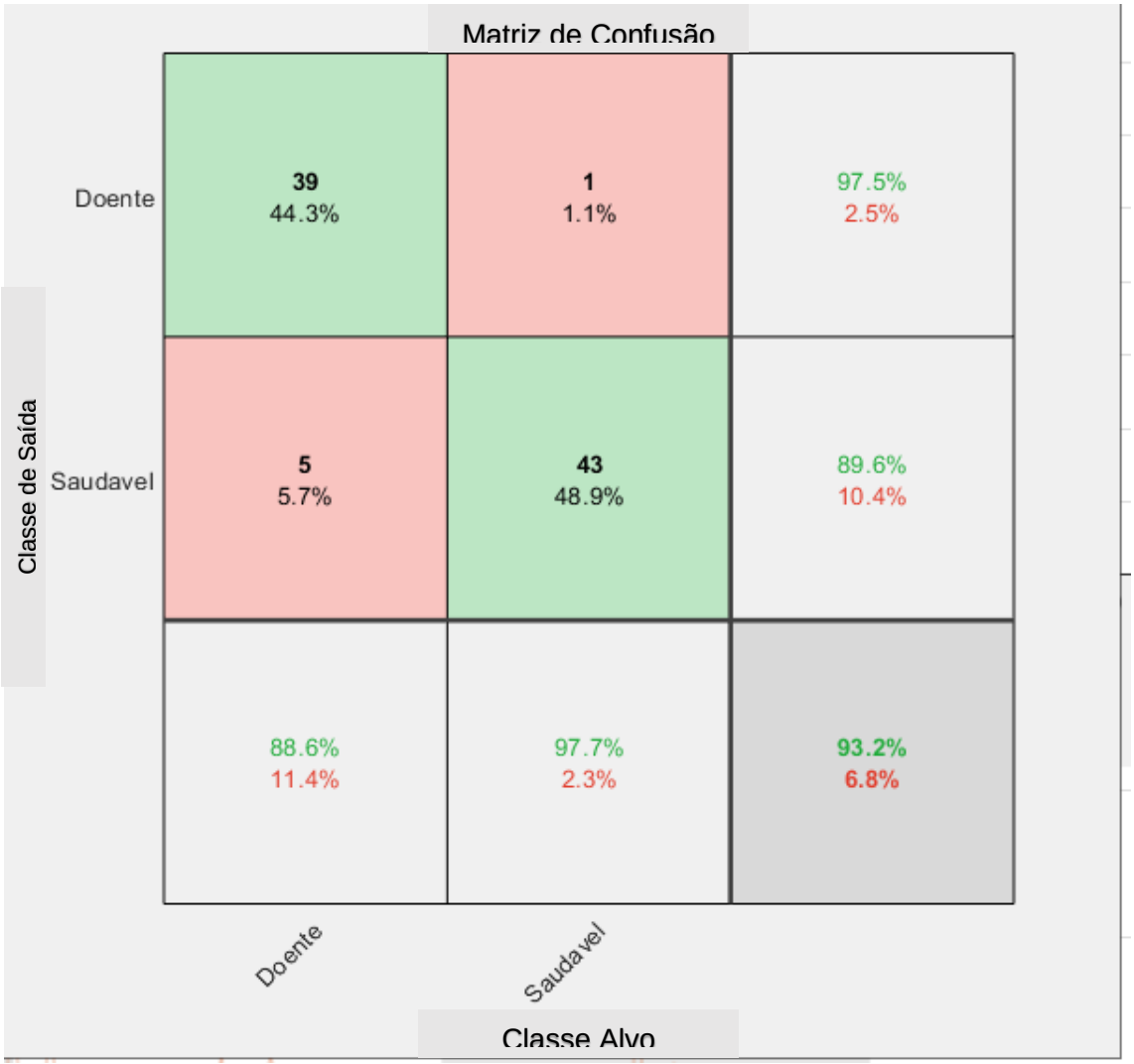


Fonte: Elaborado pelo Autor. (2020).

Na Figura 22 temos a matriz de confusão com 5 épocas com 40 classificações como sendo classe 1, em 97,5% das vezes a predição estava correta (39 vezes) e em 2,5% estavam erradas. Para a classe 2, das 48 amostras classificadas como pertencentes da classe 2, em 89,6% ele acerta e 10,4% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 93,2%. A

sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 88,6% e 97,7%.

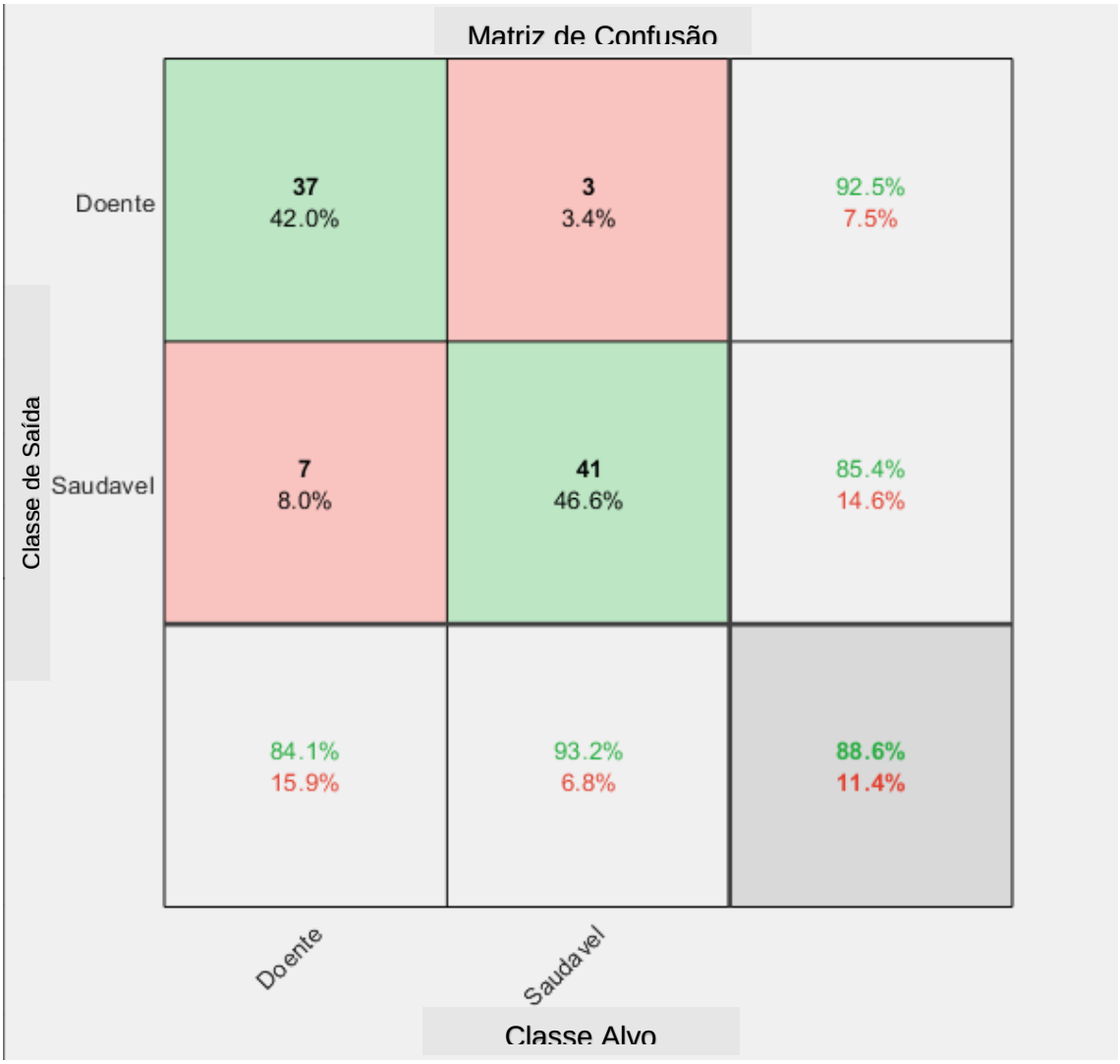
Figura 22 – Matriz de Confusão GoogleNet – 5 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 23 temos a matriz de confusão com 10 épocas com 40 classificações como sendo classe 1, em 92,5% das vezes a predição estava correta (37 vezes) e em 7,5% estavam erradas. Para a classe 2, das 48 amostras classificadas como pertencentes da classe 2, em 85,4% ele acerta e 14,6% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 88,6%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 84,1% e 93,2%.

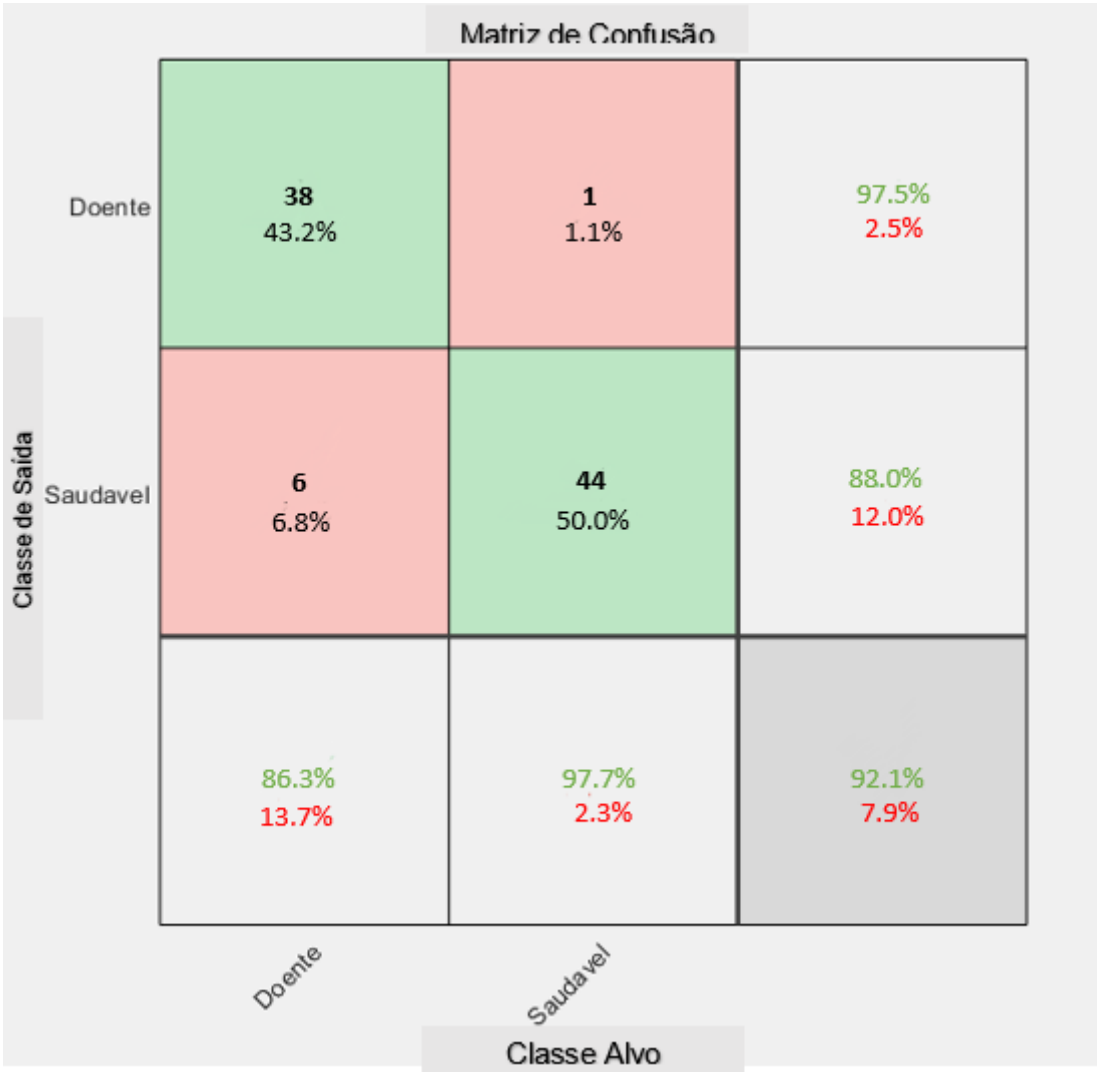
Figura 23 – Matriz de Confusão GoogleNet – 10 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 24 temos a matriz de confusão com 20 épocas com 39 classificações como sendo classe 1, em 97,5% das vezes a predição estava correta (38 vezes) e em 2.5% estavam erradas. Para a classe 2, das 50 amostras classificadas como pertencentes da classe 2, em 88,0% ele acerta e 12,0% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 92,1%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 86,3% e 97,7%.

Figura 24 – Matriz de Confusão GoogleNet – 20 épocas



Fonte: Elaborado pelo Autor. (2020).

3.3 ResNet-18

Na Tabela 6 é apresentado o resultado dos dados de teste da arquitetura ResNet-18. As épocas utilizadas foram 3, 5, 10 e 20.

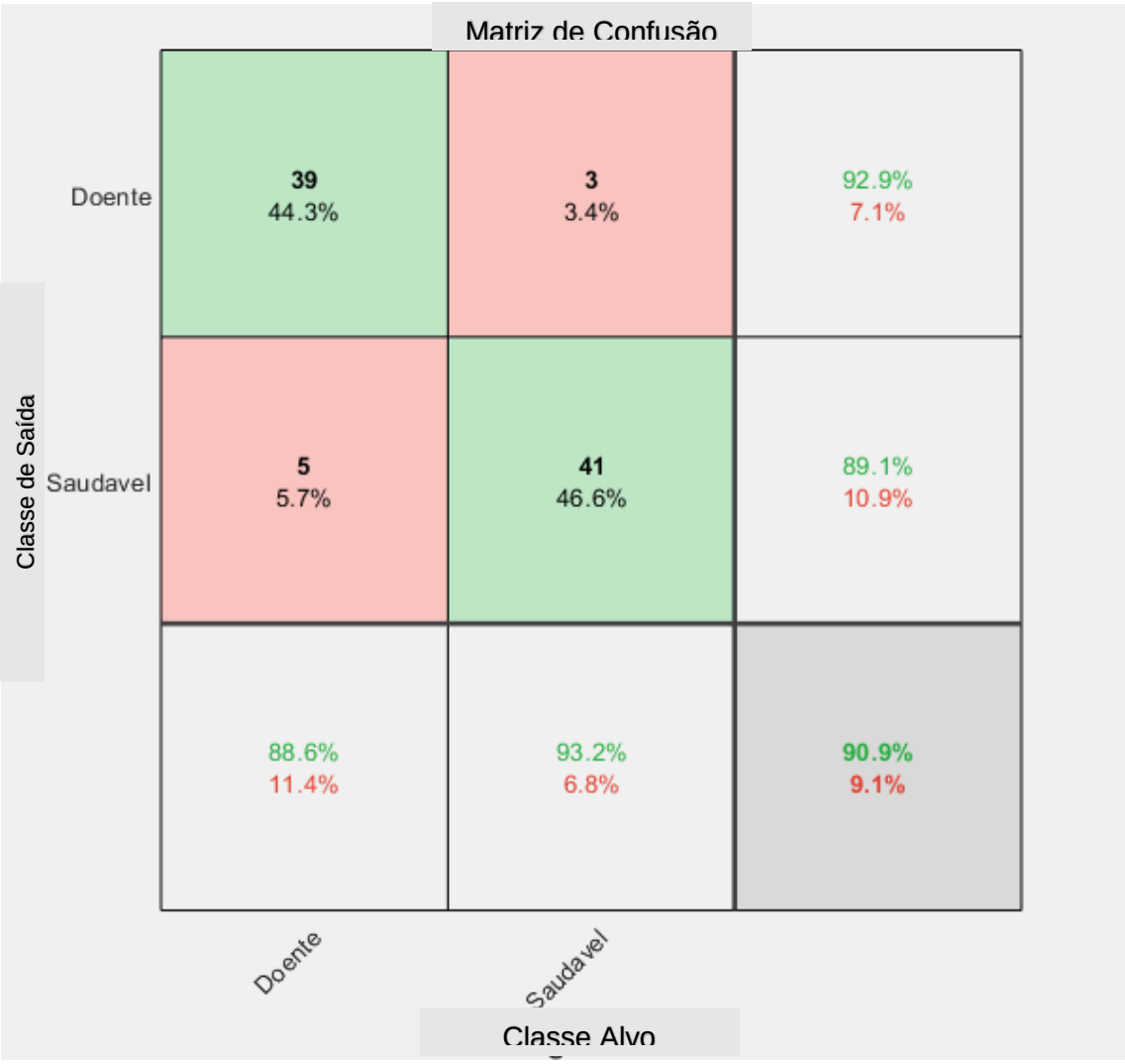
Tabela 6 – Resultado ResNet-18 – dados de teste

Número de épocas	Acurácia	Sensibilidade	Especificidade
3	90,9%	88,6%	93,2%
5	95,5%	93,2%	97,7%
10	88,6%	93,2%	84,1%
20	**96,6%**	**97,7%**	**95,5%**

Fonte: Elaborado pelo Autor. (2020).

Na Figura 25 temos a matriz de confusão com 3 épocas com 42 classificações como sendo classe 1, em 92,9% das vezes a predição estava correta (39 vezes) e em 7,1% estavam erradas. Para a classe 2, das 46 amostras classificadas como pertencentes da classe 2, em 89,1% ele acerta e 10,9% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 90,9%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 88,6% e 93,2%.

Figura 25 – Matriz de Confusão ResNet-18 – 3 épocas

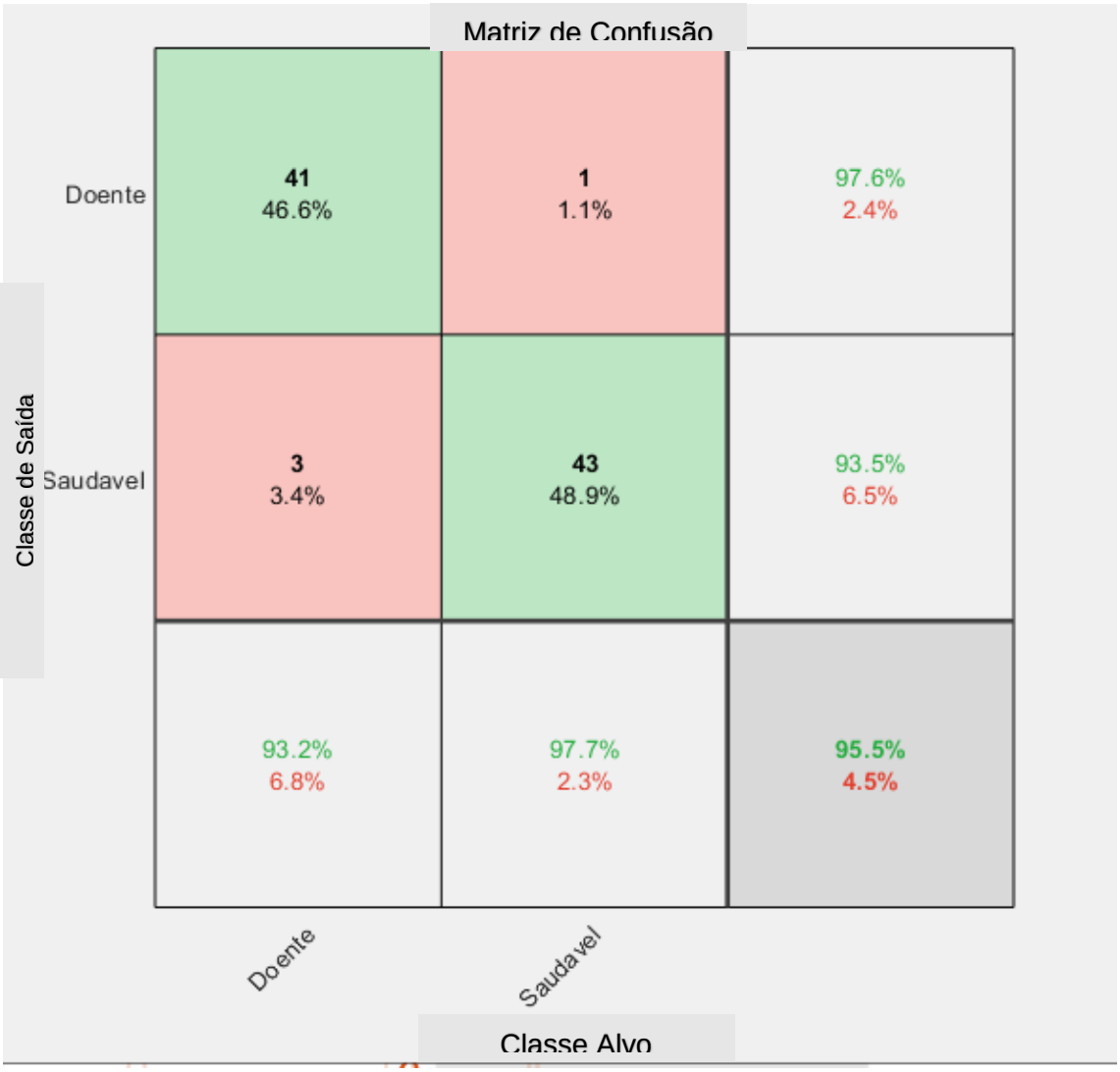


Fonte: Elaborado pelo Autor. (2020).

Na Figura 26 temos a matriz de confusão com 5 épocas com 42 classificações como sendo classe 1, em 97,6% das vezes a predição estava correta (41 vezes) e em 2,4% estavam erradas. Para a classe 2, das 46 amostras classificadas como

pertencentes da classe 2, em 93,5% ele acerta e 6,5% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 95,5%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 93,2% e 97,7%.

Figura 26 – Matriz de Confusão ResNet-18 – 5 épocas

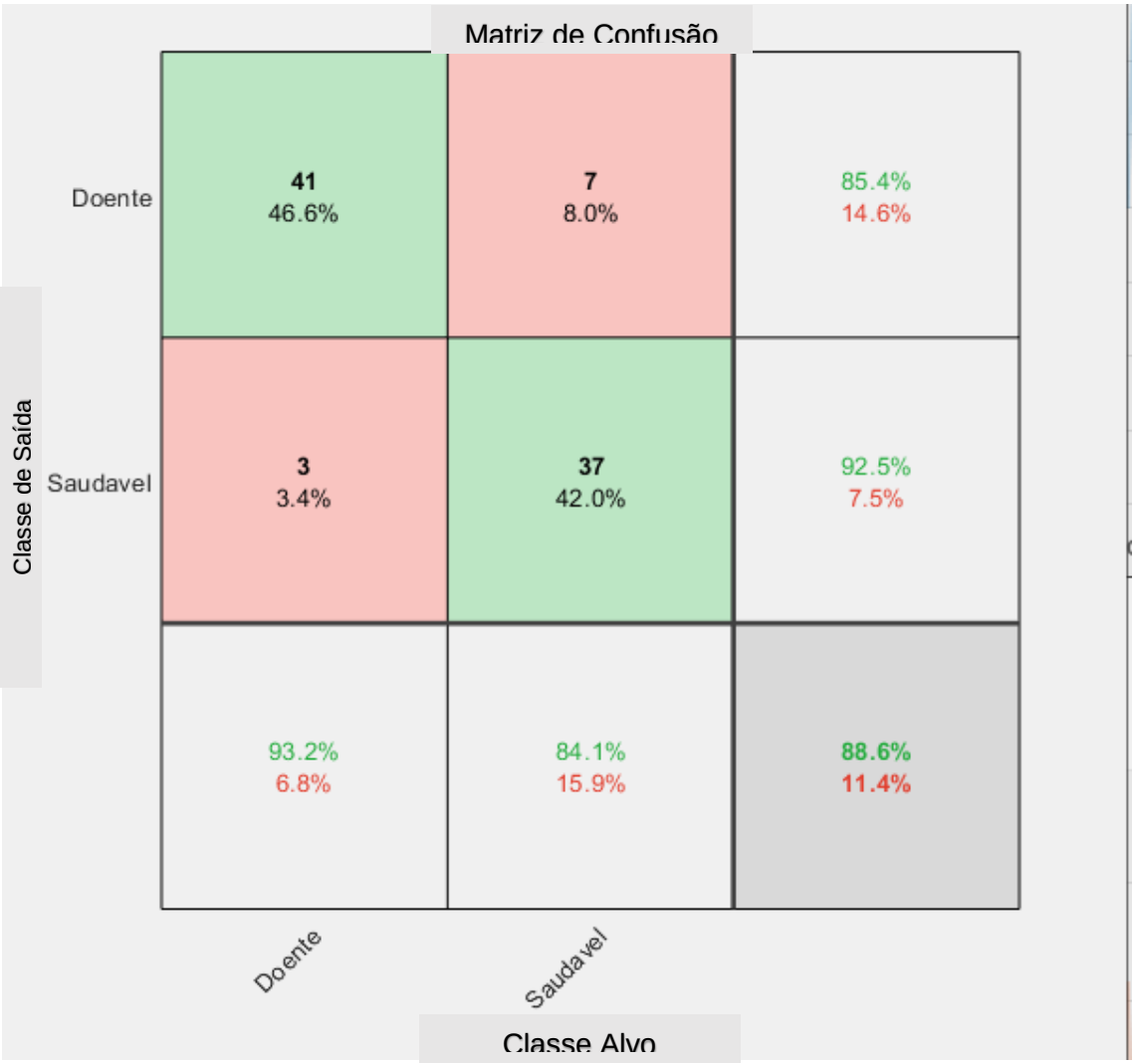


Fonte: Elaborado pelo Autor. (2020).

Na Figura 27 temos a matriz de confusão com 10 épocas com 48 classificações como sendo classe 1, em 85,4% das vezes a predição estava correta (41 vezes) e em 14,6% estavam erradas. Para a classe 2, das 40 amostras classificadas como pertencentes da classe 2, em 92,5% ele acerta e 7,5% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna:

88,6%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 93,2% e 84,1%.

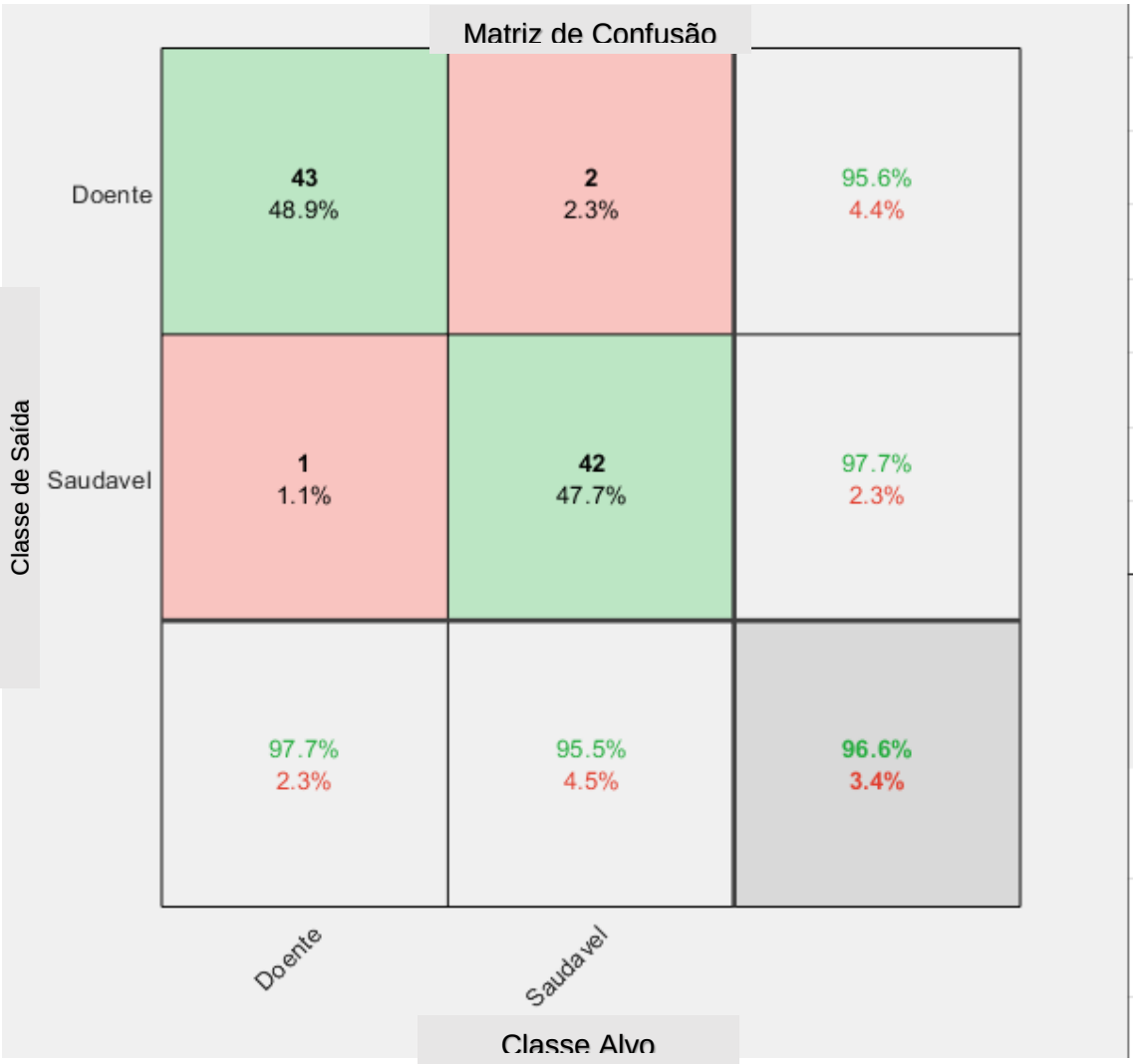
Figura 27 – Matriz de Confusão ResNet-18 – 10 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 28 temos a matriz de confusão com 20 épocas com 45 classificações como sendo classe 1, em 95,6% das vezes a predição estava correta (43 vezes) e em 4,4% estavam erradas. Para a classe 2, das 43 amostras classificadas como pertencentes da classe 2, em 97,7% ele acerta e 2,3% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 96,6%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 97,7% e 95,5%.

Figura 28 – Matriz de Confusão ResNet-18 – 20 épocas



Fonte: Elaborado pelo Autor. (2020).

3.4 VGG-19

Na Tabela 7 é apresentado o resultado dos dados de teste da arquitetura VGG-19. As épocas utilizadas foram 3, 5, 10 e 20.

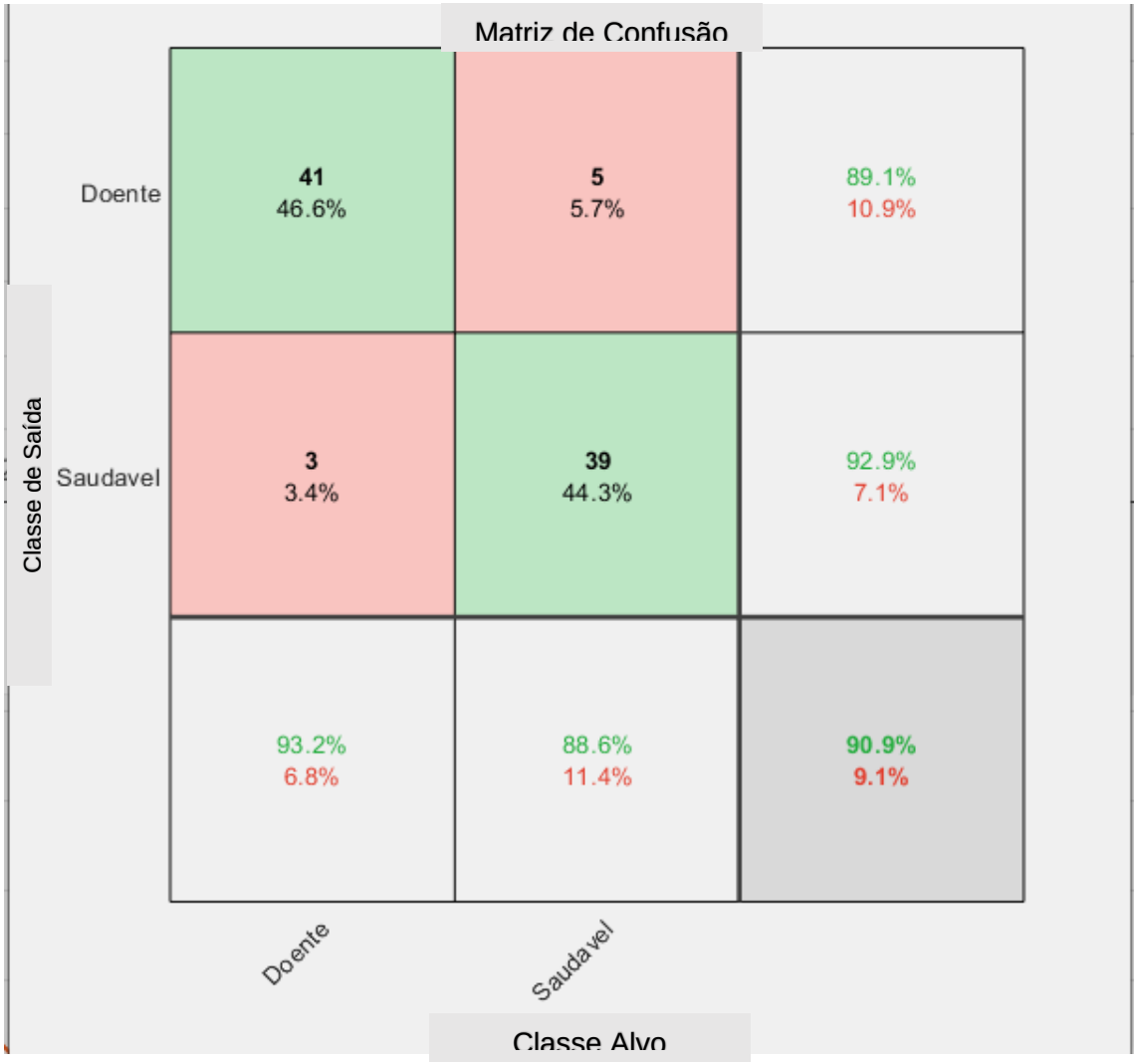
Tabela 7 – Resultado VGG-19 – dados de teste

Número de épocas	Acurácia	Sensibilidade	Especificidade
3	90,9%	93,2%	88,6%
5	94,4%	93,2%	95,6%
10	92,8%	95,2%	90,4%
20	**96,6%**	**95,5%**	**97,7%**

Fonte: Elaborado pelo Autor. (2020).

Na Figura 29 temos a matriz de confusão com 3 épocas com 46 classificações como sendo classe 1, em 89,1% das vezes a predição estava correta (41 vezes) e em 10,9% estavam erradas. Para a classe 2, das 42 amostras classificadas como pertencentes da classe 2, em 92,9% ele acerta e 7,1% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 90,9%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 93,2% e 88,6%.

Figura 29 – Matriz de Confusão VGG-19 – 3 épocas

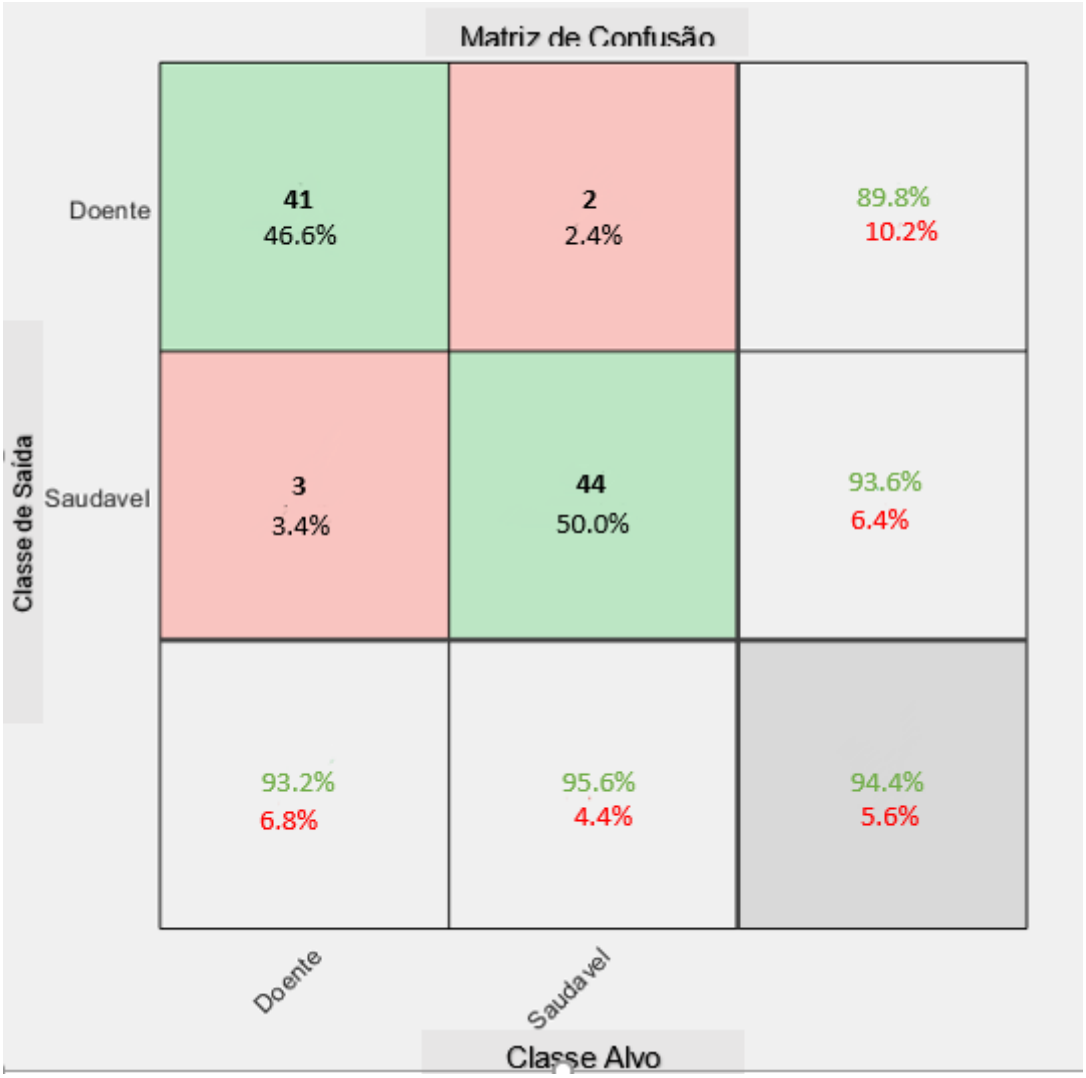


Fonte: Elaborado pelo Autor. (2020).

Na Figura 30 temos a matriz de confusão com 5 épocas com 41 classificações como sendo classe 1, em 89.8% das vezes a predição estava correta (41 vezes) e em 10.2% estavam erradas. Para a classe 2, das 47 amostras classificadas como pertencentes da classe 2, em 93,6% ele acerta e 6,4% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 94,4%. A

sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 93,2% e 95,6%.

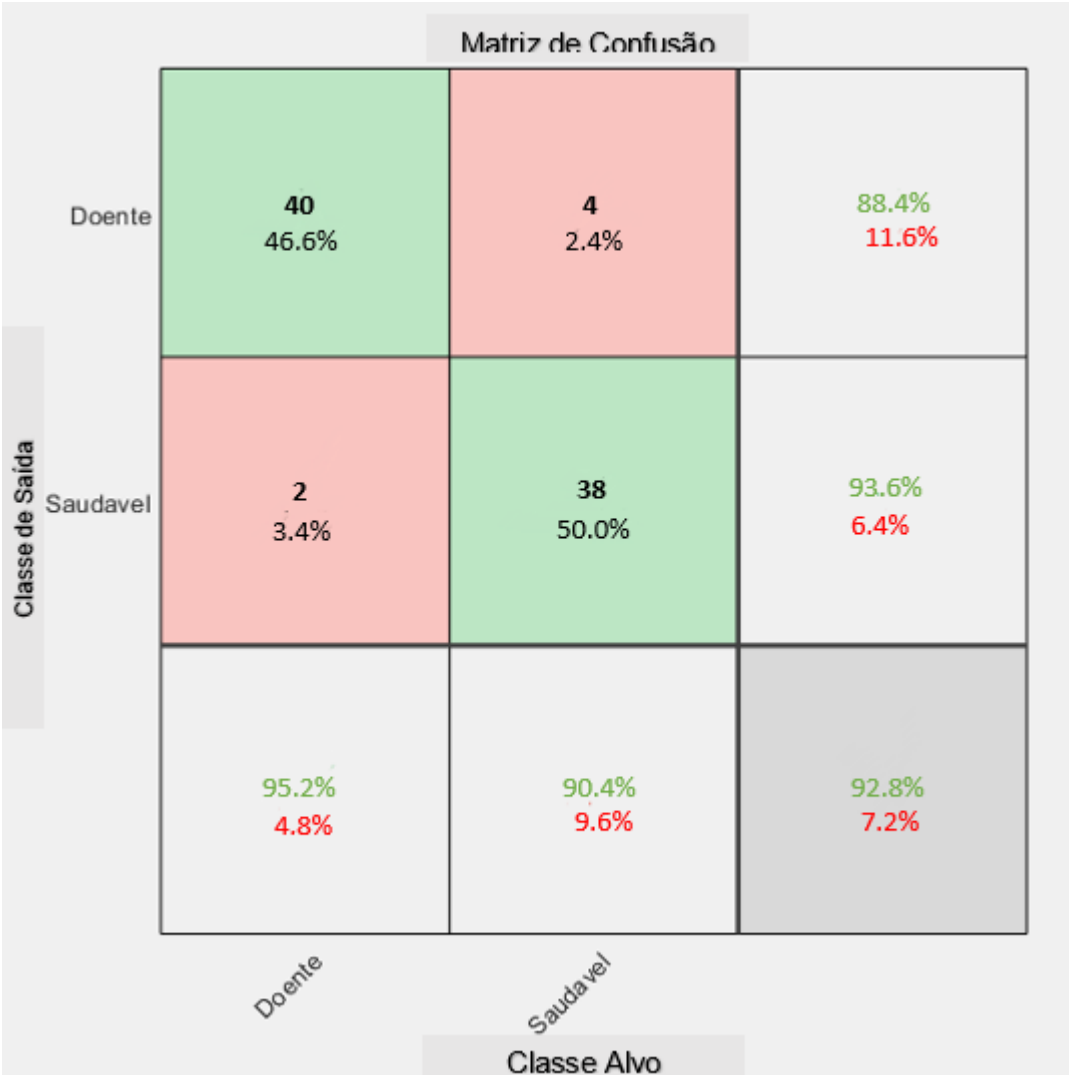
Figura 30 – Matriz de Confusão VGG-19 – 5 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 31 temos a matriz de confusão com 10 épocas com 44 classificações como sendo classe 1, em 88,4% das vezes a predição estava correta (40 vezes) e em 11,6% estavam erradas. Para a classe 2, das 40 amostras classificadas como pertencentes da classe 2, em 93,6% ele acerta e 6,4% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 92,8%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 95,2% e 90,4%.

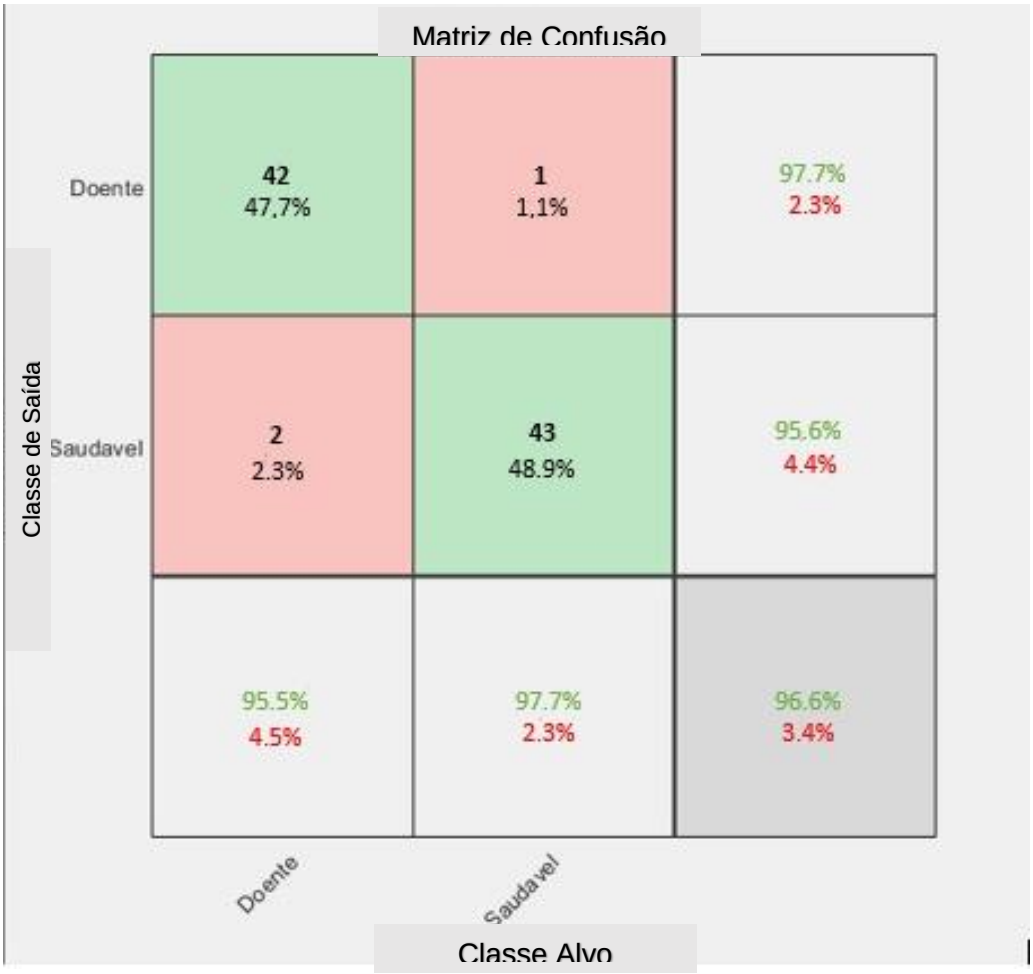
Figura 31 – Matriz de Confusão VGG-19 – 10 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Figura 32 temos a matriz de confusão com 20 épocas com 43 classificações como sendo classe 1, em 97,7% das vezes a predição estava correta (42 vezes) e em 2,3% estavam erradas. Para a classe 2, das 45 amostras classificadas como pertencentes da classe 2, em 95,6% ele acerta e 4,4% ele erra. Também existe a informação da acurácia localizada na última linha e na última coluna: 96,6%. A sensibilidade e a especificidade podem ser visualizadas na última linha da matriz de confusão, com os valores de 95,5% e 97,7%.

Figura 32 – Matriz de Confusão VGG-19 – 20 épocas



Fonte: Elaborado pelo Autor. (2020).

Na Tabela 8 podemos observar a comparação de todas as arquiteturas citadas e testadas com 20 épocas:

Tabela 8 – Resultado das arquiteturas com 20 épocas – dados de teste

Arquitetura/Número de Épocas	Acurácia	Sensibilidade	Especificidade
AlexNet / 20 épocas	93,2%	90,9%	95,5%
GoogleNet/ 20 épocas	92,1%	86,3%	97,7%
ResNet-18 / 20 épocas	**96,6%**	**97,7%**	**95,5%**
VGG-19 / 20 épocas	**96,6%**	**95,5%**	**97,7%**

Fonte: Elaborado pelo Autor. (2020).

4. Conclusão

Foi utilizado neste trabalho técnicas de arquiteturas populares de redes neurais convulsionais, como AlexNet, GoogleNet, ResNet-18 e VGG-19, com o objetivo de classificar as imagens nas classes de portadora ou não da doença (ferrugem). A base de dados utilizada possui imagens de 12 tipos de plantas, totalizando 440 imagens, nas quais metade são rotuladas como doentes (portadoras do fungo ferrugem) e a outra metade saudável, divididas entres imagens de treinamento e validação e teste, na proporção de 400 por 40 (400/40).

Foi utilizado transferência de aprendizagem nos experimentos, treinando principalmente as redes completamente conectadas, alterando o número de épocas durantes seu treinamento entre 3, 5, 10 e 20.

Os melhores resultados obtidos foram utilizados a RNC Resnet-18 e a VGG-19, ambas com treinamento de 20 épocas. A Resnet-18 atingiu uma acurácia de 96,6%, sensibilidade de 97,7% e especificidade de 95,5%, enquanto a VGG-19 atingiu uma acurácia de 96,6%, sensibilidade de 95,5% e especificidade de 97,7%. Os resultados são positivos e gratificante, pois indicam que a utilização de redes neurais convolucionais em imagens para diagnóstico de patologia fúngica no caso ferrugem, possuem resultados positivos e pode ser objeto de estudos mais profundos para atacar o problema.

Em pesquisas futuras, podemos utilizar uma base de dados maior para que possamos melhorar o treinamento da rede e também um estudo mais detalhado sobre a designação de hiperparâmetros e arquiteturas de redes neurais convolucionais usando o algoritmo genéticos ou outras técnicas.

5. Referências

ALEXNET **Layers Matlab**. 2017. <https://www.mathworks.com/help/deeplearning/ref/alexnet.html>. Acesso em : 07 de agos. 2020.

ALVES, GISELY. 2018. **Entendendo Redes Convolucionais (CNNs)** < https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184>. Acesso em: 30 de set. 2020.

ANWAR, AQEEL. 2019. **Difference between AlexNet, VGGNet, ResNet, and Inception** < https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96>. Acesso em: 18 de set. 2020.

ATTUX, R. R. F.; ATTUX, R. R. F. **Perceptron de Múltiplas Camadas** *. 1969. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia353_1s07/topico5_07.pdf>.

BARATLOO, A. et al. **Part 1: simple definition and calculation of accuracy, sensitivity and specificity. Emergency**, v. 3, n. 2, p. 48-49, 2015.

CASTRO, F. D.; CASTRO, M. D. **Redes neurais artificiais**. Porto Alegre, RS: Pontifícia Universidade Católica do Rio Grande do Sul, 2001.

CONFUSION Matrix. 2018. <https://www.mathworks.com/help/deeplearning/ref/plotconfusion.html>. Acesso em: 06 de agos. 2020.

CONFUSIONMAT. 2020. < https://www.mathworks.com/help/stats/confusionmat.html>. Acesso em: 06 de agos. 2020

CONVOLUTIONAL **Neural Network (CNN).** 2013. <https://developer.nvidia.com/discover/convolutional-neural-network> Acesso em: 18 de set. 2020.

CONVOLUTIONAL **Neural Networks (CNN): Step 1- Convolution Operation**. 2018. <https://www.superdatascience.com/blogs/ convolutional-neural-networks-cnn-step 1-convolution-operation>. Acesso em: 09 de set. 2020.

GONZALES, Rafael C; WOODS, Richard E. **Digital Image Processing Using MATLAB**. 2. ed. New York: Gatesmark Publishing, 2009. 827 p.

GONZALES, Rafael C; WOODS, Richard E. **Digital Image Processing**. 3. ed. New York: Prentice Hall, 2007. 793 p.

GOULART, IVES. 2013. **Ferrugem**< https://www.jardineiro.net/pragas/ferrugem.html> Acesso em: 09 de set. 2020.

HAYKIN, S. **Redes Neurais, princípios e práticas**. Porto Alegre: Bookman, 2001.

JAIN, A. K, MAO, J., MOHIUDDIN, K.M. **Artificial neural networks: a tutorial**. IEEE Computer, v. 29, n. 3, p. 56-63, 1996.

PL@NTNET. **Identify, explore and share your observations of wild plants**

<https://identify.plantnet.org/>. Acesso em: 15 de set. 2020.

SOUZA, A. FERNANDO. 2007. **Estratégias utilizadas para o controle químico da ferrugem do cafeeiro.** <https://www.cafepoint.com.br/noticias/tecnicas-de-producao/estrategias-utilizadas-para-o-controle-quimico-da-ferrugem-do-cafeeiro-33706n.aspx> Acesso em: 19 de out.2020.

VASCONCELLOS, PAULO. 2018. **Como saber se seu modelo de Machine Learning está funcionando mesmo** < https://paulovasconcellos.com.br/como-saber-se-seu-modelo-de-machine-learning-est%C3%A1-funcionando-mesmo-a5892f6468b>. Acesso em: 15 de set. 2020.

APÊNDICE A – CÓDIGO FONTE

```
 edit( fullfile(matlabroot,'examples','nnet','main','findLayersToReplace.m'))

myfolder = 'D:\Faculdade\TCC2\Base_Imagens\Treinamento'
imds = imageDatastore( myfolder,'IncludeSubfolders',true , 'LabelSource' ,
'foldernames' ) ;

[imdsTrain,imdsValidation] = splitEachLabel(imds,0.8,'randomized');

% Para testar com outra rede , basta trocar ' alexnet ' por:
% googlenet , resnet18 , vgg19

net = alexnet;

inputSize = net.Layers (1) .InputSize;

if isa(net,'SeriesNetwork')
  lgraph = layerGraph ( net . Layers );
else
  lgraph = layerGraph(net) ;
end

[ learnableLayer , classLayer ] = findLayersToReplace( lgraph) ;
[ learnableLayer , classLayer ]

numClasses = numel( categories ( imdsTrain.Labels ) ) ;

if isa ( learnableLayer , 'nnet.cnn.layer.FullyConnectedLayer' )
  newLearnableLayer = fullyConnectedLayer ( numClasses , 'Name' , 'new_fc' ,
'WeightLearnRateFactor' , 10 , 'BiasLearnRateFactor' ,10) ;

elseif isa ( learnableLayer , 'nnet.cnn.layer.Convolution2DLayer' )
  newLearnableLayer = convolution2dLayer (1 , numClasses , 'Name' , 'new_conv ' ,
'WeightLearnRateFactor' , 10 , 'BiasLearnRateFactor' ,10) ;
end

lgraph = replaceLayer(lgraph , learnableLayer.Name,newLearnableLayer );

newClassLayer = classificationLayer( 'Name' , 'new_classoutput');
lgraph = replaceLayer( lgraph , classLayer.Name, newClassLayer );

%Rede esta pronta para ser retreinada no novo conjunto de imagens.
%Extraimos as camadas e conexoes do grafico de camadas e selecionamos quais
%camadas congelar. As 10 camadas formam a haste inicial da rede, e a funcao
%freezeWeight define as taxas de aprendizagem como zero nas primeiras 10
%camadas, entao utiizamos a funcao createLgraphUsingConnections para
%reconectar as camadas na ordem original.
layers = lgraph.Layers;
connections = lgraph.Connections;
```

```
layers(1:10) = freezeWeights(layers(1:10));
lgraph = createLgraphUsingConnections(layers, connections);

%A rede requer imagens de entrada de tamanho 224 por 224 por 3, mas as imagens
%no armazenamento de dados de imagens têm tamanhos diferentes. Use um
%armazenamento de dados de imagem aumentada para redimensionar
automaticamente
%as imagens de treinamento. Especifique operações de aumento adicionais a
%serem executadas nas imagens de treinamento: gire aleatoriamente as imagens
%de treinamento ao longo do eixo vertical, traduza-as aleatoriamente em até
%30 pixels e dimensione-as em até 10% horizontal e verticalmente.
%O aumento de dados ajuda a evitar que a rede se ajuste demais e memorize os
detalhes exatos das imagens de treinamento.
pixelRange = [-30 30];
scaleRange = [0.9 1.1];
imageAugmenter = imageDataAugmenter('RandXReflection',
true,'RandXTranslation', pixelRange,'RandYTranslation', pixelRange,'RandXScale',
scaleRange,'RandYScale', scaleRange);
augimdsTrain = augmentedImageDatastore(inputSize(1:2), imdsTrain
,'DataAugmentation', imageAugmenter);

augimdsValidation = augmentedImageDatastore(inputSize (1:2) ,imdsValidation ) ;

%Especifique o número de épocas para treinar. Ao realizar a aprendizagem por
%transferência, você não precisa treinar por tantas épocas. Uma época é um
%ciclo de treinamento completo em todo o conjunto de dados de treinamento.
%Especifique o tamanho do minilote e os dados de validação. Calcule a precisão da
validação uma vez por época.
miniBatchSize = 10;
valFrequency = floor(numel(augimdsTrain.Files)/miniBatchSize);
options = trainingOptions('sgdm', 'MiniBatchSize' ,miniBatchSize,'MaxEpochs',5
,'InitialLearnRate',1e-4,'Shuffle','every-
epoch','ValidationData',augimdsValidation,'ValidationFrequency',valFrequency
,'Verbose', false ,'Plots','training-progress');

net = trainNetwork(augimdsTrain,lgraph, options);
[YPred,probs] = classify (net ,augimdsValidation);
accuracy = mean(YPred == imdsValidation . Labels)

plotconfusion(imdsValidation.Labels ,YPred)

oTestSet = imageDatastore('D:/Faculdade/TCC2/Base_Imagens/Validacao' ,
'IncludeSubfolders' ,true , 'LabelSource' , 'foldernames');
oAugTestSet = augmentedImageDatastore(inputSize (1:2) ,oTestSet);
YTestPred = classify ( net , oAugTestSet) ;
accuracy = mean(YTestPred == oTestSet . Labels)
plotconfusion(oTestSet . Labels ,YTestPred)
```

APÊNDICE B – TERMO DE AUTORIZAÇÃO DE PUBLICAÇÃO

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
GABINETE DO REITOR

Av. Universitária, 1069 ● Setor Universitário
Caixa Postal 86 ● CEP 74605-010
Goiânia ● Goiás ● Brasil
Fone: (62) 3946.1000
www.pucgoias.edu.br ● reitoria@pucgoias.edu.br

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O(A) estudante Wesley da Cunha Júnior
do Curso de Engenharia da Computação matrícula 2014.1.033.0175-6,
telefone: (62) 982855[illegible] e-mail wesley.da.cunha.junior@gmail.com na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Detecção de ferrugem de folhas vegetais por meio de imagens aplicando redes neurais convolucionais gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto (PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 08 de dezembro de 2020.

Assinatura do(s) autor(es): Wesley da Cunha Júnior

Nome completo do autor: Wesley da Cunha Júnior

Assinatura do professor-orientador: _____

Nome completo do professor-orientador: _____