CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

UNIDADE ARAXÁ

MARLY DA SILVA LEITE

IDENTIFICAÇÃO DE SINTOMAS DE DEFICIÊNCIAS MINERAIS E PRAGAS NA CULTURA CAFEEIRA UTILIZANDO VISÃO COMPUTACIONAL

ARAXÁ - MG

2018

MARLY DA SILVA LEITE

IDENTIFICAÇÃO DE SINTOMAS DE DEFICIÊNCIAS MINERAIS E PRAGAS NA CULTURA CAFEEIRA UTILIZANDO VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá, como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Automação Industrial.

Orientador: Prof. Me. Sérgio Luiz da Silva Pithan
Coorientador: Prof. Carlos Dias da Silva Júnior

ARAXÁ - MG

2018

MARLY DA SILVA LEITE

IDENTIFICAÇÃO DE SINTOMAS DE DEFICIÊNCIAS MINERAIS E PRAGAS NA CULTURA CAFEEIRA UTILIZANDO VISÃO COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao
Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá,
como requisito parcial para a obtenção do grau de Bacharel em
Engenharia de Automação Industrial.

Defesa: Araxá, 10 de dezembro de 2018.

BANCA AVALIADORA

_____
Me. SÉRGIO LUIZ DA SILVA PITHAN - Orientador
Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá

_____
Eng. CARLOS DIAS DA SILVA JÚNIOR- Coorientador
Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá

_____
Dra. ALINE FERNANDA BIANCO - Avaliadora Titular
Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá

_____
Dr. WANDERLEY ALVES PARREIRA - Avaliador Titular
Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá

_____
Dr. HENRIQUE JOSÉ AVELAR - Avaliador Suplente
Centro Federal de Educação Tecnológica de Minas Gerais - Unidade Araxá

DEDICATÓRIA

Dedico este trabalho a Deus
que foi minha fortaleza para conseguir cumprir esta missão.

AGRADECIMENTOS

Agradeço a Deus por sempre me amparar e dar forças para alcançar meus objetivos.

Ao meu orientador, professor Me. Sérgio Luiz Pithan, por todo conhecimento transmitido, paciência e confiança.

Ao meu coorientador, professor Carlos Dias da Silva Júnior, pelas contribuições e incentivo.

À professora, Dra. Jalmira Fiúza pela dedicação, pelo aprendizado e principalmente, pelo apoio emocional.

Ao CEFET-MG, por ter me proporcionado crescimento profissional e pessoal, especialmente, aos funcionários da biblioteca e do restaurante que me acolheram com muito carinho ao longo desses anos.

À Íris Lopes da Costa Avelar, pelo apoio e confiança.

Aos professores, pelo empenho e dedicação.

Aos meus colegas de curso pelo aprendizado, companheirismo e alegrias compartilhadas.

Aos meus pais, Jaqueline e Josias (in Memoriam), pelo amor e por sempre investirem e acreditarem em mim e lutar para que eu tivesse uma boa educação.

Ao meu esposo, Luiz Fernando, pela paciência, compreensão, carinho e amor.

À minha amiga, Fernanda Sousa, pela ajuda, atenção e companheirismo.

À equipe do PSF- Pão de Açúcar pela compreensão.

Agradeço a todos que contribuíram para que eu chegasse até aqui.

EPÍGRAFE

“Uma imagem vale mais que mil palavras.”

Autor Desconhecido

RESUMO

A visão computacional é a ciência que estuda técnicas para replicar a visão biológica através do desenvolvimento de *hardwares* e *softwares*, alcançando várias aplicações. Uma delas é a agricultura de precisão que usa tecnologia de ponta para garantir melhores condições para o manejo de culturas, podendo ser aplicada na diagnose visual utilizada para diagnosticar deficiência de elementos e incidência de pragas nas plantas. Esta pesquisa objetivou identificar deficiências minerais e pragas no cafeeiro utilizando técnicas de visão computacional e redes neurais convolucionais, valendo-se de um modelo da arquitetura AlexNet pré-treinada disponível no *software* MATLAB®. Para alcançar esse objetivo foram pesquisados conceitos de processamento digital de imagens e redes neurais artificiais, enfatizando as redes neurais convolucionais. Foram feitos ajustes na rede AlexNet pré-treinada e, objetivando classificar imagens das folhas e frutos do cafeeiro *in loco*, foi desenvolvido um classificador dinâmico. Para efeito de teste, também foi desenvolvido um classificador estático para imagens de um banco de dados. Em linhas gerais, realizou-se uma pesquisa bibliográfica em livros, artigos científicos e documentos eletrônicos; procedeu-se a aquisição de imagens de folhas e frutos do cafeeiro por meio de artigos e bancos de imagens via internet; efetuou-se a ampliação dos dados adquiridos com a aplicação das técnicas dos momentos invariantes de Hu; executou-se o retreinamento da rede neural convolucional AlexNet, utilizando MATLAB® e, testes com a rede neural retreinada para verificar seu desempenho utilizando os classificadores estáticos e dinâmico foram realizados. Analisando os resultados obtidos, constatou-se que a rede neural convolucional apresentou uma precisão de 83,76% para as imagens de teste com o classificador estático. Erros podem ter ocorrido devido à má qualidade das imagens, que em boa parte obtiveram correta classificação com o classificador dinâmico. Em suma, através do estudo realizado foi possível observar que as redes neurais convolucionais podem oferecer um aporte eficiente para a diagnose visual na cultura cafeeira.

Palavras-chave: Visão computacional. Redes neurais convolucionais. Diagnose visual. Cafeeiro.

ABSTRACT

Computer vision is a science that studies techniques to replicate a biological vision through hardwares and softwares development, reaching several applications. One of them is precision agriculture which uses the latest technology to ensure better conditions for crop management and can be applied in the visual diagnosis used to diagnose element deficiency and pest incidence in plants. This research aimed to identify mineral deficiencies and pests in coffee plants using computer vision techniques and convolutional neural networks, using a pretrained AlexNet architecture model available in MATLAB® software. In order to achieve this goal, concepts of digital image processing and artificial neural networks were researched, emphasizing convolutional neural networks. Adjustments were made to the pretrained AlexNet network and, so as to classify images of the coffee leaves and fruits *in loco*, a dynamic classifier was developed. For testing purposes, a static classifier for images of a database was also developed. In general terms, a bibliographic research was carried out in books, scientific articles and electronic documents; we proceeded to acquire images of coffee leaves and fruits through articles and banks of images via the Internet; the data acquired were amplified with the application of Hu invariant moments techniques; the re-training of the AlexNet convolutional neural network was performed using MATLAB® and tests with the retrained neural network to verify its performance using the static and dynamic classifiers were performed. Analyzing the obtained results, it was verified that the convolutional neural network presented an accuracy of 83.76% for the test images with the static classifier. Errors may have occurred due to the poor quality of the images, which for the most part obtained correct classification with the dynamic classifier. In summary, through the study, it was possible to observe that the convolutional neural networks can offer an efficient contribution to the visual diagnosis in the coffee culture.

Keywords: Computer vision. Convolutional neural networks. Visual diagnosis. Coffee.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS E TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CECAFÉ	- Conselho dos Exportadores de Café do Brasil
EMBRAPA	- Empresa Brasileira de Pesquisa e Agropecuária
MAPA	- Ministério da Agricultura, Pecuária e Abastecimento
GPU	- Unidade de Processamento Gráfico
RGB	- *Red, Green and Blue*
RNA	- Redes Neurais Artificias
RNC	- Redes Neurais Convolucionais

LISTA DE SÍMBOLOS

® Marca Registrada

™ Trade Mark (Marca Comercial)

SUMÁRIO

1 INTRODUÇÃO

Visão Computacional é a ciência que simula a visão humana através de modelagem de dados, *softwares* e *hardwares* e, a partir de dados extraídos automaticamente de imagens, utiliza a aprendizagem de máquina para fazer deduções e atuar na tomada de decisões. Esta ciência utiliza técnicas do ramo de Inteligência Artificial e suas principais funções são: reconhecimento de objetos, estimativa de movimento, reconstrução de cenas e restauração de imagens (COMSTOR AMERICAS, 2017). Para realizar essas funções são seguidos alguns passos fundamentais: obtenção de imagens, segmentação, extração de características, processamento de alto nível como o reconhecimento de padrões, entre outros (SOUZA, [200-]).

Uma das contribuições da visão computacional está na agricultura de precisão que utiliza alta tecnologia para garantir uma boa produção e lucratividade no agronegócio. Suas técnicas podem ser aplicadas na identificação de doenças e pragas e no diagnóstico do grau de lesão foliar causados por elas (BORTH et al., 2014). Nesse contexto, este trabalho busca identificar os sintomas de deficiências minerais e pragas na cultura cafeeira através da visão computacional.

O cafeeiro é uma cultura de grande importância para o setor agropecuário brasileiro. De acordo com a Revista Cafeicultura (2005), para uma boa produção de café são necessários muitos cuidados, a iniciar pelo viveiro de mudas. Dentre esses cuidados estão o controle de pragas e condições de adubação do solo (IFCE, 200-). O desenvolvimento e a produtividade do café limitam-se quando este é cultivado em um solo com quantidades insuficientes de minerais e a falta deles poderá ter seus sintomas refletidos visualmente na folha, que é o órgão onde ocorrem as atividades metabólicas das plantas (FAQUIN, 2002). Segundo Zúñiga (2012), os métodos de diagnose foliar (análises químicas) e diagnose visual das folhas são utilizados para fazer a avaliação nutricional em plantas. A diagnose visual está intimamente relacionada com a experiência do especialista que a realiza, podendo resultar em interpretações incorretas, fazendo-se necessário utilizar outras ferramentas para diagnóstico.

Vislumbrando uma possível solução este trabalho parte da hipótese de que se pode utilizar visão computacional e uma Rede Neural Convolucional (RNC) para auxiliar na diagnose visual, permitindo a redução do erro.

Segundo Szegedy et al [200-], as RNCs vêm apresentando ganhos importantes para a solução de problemas que envolvem a visão computacional, como segmentação, detecção de objetos, entre outros.

Diante disso, este trabalho tem como objetivo geral identificar deficiências minerais e pragas no cafeeiro utilizando técnicas de visão computacional e redes neurais convolucionais, usando um modelo da arquitetura AlexNet pré-treinada disponível na plataforma MATLAB®. Para tanto, foram estabelecidos os seguintes objetivos específicos: pesquisar conceitos sobre processamento e análise de imagens; assimilar conceitos básicos sobre Redes Neurais Artificiais, abordando as Redes Neurais Convolucionais; ajustar a Rede Neural Convolucional AlexNet pré-treinada para solucionar o problema de classificação das disfunções do cafeeiro; desenvolver um classificador estático para classificar imagens de um banco de dados e, desenvolver um classificador dinâmico para classificar imagens em tempo real, objetivando a identificação *in loco*.

Esta pesquisa se justifica pela dificuldade, na agricultura, em determinar os danos causados por pragas e deficiências minerais com precisão através de análise visual, impedindo a tomada de medidas para que possam ser minimizados. Diante disso, o uso de Redes Neurais Convolucionais pode apresentar um resultado expressivo para a classificação precisa de moléstias no cafeeiro, ajudando a minimizar perdas na produtividade.

Inicialmente, para o desenvolvimento do trabalho realizou-se uma pesquisa bibliográfica baseada em artigos científicos, livros, documentos e publicações eletrônicas. A abordagem utilizada para a coleta e processamento de dados, bem como para a análise de resultados foi a quantitativa.

Este trabalho estrutura-se em quatro capítulos, apresentando-se no primeiro um embasamento teórico sobre a importância do cafeeiro na economia brasileira, os sintomas dos principais desarranjos nutricionais e pragas que acometem essa cultura, conceitos de processamento digital de imagens, redes neurais artificiais e explicação sobre a arquitetura AlexNet. No segundo capítulo é explicitada a metodologia utilizada para a realização da parte experimental da pesquisa. No terceiro, apresenta-se a análise dos resultados e, no quarto as considerações finais a respeito do trabalho.

2 REFERENCIAL TEÓRICO

2.1 A Importância do Café na Economia Brasileira

O Brasil atua no agronegócio, em âmbito mundial, como o maior produtor, exportador e segundo maior consumidor de café, de acordo com o Ministério da Agricultura, Pecuária e Abastecimento (MAPA). Segundo o Conselho dos Exportadores de Café do Brasil (CECAFÉ), apesar de não ser mais o principal produto de exportação brasileiro, o café continua se destacando em vendas externas no segmento do agronegócio. Estima-se que o parque nacional cafeeiro seja 2,25 milhões de hectares, com o trabalho de 290 mil produtores, abrangendo, aproximadamente, 1900 municípios.

Duas principais espécies de café são cultivadas no país:

- Coffea Arabica: espécie responsável por 70% do café comercializado no mundo, consumido puro ou utilizado em *blends* (mistura de grãos) de alta qualidade.
- Coffea Canephora: também chamado de café robusta, é utilizado com maior frequência em *blends*, nas quais é misturado ao café arábica, compondo até 30% da sua composição. Como apresenta maior solubilidade e rendimento maior que o da outra espécie, após ser torrado, o *conilon* é fundamental na fabricação dos cafés solúveis (SOUZA, 2004, p. 11).

2.2 Métodos de Diagnóstico de Deficiências Nutricionais no Cafeeiro

A nutrição do cafeeiro é um fator de produção muito importante, pois esta cultura retira grandes quantidades de nutrientes do solo, sendo necessária a adubação correta para que a produtividade atinja os níveis desejados (VOLTOLINI; ALECRIM; SOUSA, 2015). A nutrição mineral do cafeeiro interfere diretamente na produção dos grãos, influenciando o lucro a ser obtido com a safra. Com uma nutrição mineral equilibrada, o cafeeiro estará menos suscetível ao desenvolvimento de doenças, pois compostos inibidores podem ser acumulados, dificultando a infiltração e instalação dos seus agentes causadores (REZENDE, 2015).

Assim sendo, é imprescindível uma avaliação nutricional do cafeeiro, sendo este procedimento importante para que seja identificada a carência de algum mineral que possa prejudicar o desenvolvimento e a produção do cultivo. Essa

avaliação geralmente é feita comparando-se uma planta ou amostras de uma população de plantas a uma amostra que seja considerada saudável. Essas amostras podem ser adquiridas através de experimentos controlados ou podem ser colhidas em campo (FAQUIN, 2002).

Ainda segundo Faquin (2002), existem vários métodos para fazer o diagnóstico nutricional, porém os que mais se destacam são a Diagnose Visual e a Diagnose Foliar (análise química do teor de nutrientes em determinadas folhas), devido à relevância e praticidade de suas aplicações. Em ambos os procedimentos, o principal órgão de análise é a folha, pois a maioria das atividades metabólicas de uma planta se realiza nela, fazendo com que qualquer alteração nutricional manifeste seu sintoma com maior frequência.

2.2.1 DIAGNOSE VISUAL

Para Rosseto e Santiago [20--], a Diagnose Visual identifica a falta ou o excesso de minerais observando-se aspectos como cor, formato e textura da folha de uma planta e identificar deficiências nutricionais pode auxiliar na correção de adubação, especialmente na adubação do próximo ciclo de plantio.

Esse método requer grande conhecimento técnico, pois deficiências nutricionais podem ser confundidas com doenças, danos causados por pragas, por agentes químicos, alterações causadas pelo sol, ventos frios, falta de umidade e condições físicas do solo como compactação e erosão. Para tanto, algumas indicações devem ser observadas visando diminuir quaisquer equívocos:

1) *Alastramento do sintoma*: O sintoma de origem nutricional aparece de uma forma geral em todas as plantas do terreno, não aparecendo isoladamente ou na parte mais densa do plantio.
2) *Características do sintoma*: Os sintomas nutricionais manifestam duas características que os diferenciam dos demais:

- *Simetria*: Sintomas de origem nutricional aparecem de modo simétrico na planta, entre folhas opostas ou próximas no galho e surge independente do lado exposto ao agente causador de agravos. Danos simétricos causados por exposição a esse agente aparecerão somente no lado que foi exposto, estando o outro livre de agressões.

- *Gradiente*: Considerando um mesmo galho, há uma diferença de coloração entre folhas novas e folhas velhas, causada pela redistribuição dos nutrientes na planta. Se o nutriente for móvel (N, P, K e Mg), em condições de falta desse nutriente, a planta fará o seu deslocamento para as folhas mais novas e para os frutos, causando o aparecimento dos sintomas da deficiência nutricional nas folhas velhas. Para nutrientes pouco móveis (S, Cu, Fe, Mn, Zn e Mo) e imóveis (B e Ca), sucede o contrário e os sintomas aparecem nas folhas mais novas (FAQUIN, 2002).

2.3 Sintomas Visuais de Deficiências Minerais no Cafeeiro

Para Faquin (2002), cada mineral provoca um sintoma característico quando está em deficiência na planta. Esses efeitos podem variar de acordo com a espécie. São apresentados a seguir, os indícios de deficiência dos minerais essenciais no ciclo de vida da cultura cafeeira:

- Nitrogênio: Segundo Mesquita et al (2016), por ser um nutriente móvel no cafeeiro, os sintomas de sua deficiência se manifestam nas folhas velhas. A insuficiência de nitrogênio apresenta-se através de clorose em toda a superfície da folha, conforme demonstrado na FIG. 1.

FIGURA 1 - Deficiência de nitrogênio.

Fonte: MESQUITA et al (2016, p. 14).

- Fósforo: Os sintomas de insuficiência de fósforo também irão ocorrer nas folhas velhas do cafeeiro, inicialmente, devido à mobilidade desse nutriente. A deficiência desse mineral apresenta-se pela alteração gradativa da coloração na ponta e no centro da folha. A princípio, observa-se um amarelecimento,

avançando para um amarelo rosado, até chegar ao vermelho escuro e marrom violeta. A deficiência de fósforo manifesta-se pela necrose no formato de “V” invertido no extremo da folha, por vezes em formato assimétrico, que pode atingir toda a sua superfície (MESQUITA et al, 2016).

FIGURA 2 - Deficiência de fósforo.

Fonte: MESQUITA et al (2016, p. 15).

- Potássio: Por também ser um nutriente móvel na planta, seus sintomas irão surgir, primeiramente, nas folhas velhas. O sintoma de sua deficiência apresenta-se, a princípio, com o amarelecimento das extremidades das folhas, que, em seguida secam apresentando tons escuros. Circunjacente à área com necrose observa-se um contorno amarelo (MESQUITA et al, 2016).

FIGURA 3 - Deficiência de potássio.

Fonte: MESQUITA et al (2016, p. 16).

- Cálcio: Como é um nutriente imóvel no cafeeiro, os seus sintomas de deficiência irão aparecer nas folhas novas da planta. Quando em carência, ocorre a clorose

na borda da folha, que pode avançar entre as nervuras alcançando o centro, porém estas e seus contornos conservam-se verdes (MESQUITA et al, 2016).

FIGURA 4 - Deficiência de cálcio.

Fonte: MESQUITA et al (2016, p. 17).

- Magnésio: É um mineral móvel no cafeeiro, portanto, no início, seus sintomas irão surgir nas folhas mais velhas. Apresenta-se com um amarelecimento somente entre as nervuras da folha podendo atingir um tom marrom à medida que a deficiência avança (MESQUITA et al, 2016).

FIGURA 5 - Deficiência de magnésio.

Fonte: MESQUITA et al (2016, p. 18).

2.4. Incidência de Pragas na Cultura Cafeeira

De acordo com a Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA), as pragas constituem outro fator limitante para o desenvolvimento e a produção do cafeeiro. Elas devem ser examinadas para que sejam contidas, evitando maiores prejuízos no plantio. Destacam-se as principais pragas que acometem o cafeeiro influenciando na economia:

2.4.1 BROCA-DO-CAFÉ

A broca é causada por um pequeno besouro (*Hypothenemus hampei*) que ataca os frutos do cafeeiro perfurando sua coroa e depositando suas larvas nas sementes. Ele pode atacar os frutos do cafeeiro em seus diversos estágios de desenvolvimento, diminuindo a quantidade de frutos saudáveis e aumentando o número de frutos brocados e causando, também, a queda destes (AGROBYTE, [200-]).

FIGURA 6 - Broca-do-café.

Fonte: LEITE e MOTA ([200-]).

2.4.2 BICHO MINEIRO

Essa praga é causada por uma mariposa *(Leucoptera coffeella)* que põe seus ovos na folha do cafeeiro no período noturno, ficando na parte inferior delas no

período diurno. Após a desova surgem larvas que penetram na folha e se instalam entre as suas epidermes, ocasionando a destruição do tecido foliar, consequentemente levando à redução da área fotossintética e à queda das folhas. Em decorrência desse ataque, a folha apresenta áreas ressecadas de cor marrom-avermelhada (BIOCONTROLE, [20--]).

FIGURA 7 - Bicho Mineiro.

Fonte: Revista Attalea Agronegócios (2018).

2.4.3 FERRUGEM DO CAFÉ

É uma praga provocada pelo fungo *Hemileia vastatrix*, que prejudica o cafeeiro causando a queda prematura das folhas, secando os ramos e influenciando na perda de frutos no ciclo de plantio posterior. Essa doença reflete seus sintomas na folha, que apresenta manchas amarelo-alaranjadas com aparência bolorenta (AGROLINK, 2017).

FIGURA 8 - Ferrugem do café.

Fonte: AGROLINK (2017).

2.4.4 CERCOSPORIOSE

A cercosporiose é causada pelo fungo *Cercospora caffeicola*, igualmente conhecida por mancha-de-olho-pardo. Causa lesões nas folhas e frutos de formato parecido a um círculo, com coloração amarela clara e cor branca-acinzentada na parte central (MATIELLO, 2018).

FIGURA 9 - Cercosporiose.

Fonte: AGROLINK (2017).

2.4.5 MANCHA DE PHOMA

A mancha de phoma é causada pelo fungo do gênero *Phoma* que abrange mais de 2000 espécies. Esta praga incide nas folhas, causando sua queda e nas flores e ramos do cafeeiro, secando suas terminações. O fungo deixa lesões escuras onde se encontra, usualmente, nas margens da folha, impossibilitando o crescimento nessa área e deixando a folha torcida (SOUZA, 2007).

FIGURA 10 - Mancha de Phoma.

Fonte: SOUZA (2007).

2.5 Processamento Digital de Imagens

Segundo Gonzalez e Woods (2010), pode-se interpretar uma imagem como uma função bidimensional *f(x,y)*, onde *x* e *y* representam coordenadas espaciais no plano cartesiano, enquanto a amplitude de *f* para esses pares de coordenadas corresponde à escala de cinza da imagem. As imagens também podem ter dimensões maiores, como as coloridas *Red*, *Green and Blue* (RGB) e as multiespectrais. Uma imagem digital é formada por uma quantidade finita e discreta de elementos da função *f*, os quais ocupam um lugar exclusivo, possuindo valor distintos e são denominados *pixels*.

A amostragem é o processo no qual é feita a discretização dos valores contínuos das coordenadas *x* e *y* da função *f*. Já a quantização é o processo de discretização dos valores contínuos de amplitude dessa função. Ao se digitalizar uma imagem, é possível representar seus valores inteiros na forma de uma matriz bidimensional para monocromáticas ou tridimensionais no caso das coloridas, para que possa ser processada e utilizada na elaboração de algoritmos (GONZÁLEZ e WOODS, 2010).

FIGURA 11 - Imagem em escala de cinza.

Fonte: PIXABAY (2018).

FIGURA 12 - Imagem RGB.

Fonte: MESQUITA et al (2016, p. 17).

O processamento digital de imagens (PDI) é aplicado visando melhorar o aspecto visual, suavizando imperfeições, tornando a imagem melhor compreensível para o analista humano e favorecendo a posterior extração de informações (características) (BRYS, [20--]).

Ainda de acordo com Brys [20--], existem inúmeras formas para a manipulação de imagens, porém, alguns procedimentos podem ser utilizados de uma maneira geral, como a retificação e restauração, realce e classificação. A retificação e o realce são utilizados para minimizar distorções e eliminar ruídos, melhorando a representação da imagem. O realce visa melhorar ainda mais a qualidade da imagem e a classificação serve para identificar regiões que a compõe, atribuindo-lhes uma classe, substituindo a análise visual pela análise automática.

Segundo UFF [20--], os filtros são aplicados nas imagens objetivando realçar suas particularidades e podem operar no domínio da frequência (transformada de Fourier) ou espacial. A filtragem no domínio espacial utiliza máscaras de convolução (filtros espaciais) agindo pontualmente nos *pixels* da imagem. A operação de convolução pode ser representada pela Eq.(1):

$$g[x,y] = \sum_{i=1}^{n}\sum_{j=1}^{m} f[x-i, y-j]h[i,j] \quad (1)$$

Em que $g[x,y]$representa a imagem convoluída, $f[x,y]$é a imagem original e $h[i,j]$, o filtro de convolução.

Os filtros lineares realizam uma soma ponderada, considerando a vizinhança de um *pixel* para alterar a intensidade do *pixel* de saída. Filtros não lineares não utilizam esse tipo de operação. Como exemplo de filtros espaciais temos os filtros Passa Baixa, que suavizam as imagens reduzindo a alta frequência, consequentemente, diminuindo ruídos. São representados pelos filtros de média, mediana, de ordem, moda, entre outros (UFF, 20--).

Para fazer a classificação, podem-se aplicar técnicas de segmentação de imagens que consiste em separar objetos ou regiões específicas, colocando-os em primeiro plano e deixando o restante dos elementos que a compõe em segundo plano. Esse processo apresenta-se de grande importância, pois é o primeiro passo para que outras etapas dos sistemas de visão artificial possam ser executadas. Esse método busca relacionar um pixel com seus vizinhos e outros na imagem para que sejam separados em uma mesma região. O processo de segmentação possui, basicamente, duas abordagens:

a) Descontinuidade: busca detectar bordas para separar regiões. Essa abordagem trabalha com *pixels* que possuem grandes diferenças entre si. É aplicada principalmente em imagens em escala de cinza.
b) Similaridade: busca separar regiões agrupando *pixels* que tenham características semelhantes, como na limiarização e crescimento de regiões (GONZALEZ e WOODS, 2010).

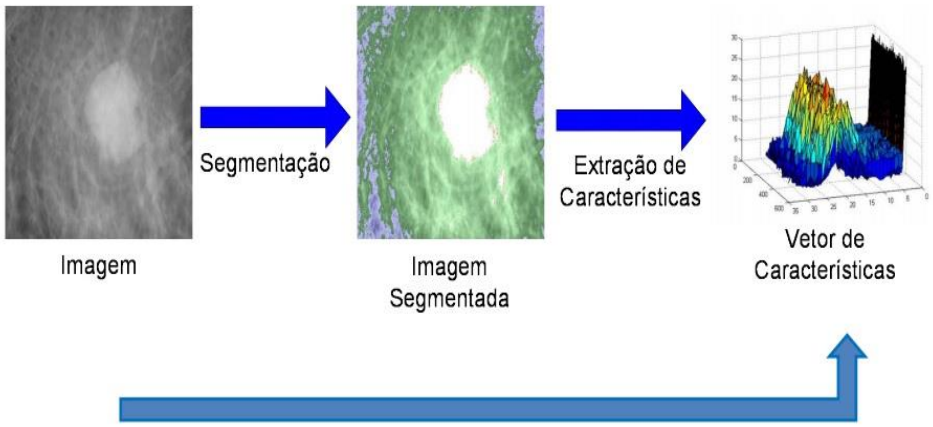
De acordo com Dias, Soares e Fonseca (2011), os filtros mais relevantes para detecção de bordas em imagens são os filtros de Roberts, Sobel, Prewitt, Canny e Laplaciano.

FIGURA 13 - Exemplo de aplicação do filtro Sobel.

Fonte: Willer Júnior et al. [20--].

Segundo USP [20--], é necessário representar as características extraídas de uma imagem para que possam ser processadas de forma eficiente por algoritmos de visão artificial. Essas características podem ser representadas por vetores (*feature vectors*), comparados a um parâmetro de semelhança. Esses vetores podem representar cores (histogramas), textura (descritores de Haralick), forma de objetos (Momentos Invariantes de Hu), entre outros.

FIGURA 14 - Processo de extração de característica de imagens.



Fonte: USP [20--].

2.6 Momentos Invariantes

Os momentos invariantes de Hu constituem um método bastante utilizado na extração de características de imagens, possibilitando o reconhecimento de um objeto mesmo quando sua imagem sofre alterações quanto à translação, escala e rotação (CORSO; [20--]).

Para Solomon e Breckon (2013) os momentos invariantes de Hu são definidos a partir dos momentos centrais e momentos centrais normalizados. Na teoria elementar de probabilidade, o n-ésimo momento pode ser calculado a partir da Eq. (2):

$$m_n = \int_{-\infty}^{\infty} x^n \, p(x) dx \qquad (2)$$

Em que:

- m_n representa o *n*-ésimo momento da função densidade de probabilidade para a variável aleatória;
- *x* é variável aleatória;
- *p(x)* é a função densidade de probabilidade.

Podem-se definir os momentos centrais através do cálculo da variação em torno do valor médio da variável aleatória, portanto:

$$M_n = \int_{-\infty}^{\infty} (x - \bar{x})^n \, p(x) dx \tag{3}$$

Em que:

- M_n representa o momento central da função densidade de probabilidade para a variável aleatória;
- *x* é variável aleatória;
- *p(x)* é a função densidade de probabilidade.

Aplicando o conceito de momento em funções 2-D, o *pq*-ésimo momento para as variáveis *x* e *y* de uma função densidade é definido por:

$$m_{pq} = \int_{-\infty}^{\infty} x^p y^q p(x,y) dx dy \tag{4}$$

De acordo com González e Woods (2010), para calcular momentos de uma imagem, substitui-se a função densidade por uma imagem f(x,y) de tamanho M x N:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q \, f(x,y) dx dy \tag{5}$$

O pq-ésimo momento normalizado é determinado por:

$$\eta_{pq} = \frac{M_{pq}}{M_{00}^{\beta}} \; em \; que \; \beta = \frac{p+q}{2} + 1 \; e \; p + q \geq 2 \tag{6}$$

Conhecendo-se os momentos centrais, um conjunto de sete momentos pode ser derivado, conhecido como momentos de Hu:

$$\Lambda_1 = \eta_{20} + \eta_{02} \tag{7}$$

$$\Lambda_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{8}$$

$$\Lambda_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{9}$$

$$\Lambda_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{30})^2 \tag{10}$$

$$\begin{aligned}\Lambda_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} - \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{03} + \eta_{21}) \times [3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]\end{aligned} \tag{11}$$

$$\begin{aligned}\Lambda_6 = {} & (\eta_{20} - \eta_{02})[(\eta_{12} + \eta_{30})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{21} + \eta_{03})(\eta_{12} + \eta_{30})\end{aligned} \tag{12}$$

$$\begin{aligned}\Lambda_7 = {} & 3(\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2] + (3\eta_{21} \\ & - \eta_{30})(\eta_{21} + \eta_{03}) \times [3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2]\end{aligned} \tag{13}$$

2.7 Redes Neurais Artificiais

Para Vellasco (2007), uma Rede Neural Artificial (RNA) é um conjunto de vários elementos de processamento que representam, matematicamente, neurônios de um cérebro biológico. Ela tem essa denominação por pretender se igualar ao cérebro humano, sendo capaz de identificar, verificar e classificar dados.

A aprendizagem da RNA acontece através da combinação de pesos que armazenam o conhecimento adquirido ao serem apresentados a exemplos (padrões). Nesse processo, os pesos são associados a não linearidade tornando a RNA capaz de resolver problemas complexos. A partir disso, por meio de um algoritmo de treinamento, o aprendizado é adquirido pela rede através da atualização dos pesos, na execução do mesmo, objetivando atingir um determinado resultado (VELLASCO, 2007).

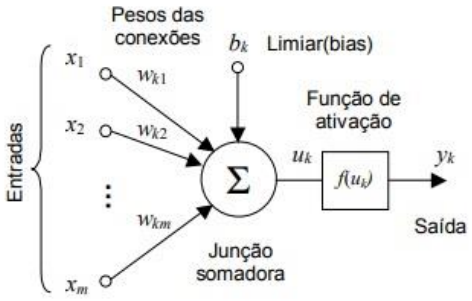
Ainda segundo Vellasco (2007), as Redes Neurais atuam especialmente na classificação de padrões e previsões, possuindo capacidade de generalização,

no entanto, é importante ressaltar que é preciso ter um conjunto de dados suficiente e relevante, para que uma Rede Neural apresente bons resultados.

2.7.1 NEURÔNIO ARTIFICIAL

A unidade básica de uma Rede Neural é o neurônio artificial (VELLASCO, 2007). A FIG. 15 mostra a representação do funcionamento de um neurônio artificial.

FIGURA 15 - Representação de um neurônio artificial.



Fonte: CASTRO e ZUBEN (1998).

Para Castro e Zuben (1998) os neurônios artificiais genéricos possuem três partes elementares:

• As conexões ou sinapses, representadas pelos pesos (w_{k1}, w_{k2}... w_{km}).

• Um somador, que pode ser caracterizado pela Eq. (14):

$$u_k = \sum_{j}^{m} w_{kj}\, x_j + b_k \tag{14}$$

Em que u representa a junção somadora da multiplicação entre as entradas x e os pesos w atribuídos, que faz uma combinação linear com o limiar *bias* b_k. O *bias* diminui ou aumenta a ação do valor da entrada na função de ativação. Isto é, a soma produz um nível de atividade que produzirá uma resposta de saída sempre que um limiar for ultrapassado.

• A função de ativação:

$$y_k = f(u_k) \tag{15}$$

Sendo y a saída do neurônio e $f(u_k)$, a função de ativação. Essa função insere não-linearidade à rede neural e limita a saída do neurônio. Os principais tipos de função de ativação são:

a) Função sigmóide ou logística: apresenta valores positivos (VELLASCO, 2007) e é representada matematicamente por:

$$y_k = \frac{1}{1 + e^{-u_k}} \tag{16}$$

Onde$e$ equivale ao número de Euler (2.718281).

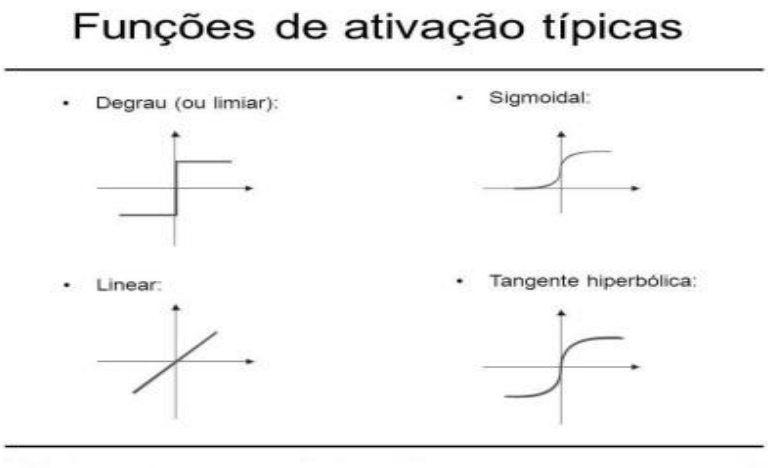
b) Função tangente hiperbólica: a resposta pode apresentar valores negativos e positivos, assumindo valores entre -1 e 1 (SÉRGIO NETO, 2017). É representada por:

$$y_k = \frac{1 - e^{u_k}}{1 + e^{u_k}} \tag{17}$$

c) Função linear: geralmente utilizada em neurônios de saída em que não é pretendido aplicar a saturação das funções sigmoide e tangente hiperbólica (VELLASCO, 2007).

d) Função degrau: se o valor calculado for maior que zero, a função retornará o valor 1, senão retornará -1 (SÉRGIO NETO, 2017)

FIGURA 16 - Funções de ativação.



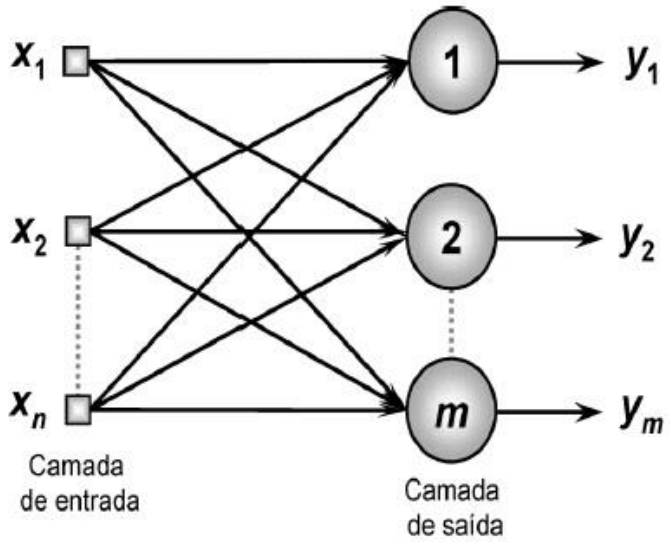
Fonte: Simas (2013) apud Sérgio Neto (2017, p. 34).

2.7.2 ARQUITETURAS DE REDE

Segundo Castro e Zuben (1998), são três os principais tipos de arquitetura para as Redes Neurais Artificiais:

•Redes *Feedforward* com Uma Única Camada: é uma rede simples (*perceptron*), composta apenas por uma camada de entrada e uma camada de saída. Possui essa denominação, pois a propagação dos dados acontece sempre da entrada para a saída, como é possível verificar na FIG. 17:

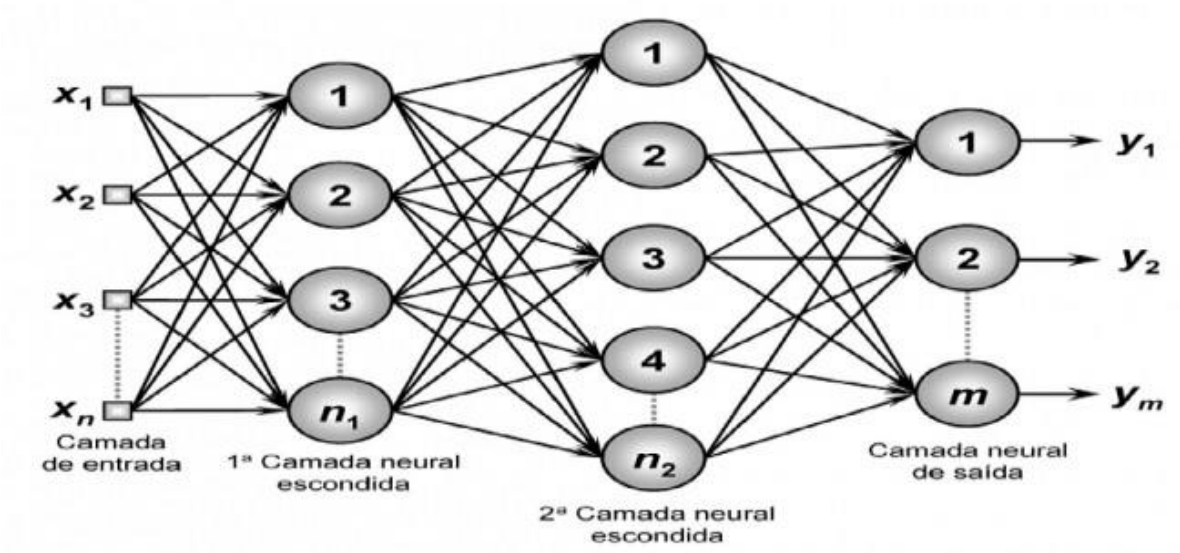
FIGURA 17 - Rede Feedforward de Camada Única.



Fonte: Palmiere (2016).

•Rede *Feedforward* com Múltiplas Camadas: é uma rede que pode possuir mais de uma camada intermediária ou escondida. Como é possível observar na FIG.18, essas camadas estão interconectadas, sendo a saída de uma camada, a entrada da camada posterior. Este tipo de rede permite o aumento da capacidade de processamento por possuir camadas intermediárias não-lineares. Utiliza-se a retroprapagação (*backpropagation*) do erro entre a saída da rede e uma saída esperada para o ajuste de parâmetros.

FIGURA 18 - Rede Feedforward com Múltiplas Camadas.



Fonte: Palmiere (2016).

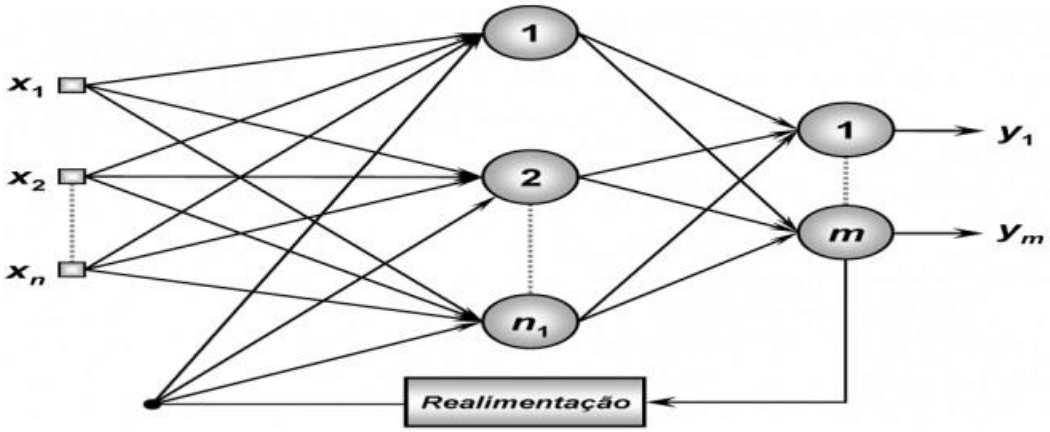
- Função Softmax: é utilizada na camada de saída onde ocorre a classificação. O objetivo desta função é apresentar a probabilidade de pertencimento de uma entrada a uma determinada classe, portanto, a classe indicada como correta pela rede é a que possui probabilidade de maior valor. A função Softmax assume apenas valores entre 0 e 1 (MIYAZAKI, 2017).

$$y_i = \frac{ez_i}{\sum_j e^{z_j}} \tag{18}$$

Ondey_i representa a saída da função e z_i é saída do neurônio.

•Redes Recorrentes: este tipo de rede possui realimentação das saídas para os demais neurônios, conforme exemplificado na FIG.19.

FIGURA 19 - Rede neural Recorrente.



Fonte: Palmiere (2016).

2.7.3 TIPOS DE APRENDIZAGEM

De acordo com Vellasco (2007), a aprendizagem de uma Rede Neural refere-se ao ajuste dos parâmetros, realizados através de estímulos recebidos do ambiente no qual ela está implantada. O ajuste dos pesos é feito através de algoritmos de aprendizagem, que são definidos para solucionar um problema específico. O aprendizado da Rede Neural ocorre, basicamente, de duas maneiras:

a) Supervisionada: neste tipo de treinamento há uma comparação entre a saída da rede e o valor de saída estimado. O erro encontrado a partir desta comparação é propagado através da rede, fazendo com que o algoritmo de aprendizagem ajuste os pesos, visando minimizar este erro. O algoritmo mais utilizado para treinamento supervisionado é o *Backpropagation*.

- Função de Custo (*Loss Function*): essa função estima o quanto a resposta da rede neural está distante do resultado desejado. Segundo Nicolau [20--], um exemplo de função de custo é o calculo do erro quadrático médio:

$$E_{QM} = \frac{\sum_{i=1}^{N}(e_i)^2}{N} \tag{19}$$

Onde e_i é o erro para cada neurônio e N é o número de padrões de entrada.

- *Cross Entropy*: mede o desempenho de uma rede neural ao fazer a classificação aplicando a função Softmax. A perda de entropia cruzada aumenta à medida que

a probabilidade prevista se distancia da classe pertencente. O cálculo da *cross entropy* para uma rede multi-classes, pode ser obtido pela Eq.(20):

$$-\sum_{c=1}^{M} y_{x,c} \log(p_{x,c}) \tag{20}$$

Onde$M$ indica o número de classes, y assume o valor 0 ou 1, caso a classe c seja a classificação correta para o objeto a ser classificado$x$ e *p* representa a predição da probabilidade de $x$pertencer à classe c. (VADHADIA; FORTUNER, 2017).

- Gradiente Descendente: essa técnica objetiva minimizar o erro da rede neural. A minimização ocorre através da modificação dos pesos e *bias* buscando encontrar o mínimo local da função de custo:

$$\Delta w_i = -\alpha \frac{\partial E(w_i)}{\partial w_i} \tag{21}$$

OndeΔw_i representa a atualização dos pesos e *bias* durante uma iteração calculada através da multiplicação da taxa de aprendizadoα pela derivada parcial da função de custo $E(w_i)$ com relação ao peso w_i(NICOLAU, ([20--])).

- Gradiente Descendente Estocástico: é um método utilizado para a atualização de parâmetros em processos nos quais o conjunto de treinamento é muito grande, como na Aprendizagem Profunda. A diferença deste método para o Gradiente Descendente é que a atualização ocorre com apenas uma observação a cada iteração, enquanto aquele utiliza a amostragem completa para fazer o mesmo (FERREIRA, [20--]).

- *Dropout*: esta técnica desativa um conjunto de neurônios selecionados aleatoriamente, fazendo com que a rede desenvolva a capacidade de aprender

mesmo que algumas unidades sejam zeradas. Esse método ajuda a minimizar o sobre-ajuste (*overfitting*) (FLORINDO, [20--]).

b) Não supervisionada: neste tipo de aprendizagem não é utilizado um sinal de erro. O treinamento ocorre a partir da apresentação de padrões associados à categoria as quais pertencem. A partir disso, ao ser apresentado um padrão que nunca foi visto pela rede, mas que pertence ao conjunto de treinamento será capaz de classificá-lo corretamente com base nas características aprendidas. Ainda segundo Vellasco (2007), é importante garantir essa boa capacidade de generalização, evitando o *overfitting* que acontece quando há um excesso de treinamento, onde a Rede Neural memoriza os padrões de treinamento.

Segundo Carvalho (2009), um ciclo de aprendizagem (época) é a apresentação dos pares de entrada e saída do processo. O ajuste dos pesos em um ciclo de aprendizagem pode ser feito das seguintes formas:

- Modo Padrão: o ajuste dos pesos é feito a cada apresentação de um exemplo à RNA. A correção do erro é feita baseada no exemplo apresentado no ciclo. A cada ciclo pode ocorrer inúmeras correções.
- Modo Batch: é feito um ajuste por ciclo. Todos os exemplos de treinamento são apresentados à RNA, o erro médio é calculado e com base nele são feitos os ajustes nos pesos (CARVALHO, 2009).

2.8 Redes Neurais Convolucionais

As Redes Neurais Convolucionais (RNC) são um modelo de redes neurais de múltiplas camadas que foram desenvolvidas para se assemelhar ao processo de visão biológico. Seu processo de reconhecimento e análise de imagens consiste em aplicar filtros nas imagens de entrada, realizando uma operação de convolução. O resultado dessa operação representa as características mais relevantes extraídas da imagem. Após esse processo é feita uma subamostragem para reduzir o tamanho desses dados e facilitar o seu processamento. Por fim, para realizar a classificação essas informações passam por camadas totalmente conectadas e por último, por uma camada de classificação, onde será indica a probabilidade de pertencimento de

uma imagem a uma determinada classe (VARGAS; CARVALHO; VASCONCELOS, [200-]).

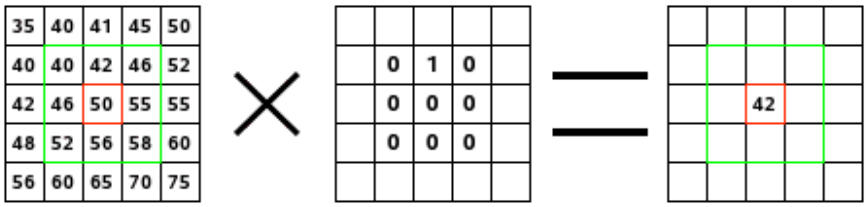
De acordo com Miyazaki (2017), as RNCs são invariantes às transformações geométricas (escala, rotação, espelhamento, etc.), que as tornam capazes de processar imagens com esse tipo de variação.

2.8.1 CAMADA CONVOLUCIONAL

Segundo Karpathy (2015), a camada convolucional é um conjunto de filtros (*kernels*) que aprendem características extraídas de uma imagem, fazendo uma espécie de varredura na mesma. A convolução entre a imagem e o *kernel* gera um mapa de características (*feature maps*). Essa operação é feita através da multiplicação da matriz da imagem pela matriz *kernel*.

Na FIG.20, é possível visualizar como é realizada a operação de convolução, em que um filtro é aplicado em uma determinada região da imagem.

FIGURA 20 - Operação de convolução.



Fonte: Willer Júnior, [20--].

A convolução é feita pela soma de todas as multiplicações do valor dos pixels da imagem pelo valor equivalente no filtro (Willer Júnior, 20--), como demonstrado na Eq. (22):

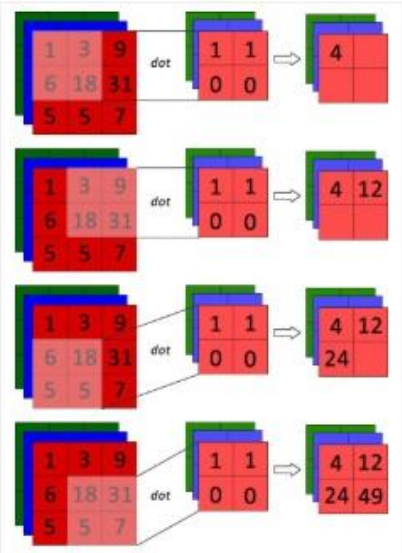
$$(40*0)+(42*1)+(46*0)+(46*0)+(50*0)+(55*0)+(52*0)+(56*0)+(58*0) = 42$$

No processo de treinamento, a rede neural aprende os filtros que são ativados quando detectam alguma característica importante, como bordas e manchas de cor. Dois parâmetros controlam o volume de saída da camada convolucional: o *stride* e o *padding* (KARPATHY, 2015).

2.8.2 PASSO (*STRIDE*)

Conforme citado anteriormente, os filtros de convolução se deslocam pela matriz da imagem para extrair características. Esse deslocamento é feito de modo espaçado, movendo o filtro *pixel* por *pixel*, caso o passo (*stride*) seja 1 ou saltando dois *pixels* por vez, caso o *stride* seja 2. Esse procedimento fará com que a saída tenha um volume reduzido (Karpathy, 2015). Na FIG.21é possível visualizar um exemplo da aplicação de um filtro 2x2x3 com stride 1.

FIGURA 21 - Aplicação do stride.

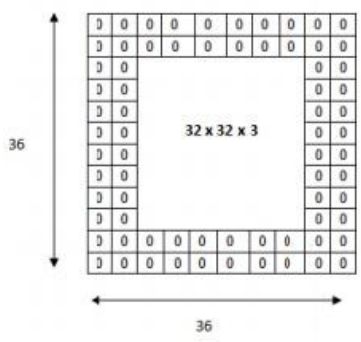


Fonte: Pacheco (2017).

2.8.3 PREENCHIMENTO (PADDING)

Segundo Florindo [20--], para que a operação de convolução seja possível, caso o espaçamento do *stride* ultrapasse o tamanho da imagem, é necessário aplicar um preenchimento, como o *zero-padding* (preenchimento com zeros), tornando a imagem convoluída do tamanho da original. A FIG.22 apresenta uma imagem com tamanho 32x32x3 preenchida com zeros para aplicação de filtro 5x5x3 com *stride* 1.

FIGURA 22 - Padding.



Fonte: Florindo ([20--]).

Ainda de acordo com Florindo [20--], após serem definidos os parâmetros para a convolução é possível calcular o tamanho da saída, conforme Eq.(22):

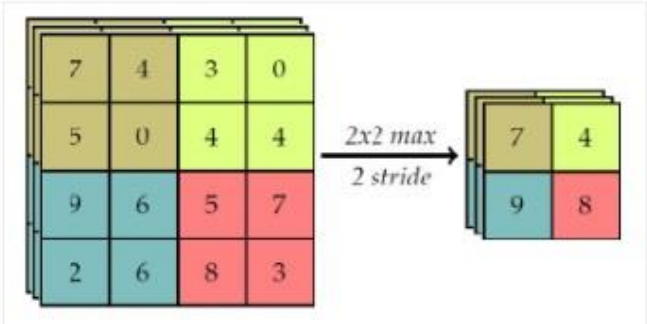
$$0 = \frac{W - K + 2P}{S} + 1 \tag{22}$$

Em que W é o tamanho (altura da imagem), K é a dimensão do filtro aplicado, P é o tamanho do *padding* e S representa o tamanho do *stride*.

2.8.4 CAMADA DE *POOLING*

Conforme Pacheco (2017), a camada de *pooling* é aplicada após a camada de convolução com o objetivo de reduzir o tamanho da imagem de saída, diminuindo também o processamento computacional e evitando o *overfitting*. É necessário definir o tamanho do filtro de *pooling* e o valor do *stride* que será aplicado. A função *Max Pooling* tem sido a mais utilizada, pois diminui a quantidade de elementos em cada camada, armazenando apenas o valor máximo em cada região. A FIG. 23 mostra a aplicação do max pooling de tamanho 2x2 com stride 2.

FIGURA 23 - Pooling.



Fonte: Pacheco (2017).

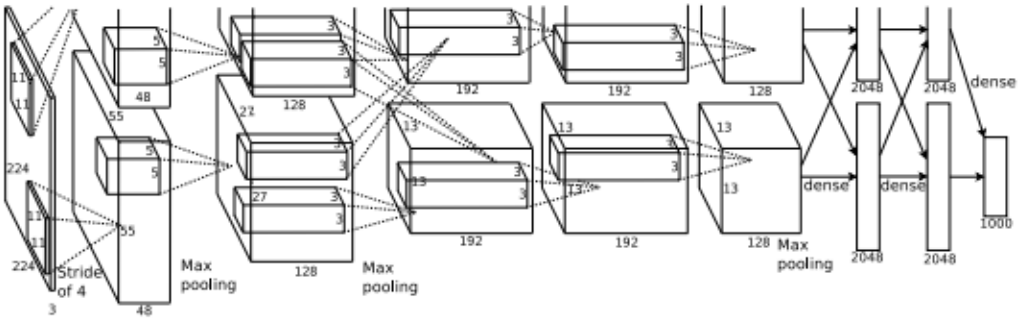
2.8.5 CAMADA TOTALMENTE CONECTADA

Os neurônios em uma camada totalmente conectada possuem conexões com todos os neurônios da camada anterior. Por seguir as camadas convolucionais e de *pooling*, essa camada realiza a classificação, utilizando nas unidades de saída uma função *softmax* que obtém a probabilidade de uma imagem de entrada pertencer a uma determinada classe (PACHECO, 2017).

2.9 AlexNet

A AlexNet é uma arquitetura de rede neural convolucional desenvolvida por Krizhevsky, Sutskever, Hinton (2012). Foi vencedora do desafio ImageNet em 2012. Possui arquitetura profunda com oito camadas de aprendizagem, cinco camadas de convolução e três camadas totalmente conectadas.

FIGURA 24 - Arquitetura da AlexNet.



Fonte: Krizhevsky, Sutskever, Hinton (2012).

A camada de entrada recebe uma imagem de dimensão 224x224x3. Em seguida, é feita uma convolução com a aplicação de 96 *kernels* de dimensão

11x11x3 na imagem de entrada. Na segunda camada de convolução, a imagem de saída da primeira camada, após passar pela camada de *pooling*, é convolucionada com 256 *kernels* de tamanho 5x5x48. As outras camadas de convolução estão conectadas entre si sem aplicação de *pooling*. A terceira camada convolucional tem 384 *kernels* de tamanho 3×3×256 que são aplicados às saídas da segunda camada convolucional. A quarta camada convolucional tem 384 *kernels* de tamanho 3×3×192, e a quinta camada convolucional tem 256 *kernels* de tamanho 3×3×192. Há 4096 neurônios em cada camada totalmente conectada (KRIZHEVSKY, SUTSKEVER, HINTON, 2012).

O treinamento da rede foi executado em duas Unidades de Processamento Gráfico (GPUs). Conforme FIG. 24 pode ser observado que o treinamento foi realizado em duas partes, uma GPU executou a parte superior e a outra executou a parte inferior da rede, comunicando-se só em algumas camadas.

A AlexNet utiliza a unidade linear retificada (função ReLU) como função de ativação por apresentar melhor desempenho de treinamento em relação ás funções tangente hiperbólica (Tanh) e sigmoide. Além disso, utiliza a técnica de *dropout* para evitar o *overfitting*.

3 METODOLOGIA

Para desenvolver este projeto foi necessário realizar, primeiramente, uma pesquisa bibliográfica em livros, artigos científicos e documentos eletrônicos de forma a se obter referências por aplicações de técnicas de visão computacional e aprendizagem profunda, em aplicações relacionadas a este trabalho.

A parte experimental do projeto seguiu as etapas:

- Aquisição de imagens de folhas e frutos do cafeeiro que apresentam sintomas das principais deficiências minerais e pragas desta cultura para a composição de um banco de imagens com várias categorias. Os dados do banco de imagens foram ampliados com a aplicação do método dos Momentos Invariantes de Hu.
- Retreinamento do modelo de Rede Neural Convolucional - AlexNet, disponível no *software* MATLAB®, para a classificação das deficiências minerais e pragas do cafeeiro.
- Realização de testes utilizando a rede neural retreinada, para verificar a eficiência da mesma, em imagens estáticas do banco de dados e também com a classificação dinâmica utilizando a *webcam*.

3.1 Materiais

O trabalho foi desenvolvido com utilização do *software* MATLAB® na versão 2017a e foi atualizado de acordo com novas funções da versão 2018a.

O hardware utilizado foi o computador portátil Intel® Core™ i5-3230M, processador de 2,60GHz com 4,00 Gigabytes de memória RAM e 346 Gigabytes de espaço em disco rígido e câmera Logitech HD Pro Webcam C920, com conexão USB, sensores e lentes de vidro, foco automático e resolução óptica real de 3 MP.

3.2 Aumento de Dados

As imagens utilizadas para retreinar a rede neural foram adquiridas através de pesquisa em artigos, como o Manual do Café - Manejo de Cafezais em Produção (MESQUITA, 2016), e bancos de imagens encontrados na internet, como em YARA BRASIL (2018). Além da qualidade das imagens, foram levadas em

consideração para a seleção, características como a categoria de deficiência mineral ou praga do cafeeiro em que cada imagem se enquadraria.

Conforme citado na seção Referencial Teórico, a quantidade de dados interfere na eficiência da rede neural ao determinar a categoria para classificação, portanto, procurou-se balancear a quantidade de dados, dispondo-se da mesma quantidade de imagens para todas as categorias. Porém, para algumas delas foi possível adquirir apenas um pequeno número de imagens diferentes. A solução para este problema foi gerar variações das imagens já existentes.

Essas variações permitem que as deficiências minerais e pragas do cafeeiro possam ser reconhecidas, independente das transformações geométricas sofridas pelas imagens tais como, redução na escala, rotação, translação e reflexão. Essa característica é garantida pelos Momentos Invariantes de Hu, que devem ser constantes, verificadas para cada uma das componentes de cor (RGB). Com esta técnica foram adquiridas 32 imagens para cada categoria.

Para que as imagens pudessem ser transladadas, reduzidas, espelhadas e rotacionadas, foi implementada a função “GerarImagens” no MATLAB®, conforme código fonte do APÊNDICE A. Todos os comandos utilizados para implementar as funções deste projeto foram pautados em MATHWORKS DOCUMENTATION (2018).

Na função “GerarImagens”, inicialmente, os valores de intensidade das imagens de entrada foram transformados em pontos flutuantes através do comando *tofloat*, sendo convertidas do intervalo de intensidade [0,255] para o intervalo [0,1], a fim de simplificar o seu processamento.

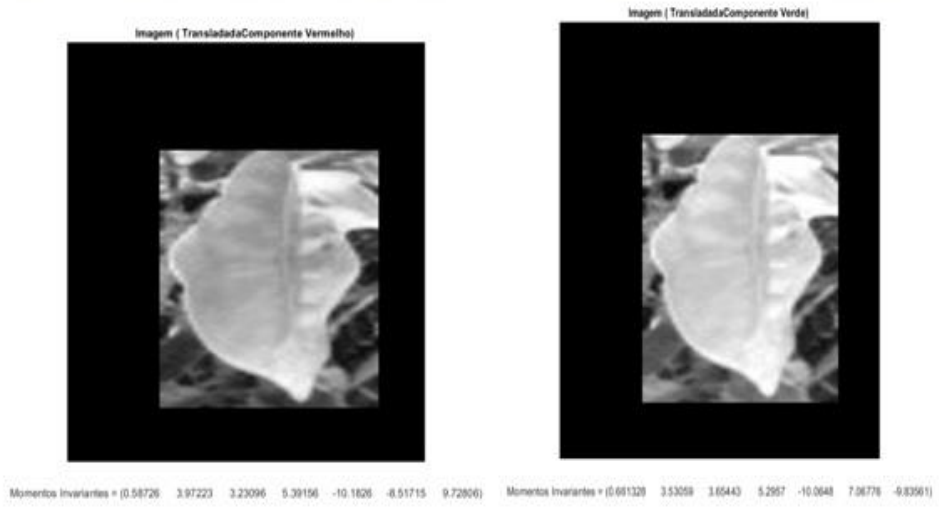
Com o intuito de aplicar as transformações geométricas nas imagens e calcular os Momentos Invariantes de Hu, conforme exemplificado por Gonzalez e Woods (2010, p.554) foi necessário redimensioná-las para o tamanho 400x400 através do comando *imresize* e adicionar uma moldura à sua volta, com preenchimento em zero, de dimensão 84x84 através do comando *padarray* para que tivessem o mesmo tamanho. Como a rede neural implementada espera que as imagens de entrada tenham dimensão 227x227, foi necessário redimensionar a imagem gerada com moldura, utilizando novamente o comando *imresize*. Um exemplo de resultado dessas operações é mostrado na FIG. 25.

FIGURA 25 - Comparação entre a imagem original e a com moldura.

Fonte: PROMIP (2007) apud SFAgro ([20--]).

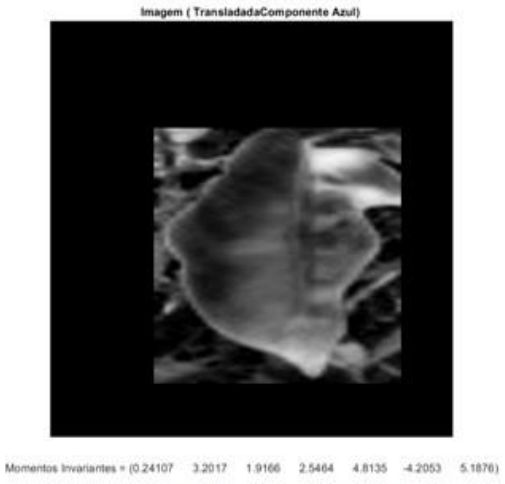
Utilizando o comando *size*, foram atribuídos os tamanhos de linhas, colunas e componentes de cor das imagens com moldura a variáveis, para que pudessem ser criadas matrizes em zero do tamanho de suas dimensões e componentes através do comando *zero*. Para transladar a imagem para a direita, as componentes de cor da imagem foram copiadas para a posição deslocada de 84 linhas para baixo e 84 colunas para a direita sobre a matriz de zeros (FIG. 26 e 27).

FIGURA 26 - Componente vermelha e verde da imagem transladada.



Fonte: Adaptado de Martins, ([20--]).

FIGURA 27 - Componente azul da imagem transladada.



Fonte: Adaptado de Martins, ([20--]).

Em seguida, através do comando *cat,* as componentes de cor deslocadas foram unidas em uma matriz de três dimensões. A imagem gerada foi redimensionada através do comando *imresize* para o tamanho 227x227 para que pudesse ser reconhecida pela rede neural como, por exemplo, na FIG. 28.

FIGURA 28 - Imagem transladada para a direita.

Fonte: Adaptado de Martins, ([20--]).

Para transladar a imagem gerada anteriormente totalmente para o canto inferior direito, foi necessário retirar a moldura nessas laterais da imagem. Um exemplo de resultado é mostrado na FIG. 29.

FIGURA 26 - Imagem transladada para o canto.

Fonte: PROMIP (2007) apud SFAgro ([20--]).

Na transformação de redução foi necessário saltar uma linha e uma coluna em toda a matriz da imagem de entrada (original). Em seguida, aplicou-se uma moldura com dimensão 184x184 utilizando o comando *padarray*. O resultado é mostrado na FIG. 30.

FIGURA 27 - Imagem original de entrada e reduzida com moldura.

Fonte: PROMIP (2007) apud SFAgro ([20--]).

Com o objetivo de refletir a imagem original, foi aplicado o comando *fliplr* que retorna as colunas da imagem invertidas da esquerda para a direita. Em seguida, a moldura de tamanho 84x84 é reaplicada na imagem através do comando *padarray*, conforme FIG. 28.

FIGURA 28 - Comparação entre a imagem original e a espelhada com moldura.

Fonte: PROMIP (2007) apud SFAgro ([20--]).

A fim de rotacionar a imagem nos ângulos de 45º e 90º foi utilizado o comando *imrotate* que gira a imagem no sentido anti-horário, utilizando o método bilinear de interpolação, cujo valor do *pixel* de saída é uma média ponderada de pixels na adjacência 2x2 mais próxima. O resultado é mostrado na FIG. 29.

FIGURA 29 - Imagem rotacionada em 45° e 90°.

Fonte: PROMIP (2007) apud SFAgro ([20--]).

Os momentos invariantes das componentes RGB foram obtidos através da implementação da função “MomentosInvariantesCor”, que utiliza a rotina “invmoments” para calcular os sete momentos invariantes e retornar os valores absolutos desses momentos, mantendo o seu sinal, conforme código fonte do APÊNDICE B.

Na TAB. 1 são apresentados os valores dos Momentos Invariantes de Hu para cada transformação geométrica e componente de cor para a imagem da FIG. 28.

TABELA 1 - Momentos Invariantes das componentes de cor.

Posição	Componentes de cor	Λ 1	Λ 2	Λ 3	Λ4	Λ 5	Λ 6	Λ 7
Original com Moldura	R (Red)	0,58726	3,97223	3,23096	5,39156	-10,1826	-8,51715	9,72806
	G (Green)	0,661328	3,53059	3,65443	5,2957	-10,0648	7,06776	-9,83561
	B (Blue)	0,24107	3,2017	1,9166	2,5464	4,8135	-4,2053	5,1876
Translada parcialmente	R (Red)	0,58726	3,97223	3,23096	5,39156	-10,1826	-8,51715	9,72806
	G (Green)	0,661328	3,53059	3,65443	5,2957	-10,0648	7,06776	-9,83561
	B (Blue)	0,24107	3,2017	1,9166	2,5464	4,8135	-4,2053	5,1876
Transladada totalmente	R (Red)	0,58726	3,97223	3,23096	5,39156	-10,1826	-8,51715	9,72806
	G (Green)	0,661328	3,53059	3,65443	5,2957	-10,0648	7,06776	-9,83561
	B (Blue)	0,24107	3,2017	1,9166	2,5464	4,8135	-4,2053	5,1876
Reduzida	R (Red)	0,587305	3,98849	3,23153	5,35489	-10,3377	-8,09951	9,65736
	G (Green)	0,6614	3,5366	3,6567	5,2931	-9,9763	7,0779	-9,873
	B (Blue)	0,24077	3,2071	1,9132	2,5415	4,8043	-4,2171	5,1794
Espelhada	R (Red)	0,58726	3,97223	3,23096	5,39156	-10,1826	-8,51715	-9,72806
	G (Green)	0,66138	3,53059	3,65443	5,2957	-10,0648	7,06776	9,83561
	B (Blue)	0,24107	3,2017	1,9166	2,5464	4,8135	-4,2053	-5,1876
Rotacionada 45º	R (Red)	0,587229	3,97279	3,23094	5,39069	-10,1786	-8,52484	9,72709
	G (Green)	0,661294	3,5305	3,65443	5,29484	-10,0632	7,0669	-9,83442
	B (Blue)	0,24104	3,2033	1,9167	2,5463	4,8136	-4,2064	5,1867
Rotacionada 90º	R (Red)	0,58726	3,97223	3,23096	5,39156	-10,1826	-8,51715	9,72806
	G (Green)	0,661328	3,53059	3,65443	5,2957	-10,0648	7,06776	-9,83561
	B (Blue)	0,24107	3,2017	1,9166	2,5464	4,8135	-4,2053	5,1876

Fonte: Autoria Própria.

Como pode ser observado a partir dos dados apresentados na TAB. 1, a aplicação dos Momentos Invariantes de Hu valida as transformações das imagens das folhas e frutos do café, pois nenhum defeito é gerado nas imagens. Mesmo para a imagem transladada, que não é uma transformação linear, pois a origem não é fixa, os momentos invariantes se mantiveram para cada componente de cor da imagem.

3.3 AlexNet Pré-treinada

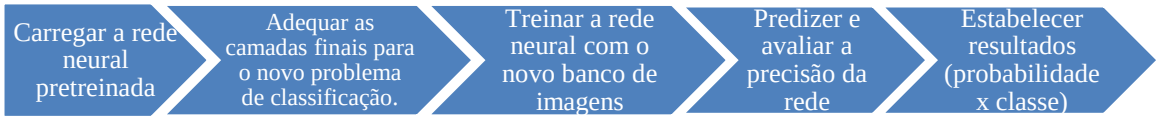
Para o desenvolvimento desta pesquisa foi utilizada a rede neural AlexNet pré-treinada disponível no MATLAB®, que processa uma imagem de entrada e gera um rótulo a ser atribuído a esta imagem na saída. Essa rede neural convolucional foi treinada originalmente com o extenso banco de imagens ImageNet, aprendendo diversos padrões para o reconhecimento de um número abrangente de imagens (MATHWORKS, 2018).

É possível fazer o ajuste de uma rede neural convolucional pré-treinada usando novas imagens ao aplicar os padrões que foram aprendidos, treinando um classificador com o propósito de solucionar um novo problema. Essa técnica é chamada de aprendizado de transferência (*transfer learning*) e apresenta-se mais

rápida e simples que treinar uma nova rede, podendo ser usado uma quantidade menor de imagens para retreinamento (MATHWORKS, 2018).

A FIG. 30 mostra as etapas para o ajuste de uma rede neural pretreinada a um novo problema de classificação.

FIGURA 30 - Etapas para ajuste de uma rede neural pré-treinada.



Fonte: Adaptado de MATHWORKS (2018).

3.4 Aprendizado de Transferência

3.4.1 ARQUITETURA

Para ajustar a AlexNet pré-treinada no MATLAB® ao processo de classificação das deficiências minerais e pragas das folhas e frutos do cafeeiro, foi criada a função “Retreinando_ALEXNET”, para carregar um modelo disponibilizado pela plataforma. Através do comando *Rede=alexnet*, esse modelo é carregado, conforme código fonte do APÊNDICE C.

O comando *layer=Rede.Layers* exibe a arquitetura da rede, conforme FIG 31. Ela possui 25 camadas, sendo cinco camadas convolucionais e três camadas totalmente conectadas.

FIGURA 31 - Arquitetura da rede pré-treinada AlexNet.

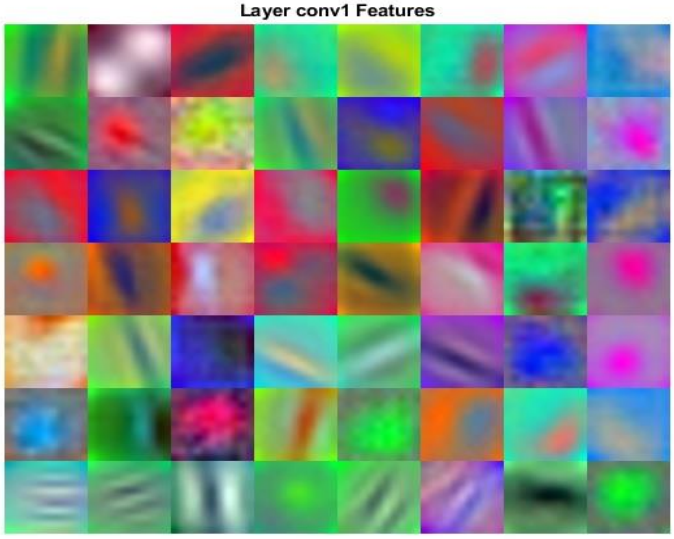
```
ans =
  25x1 Layer array with layers:

     1   'data'     Image Input                 227x227x3 images with 'zerocenter' normalization
     2   'conv1'    Convolution                 96 11x11x3 convolutions with stride [4  4], dilation factor [1  1] 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'    Convolution                 256 5x5x48 convolutions with stride [1  1], dilation factor [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], dilation factor [1  1] and padding [1  1  1  1]
    11   'relu3'    ReLU                        ReLU
    12   'conv4'    Convolution                 384 3x3x192 convolutions with stride [1  1], dilation factor [1  1] and padding [1  1  1  1]
    13   'relu4'    ReLU                        ReLU
    14   'conv5'    Convolution                 256 3x3x192 convolutions with stride [1  1], dilation factor [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: MATHWORKS, 2018.

- *Image Input*: A primeira camada é a de entrada das imagens e normalização de dados para que tenham aproximadamente a mesma escala. A rede neural requer que as imagens tenham tamanho [227 227 3], em que 227 é o valor da altura e largura das imagens e 3 é o número de canais de cor (imagens RGB). A normalização é feita pelo centro de zero, que subtrai a imagem média de cada imagem de entrada, especificada pela propriedade *AverageImage*, centralizando o conjunto de dados em relação à origem. Essa propriedade é calculada automaticamente pela função *trainNetwork* durante o treinamento. Ao retreinar a rede neural, o comando *dimEntrada=Rede.Layers(1)InputSize* atribui à variável *dimEntrada* as dimensões das imagens utilizadas para treinar a rede original, garantindo que ao redimensionar as imagens para treino e validação elas tenham a dimensão exigida pela rede.
- *Convolution*: As cinco camadas convolucionais realizam o processamento das imagens através da aplicação de um conjunto de filtros responsáveis por ativar características significantes. A primeira camada convolucional, denominada ‘conv1’, contém filtros detectores de borda e filtros de cores. Conforme FIG.32 esses filtros estão em diversas disposições, o que permite que a rede desenvolva padrões mais complexos nas camadas posteriores, gerando filtros mais detalhados. Nessa camada convoluvional são usados 96 filtros de tamanho 11x11x3. O valor do *stride* utilizado nessa camada convolucional é [4 4]. O vetor de *padding* utilizado é o de preenchimento em zero [0 0 0 0].

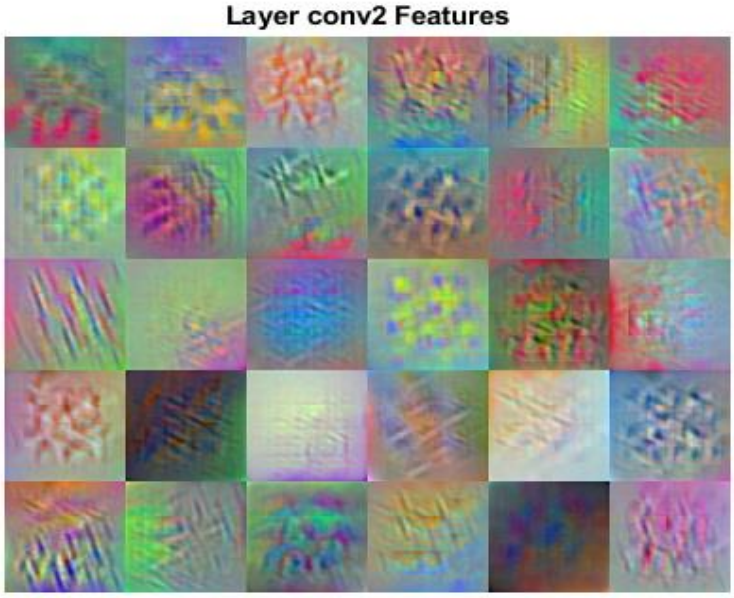
FIGURA 32 - Exemplo de padrão aprendido na primeira camada convolucional.



Fonte: MATHWORKS, 2018.

A segunda camada convolucional, ‘conv2’, utiliza 256 filtros de tamanho 5x5x48, stride [1 1] e padding [2 2 2 2]. Na FIG. 33 é possível visualizar exemplos de padrões gerados nessa camada convolucional.

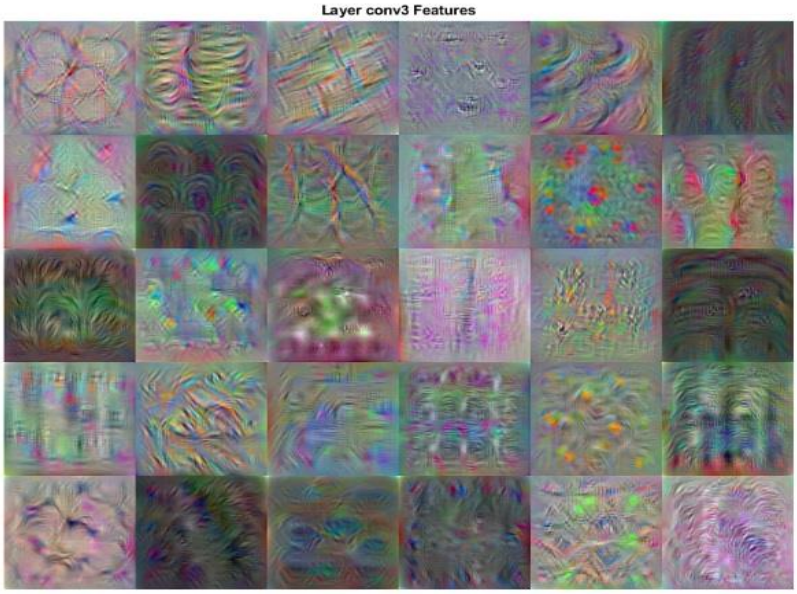
FIGURA 33 - Exemplo de padrões aprendidos na segunda camada convolucional.



Fonte: MATHWORKS, 2018.

A terceira camada convolucional, ‘conv3’, possui 384 filtros de tamanho 3x3x256, stride [1 1] e padding [1 1 1 1]. Um exemplo de padrões gerados pelo processo de convolução nessa camada pode ser visto na FIG. 34.

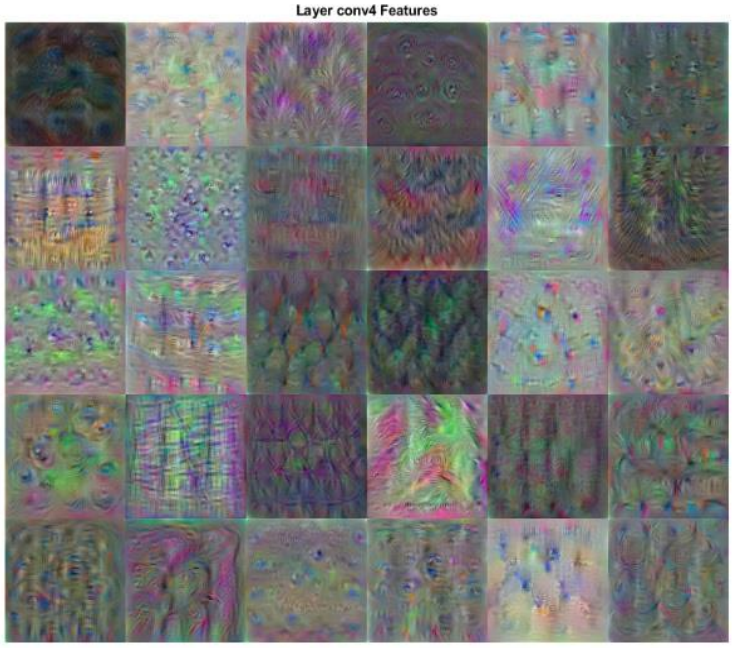
FIGURA 34 - Exemplo de padrões aprendidos na terceira camada convolucional.



Fonte: MATHWORKS, 2018.

A quarta camada convolucional, ‘conv4’, possui 384 filtros de dimensões 3x3x192, stride [1 1] e padding [1 1 1 1]. Na FIG.35 é apresentado um exemplo de recursos que podem ser aprendidos nessa camada.

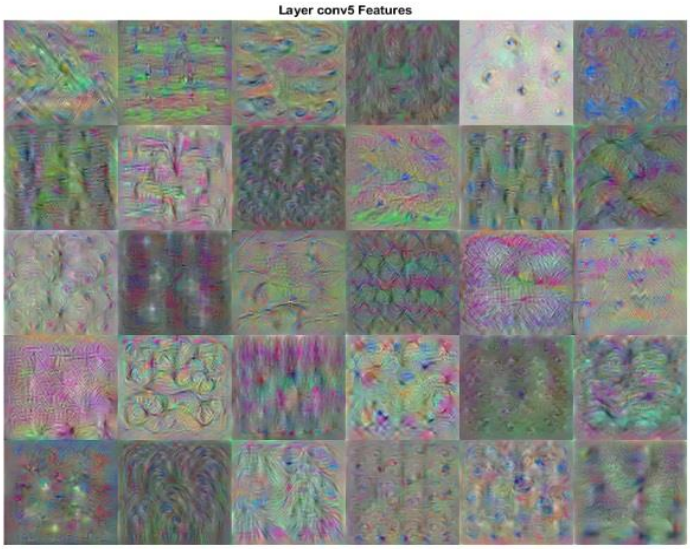
FIGURA 35 - Exemplo de padrões aprendidos na quarta camada convolucional.



Fonte: MATHWORKS, 2018

A quinta camada convolucional, ‘conv5’, possui 256 filtros 3x3x192, stride [1 1] e padding [1 1 1 1]. Um exemplo de características que podem ser extraídas nessas camadas pode ser visto na FIG. 36.

FIGURA 36 - Exemplo de padrões aprendidos na quinta convolucional.



Fonte: MATHWORKS, 2018.

- ReLU: As camadas convolucionais são seguidas pela unidade linear retificada (ReLU), que é uma função de ativação não linear. Essa camada realiza uma operação de limite para cada unidade, em que qualquer valor de entrada *x* que seja negativo é estabelecido como zero, de acordo com a função da Eq. (23).

$$f(x) = \begin{cases} x, x \geq 0 \\ 0, x < 0 \end{cases} \tag{23}$$

- *Cross Channel Normalization*: Na quarta e oitava camadas, após as camadas de ativação ReLU, encontra-se uma camada de normalização de resposta local em canal. Nessa camada acontece a troca de cada elemento gerado pela camada de convolução e que passou pela função de ativação por um valor normalizado, obtido através dos elementos de um determinado número de canais adjacentes (elementos na janela de normalização). Para cada elemento na entrada, a função *trainNetwork* calcula um valor normalizado. A *AlexNet* pré-treinada faz normalização de resposta local através de uma janela de cinco canais para cada elemento.

- *Max pooling*: Essa camada reduz a amostragem de dados (tamanho das imagens) dividindo a entrada em regiões menores e calculando o máximo de cada região. A rede neural utiliza um filtro de dimensão 3x3 para realizar o *pooling*. O valor do

stride utilizado na rede neural é [2 2] e o vetor utilizado para o *padding* na camada de *max pooling* é [0 0 0 0].

- *Fully Connected*: As camadas totalmente conectadas realizam a multiplicação das informações de entrada por uma matriz de peso e acrescenta um vetor *bias*. A rede neural possui três camadas totalmente conectadas que foram reconfiguradas para as 11 classes de deficiências minerais e pragas no cafeeiro trabalhadas neste projeto. Os neurônios das camadas totalmente conectadas estão ligados a todos os neurônios da camada anterior. Os padrões identificados nessas camadas são mais intricados que aqueles aprendidos nas camadas inciais da rede neural. A taxa de aprendizado e os parâmetros de regularização podem ser ajustados ao criar as camadas de transferência:

  - *WeightLearnRateFactor*: É o fator de taxa de aprendizado para os pesos. O algoritmo multiplica esse fator pela taxa de aprendizado global para determinar a taxa de aprendizado para os pesos nessa camada. Para o retreinamento da rede neural esse fator foi definido com valor 20, significando que a taxa de aprendizado para os pesos nessa camada será 20 vezes maior que a taxa de aprendizado global.
  - *BiasLearnRateFactor*: É o fator de taxa de aprendizado para os *bias.* Assim como no fator citado anteriormente, é feita a multiplicação desse fator pela taxa de aprendizado global para determinar a taxa de aprendizado para os *bias* nessa camada. No retreinamento da rede neural, o BiasLearnRateFactor foi definido com o valor 20, portanto, a taxa de aprendizado para os *bias* na camada será 20 vezes maior que a taxa de aprendizado global. A taxa de aprendizagem global é definida nas opções da função *trainingOptions*.

- *Dropout*: Após a primeira camada totalmente conectada é empregada a camada de *dropout* que desativa aleatoriamente alguns neurônios em certa probabilidade. Deste modo, para cada novo elemento de entrada, a função *trainNetwork* escolhe aleatoriamente um subconjunto de neurônios, compondo uma camada diferente. A probabilidade para a escolha da unidade a ser inibida temporariamente é de 50% para o retreinamento da rede neural.

- *Softmax*: Essa camada aplica uma função *softmax* às imagens de entrada. Essa camada, junto a uma camada de classificação, deve seguir a camada final totalmente conectada. Essa função de ativação da unidade de saída transforma as saídas para cada classe em valores entre 0 e 1. Isso corresponde à probabilidade de uma imagem pertencer a uma determinada classe.

- *Classification Output*: A camada de classificação calcula a perda de entropia cruzada para problemas de classificação de múltiplas classes exclusivas. Na camada de classificação, *trainNetwork* pega os valores da função *softmax* e atribui cada entrada a uma das 11 classes de deficiências minerais e pragas do cafeeiro, usando a função de entropia cruzada para um esquema de codificação.

3.4.1 OPÇÕES DE TREINAMENTO

As imagens para o retreinamento da rede neural convolucional foram organizadas em pastas, separadas por cada deficiência mineral e praga do cafeeiro, sendo possível usar o *Data Store* do MATLAB. Essa propriedade rotula automaticamente as imagens com base nos nomes das pastas e armazena os dados como um objeto *ImageDatastore*. Esse tipo de armazenamento apreende com eficiência lotes de imagens durante o treinamento de uma rede neural convolucional. As imagens foram separadas aleatoriamente em um conjunto com 80% para treinamento e 20% para testar a precisão da rede.

As condições para o retreinamento foram especificadas no comando *trainingOptions*. Para minimizar a função de custo, foi utilizada a abordagem do gradiente descendente estocástico com momentum - '*sgdm'*. O algoritmo de descida de gradiente estocástico pode oscilar no percurso de descida mais abrupta em direção ao ótimo. Acrescentar um termo de momentum à atualização dos parâmetros é uma forma de reduzir essa oscilação. A descida de gradiente estocástica com atualização de momentum é definida pela Eq. (24):

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \tag{24}$$

Em que *l* representa o número da iteração, $\alpha > 0$ é taxa de aprendizado, θ é o vetor de parâmetros (pesos e *bias*), $\nabla E(\theta)$ é o gradiente da função de custo e γ determina a contribuição do passo de gradiente anterior para a iteração atual. Foram utilizados 0,9 (valor padrão) para o momentum (MATHWORKS, 2018).

A taxa de apredizado inicial (‘*InitialLearnRate*’) usada para o treinamento foi definida como 0,001, sendo o valor padrão para o otimizador ‘*sgdm*’.

O algoritmo de descida de gradiente estocástico com momentum avalia o gradiente e atualiza os parâmetros usando um subconjunto do conjunto de treinamento chamado de mini-lote (*mini-batch*). Cada avaliação do gradiente usando o mini-lote é uma iteração. Em cada iteração, o algoritmo avança para minimizar a função de custo. A passagem completa do algoritmo de treinamento sobre todo o conjunto de treinamento usando mini-lotes corresponde a uma época.

O número máximo de épocas (‘*MaxEpochs*’) a serem usadas para treinamento foi definido como 100. Foram feitas várias tentativas com números maiores e menores de épocas, porém o resultado não foi satisfatório. O tamanho do mini-lote (‘*MiniBatchSize*’) definido para cada iteração de treinamento foi 64.

Os dados a serem usados para validação (‘*ValidationData*’) durante o treinamento são as imagens especificadas para validação reconfiguradas e redimensionadas.

O comando *redRet = trainNetwork* inicia o treino da rede com as imagens especificadas para treino, as camadas de transferência e as opções especificadas para o treinamento. Logo após, a acurácia foi verificada, classificando as imagens de validação com a rede retreinada e apresentando a média (*mean*) de acertos da classificação.

FIGURA 37 - Progresso do retreinamento da rede neural.

```
>> Retreinando_ALEXNET_PithanV2(100);
Training on single CPU.
Initializing image normalization.
|=========================================================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Validation  |  Mini-batch  |  Validation  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |   Accuracy   |     Loss     |     Loss     |      Rate       |
|=========================================================================================================================|
|       1 |           1 |       00:00:05 |        7.81% |       22.73% |       2.6831 |       2.3153 |          0.0010 |
|      13 |          50 |       00:04:52 |       96.88% |       92.42% |       0.0558 |       0.2736 |          0.0010 |
|      25 |         100 |       00:09:54 |      100.00% |       93.94% |       0.0002 |       0.1068 |          0.0010 |
|      38 |         150 |       00:15:45 |      100.00% |      100.00% |       0.0007 |       0.0139 |          0.0010 |
|      50 |         200 |       00:21:01 |       98.44% |       98.48% |       0.0327 |       0.0347 |          0.0010 |
|      63 |         250 |       00:27:27 |      100.00% |      100.00% |       0.0080 |       0.0108 |          0.0010 |
|      75 |         300 |       00:33:38 |      100.00% |      100.00% |       0.0018 |       0.0425 |          0.0010 |
|      88 |         350 |       00:40:04 |      100.00% |       98.48% |       0.0009 |       0.0173 |          0.0010 |
|     100 |         400 |       00:46:33 |      100.00% |      100.00% |       0.0003 |       0.0141 |          0.0010 |
|=========================================================================================================================|
    1
```

Fonte: Autoria Própria

A FIG. 37 apresenta os dados no progresso de treinamento. A conclusão do treinamento ocorreu em um tempo total de 46 minutos e 33 segundos, sendo rápido devido à pequena quantidade de dados. São apresentadas a acurácia para treinamento dos mini-lotes e da validação a cada iteração. O erro no treinamento (*Mini-batch Loss*) e na validação (*Validation Loss*) é representado pela perda de entropia cruzada. À medida que a rede é apresentada aos dados de treinamento, a cada iteração, a função de perda é minimizada e a acurácia aumenta, até atingir a quantidade de épocas especificadas.

3.5 Classificador dinâmico

Com o objetivo de identificar doenças e pragas do cafeeiro *in loco,* foi desenvolvido um sistema de classificação em tempo real. Conectou-se a *webcam* à rede neural retreinada e a partir disso, foi possível apontá-la para as imagens para que a rede neural pudesse fazer o reconhecimento. São apresentados os comandos implementados para essa função:

- *webcamlist*: provê uma lista das *webcams* reconhecidas pelo MATLAB®.

FIGURA 38 - Função webcamlist no MATLAB®.

Fonte: Autoria Própria.

- cam= webcam(devicenumber): conecta a câmera à *webcam*. Cria no *workspace* do MATLAB® o objeto “cam” como sendo a webcam do índice ‘*devicenumber*’ definido e mostra todas as propriedades de *hardware* da *webcam*.

FIGURA 39 - Propriedades da webcam.

```
>> cam

cam =

  webcam with properties:

                    Name: 'Logitech HD Pro Webcam C920'
              Resolution: '640x480'
    AvailableResolutions: {1×19 cell}
                     Pan: 0
              Saturation: 128
            ExposureMode: 'auto'
              Brightness: 128
               Sharpness: 128
                    Tilt: 0
                Contrast: 128
                   Focus: 0
                    Zoom: 100
                    Gain: 79
            WhiteBalance: 4336
                Exposure: -5
   BacklightCompensation: 0

fx >>
```

Fonte: Autoria Própria.

- nnet1 = net: carrega a rede neural retreinada.

- Imag = cam.snapshot: programa a câmera para tirar uma foto para análise. Uma imagem é capturada pela *webcam* e a atribuição dessa imagem é feita à variável Imag.

- image(Imag): exibe a imagem adquirida no comando anterior. A cada vez que estas funções são executadas, uma imagem é capturada pela webcam e exibida na tela. De forma a se obter diversas imagens, enquanto o software estiver em execução, foi necessário criar uma rotina de repetição e adicionar o comando *drawnow,* conforme FIG. 43.

FIGURA 40 - Rotina de repetição.

```
while true
    Imag = cam.snapshot;

    drawnow

end
```

Fonte: Autoria Própria.

- Imag = imresize (Imag, [227,227]): redimensiona as imagens adquiridas para o tamanho exigido pela rede neural, antes de iniciar a classificação.

- label = classify (nnet1, Imag) : atribui à variável *label* a classificação das imagens obtidas feita pela rede neural.

- title (char (label)): converte a *label* para uma *string* para usá-la no título da imagem.

Ao executar esses comandos, as imagens capturadas através da câmera são classificadas e é exibida ao usuário uma interface com a imagem em tempo real e a *label* da categoria da imagem exibida.

3.6 Classificador estático

A fim de saber qual a probabilidade de pertencimento de uma imagem a uma determinada categoria da rede neural retreinada e otimizar os testes, foi desenvolvido um classificador para imagens contidas em um banco de dados. Para tal, foi implementada uma função utilizando os seguintes comandos no MATLAB®:

- nnet1 = net: carrega a rede neural retreinada.
- dir = strcat: atribui à variável *dir* a associação da imagem ao nome da categoria pertencente e sua ordenação no banco de imagens.
- f = imread (dir): atribui à varável *f* a imagem extraída do banco de imagens para classificação.
- f = imresize (f, [227,227]): redimensiona as imagens a serem classificadas para o tamanho exigido pela rede neural.
- [Classe, Prob] = classify ((nnet1, f) = classifica as imagens utilizando a rede neural retreinada, retornando a classe (uma das 11 categorias utilizadas no projeto) e a probabilidade de pertencimento a essa classe.
- Probabilidade = round (Prob*100)': Calcula a probabilidade percentual para cada categoria.

Se a probabilidade de uma classe não for 100%, o vetor probabilidade é mostrado como tabela, apresentando o percentual de probabilidade para cada uma das categorias. Para isso foi necessário inserir um *if loop*:

FIGURA 41 - *if loop* para cálculo da probabilidade.

```
if max(Probabilidade)<100
CLASSES_CAFE={'BICHO  MINEIRO';'BROCA';'CERCOSPORA';...
    'D.  CÁLCIO';'D.  FÓSFORO';'D.  MAGNÉSIO';...
    'D.  NITROGÊNIO';'D.  POTÁSSIO';'FERRUGEM';'F.  SAUDÁVEL';'PHOMA'};
T  =  table(CLASSES_CAFE,Probabilidade);
disp(T)
end
```

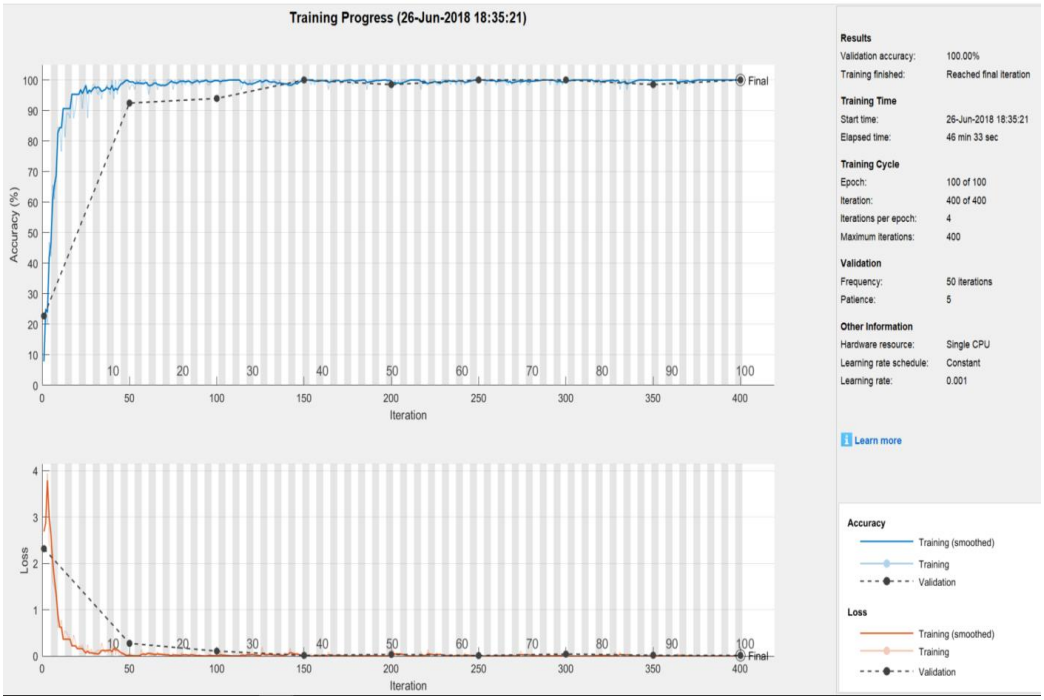
Fonte: Autoria Própria.

Em seguida, a imagem é exibida através do comando *imshow* com o nome da classe real, a atribuição de classe e a probabilidade percentual de pertencimento.

4 RESULTADOS E DISCUSSÕES

Ao aplicar o método de aprendizagem de transferência para retreinar a rede neural convolucional, a eficiência da rede é otimizada em cada iteração. Esse fato pode ser observado no gráfico da FIG. 42.

FIGURA 42 - Curvas de acurácia, erro e validação.



Fonte: Autoria Própria.

As curvas de acurácia e erro (*loss*) apresentadas na FIG. 42 mostram a relação das predições e acertos a cada iteração e época pelo método de entropia cruzada. É possível observar que as curvas para treinamento e validação tendem a convergir. A técnica de *dropout* e o aumento dos dados para treinamento com a aplicação dos Momentos Invariantes de Hu contribuiu para evitar o *overfitting*, aumentando a eficiência da rede.

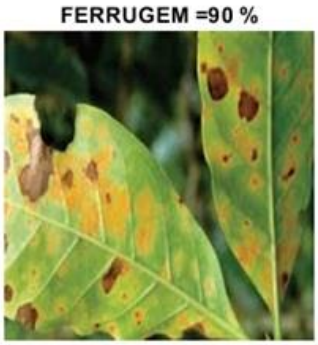
Foram testadas 117 imagens com o classificador estático, entre as quais a rede neural conseguiu classificar 98 corretamente. Algumas imagens obtiveram correta classificação, porém sem indicação de 100% para a classe real, conforme exemplificado na TAB. 2 e FIG. 46.

TABELA 2 - Probabilidade de classificação para folha com sintomas de ferrugem.

Classes Café	Probabilidade (%)
Bicho Mineiro	0
Broca	0
Cercospora	0
Deficiência de Cálcio	0
Deficiência de Fósforo	6
Deficiência de Magnésio	0
Deficiência de Nitrogênio	2
Deficiência de Potássio	2
Ferrugem	90
Folha Saudável	0
Phoma	0

Fonte: Autoria Própria.

FIGURA 46- Classificação de folha com sintomas de ferrugem.



NOME =FERRUGEM$_1$5

Fonte: Yara Brasil ([20--]).

A fim de validar a quantidade de imagens corretamente classificadas foi desenvolvida a TAB. 3. Esta tabela relaciona a classe real de cada imagem apresentada à rede neural com a *label* indicada por esta. As células da diagonal, nessa tabela, representam predições corretas. Conforme TAB. 3, as classes “bicho mineiro”, “broca”, “phoma” e “folha saudável” não apresentaram confusão. Essas classes têm como característica em comum conter imagens com boa qualidade, o que permite melhor visualização dos sintomas causados pelas pragas.

TABELA 3 - Tabela de confusão.

		PREDIÇÃO											
		Bicho Mineiro	Broca	Cercospora	Def. Cálcio	Def. Fósforo	Def. de Magnésio	Def. Nitrogênio	Def. de Potássio	Ferrugem	Phoma	Folha Saudável	FN
CLASSE REAL	Bicho Mineiro	4											
	Broca		6										
	Cercospora			11						1	1		
	Def. Cálcio				5							2	
	Def. Fósforo					13		1	1				
	Def. de Magnésio						6		1				
	Def. Nitrogênio				2		1	10		3	2		
	Def. de Potássio						1		10	1			
	Ferrugem					2				19			
	Phoma										10		
	Folha Saudável											4	
FP					2	2	2	1	2	5	3	2	19

Fonte: Autoria Própria.

Com base na TAB. 3, calcula-se a precisão para medir o desempenho da rede neural, conforme Silva, Almeida e Yamakami, 2012:

$$Precisão = \frac{VP}{VP + FP} \tag{25}$$

Em que *VP* são verdadeiros positivos (imagens classificadas corretamente) e *FP* falsos positivos (imagens classificadas incorretamente). Portanto, a rede neural atingiu uma precisão de 83,76%.

Entre as predições incorretas destacam-se, como por exemplo, a confusão de uma imagem da categoria “cercospora” classificada como “ferrugem”, apresentada na FIG. 43.

FIGURA 43 - Classificação de folha com sintomas de cercosporiose.

Fonte: Marangoni e Lima (2010).

Conforme dados da TAB. 4 e a FIG. 47 houve indicação de maior probabilidade para a classe “ferrugem”. Observa-se que, nesta imagem, a folha apresenta manchas escuras delimitadas por manchas amarelo-alaranjadas que podem indicar sintomas parecidos com a presença do fungo causador da classe predita pela rede neural (AGROLINK, 2017).

TABELA 4 - Probabilidade de classificação para folha com sintomas de cercosporiose.

Classes Café	Probabilidade (%)
Bicho Mineiro	0
Broca	0
Cercospora	1
Deficiência de Cálcio	1
Deficiência de Fósforo	0
Deficiência de Magnésio	0
Deficiência de Nitrogênio	0
Deficiência de Potássio	0
Ferrugem	92
Folha Saudável	0
Phoma	6

Fonte: Autoria Própria.

FIGURA 48-Classificação de folha com sintomas de deficiência de cálcio.

NOME =DEFICIÊNCIA DE CÁLCIO$_4$

Fonte: Lira ([20--]).

A rede neural classificou erroneamente a imagem da FIG. 48 pertencente à classe “deficiência de cálcio” atribuindo maior probabilidade de pertencimento à classe “folhas saudáveis”, conforme TAB. 5. Isso pode ter ocorrido devido aos sintomas de deficiência de cálcio se apresentar de forma sutil com um leve amarelamento ao longo da borda da folha sem se estender ao longo das nervuras,

apresentando a maior área da folha com verde mais intenso, o que caracteriza uma folha saudável (MESQUITA, 2016).

TABELA 5 - Probabilidade de classificação para folha com sintomas de deficiência de cálcio.

Classes Café	Probabilidade (%)
Bicho Mineiro	1
Broca	0
Cercospora	1
Deficiência de Cálcio	45
Deficiência de Fósforo	0
Deficiência de Magnésio	0
Deficiência de Nitrogênio	0
Deficiência de Potássio	0
Ferrugem	0
Folha Saudável	53
Phoma	0

Fonte: Autoria Própria.

FIGURA 44 - Classificação de folha com sintomas de deficiência de nitrogênio.

Fonte: Nelson Sobrinho (2012).

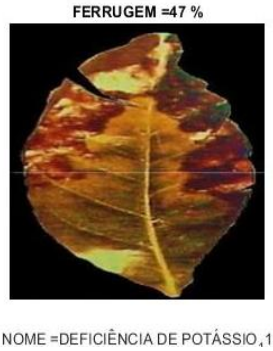
A imagem da FIG. 44 pertencente à classe “deficiência de nitrogênio” foi classificada incorretamente, sendo atribuída maior probabilidade de pertencimento à classe “deficiência de cálcio”, TAB. 6. Isso pode ter ocorrido devido à qualidade ruim da imagem e ao destaque das nervuras com presença de clorose em seu entorno, que podem caracterizar a classe predita pela rede neural (MESQUITA, 2016).

TABELA 6 - Probabilidade de classificação para folha com deficiência de nitrogênio.

Classes Café	Probabilidade (%)
Bicho Mineiro	0
Broca	0
Cercospora	0
Deficiência de Cálcio	96
Deficiência de Fósforo	0
Deficiência de Magnésio	4
Deficiência de Nitrogênio	0
Deficiência de Potássio	0
Ferrugem	0
Folha Saudável	0
Phoma	0

Fonte: Autoria Própria

FIGURA 45 - Classificação de folha com sintomas de deficiência de potássio.



Fonte: Nelson Sobrinho (2012).

Na imagem da FIG. 45 pertencente à classe “deficiência de potássio” houve confusão com a classe “ferrugem”, atribuindo-lhe maior probabilidade, TAB. 7. Esta imagem também apresenta má qualidade e mostra em sua maior parte, tons alaranjados e manchas mais escuras em sua extremidade, que podem indicar sintomas da presença de ferrugem (AGROLINK, 2017).

TABELA 7 - Probabilidade de classificação para folha com deficiência de potássio.

Classes Café	Probabilidade (%)
Bicho Mineiro	0
Broca	0
Cercospora	0
Deficiência de Cálcio	1
Deficiência de Fósforo	1
Deficiência de Magnésio	0
Deficiência de Nitrogênio	2
Deficiência de Potássio	45
Ferrugem	47
Folha Saudável	0
Phoma	4

Fonte: Autoria Própria.

FIGURA 46 - Classificação de folha com sintomas de deficiência de ferrugem.

Fonte: Fugimoto (2011).

Na imagem da FIG. 46, houve confusão com maior probabilidade para a classe “deficiência de fósforo”, TAB. 8. O erro de classificação pode ter ocorrido devido à presença de mancha escura e manchas amareladas, que podem indicar sintomas iniciais da deficiência indicada pela rede neural (MESQUITA, 2016).

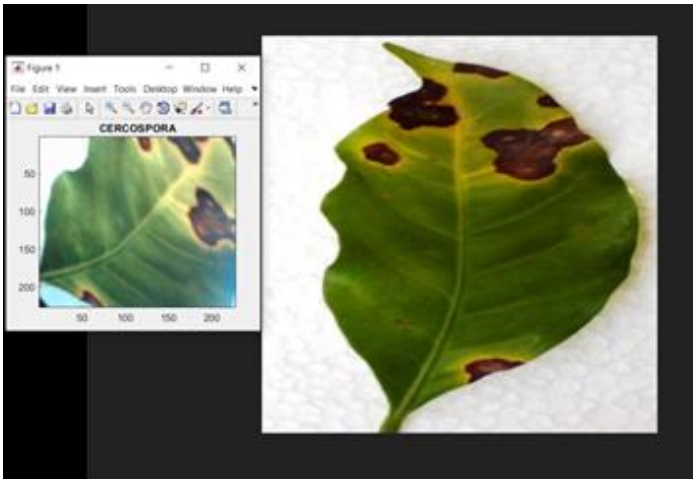
TABELA 8 - Probabilidade de classificação para folha com incidência de ferrugem.

Classes Café	Probabilidade (%)
Bicho Mineiro	0
Broca	0
Cercospora	22
Deficiência de Cálcio	0
Deficiência de Fósforo	78
Deficiência de Magnésio	0
Deficiência de Nitrogênio	0
Deficiência de Potássio	0
Ferrugem	0
Folha Saudável	0
Phoma	0

Fonte: Autoria Própria.

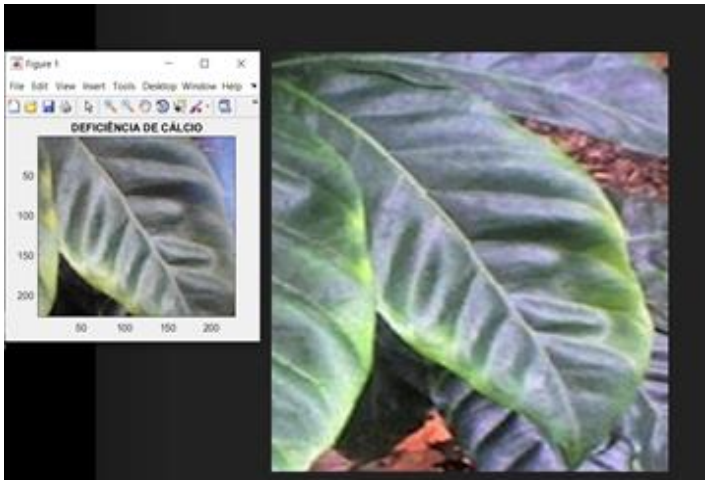
Algumas das predições incorretas da RNC deste projeto podem ser solucionadas realizando algumas modificações nas imagens. Com o classificador dinâmico, por exemplo, é possível utilizar a câmera para focar nos sintomas das deficiências minerais e pragas e obter a indicação da classe correta. Esse procedimento gerou classificações corretas, como por exemplo, para as imagens da FIG. 47, 48 e 51 conforme FIG. 52, 53 e 54.

FIGURA 47 - Classificação da imagem da FIG.47 com o classificador dinâmico.



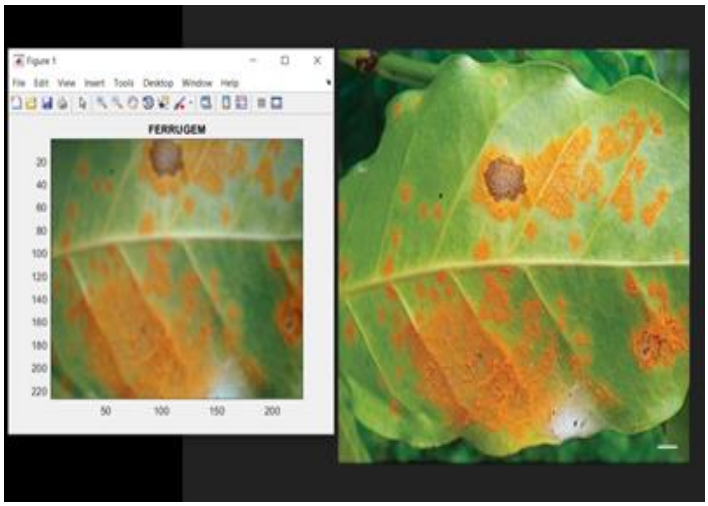
Fonte: Marangoni e Lima (2010).

FIGURA 48 - Classificação da imagem FIG.48 com o classificador dinâmico.



Fonte: Lira ([20--]).

FIGURA 49 - Classificação da imagem FIG. 51 com o classificador dinâmico.



Fonte: Fugimoto (2011).

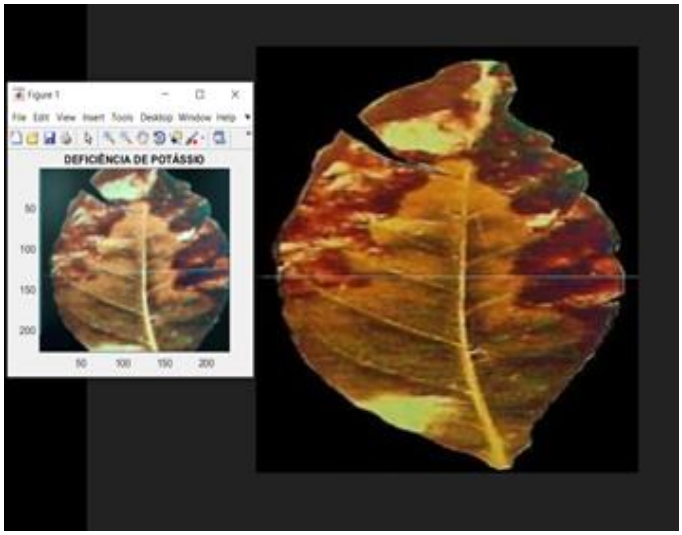
Nas imagens da FIG. 49 e FIG. 50 não foi possível obter um bom resultado com o classificador estático devido à baixa qualidade (resolução) das imagens e às más condições em que as imagens foram digitalizadas. Essas imagens foram mantidas propositalmente para simular problemas durante a captura de imagens em campo.

Conforme FIG. 55 e FIG. 56 com auxílio do classificador dinâmico, melhorando condições de iluminação, contraste e reposicionando a câmera, com foco nos sintomas das deficiências, foi possível obter a classificação correta.

FIGURA 50 - Classificação da imagem FIG. 49 com o classificador dinâmico.

Fonte: Nelson Sobrinho (2012).

FIGURA 51 - Classificação da imagem FIG. 50 com o classificador dinâmico.



Fonte: Nelson Sobrinho (2012).

5 CONSIDERAÇÕES FINAIS

O desenvolvimento do presente estudo possibilitou uma análise de como a visão computacional em conjunto com a aprendizagem profunda podem ajudar a diagnosticar sintomas visuais de deficiências minerais e pragas no cafeeiro, cogitando a aplicabilidade das Redes Neurais Convolucionais como possível solução para este problema.

A aplicação do método de aprendizagem de transferência, utilizando as camadas iniciais como extratores de características e alterando as camadas finais da rede neural convolucional AlexNet pré-treinada disponível no *software* MATLAB® se mostrou uma boa solução para um banco de imagens pequeno e distinto do banco de imagens utilizado para treinar a rede original, conseguindo classificar 83,76% das imagens usadas para teste com o classificador estático. A má qualidade das imagens contribuiu para erros de classificação, sendo utilizado o classificador dinâmico como tentativa de solucionar este impasse, ao reposicionar a câmera e focalizar nos sintomas, sendo assim possível obter a classificação correta para algumas imagens.

Para melhores resultados seria necessário adquirir mais imagens para ampliar o banco de dados e fazer um pré-processamento nas imagens adquiridas. Além disso, orientações de um especialista na cultura cafeeira seriam de grande valia, pois poderiam destacar com mais exatidão as áreas sintomáticas das folhas ou frutos da planta atingida por moléstias.

Conclui-se que as redes neurais convolucionais são viáveis para a extração de características e classificação dos sintomas de deficiências minerais e pragas nas imagens das folhas e frutos do café, podendo ser um passo importante para uma diagnose visual mais precisa, sem erros de interpretações humanas.

5.1 Trabalhos Futuros

Para trabalhos futuros sugere-se aumentar o número de classes, pois algumas deficiências e pragas não foram inclusas neste trabalho, devido à dificuldade de adquirir imagens. Além disso, cada classe pode ser subdividida pelo nível de severidade das deficiências ou pragas, disponibilizando, automaticamente, sugestões para resolver o problema e evitar que o mesmo se alastre.

REFERÊNCIAS

AGROBYTE. **Pragas do Café.** [20--]. Disponível em: <http://www.agrobyte.com.br/cafe.htm>.Acesso em: 27 jun. 2018.

AGROLINK. **Ferrugem do Cafeeiro**. 2017. Disponível em: https://www.agrolink.com.br/problemas/ferrugem-do-cafeeiro_1532.html. Acesso 27 jun. 2018.

AGROLINK. **Olho pardo**. 2017. Disponível em: https://www.agrolink.com.br/problemas/olho-pardo_1579.html 2017. Acesso em: 27 jun. 2018.

BIOCONTROLE. Métodos de Controle de Pragas Ltda. **Bicho-mineiro-do-café/Leucoptera Coffella**, [20--]. Disponível em: http://www.biocontrole.com.br/produto/bicho-mineiro-do-cafe-leucoptera-coffeella/**.** Acesso em: 27 jun.2018.

BORTH, M.R. et al. **A Visão Computacional no Agronegócio: Aplicações e Direcionamentos.** 2014. Disponível em: http://www.gpec.ucdb.br/pistori/publicacoes/borth_ecaeco2014.pdf>. Acesso em: 30 abr. 2018.

BRYS, L.M. **Processamento Digital de Imagens**. Centro Estadual de Pesquisas em Sensoriamento Remoto e Meteorologia. UFRGS, [20--]. Disponível em: http://www.ufrgs.br/engcart/PDASR/pdi.html. Acesso em: 07 nov. 2018.

CARVALHO, A. P. L. F. de. **Redes Neurais Artificias.** Departamento de Ciência da Computação - Universidade de São Paulo, 2009. Disponível em: http://conteudo.icmc.usp.br/pessoas/andre/research/neural/. Acesso: 30 abr. 2018.

CASTRO, L.N.; ZUBEN, F.J.V. **Redes Neurais Artificiais**. Campinas: FEEC, UNICAMP, 1998. Disponível em: ftp://calhau.dca.fee.unicamp.br/pub/docs/vonzuben/ia006_03/topico5_03.pdf. Acesso em: 30 abr. 2018.

COMSTOR AMERICAS. **O que é Visão Computacional?** 2017. Disponível em: https://blogbrasil.comstor.com/o-que-e-visao-computacional. Acesso em: 24 abr.2018.

CONSELHO DOS EXPORTADORES DE CAFÉ DO BRASIL. **História do Café.** Disponível em: http://www.cecafe.com.br/sobre-o-cafe/historia-do-cafe/. Acesso em: 26 mar. 2018.

CONSELHO DOS EXPORTADORES DE CAFÉ DO BRASIL. **Produção**. Disponível em: http://www.cecafe.com.br/sobre-o-cafe/producao/. Acesso em: 26 mar. 2018.

CORSO, D.A. et al. **Extração de Características Baseadas em Forma Para o Reconhecimento de Padrões em um Sistema de Visão Computacional.** Pontifícia Universidade Católica, [20--]. Disponível em: http://www.ppgia.pucpr.br~alceu/pdi/Momentos%20HU/Artigo%20Diego%20e%20Rubens%20-%20Hu.pdf. Acesso em: 02 mai. 2018.

DIAS, F.G. et al. **Estudo da aplicação de filtros de detecção de bordas na identificação da frente termal da Corrente do Brasil**. Instituto Nacional de Pesquisas Espaciais, 2011. Disponível em: http://marte.sid.inpe.br/rep/dpi.inpe.br/marte/2011/07.29.17.38?mirror=dpi.inpe.br/banon/2003/12.10.19.30.54&metadatarepository=dpi.inpe.br/marte/2011/07.29.17.38.15. Acesso em: 07 nov. 2018.

EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIA. **Principais pragas do cafeeiro no contexto do manejo integrado de culturas.** 2015. Disponível em: https://www.embrapa.br/busca-de-noticias/-/noticia/6694669/artigo---principais-pragas-do-cafeeiro-no-contexto-do-manejo-integrado-de-pragas.Acesso em: 27 jun. 2018.

FAQUIN, V. **Diagnose do estado nutricional das plantas**. 2002. Curso de Pós-Graduação “Latu Sensu” (Especialização) a Distância: Fertilidade do Solo e Nutrição das Plantas no Agronegócio. Universidade Federal de Lavras e Fundação de Apoio ao Ensino Pesquisa e Extensão, 2002. Disponível em: http://www.dcs.ufla.br/site/adm/upload/file/pdf/Prof_Faquin/Diagnose%20do%20Estado%20Nutricional%20das%20Plantas.pdf>. Acesso em: 15 abr. 2018.

FERREIRA, E.V. **Gradiente descendente (batch, stochastic e boosting).** Universidade Federal do Paraná, [20--]. Disponível em: http://www.leg.ufpr.br ~eferreira/CE064/gradiente%20descendente.pdf. Acesso em: 07 nov. 2018.

FLORINDO, J.B. **Tópico 10 – Redes Neurais Convolucionais – Deep Learning**. Universidade Estadual de Campinas, [20--]. Disponível em: https://www.ime.unicamp.br/~jbflorindo/Teaching/2018/MT530/T10.pdf. Acesso em: 07 nov. 2018.

FUGIMOTO, G. **Pesquisadores da UFV esclarecem mistério sobre ferrugem no café**. Rede Agronomia - Rede dos Engenheiros Agrônomos do Brasil, 2011. Disponível em:http://agronomos.ning.com/profiles/blogs/pesquisadores-da-ufv-esclarecem-misterio-sobre-ferrugem-no-cafe. Acesso em: 26 nov. 2018.

GONZALEZ, R. C.; WOODS, R. E. **Processamento Digital de Imagens.** 3. ed. São Paulo: Pearson Prentice Hall, 2010. 624 p.

INTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ. **Aula 10 - Cultura do Café (Coffea arabica L.)-II.** [200-]. Disponível em: http://proedu.ifce.edu.br/bitstream/handle/123456789/579/Aula_10.pdf?sequence=10&isAllowed=y. Acesso em: 30 abr. 2018.

JÚNIOR, Willer Gomes et al. **Filtros de Detecção de Borda: Sobel.** [20--]. Disponível em: https://docs.gimp.org/2.8/pt_BR/plug-in-sobel.html**.** Acesso em: 07 nov. 2018.

JÚNIOR, Willer Gomes et al.**Matriz de Convolução.** [20--]. Disponível em: https://docs.gimp.org/2.8/pt_BR/plug-in-convmatrix.html**.** Acesso em: 07 nov. 2018.

KARPATHY, A. **Convolutional Neural Networks:** Archtectures, Convolution/Pooling Layers.CS231n Convolutional Neural Networks Recognition, 2015. Disponível em: http://cs231n.github.io/convolutional-networks/. Acesso: 07 nov. 2018.

KRIZHEVSKY, A. et al. **ImageNet Classification with Deep Convolutional Neural Networks** .Disponível em:https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf. Acesso em: 25 jun. 2018.

LEITE, G.L.D.; MOTA, V.A. **Pragas do Cafeeiro. Universidade Federal de Minas Gerais.** [20--].Disponível em:https://halley.adm-serv.ufmg.br/ica/wp-content/uploads/2017/06/Pragas_de_cafe.pdf**.** Acesso em: 27 jun.2018.

LIRA, E. A. **Nutrición foliar en el cultivo de Café**. [20--] Disponível em: https://slideplayer.es/slide/10365986/. Acesso em: 26 nov. 2018.

MARANGONI, R. E; LIMA, M. L. P. **Doenças Incidentes na Cultura do Café (Coffea sp.)**.2010. Disponível em: https://fitopatologia1.blogspot.com/2010/12/doencas-incidentes-na-cultura-do-cafe.html. Acesso em: 26 nov.2018.

MARTINS, A. G. **Avanços na nutrição para o café conilon.** [20--]. Disponível em:https://pt.slideshare.net/cafeicultura/andr-guaroni-minicurso-avanos-na-nutrio-para-o-caf-conilon**.** Acesso em: 27 nov. 2018.

MATHWORKS. **MATLAB - The Language of Technical Computing**. 2018. Disponível em: https://www.mathworks.com/help/matlab/. Acesso em: 28 nov.2018.

MATIELLO, J. B. **Cercosporiose no cafeeiro, doença secundária, mas importante.** 2018. Disponível em:https://www.cafepoint.com.br/noticias/tecnicas-de-producao/cercosporiose-no-cafeeiro-doenca-secundaria-mas-importante-207823/.Acesso em: 27 jun. 2018.

MESQUITA, Carlos Magno de et al. **Manual do café: manejo de cafezais em produção**. Belo Horizonte: EMATER - MG, 2016. 72p. il.

MINISTÉRIO DA AGRICULTURA, PECUÁRIA E ABASTECIMENTO. **Café no Brasil**. Disponível em:http://www.agricultura.gov.br/assuntos/politica-agricola/cafe/cafeicultura-brasileira. Acesso em: 26 mar. 2018.

MIYAZAKI, C.K. **Redes Neurais Convolucionais para aprendizagem e reconhecimento de objetos 3D**. 2017. Monografia, Universidade de São Paulo, 2017. Disponível em: file:///C:/Users/cliente/Downloads/Miyazakicaio_tcc%20(10).pdf. Acesso em: 07 nov. 2018.

NETO, Sérgio Arthur de La Hidalga Crespo. **Reconhecimento de Tumores Cerebrais Utilizando Redes Neurais Convolucionais**. Universidade Federal do Pampas, 2017. Disponível em: http://dspace.unipampa.edu.br:8080/bitstream/riu/1911/1/Sergio%20Arthur%20de%20La%20Hidalga%20Crespo%20Neto%20-%202017.pdf.Acesso em: 7 nov. 2018.

NICOLAU, L. **Redes Neurais Artificiais**. Universidade Federal da Paraíba, [20--]. Disponível em: http://www.cear.ufpb.br/juan/wp-content/uploads/2016/08/Aula-3c-Fun%C3%A7%C3%B5es-de-Ativa%C3%A7%C3%A3o-e-Gradiente-Descendente.pdf. Acesso em: 07 nov. 2018.

PACHECO, A. **Redes Neurais Convolucionais - CNNs.** Computação Inteligente - Máquinas aprendendo a solucionar problemas complexos, 2017. Disponível em: http://www.computacaointeligente.com.br/artigos/redes-neurais-convolutivas-cnn/. Acesso em: 02 mai. 2018.

PALMIERE, S. E. **Arquiteturas e Topologias de Redes Neurais Artificiais**. EMBARCADOS, 2016. Disponível em: https://www.embarcados.com.br/redes-neurais-artificiais/. Acesso em: 7 nov. 2018.

PIXABAY. **Folha – natureza – preto e branco**. Disponível em: https://pixabay.com/pt/folha-natureza-preto-e-branco-1979281/. Acesso em: 07 nov. 2018.

PROMIP - MANEJO INTEGRADO DE PAGRADAS. **Broca-do-café: saiba como fazer o monitoramento correto da praga.** 2007. Disponível em:http://promip.agr.br/blog/2018/03/broca-do-cafe-saiba-como-fazer-o-monitoramento-correto-da-praga**.** Acesso em: 27 nov. 2018.

REVISTA ATTALEA AGRONEGÓCIOS. **Inseticida reduz em 76% a incidência do Bicho-Mineiro no Café.** 2018. Disponível em: https://revistadeagronegocios.com.br/inseticida-reduz-em-76-a-incidencia-do-bicho-mineiro-no-cafe/.Acesso em: 27 ago. 2018.

REVISTA CAFEICULTURA. **Identificação de Sintomas, Deficiências e Excessos no Cafeeiro.** 2005. Disponível em: http://revistacafeicultura.com.br/?mat=3621. Acesso em: 24 abr.2018.

REZENDE, R. **Nutrição do cafeeiro - Adubação começa com análise do solo**. REVISTA CAMPO E NEGÓCIOS, 2015. Disponível em: http://www.revistacampoenegocios.com.br/nutricao-do-cafeeiro-adubacao-comeca-com-analise-do-solo/. Acesso em: 27 mar.2018.

ROSSETO, R.; SANTIAGO, A. D. **Diagnose visual.** Agência Embrapa de Informação Tecnológica, [20--]. Disponível em: http://www.agencia.cnptia.embrapa.br/gestor/cana-de-acucar/arvore/CONT000fhvuyvaq02wyiv80v17a09ehcm584.html. Acesso em: 15 abr. 2018.

SOBRINHO, Nelson. M. **Manejo da Fertilidade dos Solos e Adubação Equilibrada para a Cultura do Café**. 2012. Disponível em: https://pt.slideshare.net/cafeicultura/adubao-cafeeiro-manejo-da-fertilidade-dos-solos-e-adubao-equilibrada-para-cultura-do-caf. Acesso em: 26 nov. 2018.

SOUZA, G. S. de. **Extração de Feições e Características.** [200-]. Disponível em: https://webserver2.tecgraf.puc-rio.br/~mgattass/ra/trb09/Guilherme/VisaoComputacional%20-%20Trabalho%201.htm.Acesso: 30 abr. 2018.

SZEGEDY, C. et al. **Rethinking the Inception Architecture for Computer Vision.** [200-]. Disponível em: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf. Acesso em: 30 abr. 2018.

SOUZA, A.F. **Manejo integrado da Mancha de Phoma no cafeeiro. 2007.** Disponível em:https://www.cafepoint.com.br/noticias/tecnicas-de-producao/manejo-integrado-da-mancha-de-phoma-no-cafeeiro-35756n.aspx. Acesso em: 27 jun.2018.

SOUZA, F. et al. **Características das principais variedades de café cultivadas em Rondônia.** Porto Velho: Embrapa Rondônia, 2004. 21 p.

SOLOMON, C.; BRECKON, T. **Fundamentos de processamento digital de imagens: uma abordagem prática com exemplos em Matlab.** Rio de Janeiro: LTC, 2013. 289 p.

SILVA, R. M.; ALMEIDA, T.A; YAMAKAMI, A. Análise de desempenho de redes neurais artificiais para classificação automática de web spam. **Revista Brasileira de Computação Aplicada, Passo Fundo**, v. 4, n. 2, p. 42-57, out. 2012.

USP. **Extração de Características de Imagens – Descritores de cor**. [20--]. Disponível em: https://edisciplinas.usp.br/pluginfile.php/1364034/mod_resource/content/1/Representa%C3%A7%C3%A3oCor.pdf. Acesso em:07 nov. 2018.

UNIVERSIDADE FEDERAL FLUMINENSE. **Capítulo 5 - Filtragem de Imagens**. [20--]. Disponível em: http://computacaografica.ic.uff.br/transparenciasvol2cap5.pdf. Acesso em:07 nov. 2018.

VOLTOLINI, G. et al. **Fertilidade e nutrição do cafeeiro: quebrando paradigmas - Parte 1**. CAFEPOINT, 2015. Disponível em: https://www.cafepoint.com.br/radares-tecnicos/nucleo-de-estudos-de-cafeicultura-ufla/fertilidade-e-nutricao-do-cafeeiro-quebrando-paradigmas-parte-1-96114n.aspx. Acesso em: 27 mar. 2018.

VELLASCO, M. M. B. R. **Redes Neurais Artificiais.** Pontifícia Universidade Católica do Rio de Janeiro, 2007. Disponível em: http://www2.ica.ele.puc-rio.br/Downloads/33/ICA-introdu%C3%A7%C3%A3o%20RNs.pdf. Acesso em: 30 abr. 2018.

VARGAS, A. C. G. et al. **Um estudo sobre Redes Neurais Convolucionais e sua aplicação em detecção de pedestres.** UFF [200-]. Disponível em: http://gibis.unifesp.br/sibgrapi16/eproceedings/wuw/7.pdf. Acesso em: 07 nov. 2018.

VADHADIYA, P.; FORTUNER, B. **Loss Functions**. 2017. Disponível em: https://ml-cheatsheet.readthedocs.io/en/latest/index.html. Acesso em: 07 nov. 2018.

YARA BRASIL. **Deficiências-Café**. 2018. Disponível em: https://www.yarabrasil.com.br/nutricao-de-plantas/cafe/deficiencias---cafe/. Acesso em: 26 nov. 2018.

YARA BRASIL. **Manejo das plantas de café e proteção da cultura**. [20--] Disponível em: https://www.yarabrasil.com.br/nutricao-de-plantas/cafe/manejo-das-plantas-de-cafe-e-protecao-da-cultura/. Acesso em: 26 nov. 2018.

ZÚÑIGA, A.M.G. **Sistema de Visão Artificial para Identificação do Estado Nutricional de Plantas.** 2012. 142p. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação, USP, 2012. Disponível em: file:///C:/Users/cliente /Downloads/AlvaroGomezZuniga%20(2).pdf. Acesso: 30 abr. 2018.

APÊNDICE A

FUNÇÃO: *GerarImagens*

```
function  GerarImagens(nome) %  Momentos Invariantes
f = imreAd(strcat(nome,'.jpg'));
% clear;close all;clc;
nome = 'Enxofre 1.jpg'; f=imread(nome);
f = tofloat(f);
f = imresize(f,[400,400]);         % Redimensiona a imagem
fp = padarray(f,[84,84],'both');   % coloca moldura na imagem
nome0 = 'Original com Moldura';
Imag(fp,nome0);                    % Mostra a imagem
MomentosInvariantesCor(fp,nome0);  % Calcula os momentos invariantes para
cadacomponente R G B

[Mp,Np,d] = size(fp);
ftr = zeros(Mp,Np,1,'single'); % Cria matrizes com zeros em cada componente e
nasdimensões da imagem
ftr = zeros(Mp,Np,2,'single');
ftr = zeros(Mp,Np,3,'single');


% Copia a componente da imagem para a posição deslocada de 84 linhas para
% baixo e 84 colunas para a direita sobre a matriz de zeros
ftr(85:1:Mp+84,85:1:Np+84,1) = fp(:,:,1);
ftr(85:1:Mp+84,85:1:Np+84,2) = fp(:,:,2);
ftr(85:1:Mp+84,85:1:Np+84,3) = fp(:,:,3);


% junta as 3 componenetes de cor
ftrans=cat(3,ftr(:,:,1),ftr(:,:,2),ftr(:,:,3));
nome1='Transladada';
Imag(ftrans,nome1);
fp2 = imresize(ftrans,[227,227]); % Redimensiona para treinar a rede
% imwrite(fp2,strcat(nome,'_2.jpg')); Salva a imagem para treinar a rede
```

```
MomentosInvariantesCor(ftrans,nome1);

% Retira a moldura lateral direita e inferior
ftra=(ftrans(1:1:568,1:1:568,:));
nome1_1='Transladada Canto';
Imag(ftra,nome1_1);
fp3=imresize(ftra,[227,227]);
% imwrite(fp3,strcat(nome,'_3.jpg'));
MomentosInvariantesCor(ftra,nome1_1);
%
fhs = f(1:2:end,1:2:end,:); %"Pula" uma linha e uma coluna para reduzir
fhsp = padarray(fhs,[184 184],'both');
nome2='Reduzida';
Imag(fhsp,nome2);
fp4=imresize(fhsp,[227,227]);
% imwrite(fp4,strcat(nome,'_4.jpg'));
MomentosInvariantesCor(fhsp,nome2);

fm  = fliplr(f);
fmp = padarray(fm,[84 84],'both');
nome3='Espelhada';
Imag(fmp,nome3);
fp5=imresize(fmp,[227,227]);
% imwrite(fp5,strcat(nome,'_5.jpg'));
MomentosInvariantesCor(fmp,nome3);
% Rotaciona +45° graus utilizando interpolação bilinear
fr45 = imrotate(f,45,'bilinear');
nome4='Rotacionada +45°';
Imag(fr45,nome4);
fp6=imresize(fr45,[227,227]);
% imwrite(fp6,strcat(nome,'_6.jpg'));
MomentosInvariantesCor(fr45,nome4);


% Rotaciona +90° graus utilizando interpolação bilinear
fr90 = imrotate(f,90,'bilinear');
```

```
fr90p = padarray(fr90,[84,84],'both');
nome5='Rotacionada +90°';
Imag(fr90p,nome5);
fp7=imresize(fr90,[227,227]);
% imwrite(fp7,strcat(nome,'_7.jpg'));
MomentosInvariantesCor(fr90p,nome5);
```

APÊNDICE B

FUNÇÃO: *MomentosInvariantesCor*

```
function MomentosInvariantesCor(f,name)
[~,~,d]=size(f);
if d>=3
    r=f(:,:,1);
    g=f(:,:,2);
    b=f(:,:,3);
end
I = invmoments(r);
Mi = -sign(I).*(log10(abs(I)));
figure,imshow(r);
nomer=strcat(name,'Componente Vermelho');
title({(['Imagem ( ',(nomer),') '])});
xlabel({([' Momentos Invariantes = (',num2str(Mi),') '])})

I = invmoments(g);
Mi = -sign(I).*(log10(abs(I)));
nomeg=strcat(name,'Componente Verde');
figure,imshow(g);
title({(['Imagem ( ',(nomeg),') '])});
xlabel({([' Momentos Invariantes = (',num2str(Mi),') '])})

I = invmoments(b);
Mi = -sign(I).*(log10(abs(I)));
nomeb=strcat(name,'Componente Azul');
figure,imshow(b);
title({(['Imagem ( ',(nomeb),') '])});
xlabel({([' Momentos Invariantes = (',num2str(Mi),') '])})
```

FUNÇÃO: *Invmoments*

```
function phi = invmoments(F)
%INVMOMENTS Compute invariant moments of image.
%   PHI = INVMOMENTS(F) computes the moment invariants of the image
%   F. PHI is a seven-element row vector containing the moment
%   invariants as defined in equations (11.3-17) through (11.3-23) of
%   Gonzalez and Woods, Digital Image Processing, 2nd Ed.
%
%   F must be a 2-D, real, nonsparse, numeric or logical matrix.

%   Copyright 2002-2009 R. C. Gonzalez, R. E. Woods, and S. L. Eddins
%   From the book Digital Image Processing Using MATLAB, 2nd ed.,
%   Gatesmark Publishing, 2009.
%
%   Book web site: http://www.imageprocessingplace.com
%   Publisher web site: http://www.gatesmark.com/DIPUM2e.htm

if (ndims(F) ~= 2) || issparse(F) || ~isreal(F) || ...
        ~(isnumeric(F) || islogical(F))
   error(['F must be a 2-D, real, nonsparse, numeric or logical' ...
         'matrix.']);
end

F = double(F);

phi = compute_phi(compute_eta(compute_m(F)));

%--------------------------------------------------------------------%
function m = compute_m(F)

[M, N] = size(F);
[x, y] = meshgrid(1:N, 1:M);
```

```
% Turn x, y, and F into column vectors to make the summations a bit
% easier to compute in the following.
x = x(:);
y = y(:);
F = F(:);


% DIP equation (11.3-12)
m.m00 = sum(F);
% Protect against divide-by-zero warnings.
if (m.m00 == 0)
   m.m00 = eps;
end
% The other central moments:
m.m10 = sum(x .* F);
m.m01 = sum(y .* F);
m.m11 = sum(x .* y .* F);
m.m20 = sum(x.^2 .* F);
m.m02 = sum(y.^2 .* F);
m.m30 = sum(x.^3 .* F);
m.m03 = sum(y.^3 .* F);
m.m12 = sum(x .* y.^2 .* F);
m.m21 = sum(x.^2 .* y .* F);


%-----------------------------------------------------------------%

function e = compute_eta(m)

% DIP equations (11.3-14) through (11.3-16).

xbar = m.m10 / m.m00;
ybar = m.m01 / m.m00;

e.eta11 = (m.m11 - ybar*m.m10) / m.m00^2;
```

```
e.eta20 = (m.m20 - xbar*m.m10) / m.m00^2;
e.eta02 = (m.m02 - ybar*m.m01) / m.m00^2;
e.eta30 = (m.m30 - 3 * xbar * m.m20 + 2 * xbar^2 * m.m10) / ...
          m.m00^2.5;
e.eta03 = (m.m03 - 3 * ybar * m.m02 + 2 * ybar^2 * m.m01) / ...
          m.m00^2.5;
e.eta21 = (m.m21 - 2 * xbar * m.m11 - ybar * m.m20 + ...
         2 * xbar^2 * m.m01) / m.m00^2.5;
e.eta12 = (m.m12 - 2 * ybar * m.m11 - xbar * m.m02 + ...
2 * ybar^2 * m.m10) / m.m00^2.5;


%-----------------------------------------------------------------%

function phi = compute_phi(e)

% DIP equations (11.3-17) through (11.3-23).

phi(1) = e.eta20 + e.eta02;
phi(2) = (e.eta20 - e.eta02)^2 + 4*e.eta11^2;
phi(3) = (e.eta30 - 3*e.eta12)^2 + (3*e.eta21 - e.eta03)^2;
phi(4) = (e.eta30 + e.eta12)^2 + (e.eta21 + e.eta03)^2;
phi(5) = (e.eta30 - 3*e.eta12)*(e.eta30 + e.eta12)* ...
        ((e.eta30 + e.eta12)^2 - 3*(e.eta21 + e.eta03)^2 ) + ...
        (3*e.eta21 - e.eta03)*(e.eta21 + e.eta03) * ...
        (3*(e.eta30 + e.eta12)^2 - (e.eta21 + e.eta03)^2 );
phi(6) = (e.eta20 - e.eta02) * ((e.eta30 + e.eta12)^2 - ...
        (e.eta21 + e.eta03)^2 )+ ...
        4 * e.eta11*(e.eta30 + e.eta12)*(e.eta21 + e.eta03);
phi(7) = (3*e.eta21 - e.eta03)*(e.eta30 + e.eta12)*...
        ((e.eta30 + e.eta12)^2 - 3*(e.eta21 + e.eta03)^2 ) + ...
        (3*e.eta12 - e.eta30)*(e.eta21 + e.eta03)* ...
        (3*(e.eta30 + e.eta12)^2 - (e.eta21 + e.eta03)^2 );
```

APÊNDICE C

FUNÇÃO: Retreinando_ALEXNET

```
function [netPiCafe] = Retreinando_ALEXNET(ne)
% Retreinando_ALEXNET Usando a rede pré treinada ALEX NET
% N° de épocas de retreinamento (ne)
Rede = alexnet;  % Carrega a rede ALEXNET


% dimEntrada = Dimensão das imagens utilizadas para treinar a rede original
% As imagens para retreinar necessáriamente devem ter as mesmas dimensões
dimEntrada = Rede.Layers(1).InputSize;


% Carrega as imagens p/ Treino e Validação
todasImg = imageDatastore('CafeRec','IncludeSubfolders',true,...
'LabelSource','foldernames');


% Separa aleatóriamente 80% das imagens para treinar a rede
[imgTreino,imgValidacao] = splitEachLabel(todasImg,.8,'randomize');


numClasses = numel(categories(imgTreino.Labels)); % Número de classes

% Reconfigurando as 3 últimas camadas para o novo número de classes
% camadas totalmente conectadas
camadas_de_Transferencia = Rede.Layers(1:end-3);


camadas = [
    camadas_de_Transferencia
    fullyConnectedLayer(numClasses,'WeightLearnRateFactor',20,...
    'BiasLearnRateFactor',20)
    softmaxLayer
    classificationLayer];
```

```
numImgTreino = numel(imgTreino.Labels);
idx = randperm(numImgTreino,16);
figure
for i = 1:16
    subplot(4,4,i)
I = readimage(imgTreino,idx(i));
    imshow(I)
end


% Images aumentadas e reconfigurada para minimizar um sobre treinamento
pixelRange = [-30 30];
imgAumentada = imageDataAugmenter( ...
    'RandXReflection',true, ...
    'RandXTranslation',pixelRange, ...
    'RandYTranslation',pixelRange);


% Imagens p/ treino reconfiguradas e redimensionadas para
% as dimensões da rede
imgTreinoRedim = augmentedImageDatastore(dimEntrada(1:2),imgTreino, ...
    'DataAugmentation',imgAumentada);


%  Imagens p/ validação reconfiguradas e redimensionadas para
% as dimensões da rede
imgValidacaoRedim = augmentedImageDatastore(dimEntrada(1:2),imgValidacao);

% Opções para Retreinar a rede
opcao = trainingOptions('sgdm','CheckpointPath',...
'C:\Users\Cliente\Desktop\CafeRec',...
    'InitialLearnRate',1e-3,'MaxEpochs',ne,'MiniBatchSize',64,...
    'ValidationData',imgValidacaoRedim, ...
```

```
    'Plots','training-progress');
%'C:\Users\Cliente\Desktop\CafeRec', Salva todos os passos do
    % treinamento para uso posterior


% Treinamento da Rede
redeRet = trainNetwork(imgTreinoRedim,camadas,opcao);
% redeRet = rede Retreinada



% Medindo a acuracia da rede
[predCamadas,~] = classify(redeRet,imgValidacaoRedim);

acuracia = mean(predCamadas == imgValidacao.Labels);
disp(acuracia)

%save('netPiCafe1','net')
netPiCafe=redeRet;
```

APÊNDICE D

FUNÇÃO: Classificador Estático

```
function [Classe] = classificadorCafe(nome,net)
nnet1 = net; % Carrega a rede retreinada
dir=strcat('C:\Users\Cliente\Dropbox\Análise Imagens\CAFÉ_1/Análise\',nome,'.jpg');
f = imread(dir);
f = imresize(f,[227,227]); % Redimensionar as imagens


[Classe,Prob] = classify(nnet1,f); % Classifica utilizando a rede retreinada
% Classe = Uma das 11 categorias
% Probaliidade percentual = Vetor com a probalilidade de cada categoria
Probabilidade=round(Prob*100)';
%disp(Classe)
disp('    ')


% Caso não seja 100% a probabilidade de uma das classes, mostra o vetor
% probabilidade como tabela
if max(Probabilidade)<100
CLASSES_CAFE={'BICHO MINEIRO';'BROCA';'CERCOSPORA';'D. CÁLCIO';'D.
FÓSFORO';'D. MAGNÉSIO';'D. NITROGÊNIO';'D. POTÁSSIO';'FERRUGEM';'F.
SAUDÁVEL';'PHOMA'};
T = table(CLASSES_CAFE,Probabilidade);
disp(T)
end


% Plota a imagem com a classe e a probabilidade e no "x" o nome da imagem
Name=strcat('NOME = ',nome);
classes = strcat(char(Classe),' = ',num2str(round(max(Prob)*100)),' %');
figure, imshow(f),
title(classes),xlabel(Name)
```

FUNÇÃO: Classificador Dinâmico

```
function [classe] = classificadorDinamico(nome)

cam = webcam(2);%conecta a câmera
nnet1 = net; % Carrega a rede retreinada
while true
Imag = cam.snapshot; %Imagem da Câmera para análise
Imag = imresize(Imag,[227,227]); % Redimensiona a imagem
image(Imag)
label = classify(nnet1,Imag); % Classifica
title(char(label))
drawnow

end
```