VISÃO COMPUTACIONAL PARA DETECÇÃO DE DOENÇAS FÚNGICAS NA AGRICULTURA

Geferson Galvão[1]
Wendel Carvalho[2]
Willian Rocha[3]
Orientador: Prof. Júlio Cézar da Silva Costa[4]

Resumo: A tecnologia vem sendo aplicada em diversas áreas, devido a sua ampla gama de recursos, algumas atividades se tornam mais simples com seu auxilio. A agricultura também adotou a tecnologia em algumas de suas subáreas. Neste ramo é encontrada a "Agricultura de precisão" uma pratica agrícola, que mesmo fazendo uso de suas técnicas de tecnologia da informação, ainda há agricultores sofrendo com os fungos e pragas que devastam plantações em pouco tempo, ocasionando enormes prejuízos. O trabalho descrito tem por objetivo, ajudar e auxiliar os agricultores no combate contra duas doenças cancerígenas muitos frequentes em plantações de café, a cercosporiose e a ferrugem. Com o auxílio de técnicas de visão computacional, aprendizado de máquina através de redes neurais e reconhecimento de padrões desenvolvemos um algoritmo capaz de identificar plantas que contenham as doenças mencionadas. A partir deste diagnóstico fornecido pelo algoritmo, o agricultor terá condições de tomar as medidas necessárias para tratamento de sua plantação evitando prejuízos futuros.

Palavras Chave: Visão computacional, redes neurais, inteligência artificial.

Abstract: The technology has been applied in several areas, due to its wide range of resources, some activities become simpler with your help. Agriculture has also adopted technology in some of its subareas. In this branch is found the "Precision agriculture" an agricultural practice, that even making use of its techniques of the information, there are still farmers suffering with the fungi and plagues that devastate plantations in a short time, causing enormous damages. The objective of this paper is to help and assist farmers in combating two carcinogenic diseases that are common in coffee plantations, cercosporiosis and rust. With the aid of computer vision techniques, machine learning through neural networks and pattern recognition, we have developed an algorithm capable of identifying plants that contain the mentioned diseases. From this diagnosis provided by the algorithm, the farmer will be able to take the necessary measures to treat his plantation avoiding future losses.

Keywords: Computer vision, neural networks, artificial intelligence.

[1] Bacharel em Ciência da Computação – Faculdade Única, geferson.fp@hotmail.com.
[2] Bacharel em Ciência da Computação – Faculdade Única, w_cn@hotmail.com.
[3] Bacharel em Ciência da Computação – Faculdade Única, willianrocha@gmail.com.
[4] Bacharel em Sistemas da Informação – Unileste, Ipatinga, MG, MBA Executivo de Gerenciamento de Projetos – Fundação Getúlio Vargas, Docente na Faculdade Única, jotaulio@hotmail.com.

1. INTRODUÇÃO

A computação é de extrema importância para a indústria em geral, naturalmente técnicas de inteligência artificial são cada vez mais utilizadas nas diversas áreas existentes.

Através de visão computacional, robôs industriais, por exemplo, podem se situar e identificar com eficiência padrões defeituosos, trabalhando como um mecanismo de "escolha" e seleção de produtos.

Em alguns casos é utilizado também algoritmos de aprendizado de máquina para que os robôs possam reconhecer novos padrões não especificados previamente.

A inteligência artificial (IA) é uma ciência recente e pode ser dividida em diversas subáreas, didaticamente possui diversas definições e gera implicações filosóficas que podem fugir ao tema proposto. Derivando o termo inteligência a uma das características como: racionalidade é possível ter uma compreensão mais elementar, um sistema é racional se "faz tudo certo" com os dados que tem (NORVIG 2004).

Assim como a IA compreende uma área de estudo entorno do que é inteligência e em meios de reproduzi-la sinteticamente, a visão computacional se organiza dentro dos estudos da computação voltada a fazer a máquina "enxergar" ou ao menos ter percepção dos objetos à sua volta através de programação.

Visto que a visão é um dos principais sentidos humanos e nos permite perceber onde estamos e tomar decisões sobre o que faremos a partir do que vemos, é natural criar um mecanismo que possa oferecer as máquinas um "sentido" semelhante.

Notoriamente os estudos e aplicações sobre IA e visão computacional se complementam principalmente na área da robótica que está cada vez mais presente no dia a dia, robôs industriais já utilizam de ambas as áreas para melhorar seu desempenho, e são utilizadas em larga escala para controle de qualidade.

Portanto, neste trabalho, desenvolvemos um algoritmo para reconhecer padrões criados por fungos, como Cercosporiose, causadora da mancha de olho pardo e *Hemileia Vastatrix*, responsável pela ferrugem.

Esses padrões foram escolhidos inicialmente por poderem ser encontrados em diferentes tipos de plantas e comuns em viveiros, onde, o espaço é mais limitado, podendo contribuir para a realização dos testes da aplicação proposta neste trabalho.

2. REFERÊNCIAL TEÓRICO

2.1. Fungos na agricultura

Fungos e infecções são famosos por destruírem grande parte e trazerem consigo enormes prejuízos aos agricultores. Plantas geneticamente fortalecida são criadas, mas suportam por pouco tempo a esses "predadores".

Um fator comum entre fungos, vírus e bactérias é causar alterações no metabolismo, na cor da folhagem, alterações nos órgãos e anatomia. Alguns sinais são pó branco nas folhagens, bolores cinzentos ou pretos, bolhas cor de ferrugem, uma massa ou crescimento preto, pintas pretas, o aparecimento de cogumelos, entre outros. Sendo estes os padrões de pesquisas propostos pelo algoritmo a ser desenvolvido.

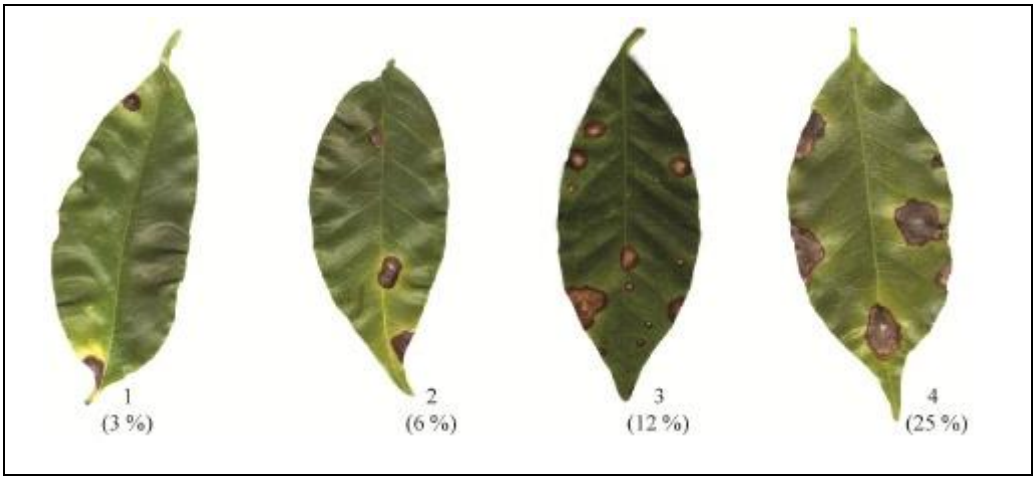
2.1.1. Cercosporiose

Cercosporiose também conhecida como mancha de olho pardo, apesar de ter surgido como uma doença secundária tem se tornado uma das principais preocupações para agricultores em geral, porém mais especificamente no café se tornou alvo de combate imediato por não só danificar como diminuir a produção das plantas também podendo se expandir na plantação. "A mancha de olho pardo é uma doença grave em cafeeiros estabelecidos em plena exposição solar ou com nutrição deficiente" (SOUZA, A F, 2003).

O ataque dos fungos ocorre principalmente em fase de desenvolvimento das plantas, nos primeiros três anos e em viveiros. Ainda não existem variedades resistentes a doença, e o controle é feito basicamente envolvendo adubação equilibrada e controle químico, principalmente com fungicidas cúpricos aplicados no período de dezembro a março (SOUZA, A F, 2003).

A figura 1 representa a escala de severidade da evolução da cercosporiose nas folhas de café.

Figura 1: Escala de severidade da cercosporiose.



Fonte: (Marcelo, 2016), Modificado.

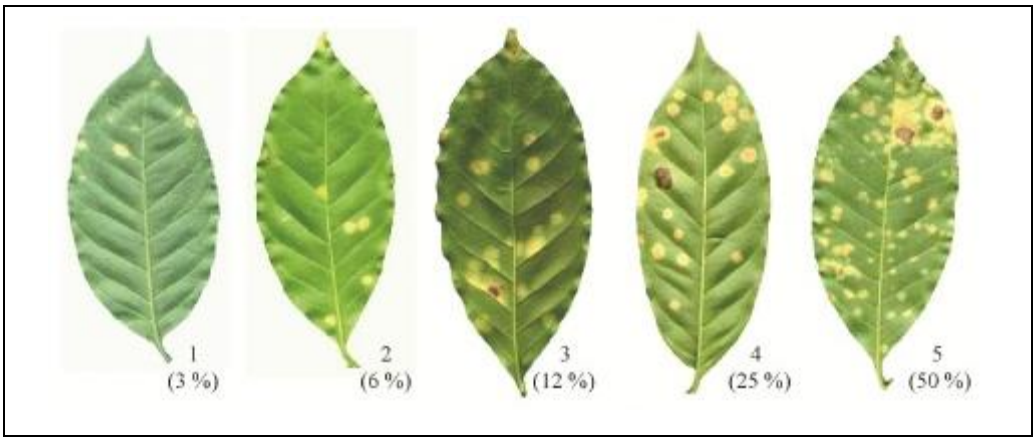
2.1.2. Ferrugem

A ferrugem tem como seus sintomas iniciais pontos escuros no tecido das folhas com uma coloração cinza, devido ao hábito biotrófico do fungo (nutre-se do tecido vivo das plantas) e as células infectadas morrem somente após ter ocorrido abundante esporulação e por essas características iniciais a ferrugem pode também ser facilmente confundida com as lesões iniciais de mancha parda (*Septoria glycines Hemmi*) que forma um halo amarelo ao redor da lesão necrótica, que é angular e castanho-avermelhada (EMBRAPA, 2011).

"A infecção por P. pachyrhizi causa rápido amarelecimento ou bronzeamento e queda prematura das folhas, impedindo a plena formação dos grãos. Quanto mais cedo ocorrer a desfolha, menor será o tamanho dos grãos e, consequentemente, maior a perda do rendimento e da qualidade (grãos verdes) " (EMBRAPA, 2011). Causando sérios danos nas plantações seja por queda de produção ou custo de manejo, além de possuir fácil mecanismo de proliferação o que a torna uma doença que deve ser cuidadosamente analisada nas plantações tanto de café como milho, soja e cana-de-açúcar.

A figura 2 representa a escala de severidade da evolução da ferrugem nas folhas de café.

Figura 2: Escala de severidade da ferrugem.



Fonte: (Marcelo, 2016), Modificado.

2.2. Inteligência artificial

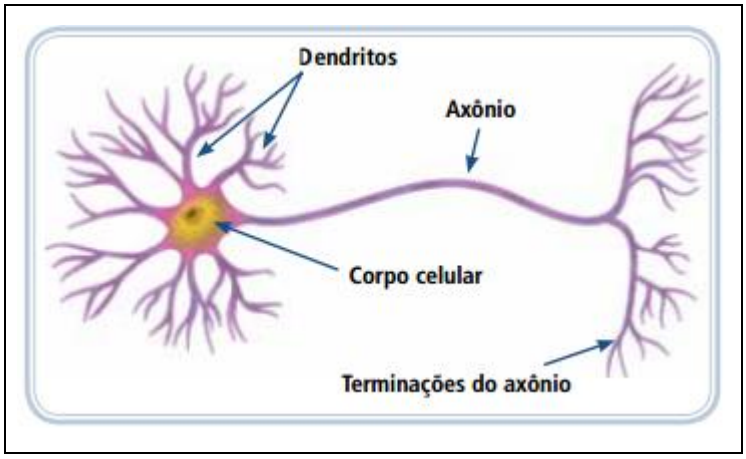
Inteligência artificial pode ser definida ao longo de duas principais definições, tais como processos de pensamento e raciocínio e as que endereçam comportamento. Os seguintes pensamentos se enquadram no âmbito de definição de processos de pensamento e raciocínio, "O excitante novo esforço para fazer computadores pensarem […], máquinas com mentes, no sentido literal e completo. " (HAUGELAND, 1985) e "O estudo de faculdades mentais por meio do uso de modelos computacionais. " (CHARNIAK E., MCDERMOTT, 1985). Os pensamentos que se enquadram no âmbito de definição comportamental são "O estudo de como fazer computadores realizarem coisas em que, no momento, pessoas são melhores. " (RICK E KNIGHT, 1991) e "Inteligência Computacional é o estudo do projeto de agentes inteligentes. " (POOLE et al., 1998). Mediante tais definições concluímos que um sistema é racional se ele faz a "coisa certa", dado o que ele sabe.

2.3. Redes neurais artificiais

Uma rede neural artificial (RNA) é um mecanismo de processamento de dados e foi nomeado dessa forma, pois possui algumas características de funcionamento em comum com as redes neurais biológicas. E devido a esse fato as RNA's tem seu estudo de forma multidisciplinar, envolvendo pesquisadores de diversas áreas, como neurofisiologia, psicologia, física, computação e engenharia (DE CASTRO SILVA, 1998).

A figura 3 abaixo possui a representação de um neurônio fisiológico.

Figura 3: Neurônio fisiológico.



Fonte: Corrêa, 2016.

O neurônio é a unidade funcional básica do sistema nervoso. Na figura 3 acima está representado três das características principais de um neurônio: (1) os dendritos; (2) o corpo celular neuronal; e (3) o axônio. Dos quais podem ser definidos como: dendritos são projeções finas do corpo neuronal que se estendem nas áreas circunvizinhas e que recebem o impulso que chega ao neurônio. Este sinal segue então por uma extensão única que é o axônio. Este, por sua vez, é capaz de se ramificar em várias terminações que irão fazer sinapses com os neurônios seguintes (CORRÊA, 2016).

Segundo Leandro Nunes (1998) as redes neurais artificiais são desenvolvidas como generalizações de modelos matemáticos da cognição humana ou biologia neural, dessa forma, possui algumas das características a seguir:

• o processamento da informação ocorre em elementos chamados neurônios;

• a informação é propagada de um elemento (neurônio artificial) a outro através de conexões;

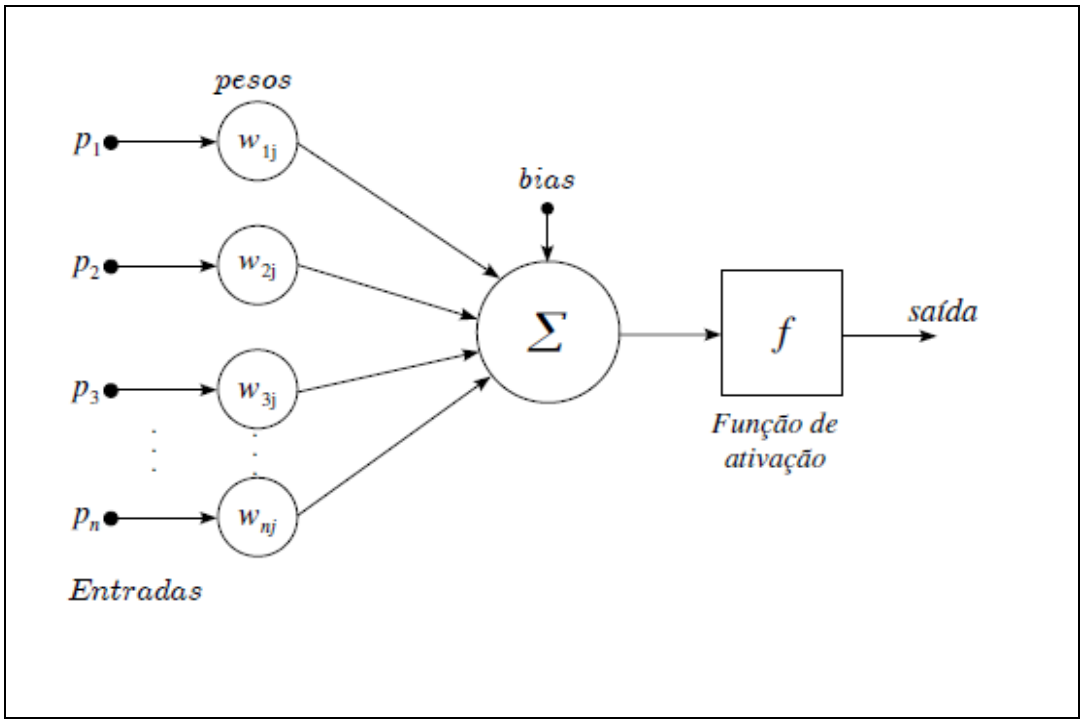
• para cada conexão há um peso associado, em uma rede neural típica, onde esse peso modula o sinal de saída que é passada, ou não, para o próximo neurônio;

• um neurônio possui uma função de ativação (normalmente não linear) que é aplicada à sua entrada de rede para determinar sua saída.

2.3.1. Representação de um neurônio artificial

O modelo artificial de neurônio é mostrado na figura 4 abaixo, sendo que esta é uma generalização do modelo de *McCulloch e Pitts* de 1943 esse modelo inclui um sinal adicional bias (b) que favorece ou limita a possibilidade de ativação do neurônio (MATSUNAGA, 2012).

Figura 4: Modelo artificial de neurônio.



Fonte: Matsunaga, 2012.

O processo sináptico é representado pelos pesos (w) que são utilizados para amplificarem cada um dos sinais recebidos. A chamada função de ativação (f) modela a forma como o neurônio responde ao nível de excitação, limitando e definindo a saída da rede neural. Sendo que a função de ativação pode ter diferentes representações (MATSUNAGA, 2012).

Matematicamente temos:

$$y = \Theta\left(\sum_{j=1}^{D} w_j x_j - \mu\right)$$

Onde y representa a rede neural e cada entrada está associada um peso Wj que reflete a importância da entrada Xj, que interage com o limiar μ e o neurônio "dispara" o valor de saída, se não ultrapassar o limiar a saída pode ficar passiva em y=0, porém o valor de saída é ponderado de acordo com a função de ativação Θ (RAUBER, 2005).

Uma rede neural possui 3 (três) aspectos que são comuns a todas elas, o padrão de conexões entre suas unidades (arquitetura), o método de determinação dos pesos das conexões, ou seja, algoritmo de treinamento ou aprendizado (Elemento de processamento) e também sua função de ativação, que determina o valor a ser representado na saída (DE CASTRO SILVA, L. N.,1998).

2.4. Visão Computacional

A Visão Computacional é uma recente área da ciência, responsável pela "visão" de máquinas e robôs, ela dedica a desenvolver teorias e métodos voltados à extração de informações significativas de uma imagem ou do ambiente ao redor, possibilitando reconhecer, manipular e analisar os objetos que compõem esse meio. Ela tem a capacidade de extrair informações relevantes a partir de imagens capturadas por câmeras fotográficas, vídeos, sensores, entre outros dispositivos, para automatizar a tomada de decisão em um sistema, por exemplo, (BORTH, APUD SHAPIRO ET AL., 2001). Tem como entrada de dados uma imagem e, como saída, a interpretação parcial ou total dessa imagem (BORTH, APUD MARENGONI ET AL., 2009).

O principal objetivo da Visão computacional é reproduzir a capacidade de reconhecimento de imagens a partir de diversas técnicas computacionais, partindo de imagens e chegando a modelos matemáticos. Para aprimorar as imagens captadas e geradas pesquisadores criaram técnicas com o objetivo de recuperar a forma tridimensional, bem como da aparência de objetos em imagens (BORTH, APUD SZELISKI, 2010).

A visão computacional é uma área de pesquisa que pode incluir métodos de aquisição de imagens, pré-processamento, segmentação, extração de atributos ou características e reconhecimento de padrões.

3. METODOS, TESTES E RESULTADOS

Este trabalho é um algoritmo que utiliza redes neurais para fazer análise de pixels e consequentemente obter resultados de visão computacional, portanto foram feitos diversos testes de técnicas e avaliações de redes neurais. E foi decidido fazer uma análise pixel a pixel das imagens fornecidas ao algoritmo, portanto para obter uma avaliação bem definida, a rede neural utilizada foi obtida sob o método de treinamento supervisionado.

3.1. Funcionamento do programa

Todo processo de treinamento assim como os resultados obtidos e o método para obter a base de dados usados para gerar o treinamento da rede neural será apresentado a seguir. Assim como o desenvolvimento de uma aplicação de suporte para apresentar os resultados ao usuário.

Em uma visão macro, o nosso programa para o reconhecimento de plantas com doenças funciona de maneira simples. Ao receber um arquivo do tipo imagem, automaticamente o algoritmo redimensiona a mesma para o tamanho de 400x550, facilitando trabalhos futuros. Depois de inserida a imagem que se deseja analisar, é selecionada qual tipo de doença será analisada na imagem, pois é a partir dessas informações que acionaremos a rede neural correta para o caso. Após passar essa etapa o algoritmo processará a imagem

Em um primeiro momento é recebido uma imagem a qual se deseja obter as informações solicitadas, a imagem já é recebida com seu tamanho original reduzido para um quadro padrão, para diminuir o tempo de processamento e também facilitar as análises e comparações.

Logo depois de obter a imagem, a mesma é dívida em *pixels*, desses pixels é extraído o RGB (*Red, Green, Blue*), cada pixel da imagem possui um vetor contendo os valores de seus respectivos

RGB's. Cada pixel correspondente a um local especifico da imagem é analisado e comparado com os valores de RGB's da doença que foi selecionada.

Os RBG's das doenças foram previamente obtidos com o auxílio de uma rede neural de 6 neurônios, à qual foi utilizada para efetuar o treinamento ao qual se obteve 95.000 entradas para a doença cercosporiose e 119.700 entradas para ferrugem.

Quando os *pixels* da imagem de entrada e os *pixels* da doença tiverem uma compatibilidade alta, esse *pixel* será identificado como um *pixel* doente, e sua cor serão alterados, a fim de facilitar a sua identificação na imagem de saída.

Os demais pixels onde não foi identificada doença não irão sofrer alterações. Por fim é finalizada essa parte de análise e é devolvida ao usuário a imagem com seus *pixels* doentes com sua coloração alterada e também uma estatística geral das informações extraídas.

3.2. Funções para obtenção dos dados

Para que seja possível o treinamento de uma rede neural, é necessário fazer o levantamento dos dados que serão fornecidos para essa rede, sendo assim foram desenvolvidas funções tanto para obtenção desses dados, que são os valores dos pixels referente ás doenças, e também uma função adicional para exportar essa informação para que seja possível ser repassadas ao ambiente de programação responsável pelo treinamento em si.

Na figura 5, está representado à função utilizada para obtenção dos dados para o treinamento e para que fosse possível fazer o levantamento dessas informações, para isso foi necessário repetir o processo duas vezes ou mais para cada doença, uma para obtenção dos valores referente aos dados positivos, que representam os *pixels* com doenças, e as demais para obter os valores referentes a *pixels* que divergem aos anteriores, ou seja, que não representam doenças.

Figura 5: Função para obter os valores dos pixels.

```
//Método para obter os balores RGB para padrão XLS
public String gerarRGBParaXLS(BufferedImage imagem) {
    String r = "", g = "", b = "", matrizfinal = "";
    if (imagem.getType() == BufferedImage.TYPE_INT_RGB) {
        int[] dataBuff = imagem.getRGB(0, 0, imagem.getWidth(),
                imagem.getHeight(), null, 0, imagem.getWidth());
        for (int i = 0; i < dataBuff.length; i++) {

            b = ((dataBuff[i] >> 0) & 0xFF) + "\t";
            g = ((dataBuff[i] >> 8) & 0xFF) + "\t";
            r = ((dataBuff[i] >> 16) & 0xFF) + "\t";
            matrizfinal = matrizfinal + r + g + b + "0\n";
        }

    }
    return matrizfinal;

}
```

Fonte: do autor (2017).

Nesta figura as imagens são convertidas para um vetor com os valores dos pixels, e os valores referentes a cada tonalidade, vermelho, verde e azul, são convertidos para formato *String* que são novamente transferidos para uma quarta variável denominada matriz final, que é utilizada para armazenar as informações da imagem.

Dessa forma é possível ter um padrão de entradas, que neste caso é semelhante ao utilizado no Microsoft Excel e outros aplicativos que utilizam tabelas como padrão visual.

Para que seja possível um treinamento supervisionado também já está incluso o valor referente à saída esperada da rede para quando encontrar os valores referentes à imagem passados a função, este valor está expresso como 0 (zero) ao fim da função mas pode ser alterado para 1 (um) de acordo com o resultado esperado da informação na imagem.

Essa informação é passada novamente para a função expressa na figura 6, que é responsável por gravar esses dados em um editor de texto.

Figura 6: Função para gravar dados.

```
/*metodo para converter dados da imagem em txt com a matriz*/
public void gerarTxt(String matriz, String CaminhoSaida) throws IOException {

    File arquivo = new File(CaminhoSaida + ".txt"); //se já existir, será sobrescrito
    FileWriter fw = new FileWriter(arquivo);
    BufferedWriter buffWrite = new BufferedWriter(fw);

    buffWrite.write(matriz);
    buffWrite.flush();
    buffWrite.close();
}
```

Fonte: do autor (2017).

Esta função utiliza duas variáveis do tipo String, uma com o vetor ou matriz, com os valores já convertidos no padrão desejado e outra com o caminho de saída que informa onde esses dados serão gravados, portanto ela grava as informações em um formato de editor de texto (txt) no caminho de saída escolhido.

As imagens utilizadas para gerar o treinamento estão expressas abaixo:

A figura 7 representa o recorte de diversas imagens, para que seja possível obter apenas dados das tonalidades de cor referente a doença da cercosporiose no cafeeiro da forma mais exclusiva possível, e os valores das imagens com doença são transmitidas para a rede neural como representações de saídas como 1 (um).

Assim como na imagem dos padrões de cercosporiose, a figura 8 abaixo representa recortes de imagens, porém com o padrão visual da ferrugem e também possui o mesmo propósito.

E a figura 9 representa um conjunto de corte de imagens de plantas sadias em diversas situações para que seja possível uma generalização na rede neural, e seus valores de *pixel* são transmitidos para esta rede como representações de saídas como 0 (zero).

Figura 7: Padrões de Cercosporiose.

Fonte: do autor (2017).

Figura 8: Padrões de Ferrugem.

Fonte: do autor (2017).

Figura 9: Conjunto de corte de imagens de plantas sadias.

Fonte: do autor (2017).

3.3. Testes iniciais no MATLAB

Após diversos testes falhos quanto aos métodos abordados, como a tentativa de fazer uma análise consistente de uma imagem inteira no Matlab e que também fosse possível ser replicada para um algoritmo em Java, foi decidido alterar a abordagem, pois as divergências quanto aos métodos de entrada dos algoritmos causaram diversos transtornos que iriam romper com o cronograma do projeto.

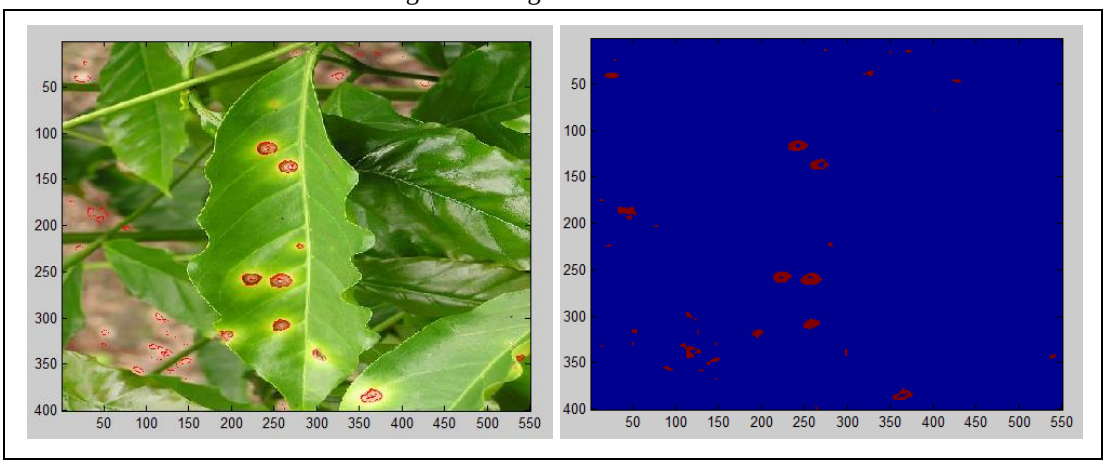
Portanto uma nova abordagem foi escolhida, e o processo de análise passou a ser *pixel* a *pixel*, assim como demonstrado no Diagrama de processamento da imagem, as imagens a seguir apresentam os resultados obtidos nos testes ainda no Matlab e são um conjunto de duas imagens, um original e outra binarizada.

Os testes no Matlab foram focados apenas em uma doença, pois tinha um alto custo de processamento sendo que as imagens demoraram mais de 20 minutos para obter os resultados. E tiveram o objetivo de avaliar o método de análise abordado.

A figura 10 representa um teste com uma rede neural, solucionando o problema de reconhecimento para estágios iniciais e avançados da doença (ambos, presentes na mesma imagem à

direita), porém algumas características do fundo também são reconhecidas, e por isso à rede foi descartada.

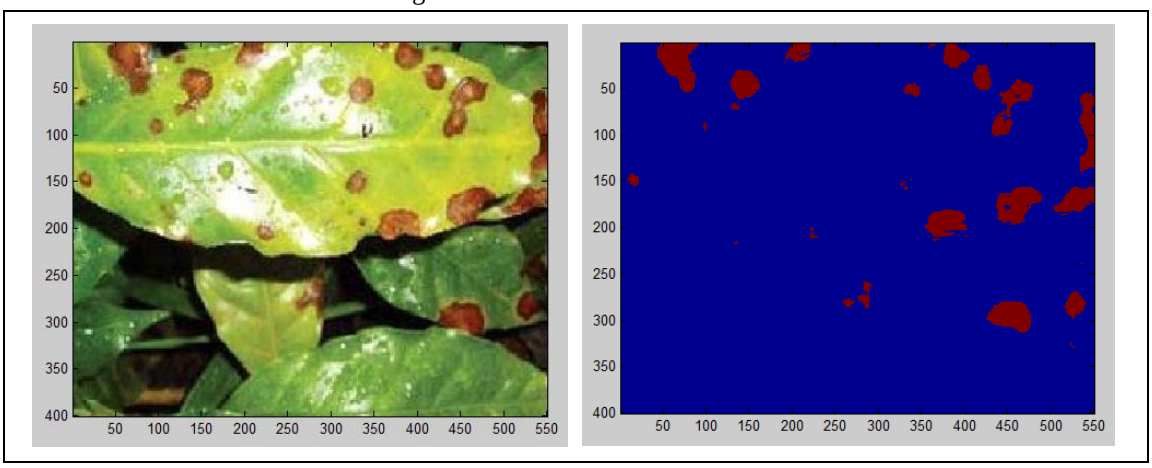
Figura 10: Segundo teste no Matlab.



Fonte: do autor (2017).

A figura 11 abaixo representa um teste mais avançado e evidência a solução de outro problema no processamento da imagem.

Figura 11: Terceiro teste no Matlab.



Fonte: do autor (2017).

Estágios anteriores dessa etapa do treinamento reconheciam caracteres mais escuros, por exemplo, a sombra de uma folha presente na outra (ruído da imagem) como doença porém isso foi contornado elevando o critério de avaliação da rede, onde a mesma possui um retorno como uma variável de ponto flutuante que varia entre zero e um, o critério para um *pixel* ser considerado com doença passa a ser 0.99, ou seja, o mais próximo de 1 possível, dado que um é o valor absoluto do que é um *pixel* com uma doença para a rede.

Devido ao tempo de execução de análise e do prazo expresso no cronograma do projeto os demais testes serão feitos já no protótipo do agente inteligente, uma vez que o método escolhido já apresenta resultados promissores e desse modo será possível iniciar a programação da rede como um algoritmo em Java e efetuar testes para ambas as doenças.

3.4. TESTES DO *SOFTWARE* EM JAVA

O software produzido consiste em um agente inteligente, que apesar de ser um protótipo apresenta resultados consistentes em relação à análise das imagens, sendo que os primeiros resultados demoravam em torno de 2 minutos e meio para completar a análise de uma imagem, mesmo para as imagens menores o algoritmo tinha um custo de processamento muito alto, ainda que menor que o presente no Matlab.

A primeira função utilizada para percorrer à imagem é semelhante a utilizada para obter os valores de *pixel* da imagem para o treinamento, contudo apresentava um alto custo de tempo de processamento.

O processo obtém um buffer com os valores de pixel da imagem e faz a avaliação de cada um através de um laço de repetição e os copia para um segundo buffer auxiliar que será transformado em uma imagem novamente em seguida, essa imagem é apresentada ao usuário.

Os resultados nessa função são expressos a seguir, e para isso as imagens tiveram que ser redimensionadas para 380x200 (largura x altura), para diminuir o tempo de processamento que passaram a ser de 81615 milissegundos aproximadamente a um minuto e vinte e dois segundos, que já é uma melhoria significativa nos resultados, porém as imagens ficaram menores ao serem apresentadas ao usuário.

A figura 12 traz, assim como no Matlab um conjunto de duas imagens, um original e outra com um processo semelhante ao de binarização, e são resultado da execução da função anterior.

Para que os resultados fossem apresentados mais rapidamente foi excluída parte do processamento presente no Matlab que era responsável por circular as partes referentes às doenças na imagem original.

É possível perceber que o reconhecimento da doença foi efetivo, porém parte do galho onde a folha está presente também foi reconhecido como doença.

Figura 12: Primeiro teste em Java.

Fonte: do autor (2017)

3.5. FUNÇÃO DE AVALIAÇÃO DE CERCOSPORIOSE

Com o melhoramento da função de avaliação foi possível obter resultados ainda mais significativos, e quase instantâneos para análise de uma única imagem, e para uma pasta com 56 imagens redimensionadas no próprio algoritmo para o tamanho de 550x400 (largura x altura), foram necessários apenas 32778 milissegundos ou aproximadamente 32 segundos.

A função melhorada para análise da cercosporiose foi obtida eliminando o máximo possível às interações dentro dos laços de repetições, tanto da própria função quanto da rede neural. O código referente à rede neural pode ser visualizado nos anexos, e a função será apresentada na figura 13 abaixo.

Figura 13: Função de avaliação da cercosporiose.

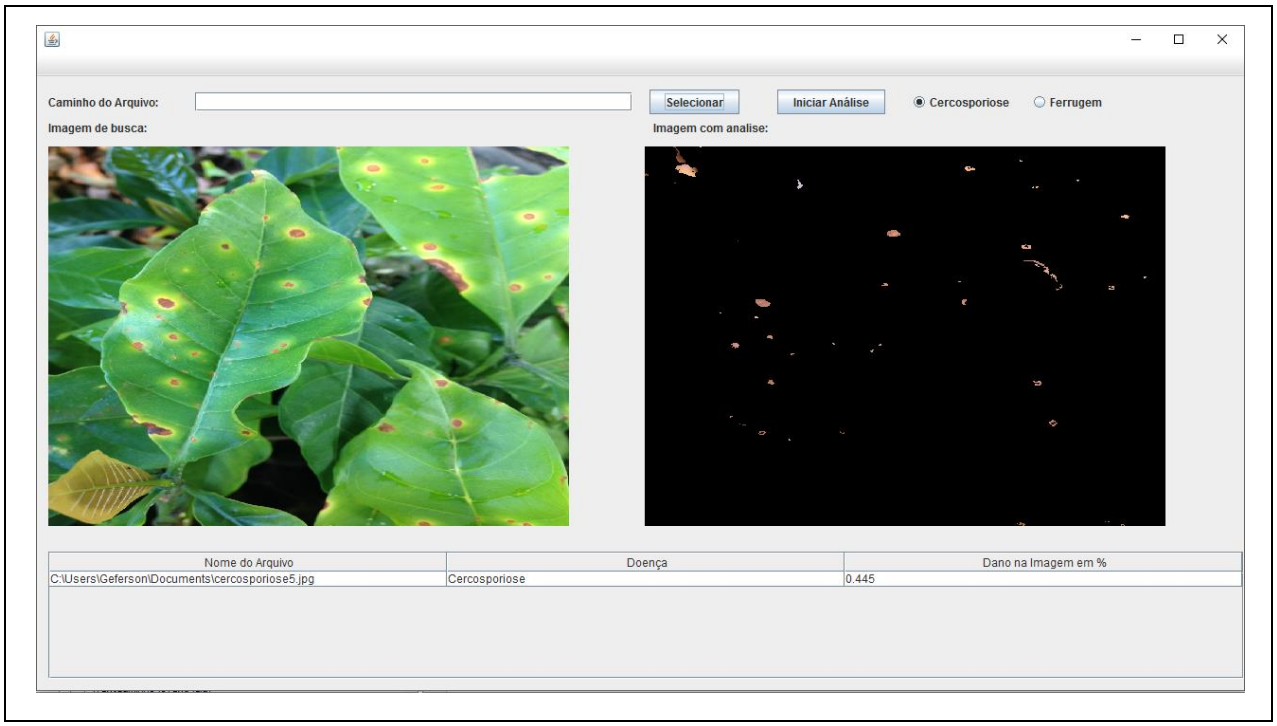
```
//Método para avaliar as imageens com indicios de doença da cercosporiose
    public void AvaliarCercosp(BufferedImage imagem, String EndImagem)
            throws AWTException {

        RedeNeural rede = new RedeNeural();
        double RetRede = 0;
        int erro = 0;
        REC_DIGIT rec = new REC_DIGIT();
        int r = 0, g = 0, b = 0;
        double[] rna = {r, g, b};
        int[] arrayPixel = new int[3];
        int[] arrayVazio = new int[3];
        arrayVazio[0] = 0;
        arrayVazio[1] = 0;
        arrayVazio[2] = 0;
        double cont = 0;
        BufferedImage imagemSaida = new BufferedImage(imagem.getWidth(),
                imagem.getHeight(), BufferedImage.TYPE_INT_RGB);
        WritableRaster raster = imagem.getRaster();
        WritableRaster raster2 = imagemSaida.getRaster();

        for (int i = 0; i < imagem.getWidth(); i++) {
            for (int j = 0; j < imagem.getHeight(); j++) {
                r = raster.getPixel(i, j, arrayPixel)[0];
                g = raster.getPixel(i, j, arrayPixel)[1];
                b = raster.getPixel(i, j, arrayPixel)[2];
                rna[0] = r;
                rna[1] = g;
                rna[2] = b;

                rec.value = rna;
                RetRede = rede.rna_digito(rec, RetRede, erro);
                if (RetRede < 0.999) {
                    raster2.setPixel(i, j, arrayVazio);
                } else {
                    raster2.setPixel(i, j, rna);
                    cont++;
                };
            }
        }
        ImageIcon imagem2;
        imagemSaida.setData(raster2);
        imagem2 = new ImageIcon(imagemSaida);
        ImagemDoenca imagemSalva = new ImagemDoenca();
        imagemSalva.setDoenca(1);
        imagemSalva.setEndImg(EndImagem);
        imagemSalva.setImagemBin(imagem2);
        double perc = (cont / (imagem.getHeight() * imagem.getWidth()));
        imagemSalva.setPercDano(perc * 100);
        listaImagem.add(imagemSalva);
        jLabelFoto1.setIcon(imagem2);
    }
```

Fonte: Do autor (2017)

Assim como na função anterior a imagem é percorrida *pixel* a *pixel* e seus valores são transferidos para uma segunda imagem que será salva, juntamente com todas as informações que serão transmitidas ao usuário, em um Arraylist próprio para isso.

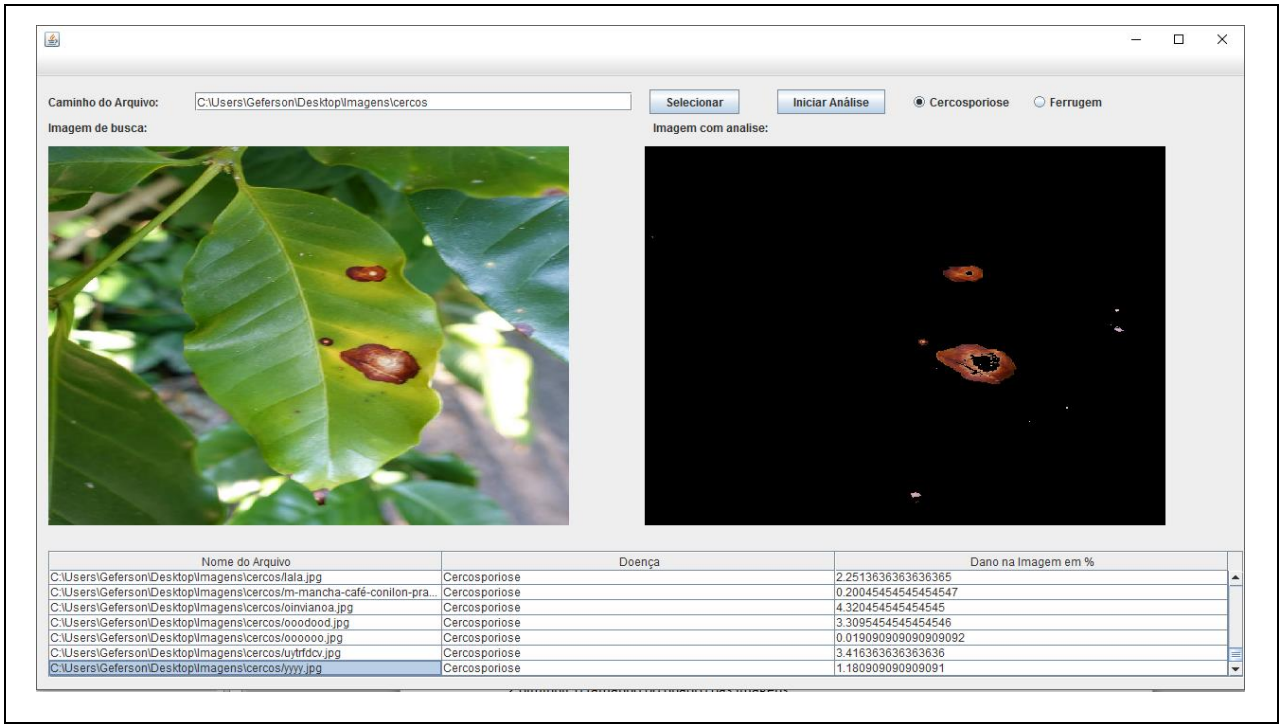
Os resultados dessa função serão expressos nas figuras 14 e 15 abaixo.

Figura 14: Resultado para uma imagem.



Fonte: do autor (2017).

Figura 15: Resultado para um arquivo.



Fonte: do autor (2017).

Para a execução de um arquivo as análises podem ser alteradas ao clicar em qualquer parte da lista das imagens. E devido à grande taxa de *pixels* presentes em uma imagem o percentual de dano, que representam o percentual de *pixels* com doença em relação ao total de *pixel* da imagem.

3.6. FUNÇÃO DE AVALIAÇÃO DA FERRUGEM

A função de avaliação de ferrugem se assemelha a de cercosporiose sendo que a única coisa a se alterar é a rede neural utilizada para retornar o resultado referente à doença da ferrugem, foi criada uma nova função ao invés de incluir uma condicional para execução de uma rede ou outra para que se possam minimizar as interações dentro de um laço de repetição, com a intenção de diminuir o tempo de execução do programa e para o caso de alterações futuras das funções elas possam passar por manutenções de forma independente.

Como expresso anteriormente após obter uma função que se adequasse bem ao método escolhida ela seria fácil de ser aplicada para ambas as doenças. A figura 16 a seguir demonstra a função utilizada para avaliação da ferrugem.

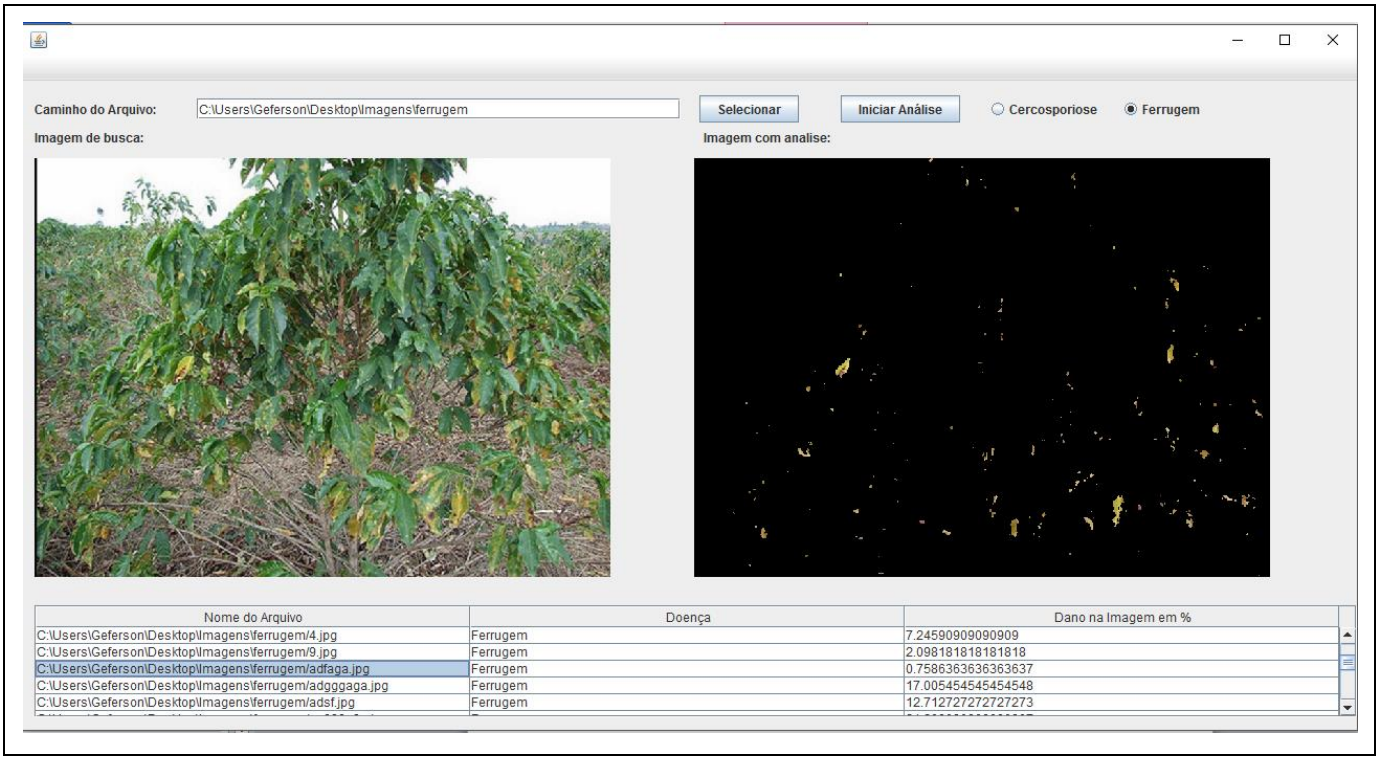
Figura 16: Função de avaliação da ferrugem.

```
//Método para avaliar as imagens com indicios de doença da ferrugem
public void AvaliarFerrugem(BufferedImage imagem, String EndImagem)
        throws AWTException {

    RNAFerrugem rede = new RNAFerrugem();
    double RetRede = 0;
    int erro = 0;
    REC_DIGIT rec = new REC_DIGIT();
    int r = 0, g = 0, b = 0;
    double[] rna = {r, g, b};
    int[] arrayPixel = new int[3];
    int[] arrayVazio = new int[3];
    arrayVazio[0] = 0;
    arrayVazio[1] = 0;
    arrayVazio[2] = 0;
    double cont = 0;
    BufferedImage imagemSaida = new BufferedImage(imagem.getWidth(),
            imagem.getHeight(), BufferedImage.TYPE_INT_RGB);
    WritableRaster raster = imagem.getRaster();
    WritableRaster raster2 = imagemSaida.getRaster();
    for (int i = 0; i < imagem.getWidth(); i++) {
        for (int j = 0; j < imagem.getHeight(); j++) {
            r = raster.getPixel(i, j, arrayPixel)[0];
            g = raster.getPixel(i, j, arrayPixel)[1];
            b = raster.getPixel(i, j, arrayPixel)[2];
            rna[0] = r;
            rna[1] = g;
            rna[2] = b;

            rec.value = rna;
            RetRede = rede.rna_digito(rec, RetRede, erro);
            if (RetRede < 0.9) {
                raster2.setPixel(i, j, arrayVazio);
            } else {
                raster2.setPixel(i, j, rna);
                cont++;
            };
        }
    }
    ImageIcon imagem2;
    imagemSaida.setData(raster2);
    imagem2 = new ImageIcon(imagemSaida);
    ImagemDoenca imagemSalva = new ImagemDoenca();
    imagemSalva.setDoenca(1);
    imagemSalva.setEndImg(EndImagem);
    imagemSalva.setImagemBin(imagem2);
    double perc = (cont / (imagem.getHeight() * imagem.getWidth()));
    imagemSalva.setPercDano(perc * 100);
    listaImagem.add(imagemSalva);
    jLabelFoto1.setIcon(imagem2);
}
```

Fonte: do autor (2017).

Os resultados obtidos por essa função estão expressos a seguir, e se demonstram eficientes uma vez que a rede neural utilizada não reconhece parte da terra ou dos galhos das plantas como parte da doença assim como apresentado na figura 17 a seguir.

Figura 17: Teste da rede neural de ferrugem.

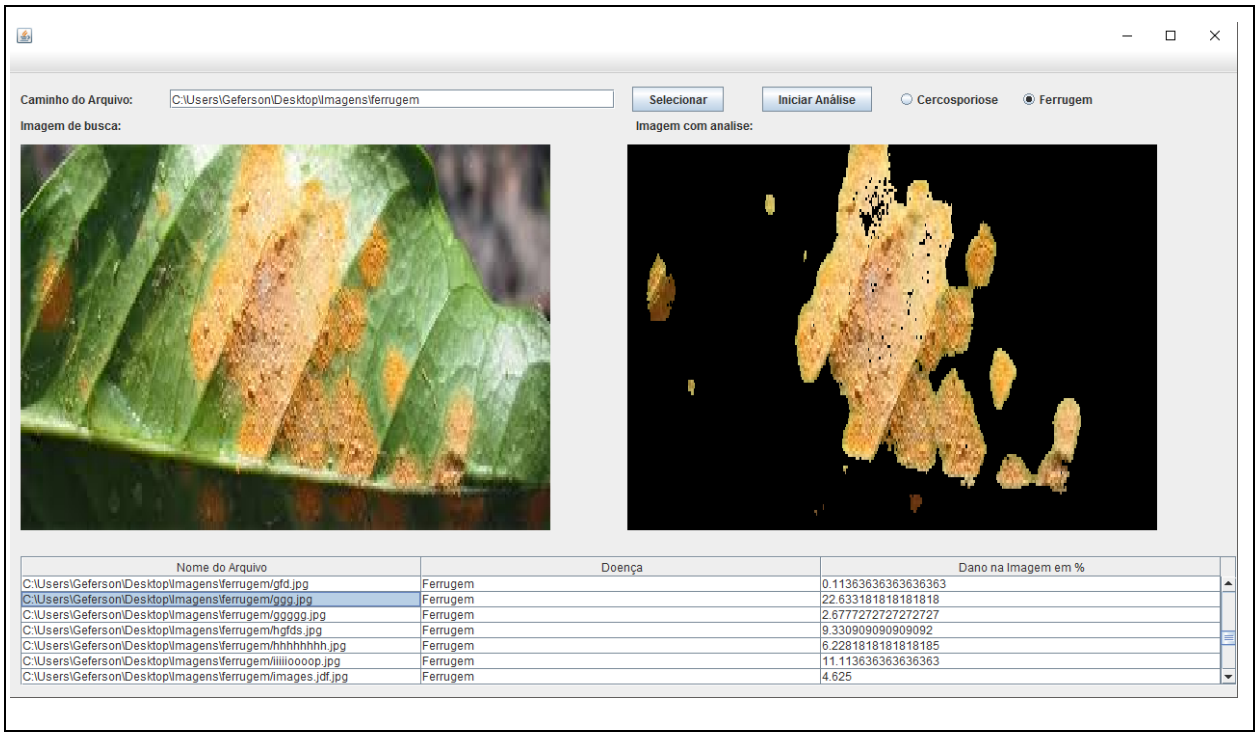


Fonte: do autor (2017).

Apesar da imagem ser de uma planta de médio porte ainda que muito comprometida com a doença da ferrugem o resultado da análise foi eficaz, uma vez que se esperava que o algoritmo funcionasse mais eficientemente para imagens de curta distância e com uma nitidez maior, porém o nível de abstração da rede foi bom o suficiente para evitar os ruídos na imagem e reconhecer apenas as características da doença. A figura 18, representa o resultado para uma única folha.

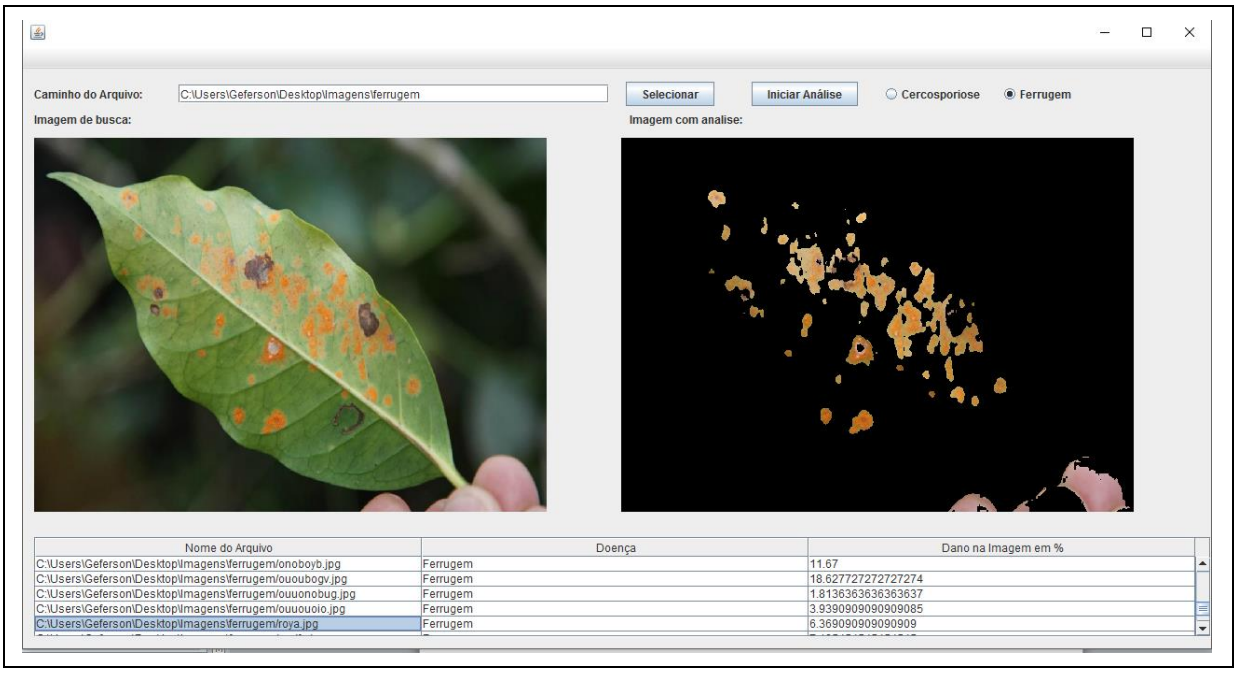
O tempo total de execução total para o diretório que continha 71 imagens com tamanho 550x400(largura x altura) foi de 55327 milissegundos, aproximadamente 55(cinquenta e cinco) segundos, porém assim como demonstrado na figura 19, mesmo para um resultado muito bom nos testes feitos pelo próprio Matlab para a rede, não foi possível extrair o processo de seleção as tonalidades referentes à mão e pele humana, pois possuem altos índices de vermelho e se encaixam perfeitamente nas abstrações de ambas as redes, tanto de cercosporiose como de ferrugem.

Figura 18: Analise de ferrugem em imagem em um diretório.



Fonte: do autor (2017).

Figura 19: Ponto falho do processo de treinamento.



Fonte: do autor (2017).

3.7. REQUISITOS DO AGENTE INTELIGENTE

Apesar de ter um ótimo nível de abstração e o método utilizado aparentemente demonstrar resultados para doenças as doenças de cercosporiose e ferrugem em outras plantas, a rede neural foi treinada especificamente para essas doenças em folhas da arvore do cafeeiro.

Devido ao tipo de abstração das redes reconhecerem algumas tonalidades da pele humana, passa a ser requisito do algoritmo que nas imagens para a análise seja evitado a presença de humanos, e também é possível que algumas tonalidades de terra assim como de pele possam interferir no resultado obtido.

Apesar de não ser usual a presença de folhas secas também pode interferir na análise, uma vez que parte da ação da doença seja extrair seus nutrientes, o que faz com que as folhas sequem no local afetado.

Em relação a isso o algoritmo pode ser mais eficiente na análise de plantas e viveiros, onde o ambiente é controlado e as plantas estão quase sempre muito próximas, isso evita que o chão seja reconhecido como doença, e também é onde, se afetado, podem gerar maiores danos as plantas, uma vez que estão muito próximas.

4. CONCLUSÃO

Para o desenvolvimento de uma aplicação que se beneficie das técnicas de reconhecimento de padrões e aprendizado de máquina, foi necessária uma grande quantidade de tempo e conhecimento, porém com o uso da plataforma Matlab e sua *toolbox* para treinamento de redes neurais, possibilitou a geração de grande quantidade de informação para alcançar o objetivo proposto.

Foi demonstrado no decorrer do trabalho que a visão computacional possui inúmeras aplicações, que auxiliam na melhoria de processos em gerais, sendo assim um ótimo objeto de estudo.

Portanto, levando em consideração empresas como a Embrapa, que financiam projetos do gênero, que buscam auxiliar na detecção e controle de doenças e pragas na agricultura, percebe-se que além de promissor, o algoritmo de detecção de cercosporiose e ferrugem, também pode ser de grande ajuda para agricultores de grande e pequeno porte.

Para trabalhos futuros serão realizadas melhorias do software e até mesmo iniciar um projeto que utiliza um meio de locomoção independente para que o agente inteligente também seja responsável por retirar as fotografias e fazer a análise diretamente no viveiro ou plantação.

Para isso serão necessários contínuos trabalhos de melhoria para obter um resultado consistente onde folhas secas e terra não sejam reconhecidas como doença.

5. REFERÊNCIAS

BORTH, M. R; IACIA, J. C.; PISTORI, H; RUVIARO, C. F; A Visão Computacional no Agronegócio: Aplicações e Direcionamentos ECAECO, 2014, disponível em:<http://www.gpec.ucdb.br/pistori/publicacoes/borth_ecaeco2014.pdf>, acessado em 30/09/2016.

CHARNIAK E., MCDERMOTT D.; *Introduction to artificial intelligence, and Addison*-Wesley Publ., 1985, ISBN 0-201-11946-3

CORRÊA, Maria Cristina Silva Montenegro. Anatomia e Fisiologia. 2016.

DE CASTRO SILVA, Leandro Nunes. "Analise e síntese de estratégias de aprendizado para redes neurais artificiais." (1998).

EMBRAPA 2011; Tecnologias de Produção de Soja - Região Central do Brasil 2012 e 2013, Sistemas de Produção 15, Londrina-PR 2011 disponível em: < http://www.cnpso.embrapa.br/download/SP15-VE.pdf>, acessado em 29/08/2016.

HAUGELAND. J; Artificial Intelligence: The Very Idea (MIT Press, Cambridge, MA, *1985*); 287 pp.

MARCELO, Severidade cercosporiose e ferrugem, disponivel em:<https://image.slideshare cdn.com/expocafemarcelo-120712083352-phpapp01/95/expocafe-marcelo-28728.jpg?cb=1342082206>, [Online 02-Setembro-2016].

MARENGONI, M.; STRINGHINI D. Introdução a visão computacional usando openCV. RITA, v.XIII, n.1, 2009.

MATLAB, informações sobre o matlab, disponível em: < https://www.mathworks.com>, [Online 05-Junho-2017].

MATSUNAGA, Victoria Yukie. Curso de Redes Neurais utilizando o MATLAB. Belém do Pará, 2012.

MONARD M. C.; BARANAUSKAS J. A., Sistemas inteligentes, 2003, cap. 4 disponível em:<http://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap4.pdf> USP. Acessado em 15/11/2016.

NETBEANS, 2017, Plataforma NETBEANS, disponível em: <https://netbeans.org/features/platform/index.html>, acessado dia 13/06/07.

NORVIG, P. RUSSELL, S. (1995), *Inteligencia Artificial*, Trad. Sob a direção de Vandenberg D. de Souza. Rio de Janeiro, Elsevier (2004) 2ª Tiragem.

POOLE D.; *Computational Intelligence: A Logical Approach,* Publ., 1998.

RICK E.; KNIGHT K.; MCGRAW HILL, *Artificial Intelligence* (2nd ed.), (*1991*). ISBN. 0-07-100894-2.

RAUBER, Thomas Walter. Redes neurais artificiais. Universidade Federal do Espírito Santo, 2005.

SOUZA, A F; Progresso da mancha de olho pardo do café em três municípios do estado de minas gerais, Universidade Federal de Viçosa, Viçosa-MG, 2003.

SHAPIRO, L.; STOCKMAN, G. *Computer vision*. New Jersey: Prentice Hall, 2001.

SZELISKI, R. Computer Vision: *Algorithms and Applications*. Springer, 2010.