UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Rodrigo Perehouskei da Costa

Identificação de Doenças em Folhas de Soja Utilizando Redes Neurais Convolucionais

Blumenau
2023

Rodrigo Perehouskei da Costa

Identificação de Doenças em Folhas de Soja Utilizando Redes Neurais Convolucionais

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Orientador, Dr. Mauri Ferrandin

Blumenau
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Perehouskei da Costa, Rodrigo
Identificação de Doenças em Folhas de Soja Utilizando Redes Neurais Convolucionais / Rodrigo Perehouskei da Costa ; orientador, Mauri Ferrandin, 2023.
72 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Inteligência artificial. 3. Rede neural convolucional. 4. Doenças em plantas. I. Ferrandin, Mauri. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

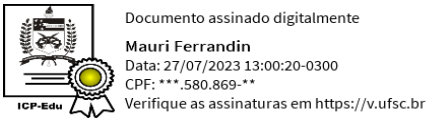
Rodrigo Perehouskei da Costa

Identificação de Doenças em Folhas de Soja Utilizando Redes Neurais Convolucionais

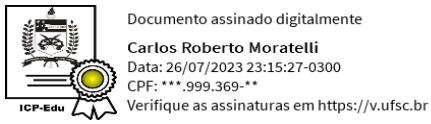
Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 10 de julho de 2023.

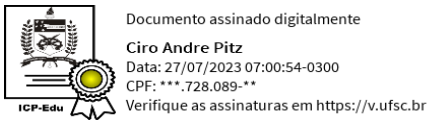
Banca Examinadora:



Prof. Dr. Mauri Ferrandin
Universidade Federal de Santa Catarina



Prof. Dr. Carlos Robeto Moratelli
Universidade Federal de Santa Catarina



Prof. Dr. Ciro André Pitz
Universidade Federal de Santa Catarina

Dedico este trabalho a todos aqueles que, de alguma forma,
auxiliaram para a concretização desta etapa.

AGRADECIMENTOS

A minha família, principalmente aos meus pais, Helena e Sidioneis, minha irmã, Daniela, por sempre me apoiarem em toda a trajetória, me proporcionando toda ajuda necessária, todo auxílio financeiro, viagens de idas e vindas, para que eu pudesse concluir esse sonho.

A Universidade de Federal de Santa Catarina, aos professores do curso de Engenharia de Controle e Automação do campus Blumenau, pelos ensinamentos e conhecimento compartilhados. Em especial ao meu orientador Mauri Ferrandin, por ter sido paciente comigo quando eu sofri meu acidente, passando por algumas cirurgias e ficando impossibilitado de dar prosseguindo no projeto por um tempo, e ter me apoiado e me auxiliado no desenvolvimento deste trabalho.

Por último, mas não menos importante, a empresa em que trabalho, por ter acreditado na minha ideia e ter me dado a oportunidade de implementação do projeto dentro da empresa.

“Não espere o futuro mudar tua vida, porque o futuro será a consequência do presente” (Racionais Mc’s)

RESUMO

Atualmente, um dos maiores problemas da agricultura mundial está interligado ao controle de doenças, sendo responsável por grandes prejuízos. A detecção das doenças nas culturas agrícolas é um fator essencial para seu controle, dessa forma, é importante uma rápida tomada de decisão, para os prejuízos serem os mínimos possíveis. Neste contexto, este trabalho investigou métodos para classificação das doenças utilizando uma área da inteligência artificial conhecida como redes neurais convolucionais. Foram utilizados quatro repositórios de imagens diferentes para compor um banco de imagem, contendo 815 imagens com seis classes, sendo cinco classes com as cinco principais doenças foliares presentes na soja e uma classe com a folha saudável. As bibliotecas *TensorFlow* e *Keras* foram utilizadas para o pré-processamento de imagem, elaboração e treinamento das duas redes neurais presentes no projeto, sendo uma rede neural convolucional chamada de MobileNetV2 e outra desenvolvida pelo autor. Mediante ao *software* Weka foram treinados algoritmos tradicionais com dados obtidos através de extração de *features* pelo descritor GIST, a fim de comparação dos resultados alcançados pelas redes neurais convolucionais. As 815 imagens foram divididas em três conjuntos de dados, estes sendo treinamento, validação e teste. Os resultados demonstraram que a rede neural MobileNetV2 alcançou um desempenho de 11,3% melhor do que a rede neural desenvolvida e 18,3% melhor do que métodos baseados em extração de *features*.

Palavras-chave: Detecção de doenças em plantas; Inteligência artificial; Redes neurais convolucionais.

ABSTRACT

Currently, one of the biggest problems in world agriculture is related to disease control, which is responsible for large-scale losses. The detection of diseases in agricultural crops is an essential factor for their control, therefore, it is important that decisions are made quickly and efficiently so that losses are drastically minimized. In this context, the present work investigated methods for classifying diseases using an area of artificial intelligence known as convolutional neural networks. Four different image repositories were used to compose an image bank, containing 815 images with six classes, five classes with the five main foliar diseases present in soybeans, and one class with a healthy leaf. The TensorFlow and Keras libraries were used for image pre-processing, elaboration and training of the two neural networks present in the project, one convolutional neural network called MobileNetV2 and another developed by the author. Using the Weka software, trained traditional algorithms were trained with data obtained through feature extraction by the descriptor GIST, in order to compare the results achieved by convolutional neural networks. The 815 images were divided into three sets of data, these being training, validation and testing. The results demonstrated that the MobileNetV2 neural network achieved a performance of 11,3% better than the developed neural network, and 18,3% better than methods based on feature extraction.

Keywords: Detection of plant diseases; Artificial intelligence; Convolutional neural networks.

LISTA DE FIGURAS

Figura 1 – Elementos para um Modelo de Aprendizagem de Máquina 19
Figura 2 – Mapa de *Machine Learning* 20
Figura 3 – Exemplo de uma Rede Neural Artificial Profunda 22
Figura 4 – Exemplo de uma Rede Neural Artificial Convolucional e suas camadas 23
Figura 5 – Exemplificação da Camada de Convolução 25
Figura 6 – Exemplificação de uma Aplicação de *max-pooling* e *avarage polling* de uma imagem 4x4 utilizando um filtro 2x2 25
Figura 7 – Exemplo de Camadas Rede Neural, Imagem Adaptada 27
Figura 8 – Diagrama para cálculo e extração do descritor GIST. (OUJAOURA *et al.*, 2014) 28
Figura 9 – Exemplo de Folha Contaminada Por Rust, Imagem de IPMIMAGES . 31
Figura 10 – Exemplo de Folha Contaminada Por Mancha-Alvo, Imagem de DIGIPHATOS 32
Figura 11 – Exemplo de Folha Contaminada Por *Oídio*, Imagem de DIGIPHATOS 32
Figura 12 – Exemplo de Folha Contaminada Por *Bacterial Blight*, Imagem de IPMIMAGES 33
Figura 13 – Exemplo de Folha Contaminada Por Míldio, Imagem de IPMIMAGES 34
Figura 14 – Recorte realizado para obter o ponto de interesse. 40
Figura 15 – Rotações e Zoom. 40
Figura 16 – Matriz de confusão. 41
Figura 17 – Arquitetura Geral da Rede Neural MobileNetV2, t= fator de expansão, c= números de canais de saída, n= número de repetição, s= strides. . . 45
Figura 18 – Bloco residual *Bottlenecks* transformando k canais para k', com stride igual a "s" e fator de expansão "t". 46
Figura 19 – Arquitetura própria da CNN desenvolvida pelo autor. 47
Figura 20 – Gráfico de *loss* da etapa 2. 55
Figura 21 – Gráfico de *loss* da etapa 4. 56
Figura 22 – Gráfico de *loss* da etapa 7. 57
Figura 23 – Gráfico de *loss* da etapa 8. 58
Figura 24 – Gráfico de *loss* da etapa 11. 59
Figura 25 – Gráfico de *loss* da etapa 12. 60
Figura 26 – Gráfico de *loss* da etapa 13. 62
Figura 27 – Gráfico de *loss* da etapa 17. 63

LISTA DE TABELAS

Tabela 1 – Tabela dos quatro bancos de imagens. 38
Tabela 2 – Tabela do banco de imagem do projeto. 38
Tabela 3 – Tabela do conjunto de imagens utilizadas no treinamento, teste e validação. 39
Tabela 4 – Tabela dos parâmetros usados de *batch-size* e *epoch* no treinamento do modelo MobileNeV2. 48
Tabela 5 – Tabela dos parâmetros usados de *batch-size*, *epoch*, e alterações de outros parâmetros no treinamento do modelo desenvolvido pelo autor. 49
Tabela 6 – Tabela de resultados das métricas utilizadas nos algoritmos tradicionais das extrações de *features* pelo descritor GIST. 50
Tabela 7 – Matriz de confusão do algoritmo IBk. 51
Tabela 8 – Tabela do Grupo 1 no treinamento do modelo MobileNeV2. 52
Tabela 9 – Tabela de resultados das métricas utilizadas referente ao Grupo 1. . . 52
Tabela 10 – Tabela do Grupo 2 no treinamento do modelo MobileNeV2. 52
Tabela 11 – Tabela de resultados das métricas utilizadas referente ao Grupo 2. . . . 53
Tabela 12 – Tabela do Grupo 3 no treinamento do modelo MobileNeV2. 53
Tabela 13 – Tabela de resultados das métricas utilizadas referente ao Grupo 3. . . 53
Tabela 14 – Tabela das etapas que apresentaram um melhor desempenho no treinamento do modelo MobileNeV2. 54
Tabela 15 – Tabela de resultados das métricas utilizadas nas etapas que tiveram os melhores desempenhos. 54
Tabela 16 – Matriz de confusão da etapa 2. 55
Tabela 17 – Tabela de precisão, recall e f1-score da etapa 2. 55
Tabela 18 – Matriz de confusão da etapa 4. 56
Tabela 19 – Tabela de precisão, recall e f1-score da etapa 4. 57
Tabela 20 – Matriz de confusão da etapa 7. 57
Tabela 21 – Tabela de precisão, recall e f1-score da etapa 7. 58
Tabela 22 – Matriz de confusão da etapa 8. 58
Tabela 23 – Tabela de precisão, recall e f1-score da etapa 8. 59
Tabela 24 – Matriz de confusão da etapa 11. 59
Tabela 25 – Tabela de precisão, recall e f1-score da etapa 11. 60
Tabela 26 – Matriz de confusão da etapa 12. 60
Tabela 27 – Tabela de precisão, recall e f1-score da etapa 12. 61
Tabela 28 – Tabela de resultados das métricas utilizadas referente a rede neural desenvolvida pelo autor 61
Tabela 29 – Matriz de confusão da etapa 13. 62
Tabela 30 – Tabela de precisão, recall e f1-score da etapa 13. 62

Tabela 31 – Matriz de confusão da etapa 17. 63
Tabela 32 – Tabela de precisão, recall e f1-score da etapa 17. 63
Tabela 33 – Tabela de resultados das métricas utilizadas nas etapas que tiveram os melhores desempenhos 64

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CEPEA	Centro de Estudos Avançados em Economia Aplicada
CIPMP	Centro de Informação de Pesquisa em Manejo de Pragas
CNN	Rede Neural Convolucional(Convolutional Neural Networks)
GIST	Global Image Features from Spatial Pyramid
GPU	Unidade de Processamento Gráfico (Graphics Processing Unit)
IA	Inteligência Artificial
MIP	Manejo Integrado de Pragas
PIB	Produto Interno Bruto
ReLU	Rectified Linear Unit
RNA	Rede Neural Artificial
TPU	Unidade de Processamento de Tensor (Tensor Processing Unit)

SUMÁRIO

1 INTRODUÇÃO 15
1.1 MOTIVAÇÃO E CONTEXTO 15
1.2 OBJETIVOS 16
1.2.1 Objetivo Geral 16
1.2.2 Objetivos Específicos 16
1.3 ESTRUTURA DO DOCUMENTO 17
2 FUNDAMENTAÇÃO TEÓRICA 18
2.1 INTELIGÊNCIA ARTIFICIAL 18
2.1.1 Aprendizagem de Máquina 18
2.1.2 Redes Neurais e Aprendizagem Profunda 21
2.1.3 Redes Neurais Convolucionais 23
2.1.3.1 Camada de Convolução 24
2.1.3.2 Camada de Pooling 25
2.1.3.3 Camada Totalmente Conectada e Saída 26
2.1.4 Descritor GIST 28
2.2 TENSORFLOW 29
2.2.1 Keras API 29
2.3 BANCO DE IMAGENS 30
2.4 IMAGENS RGB 30
2.5 CULTURA DA SOJA NO BRASIL 31
2.5.1 Utilização de CNN para Identificação de Doenças em Folhas de Soja 35
3 METODOLOGIA E DESENVOLVIMENTO 37
3.1 COMPOSIÇÃO DO CONJUNTO DE DADOS 38
3.2 PRÉ-PROCESSAMENTO 39
3.3 MODELAGEM E TREINAMENTO DE REDES NEURAIS E ALGORITMOS TRADICIONAIS 40
3.3.1 Métricas 41
3.3.1.1 Matriz de confusão 41
3.3.1.2 Acurácia 41
3.3.1.3 Precisão 42
3.3.1.4 Sensibilidade (recall) 42
3.3.1.5 F1-score 43
3.3.2 Extração de *features* e utilização de algoritmos tradicionais . 43
3.3.3 MobileNetV2 44
3.3.4 CNN com arquitetura própria 46
3.3.5 Treinamento e testes 47

4 RESULTADOS 50
4.1 TREINAMENTO DOS ALGORITMOS TRADICIONAIS ATRAVÉS DAS *FEATURES* OBTIDAS PELO DESCRITOR GIST 50
4.2 TREINAMENTO DA CNN MOBILENETV2 51
4.2.1 Resultados e análises do Grupo 1 51
4.2.2 Resultados e análises do Grupo 2 52
4.2.3 Resultados e análises do Grupo 3 53
4.2.4 Resultados e análises das etapas que obtiveram os melhores desempenhos 54
4.3 TREINAMENTO DA REDE DESENVOLVIDA PELO AUTOR . . . 61
4.4 COMPARAÇÕES ENTRE AS CNNS E ALGORITMOS TRADICIONAIS 64
5 CONCLUSÕES 65
5.1 TRABALHOS FUTUROS 65
REFERÊNCIAS 67

1 INTRODUÇÃO

O presente capítulo tem o objetivo de contextualizar o problema abordado neste trabalho, introduzindo os conteúdos tratados, definindo os objetivos esperados e apresentando a estrutura da monografia.

1.1 MOTIVAÇÃO E CONTEXTO

Nas últimas décadas, a inserção incessante de novas tecnologias nos regimes produtivos e na sociedade em geral, passou a ser algo existente que vem promovendo tantas revoluções industrias. Muitas vezes uma revolução industrial é identificada por transformações radicais, atraídas pela junção de tecnologias novas, tendo impacto nos setores econômico, social e político.

Atualmente o mundo se encontra em desenvolvimento sobre o conceito de Indústria 4.0, em que a evolução e a integração de novas tecnologias irão transformar profundamente o mundo atual, esculpindo a sociedade e a indústria pelos próximos anos (SACOMANO *et al.*, 2018).

Algumas das principais vertentes que veem impulsionando a quarta revolução industrial são tecnologias avançadas como a IA (Inteligência Artificial), robótica, visão computacional, internet das coisas, processamento digital de imagens e computação em nuvem. Vale ressaltar que a essência da industria 4.0 é fazer com que esses conjuntos de tecnologia consigam trabalhar em sincronia, ou seja, agrupando todas em apenas um sistema, associando o universo digital e físico (LEE, 2019).

Levando em conta esse cenário, este trabalho traz os resultados alcançados e a metodologia proposta para a realização de uma Rede Neural Convolucional para a detecção de doenças em folhas de soja, trabalho este, que será desenvolvido para implementação futura de um aplicativo em uma das maiores cooperativas agroindustrial do Brasil, a Cocamar.

A Cocamar é uma Cooperativa Agroindustrial fundada em 27 de março de 1963 em Maringá (PR). Hoje, com mais de 60 anos de existência, ela conta com mais de 100 unidades espalhadas por 4 estados do Brasil, sendo Goiás, Mato Grosso do Sul, São Paulo e Paraná, cerca de 3 mil colaboradores e mais de 16 mil cooperados. Sabendo que em média cada unidade da Cocamar conta com cerca de 2 engenheiros agrônomos para atender todos os cooperados da região, há a necessidade de uma maior demanda quando se tem problemas consecutivos de doenças em plantas. Com a idealização desse projeto, o próprio cooperado por meio das tecnologias propostas conseguirá ter a resposta do tipo de doença que sua cultura está sofrendo, podendo tomar uma decisão mais rápida, tendo menos perdas na safra.

A agricultura é um setor social e econômico de enorme domínio no mundo, especificamente no Brasil. Em uma pesquisa realizada em 2023, segundo o CEPEA (Centro

de Estudos Avançados em Economia Aplicada), o PIB (Produto Interno Bruto) do Brasil, resultou em 26,6% em 2021 e 24,8% em 2022 sendo compostos pelo agronegócio, ou seja, representando cerca de 1/4 da economia do país.

A soja é a principal cultura da cooperativa, gerando grande interesse na idealização deste trabalho, por ser responsável pela maior fração da economia entre os cooperados. Devido a isso, gerou uma preocupação na produção dessa cultura, não apenas na cooperativa, mas também no mundo. E um dos pontos de maior implicação dentro da produção são os problemas fitossanitários, em que as perdas, na maior parte das vezes, são devido a presença de doenças, tanto bacterianas, fúngicas ou virais. Levando em conta esse aspecto, a identificação precoce dos sintomas pode precaver perdas no acréscimo dos vetores na lavoura. Uma pesquisa realizada em 2019 pela Universidade da Califórnia, mostra que as safras têm sido diminuídas de 10% a 40% em virtude de doenças e pragas.

O tempo é uma atividade crucial no meio da produção agrícola, com isso, a classificação e detecção prematura de doenças são atividades importantes para reduzir os males que podem ser provocados pelas doenças, simplificando a tomada de decisão e viabilizando a identificação das doenças e um controle preciso .

O presente trabalho tem como objetivo trazer uma rede neural convolucional que tenha um bom desempenho para identificar as doenças nas folhas de soja, com uma junção de vários *datasets*. Para que assim, junto com a equipe de profissionais de tecnologia e inovação da cooperativa, a rede neural seja implementada em seu aplicativo no futuro.

1.2 OBJETIVOS

Na presente seção são discorridos os objetivos geral e específicos idealizados para a cumprimento deste Trabalho de Conclusão de Curso.

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral pesquisar e aplicar técnicas de visão computacional e aprendizado de máquina para classificação de folhas doentes e saudáveis na cultura de soja.

1.2.2 Objetivos Específicos

Para dar realização ao objetivo geral do presente trabalho, os seguintes objetivos específicos foram apontados:

- Desenvolver um dataset baseado na junção de imagens de vários datasets;
- Desenvolver uma rede neural convolucional (CNN) para fazer a identificação das doenças em folhas de soja;
- Treinar uma rede neural convolucional já existente para fim de comparação entre a CNN desenvolvida, técnica conhecida como Transfer Learning;

- Realizar um pré-processamento digital de imagem da folha da soja para facilitar o reconhecimento das doenças contidas nas folhas;
- Comparar os resultados obtidos das redes neurais e dos métodos básicos baseados em extração de features utilizados no trabalho;

1.3 ESTRUTURA DO DOCUMENTO

Este documento foi fragmentado em cinco capítulos a fim de discorrer os métodos aproveitados, os materiais, a teoria e os resultados atingidos.

Neste capítulo, uma sucinta introdução e demostração do problema é discorrida, além da apresentação do objetivo geral quanto específicos. No Capítulo 2, é preparada uma revisão bibliográfica demonstrando relevantes descobertas na literatura a fim de concluir o problema com as melhores ferramentas existentes e análise que serão úteis para o compreendimento dos próximos capítulos. No Capítulo 3, é demostrada a metodologia utilizada das três principais etapas para realizar a detecção de doenças em folhas de soja. No Capítulo 4, são demostrados os resultados atingidos, discorrendo os estudos considerados na evolução deste trabalho e as maiores dificuldades. No Capítulo 5, ocorre o encerramento do estudo discorrendo as conclusões relevantes e as perspectivas da implementação do trabalho na Cooperativa Cocamar. Antecipadamente pode-se concluir que os resultados obtidos foram congruentes, e validados para a implementação na Cooperativa Cocamar.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como o objetivo de descrever uma revisão teórica acerca das principais tecnologias e temas pertencentes ao desdobramento do presente trabalho. São indagadas revisões teóricas sobre a cultura da soja e inteligência artificial, como a visão computacional e aprendizado de máquina.

2.1 INTELIGÊNCIA ARTIFICIAL

A inteligência artificial pode ser delineada como a área da ciência da computação aplicada a procurar meios ou equipamentos computacionais que tenham ou multipliquem a habilidade racional do ser humano de resolver problemas, ou como o próprio nome já diz, de ser inteligente (LUGER, 2004).

Talvez o principal alvo da Inteligência Artificial é o de evoluir e utilizar máquinas para suceder atribuições humanas de modo autônomo, de maneira que se torne banal achar conexões com diversas áreas do conhecimento, como o reconhecimento de visão e de voz, robótica, e várias outras tecnologias (TEIXEIRA; GONZALES, 1983).

O termo IA é muito usado para descrever computadores que simulam a inteligência humana e imitam habilidades que os humanos associam com a mente humana. A resolução de problemas e a aprendizagem são exemplos dessas habilidades cognitivas. O campo da IA contém estudos de aprendizado de máquina, pois os sistemas de IA são capazes de aprender com as experiências. De um modo geral, as máquinas com inteligência artificial são capazes de, compreender e interpretar dados, aprender com os dados, tomar decisões inteligentes com base em *insights* e padrões extraídos de dados (YALÇIN, 2021). E com isso, a percepção de ambientes, o raciocínio e a propensão de analisar para a tomada de decisões fica mais assertiva quando uma solução de IA consegue envolver diversas tecnologias, como a rede neural artificial, sistemas de aprendizado e algoritmos genéticos (RICH; KNIGHT; CALERO, 1994).

2.1.1 Aprendizagem de Máquina

A área de aprendizado de máquina, também conhecido como *machine learning*, é uma subárea da inteligência artificial que se dedica no desenvolvimento de algoritmos e modelos que permitem que os sistemas computacionais aprendam e melhorem a partir de dados, sendo treinados usando grandes quantidades de dados para reconhecer padrões e fazer previsões ou tomar decisões com base nesses padrões(CHOLLET, 2021).

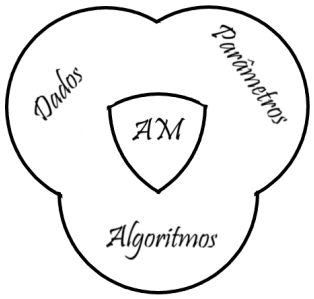
Intrinsecamente na área da IA, contem um vasto acervo de técnicas que vem sendo aprimoradas para o aprendizado de máquinas, vale ressaltar que existem três elementos principais para que o aprendizado de máquinas seja efetivo, são ele: Dados, Algoritmos e Parâmetros.

Esses três elementos em conjunto formam um modelo de aprendizado que será disposto para resolver o problema que se busca a solução, no presente trabalho, um modelo para identificar as doenças presentes em folhas de soja.

Com os testes em prática, alguns desses elementos usados para o aprendizado de máquinas, podem sofrer alterações para atingir um melhor resultado. Um dos elementos usados são os Dados, também conhecidos como Banco de Imagem ou *Datasets*. Elemento que requer uma grande diversidade, quantidade e qualidade, pois, afetará de maneira significativa no resultado. Outro elemento indispensável é o algoritmo, com o algoritmo ideal ou o conjunto dele é possível ter mais precisão no resultado. O último elemento da lista são os parâmetro, com ele são feitos os ajustes necessários dos dados.

Cada um dos elementos citados acima estão interligados, como mostrado na Figura 1, não é possível ter um aprendizado de máquina sem ter um desses três elementos, quanto maior a qualidade de dados, utilização de um algoritmo ideal e parâmetros certos, melhor será o resultado que se busca.

Figura 1 – Elementos para um Modelo de Aprendizagem de Máquina



Fonte: Elaborada pelo autor.

Aprendizagem de máquina é uma área da inteligência artificial que se dedica em desenvolver algoritmos e técnicas que possibilitam que um sistema aprenda e se aprimore a partir de dados. Ou seja, a *machine learning*, como também é chamada, é um avanço para solucionar problemas difíceis que envolvam muitos dados, por intermédios de padrões e relações.

Exitem quatro tipos principais de *machine learning*, como demostrado na Figura 2, de forma didática ela relata também alguns de seus algoritmos que as compõem. Como destacado nesta figura, os quatro principais tipos são: Aprendizado Clássico, Aprendizado por Reforço, Métodos *Ensamble* e Redes Neurais de Aprendizado Profundo.

O Aprendizado Clássico se apoia em regras e lógicas formais para aprender e tomar decisões, ela se divide em mais duas sub-classes, são: Aprendizado Supervisionado e Aprendizado Não Supervisionado.

O Aprendizado por Reforço é uma técnica em que um agente aprende a tomar as

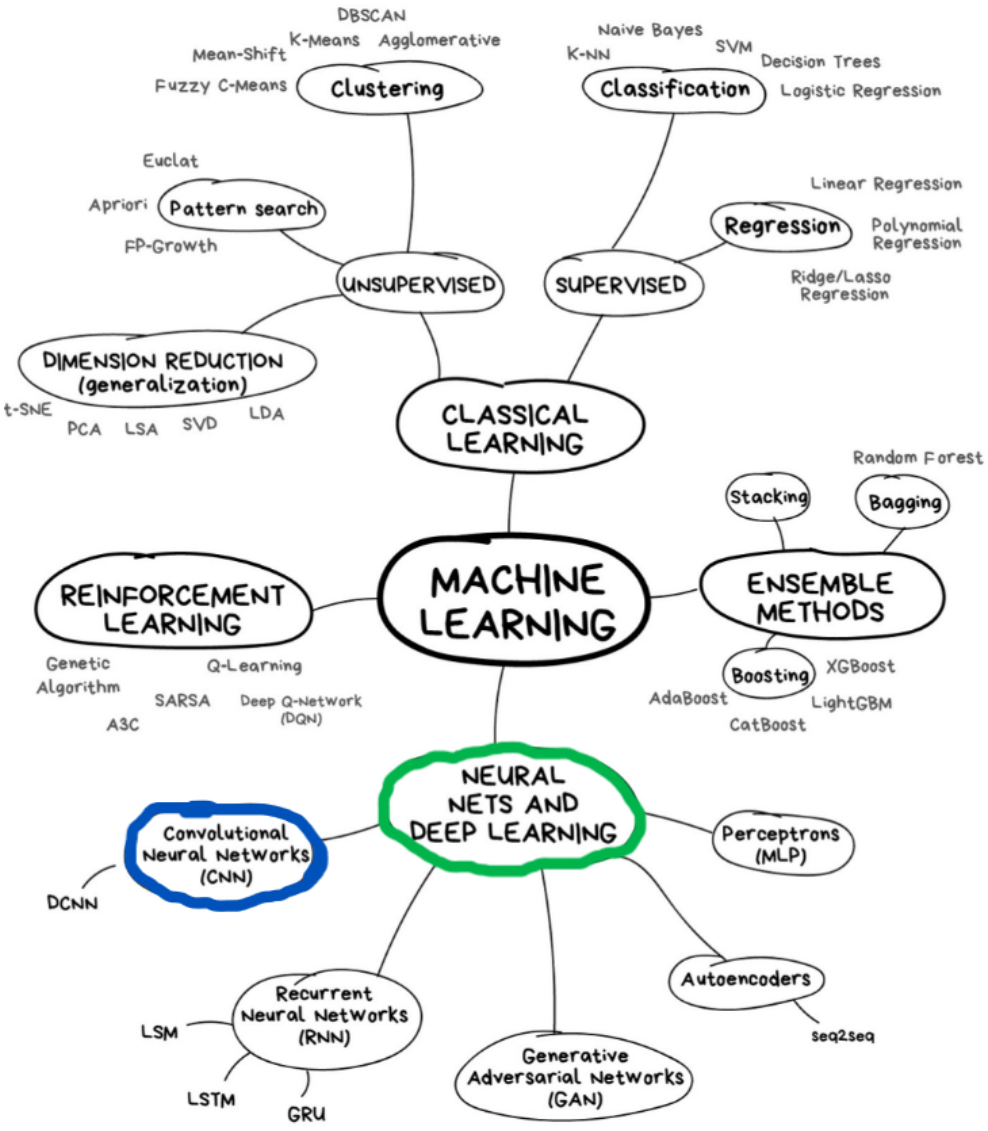
decisões em um espaço complexo por meio de tentativa e erro. O agente é penalizado ou contemplado levando em conta suas tomadas de decisões e com isso aprende a tomar as decisões que geram maiores recompensas.

O Método *Ensamble* é uma técnica que ajusta vários modelos de aprendizado para acurar a precisão e o desempenho do modelo, ele usa a sabedoria coletiva de vários modelos para traçar uma previsão mais assertiva, esse método se divide em: *Bagging, Boosting e Stacking.*

Por último, o tipo que será utilizado nesse projeto, as Redes Neurais de Aprendizado Profundo, que pode ser identificada na parte inferior da Figura 2 destacada em verde.

Nas seções seguintes são discorridas explicações sobre a sub-seção da *deep learning*, destacada em azul na parte inferior esquerda da Figura 2, as chamadas CNNs ou então, como é conhecida, as Redes Neurais Convolucionais, usadas no presente trabalho.

Figura 2 – Mapa de *Machine Learning*



Fonte: Modificada de Vas (2022).

2.1.2 Redes Neurais e Aprendizagem Profunda

As redes neurais são uma classe específica e frequentemente utilizada de algoritmos de aprendizado de máquina. RNAs são modeladas baseadas no comportamento do cérebro humano, onde o cérebro é composto por bilhões de neurônios interconectados que trabalham juntos para processar informações e tomar decisões, elas usam milhares ou até milhões de neurônios de processamento, em que são interconectados e dispostos em camadas, ou seja, os nós são conectados com cada nó processando entradas e gerando uma saída que é remetido a outros neurônios. As informações se movem pelos nós, que são estruturas matemáticas, com cada nó executando uma função diferente, para que consigam aprender a partir dos dados (VOULODIMOS *et al.*, 2018).

Cada neurônio em uma RNA recebe entradas de outros neurônios e calcula uma saída que é destinada para os neurônios seguintes. As ligações entre os neurônios são representadas por pesos, que podem ser aprimoradas no decorrer do treinamento, a fim da rede neural aprender a realizar a tarefa desejada da melhor maneira (KOVÁCS, 2002).

As RNAs podem ter várias camadas de neurônios, com as primeiras camadas recebendo os dados de entrada e as últimas camadas produzindo a saída interessada. Neste trabalho, as duas redes neurais artificiais utilizadas para fazer o projeto, possuem mais de uma camada intermediária, esse tipo de RNA são chamadas de rede neurais profundas (RAWAT; WANG, Zenghui, 2017), um exemplo da mesma se encontra na Figura 3.

Em uma rede neural artificial já treinada para identificar se uma imagem contém uma folha de soja doente ou saudável, por exemplo, os diferentes nós considerariam as informações e chegariam em uma saída que indicaria se uma imagem apresenta uma folha doente. Os modelos atuais de redes neurais desenvolvem esse tipo de problema muito bem (DATA SCIENCE ACADEMY, 2022). Elas têm o benefício de conseguirem lidar com dados não estruturados e complexos, como áudios e textos, imagens, além de terem uma capacidade de generalização para novos dados, ou seja, previsões bem sucedidas.

A aptidão de generalizar com eficácia é uma propriedade importante das redes neurais, dos sistemas de aprendizagem e de outros sistemas adaptativos. O aprendizado efetivo requer que o agente generalize a partir de dados de treinamento, para que assim consiga aplicar corretamente o aprendizado a situações ainda não vistas (LUGER, 2004).

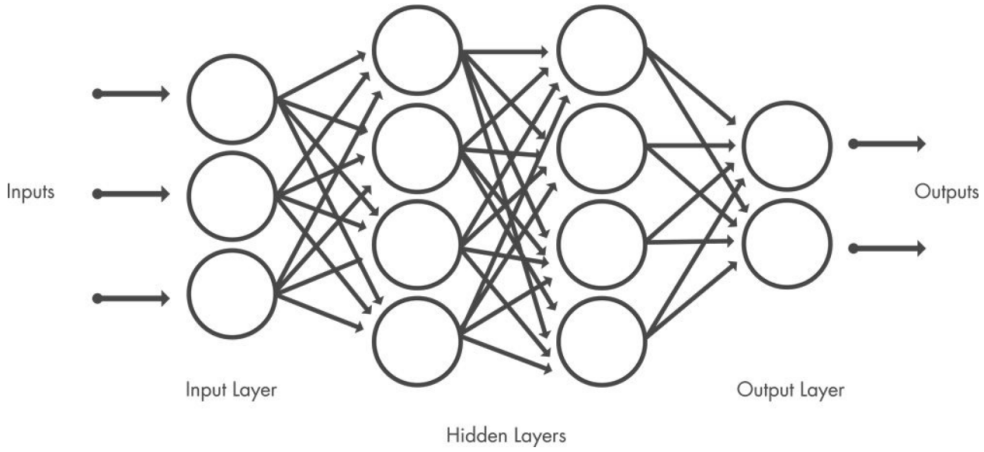
Aprendizagem de Máquina é um campo muito próspero e tem uma vasta variedade de utilizações, desde a visão computacional a previsão de mercado. A prosperidade na *machine learning* é se atentar em dados de qualidade e aderir a técnica de aprendizagem que melhor se adéque no problema que se busca a solução.

Como mencionado anteriormente, o presente trabalho tem a utilização das redes neurais de aprendizado profundo *(deep learning)*, esse tipo de rede neural apresenta várias camadas de neurônios interconectados, que são treinados em um grande grupo de dados rotulados para reconhecer padrões e com isso fazer com que o sistema tenha uma

maior precisão em suas previsões. O treinamento de uma rede neural profunda requer a atualização dos pesos de conexão entre os neurônios nela contida, a fim de reduzir o erro entre as previsões do modelo e as saídas interessadas.

No entanto, o treinamento de redes neurais pode ser computacionalmente caro por exigir recursos de *hardware*, incluindo RAM, o uso de GPUs (unidades de processamento gráfico) e TPUs (unidades de processamento tensorial) para acelerar o treinamento e para conseguir processar todos os dados existentes. Os modelos estatísticos precisam se tornar mais eficientes em termos de memória (isso normalmente é obtido adicionando não linearidades) ao mesmo tempo em que pode passar mais tempo na otimização desses parâmetros, devido ao aumento do orçamento computacional (ZHANG *et al.*, 2021).

Figura 3 – Exemplo de uma Rede Neural Artificial Profunda



Fonte: DMath (2023).

Uma das vantagens mais relevantes do *deep learning* é sua habilidade de extrair características principais dos dados de forma automática. Além de que, as redes neurais artificiais profundas são capazes de lidar com dados complexos, como por exemplo, imagens e sequências, com alta eficácia e precisão.

Determinadas aplicações do *deep learning* abrangem tradução automática, reconhecimento de fala, detecção de objetos em vídeos e imagens e diagnósticos médicos assistidos por computador assistentes virtuais inteligentes.

Entretanto, o *deep learning* possui algumas restrições, como por exemplo, a dificuldade de interpretar as decisões tomadas pelos modelos, a necessidade de muitos dados para se treinar os modelos e a possibilidade de *overfitting*, que é quando o modelo fica muito especializado nos dados de treinamento mas quando é introduzido novos dados ele não generaliza com grande eficiência, ou seja, as previsões para os treinamentos são eficazes, mas para os dados de testes não.

Novos métodos de controle de capacidade, como *dropout*, ajudaram a mitigar o perigo de *overfitting*. Isso foi conseguido aplicando injeção de ruído em toda a rede neural, substituindo pesos por variáveis aleatórias para fins de treinamento (ZHANG *et al.*, 2021).

2.1.3 Redes Neurais Convolucionais

Neste trabalho são utilizadas CNNs, esse tipo de tecnologia é amplamente usado em aplicações de visão computacional, como reconhecimento de vídeo e no caso do presente trabalho, imagem.

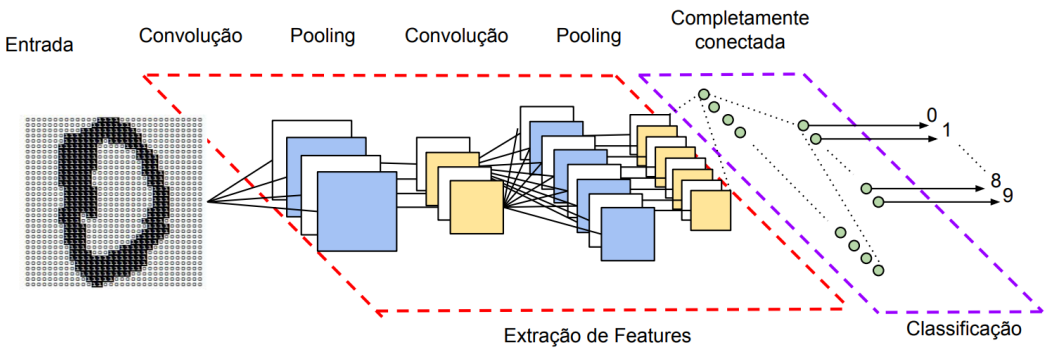
As CNNs são parecidas com às Redes Neurais clássicas, elas também possuem parâmetros que são treinados para certas funções específicas. A maior diferença entre elas são que as CNNs recebem de entrada uma única imagem, o que diminui o acervo de parâmetros utilizados, fazendo com que as funções de encaminhamento sejam melhores e a implementação mais fácil.

As CNNs usam uma operação matemática chamada convolução para processar imagens e identificar padrões nos dados, a fim de aprender características importantes da imagem, ela é composta por um grupo de filtros *(kernels)* que são aplicados em diversas regiões pequenas da imagem. Cada filtro é ajustado durante o treinamento para detectar um grupo específico de características, como formas, bordas e texturas, passando por várias camadas consecutivas que produz diversas ativações para representar as diferentes características de cada imagem (GU *et al.*, 2018).

As camadas convolucionais em uma CNN são seguidas por camadas denominadas de *pooling*, servem para reduzir a dimensão das ativações criadas pelas camadas de convolução. Ou seja, reduzir a resolução espacial para diminuir a quantidade de parâmetros necessários para a rede (O'SHEA; NASH, 2015).

Após isso, temos o que é conhecido como Classificador, o nome já é bem intuitivo, são camadas totalmente conectadas, que utilizam as características aprendidas pela rede para classificar as imagens ou efetuar outros tipos de tarefas. Essas camadas totalmente conectadas criam saídas finais que são usadas para treinar a rede neural, elas precisam que seus neurônios tenham conexões efetuadas com todas as ativações anteriores. Essa camada é responsável por traçar um caminho de decisão a partir das resposta dos filtros vindos das camadas anteriores, para cada classe de resposta (VARGAS; PAES; VASCONCELOS, 2016).

Figura 4 – Exemplo de uma Rede Neural Artificial Convolucional e suas camadas



Fonte: Vargas, Paes e Vasconcelos (2016).

As CNNs foram revolucionárias quando se diz respeito ao campo de visão computacional, ultrapassando métodos anteriores em diversas tarefas cruciais. Além disso, com a utilização e avanço de arquiteturas profundas, foi possível que as CNNs aprendessem características ainda mais complexas e melhorassem ainda mais o desempenho em diversas aplicações.

Entretanto, treinar uma CNN requer um grande número de dados e recursos de computação, especialmente em arquiteturas profundas. Além disso, há dificuldades em lidar com problemas de classificação de imagens com uma grande semelhança ou com uma qualidade baixa de imagem. Essas dificuldades são objeto de pesquisa ativa dentro da área de aprendizado profundo. Essa camada e fundamental no treinamento, pois influencia no aprendizado dos filtros e consequentemente no resultado da rede. Devido a sua simplicidade e bons resultados a função de ativação ReLU pode ser utilizada nessa etapa, contribuindo para um treinamento mais rápido sem perda de qualidade (VARGAS; PAES; VASCONCELOS, 2016).

Segundo (CHOI *et al.*, 2019), as Redes Neurais Convolucionais são uma arquitetura de rede neural usadas para fazer classificações de imagens utilizadas especialmente em técnicas de aprendizado profundo *(Deep Learning)* e visão computacional. Esta arquitetura foi inspirada e baseada na estrutura do sistema visual. A principal diferença de uma CNN de outras redes neurais são as três camadas internas que contém nela, que são, camada de convolução, camada de *poolin*ge a última, uma camada totalmente conectada onde todos os neurônios da rede se conectam, como demonstrado o na Figura 4.

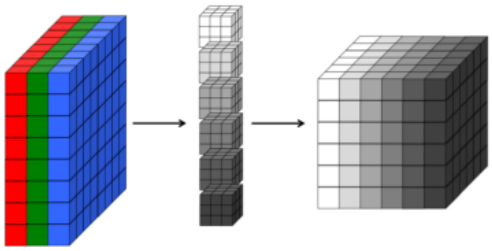
2.1.3.1 Camada de Convolução

Na camada de convolução é executado o mapeamento das principais características da imagem. São usados filtros para efetuar as convoluções, produzindo os mapas de características. Cada filtro detém dimensão reduzida, mas que se expande por toda profundidade do volume de entrada. Cada um desses filtros geram uma matriz de pesos que é deslizada em toda a imagem, realizando multiplicações elementares e somas para produzir uma ativação, como mostrado na Figura 6. Segundo (ALBAWI; MOHAMMED; AL-ZAWI, 2017), se a imagem for colorida, como é o caso do respectivo trabalho,então ela possui três canais (R, G, B) e o filtro da primeira camada convolucional terá o tamanho 5x5x3 (cinco de altura, cinco de largura e três de profundidade). Nesta etapa vários filtros de convolução podem ser aplicado, resultando em mais de uma matriz com mapas de características da imagem analisada (LECUN; BENGIO; HINTON, 2015). Na camada de convolução, na maior parte dos casos possui vários filtros, e cada um desses filtros produz um mapa de características diferente. Consequentemente, a saída de uma camada de convolução é um conjunto de mapas de características com diferentes informações sobre a imagem de entrada.

Além disso, as camadas de convolução são tipicamente acompanhadas por uma

função de ativação não linear, como a ReLU, que introduz a não linearidade na rede, permitindo que ela aprenda relações mais complexas entre as características (BUDUMA, Nithin; BUDUMA, Nikhil; PAPA, 2022).

Figura 5 – Exemplificação da Camada de Convolução



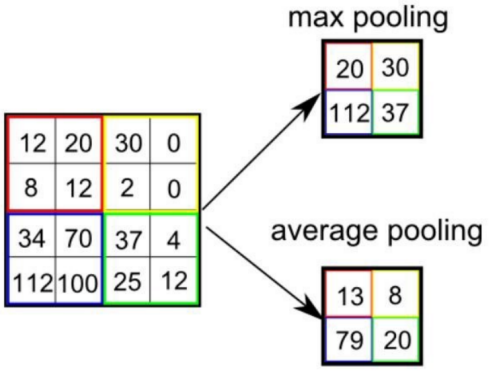
Fonte: Nithin Buduma, Nikhil Buduma e Papa (2022).

2.1.3.2 Camada de Pooling

A camada de *pooling* opera em cada mapa de características, dividindo-o em regiões (normalmente quadradas) e reduzindo a informação nessas regiões a um único valor representativo.A forma mais comum de usar o *pooling* é utilizando o *max pooling*, que é quando o valor máximo de cada região é selecionado e preservado como a representação dessa região, como exemplifica a Figura 6, que tem como entrada um bloco de dimensões 4x4, nele contém mais 4 subdivisões com blocos de 2x2, com valores em seus pontos, após a aplicação de *max-pooling*, temos uma saída com um único bloco de dimensões 2x2 que contém só os maiores valores dos sub-blocos. Ajudando a preservar as características mais proeminentes presentes em cada região.

Existe também outra forma de se usar o *pooling*, chamado de *average pooling*, também exemplificada na Figura 6, que ao invés de se utilizar os valores máximos, é calculada a média dos valores em cada região, fornecendo uma representação média daquela região (LITVAK, 2019).

Figura 6 – Exemplificação de uma Aplicação de *max-pooling* e *avarage polling* de uma imagem 4x4 utilizando um filtro 2x2



Fonte: ARAÚJO *et al.* (2017).

Algumas vantagens do *pooling* é a ajuda reduzir a dimensionalidade dos mapas de características, reduzindo o número de parâmetros na rede e o consumo de memória. Isso torna o processamento mais eficiente e ajuda a evitar o *overfitting*. Também introduz uma invariância translacional às características. Isso significa que, mesmo que a posição exata de uma característica seja alterada na imagem, a sua representação resultante no mapa de *pooling* será semelhante. Isso torna a rede mais robusta a pequenas variações e deslocamentos nas características detectadas (GUO *et al.*, 2017).

2.1.3.3 Camada Totalmente Conectada e Saída

A camada totalmente conectada, também conhecida como camada de saída ou camada densamente conectada, é a última camada em uma rede neural artificial, onde todos os neurônios estão conectados a todos os neurônios da camada anterior (SOUSA; SALAME, 2017).

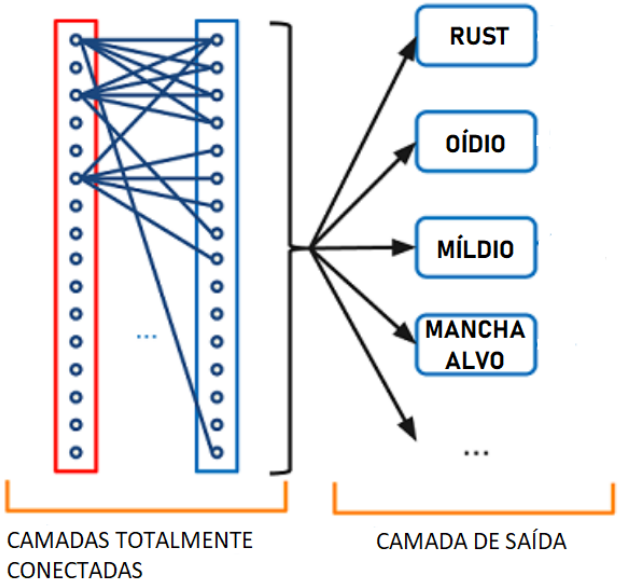
Diferente das camadas convolucionais e de *pooling*, que tem por objetivo extrair as principais características das imagens, as camadas totalmente conectadas tem por objetivo mapear as características extraídas pelas camadas anteriores para a tarefa final, como regressão, classificação, ou outra tarefa específica (VARGAS; PAES; VASCONCELOS, 2016).

Cada um dos neurônios na camada totalmente conectada adere entradas de todos os neurônios da camada anterior. Os pesos dessas conexões são ajustados ao longo de todo o treinamento da rede neural para aprimorar o desempenho da rede.

A quantidade de neurônios na camada totalmente conectada pode variar conforme o número de classes que se tem na rede para que se possa fazer a classificação, como pode ser visualizo na Figura 7.

Essa camada aplica uma transformação não linear aos dados, possibilitando que a rede neural aprenda representações mais complexas e não lineares das características extraídas nas camadas passadas. Funções de ativação, como a ReLU ou a função *sigmoide*, são muito usadas para introduzir a não linearidade. Ou seja, amplia a capacidade de aprendizado da rede neural, fazendo com que seja possível a combinação de características extraídas em etapas anteriores para realizar a tarefa final. Porém, com isso pode estender a quantidade de parâmetros da rede, fazendo com que se o tempo de treinamento seja mais longo e demandando mais imagens de treinamento para evitar o *overfitting*.

Figura 7 – Exemplo de Camadas Rede Neural, Imagem Adaptada



Fonte: Data Science Academy (2022).

No presente trabalho foram utilizadas duas funções de ativação, a função de ativação ReLU e a função de ativação *softmax* na camada de saída, elas são duas funções muito utilizadas em redes neurais.

A função de ativação ReLU é uma função não linear que mapeia os valores de entrada para zero se forem negativos e mantém os valores positivos. É comumente utilizada em camadas ocultas de redes neurais profundas por ter uma simplicidade e eficiência computacional. Resolve o problema de desvanecimento do gradiente e favorece a habilidade de aprendizado de representações não lineares mais difíceis.

Por outro lado, a função de ativação *softmax* é muito utilizada na camada de saída de uma rede neural quando se trata com problemas de classificação multiclasse. A função *softmax* mapeia os valores de entrada para uma distribuição de probabilidade, onde a soma de todas as saídas é igual a 1. Isso possibilita que o modelo atribua probabilidades para cada classe de saída, sendo útil para interpretar a saída do modelo como probabilidades. Ela é frequentemente usada em problemas de classificação onde é necessário atribuir uma probabilidade para cada classe possível.

Ambas as funções de ativação têm suas vantagens e são muito usadas em diferentes contextos de redes neurais. A função ReLU é competente e auxilia a lidar com problemas de desvanecimento do gradiente, enquanto a função *softmax* é vantajoso para atribuir probabilidades a classes em problemas de classificação multiclasse. A escolha da função de ativação cabível pode variar dependendo do problema em questão e do comportamento esperado do modelo.

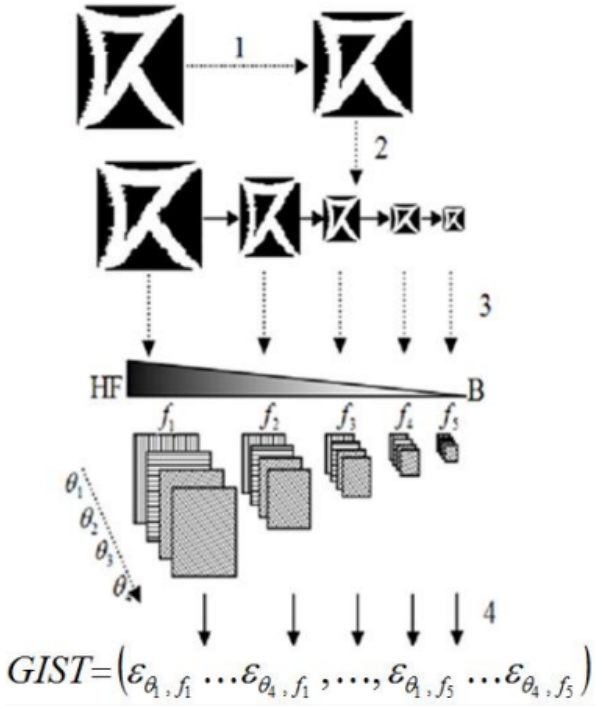
2.1.4 Descritor GIST

O descritor GIST (Global Image Features from Spatial Pyramid) é um método utilizado para extrair características globais de uma imagem, capturando informações sobre a textura e a estrutura espacial da imagem, transferindo uma representação discriminativa e compacta, como mostrado na Figura 8.

O processo de extração do descritor GIST envolve algumas etapas:

- Pré-processamento: A imagem é redimensionada para um tamanho fixo e convertida para escala de cinza;
- Filtragem: A imagem é filtrada usando filtros de Gabor em várias escalas e orientações, esses filtros são sensíveis a diferentes texturas e estruturas da imagem (DUNN; HIGGINS, 1995);
- Sub-amostragem: A imagem filtrada é subamostrada em várias resoluções para capturar informações em diferentes níveis de detalhes;
- Concatenação: Os valores resultantes da sub-amostragem são concatenados em um vetor de características para cada escala, englobando informações do gradiente.

Figura 8 – Diagrama para cálculo e extração do descritor GIST. (OUJAOURA *et al.*, 2014)



Fonte: Oujaoura *et al.* (2014).

O vetor resultante representa o descritor GIST da imagem, contendo informações globais sobre sua textura e estrutura, podendo ser usado em diversas tarefas de processa-

mento de imagens, como classificação, detecção e recuperação de imagens (DOUZE *et al.*, 2009).

2.2 TENSORFLOW

TensorFlow é uma popular biblioteca de código aberto para aprendizado de máquina e inteligência artificial, possuindo um ecossistema abrangente e flexível de ferramentas, bibliotecas e recursos da comunidade e é amplamente utilizada tanto na pesquisa acadêmica quanto na indústria (DEVELOPERS, 2022).

Com uma disponibilização de estrutura flexível e eficaz para construir e treinar redes neurais, é possível fazer a criação de modelos complexos com várias camadas e tipos de camadas diferentes, como por exemplo a utilizada nesse projeto, as redes neurais convolucionais. Oferecendo recursos para treinamento distribuído em *clusters* de computadores, aceleração por GPU e inferência em dispositivos móveis e embarcados. Fazendo com que seja possível trabalhar com uma grande quantidade de dados e tarefas computacionalmente excessivas. A arquitetura utilizada se divide principalmente em três partes, são elas (GHOTRA; DUA, 2017):

- Pré-processamento dos dados;
- Criação dos modelos;
- Estimativa e treinamento do modelo criado.

O TensorFlow possui uma API rica em recursos que possibilita aos desenvolvedores treinar e criar modelos de *machine learning* de forma satisfatória.

Atualmente, as aplicações que utilizam TensorFlow estão focalizadas em desempenhar redes neurais profundas com o propósito de classificar dígitos manuscritos, reconhecimento de imagens, tradução automática, incorporação de palavras, processamento de linguagem natural e simulações fundamentadas em equações diferenciais parciais (TENSORFLOW, 2018).

2.2.1 Keras API

A Keras API é uma biblioteca de alto nível *(high-level)* para criação e treinamento de redes neurais no TensorFlow. Disponibilizando uma interface sintetizada e intuitiva, permitindo que os usuários construam modelos de redes neurais, podendo ser de forma modular, onde as camadas são empilhadas para formar a arquitetura da rede, com três vantagens essências: *Modular and composable, User friendly e Easy to extend* (KETKAR; KETKAR, 2017). Fornecendo uma extensa gama de camadas pré-definidas, como camadas convolucionais, camadas de *pooling*, camadas totalmente conectadas, camadas de *dropout*, entre outras. Além do mais, também é possível criar camadas personalizadas para atender às exigências particulares do projeto (GULLI; KAPOOR; PAL, 2019).

Também é possível delimitar a arquitetura do modelo, compilar o modelo especificando a função de perda, métricas de avaliação e otimizador. Recursos avançados também são fornecidos, como regularização, inicialização personalizada de pesos, treinamento em lote *(mini-batch training)*, salvamento e carregamento de modelos, entre outros (MANASWI; MANASWI, 2018).

A API fornece variedades de funções de métricas e ativações que são possíveis ser utilizadas para deixar as redes neurais de aprendizado profundo mais otimizadas, podendo serem feitas medições de precisão e performance dos modelos criados.

2.3 BANCO DE IMAGENS

Um banco de imagem, é uma coleção organizada de imagens que são armazenadas eletronicamente, são muito utilizadas em diferentes setores, incluindo ciência da computação, pesquisa, negócios, saúde e etc. Refere-se a um conjunto de dados que são usados para treinar, validar e testar algoritmos de aprendizado de máquina e modelos de inteligência artificial.

A escolha de um banco de imagem pertinente depende das exigências do projeto, do tipo de imagem a serem armazenados, da escala e do desempenho esperado. Um *dataset* de imagens pode ser usado para treinar algoritmos de visão computacional, como classificação de imagens, detecção de objetos e segmentação semântica. Cada imagem no banco de imagens representa um exemplo individual dentro do *dataset*, e geralmente é acompanhada por rótulos que indicam a classe ou categoria à qual a imagem pertence (CASANOVA *et al.*, 2005). Podendo ser de uso público, fornecidos por organizações ou pesquisadores, ou podem ser criados internamente por empresas ou instituições para fins específicos, sendo uma valiosa fonte de dados para treinamento e avaliação de modelos de aprendizado de máquina e são amplamente utilizados em projetos de visão computacional (DATE, 2004).

2.4 IMAGENS RGB

Uma imagem digital é formada por certas quantias de píxeis. Um píxel é a menor unidade de uma imagem digital que representa um único ponto na matriz que compõe a imagem, responsável por armazenar informações sobre a cor e intensidade luminosa nesse ponto, cada pixel é composto por valores numéricos que representam os componentes de cor da imagem (GUTUB *et al.*, 2010).

No presente trabalho são usadas imagens coloridas RGB *(Red, Green, Blue)*, o que significa que em cada pixel tem três valores que representam a intensidade dessas três cores primárias. Cada canal de cor é representado por um valor numérico que varia de 0 a 255, o valor 0 representa a ausência de cor (ausência de intensidade), enquanto o valor 255 representa a cor mais intensa no respectivo canal. A combinação dos valores nos três

canais forma uma gama completa de cores, permitindo a exibição de uma ampla variedade de tons, o que resulta na combinação das cores para formar a imagem completa.

As imagens coloridas retem 3 planos monocromáticos que juntos formam todas as cores que existem, cada plano retrata uma intensidade de cor, a primeira camada caracteriza as tonalidades em vermelho, a segunda camada as tonalidades em verde e a última camada as tonalidades em azul, por esse motivo as siglas são no formato "RGB" *(Red-Green-Blue)* (MURILLO-ESCOBAR *et al.*, 2015).

2.5 CULTURA DA SOJA NO BRASIL

A cultura da soja é importante para o Brasil, sendo considerada uma das indispensáveis culturas agrícolas do país. O Brasil se tornou o maior produtor de soja do mundo, superando os Estados Unidos. A soja é cultivada em diferentes regiões do país, com destaque para os estados, como o Paraná, Mato Grosso e Goiás.

Contudo, a cultura da soja enfrenta desafios associados a doenças que afetam as folhas das plantas e podem atrapalhar a produtividade. Algumas das principais doenças que afetam a cultura da soja no Brasil são: Ferrugem da soja, mancha-alvo, oídio, *bacterial blight* e míldio.

1. Ferrugem da soja *(rust)*: É uma doença causada pelo fungo *Phakopsora pachyrhizi* e é uma das doenças mais alarmantes. Ocasiona o aparecimento de manchas de cor marrom na superfície inferior das folhas, que podem se espalhar e causar a queda precoce da planta, como mostrado na Figura 9. O controle da ferrugem da soja é feito por meio de medidas preventivas, como o uso de cultivares resistentes e aplicação de fungicidas (YORINORI; NUNES JÚNIOR; LAZZAROTTO, 2004).

Figura 9 – Exemplo de Folha Contaminada Por Rust, Imagem de IPMIMAGES

Fonte: ipmimage (2022).

2. Mancha-alvo: Causada pelo fungo *Corynespora cassiicola*, a mancha-alvo afeta as folhas da soja, criando manchas circulares com um centro mais escuro e bordas

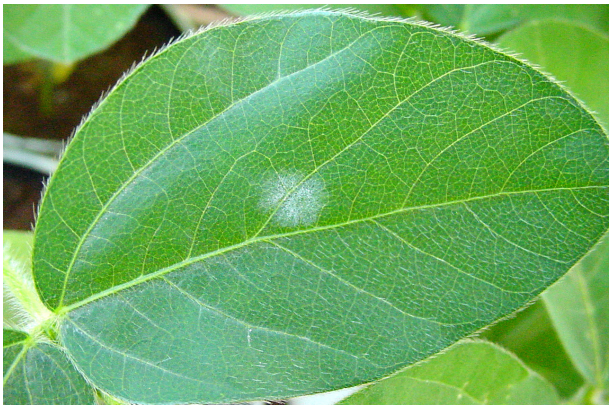
amareladas, como mostrado na Figura 10. Pode levar a desfolha prematura e diminuição da produtividade. O controle da mancha-alvo requer práticas culturais adequadas, como a rotação de culturas e o controle de plantas daninhas, além do uso de fungicidas quando preciso (SOARES; GODOY; OLIVEIRA, 2009).

Figura 10 – Exemplo de Folha Contaminada Por Mancha-Alvo, Imagem de DIGIPHATOS

Fonte: Barbedo, Koenigkan e Santos (2016).

3. Oídio: É causado pelo fungo *Erysiphe diffusa* e se caracteriza pelo aparecimento de um pó branco nas folhas da soja, como mostrado na Figura 11. Pode diminuir a área fotossintética das folhas, afetando gravemente a produção. O manejo do oídio inclui o uso de cultivares resistentes, monitoramento regular das lavouras e aplicação de fungicidas quando inevitável (YORINORI, 1997).

Figura 11 – Exemplo de Folha Contaminada Por *Oídio*, Imagem de DIGIPHATOS



Fonte: Barbedo, Koenigkan e Santos (2016).

4. *Bacterial blight*: É causada pela bactéria *Xanthomonas campestris pv. glycines*. Ela se apresenta por meio do aparecimento de manchas escuras nas folhas da soja, geralmente com um amarelado ao redor da lesão, como mostrado na Figura 12. À medida que a doença avança, as manchas podem se fundir, levando a necrose e a morte das folhas. Além disso, em condições de alta umidade, a bactéria pode produzir exsudatos de cor amarelada ou esbranquiçada. A doença *bacterial blight* pode afetar negativamente a produtividade da soja, principalmente quando ocorre em estágios iniciais de desenvolvimento da planta. O manejo dessa doença envolve o uso de sementes certificadas, rotação de culturas, controle de plantas daninhas e adoção de boas práticas agrícolas (BASU; BUTLER, 1986).

Figura 12 – Exemplo de Folha Contaminada Por *Bacterial Blight*, Imagem de IPMIMAGES

Fonte: ipmimage (2022).

5. Míldio: Por sua vez, é causado pelo fungo *Peronospora manshurica*, é mais recorrente em regiões com clima úmido e temperaturas moderadas. Os sintomas do míldio são o surgimento de manchas cloróticas nas folhas, que podem se expandir muito rápido, formando um aspecto pulverulento de coloração branca ou acinzentada na parte inferior das folhas, como mostrado na Figura 13. O míldio pode levar à queda das folhas de forma prematura e redução da produtividade. O controle do míldio requer o uso de cultivares resistentes, rotação de culturas, adoção de práticas adequadas de manejo e, se preciso, a aplicação de fungicidas (KOWATA *et al.*, 2008).

Figura 13 – Exemplo de Folha Contaminada Por Míldio, Imagem de IPMIMAGES

Fonte: ipmimage (2022).

É de extrema importância a identificação rápida dessas doenças nas folhas da soja, segue alguns dos principais motivos pelos quais a identificação precoce é crucial (HENNING *et al.*, 2014):

- Tomada de decisão rápida: Ao detectar rapidamente uma doença nas folhas da soja, os agricultores conseguem tomar medidas no mesmo instante para controlar a propagação e diminuir os danos causados;
- Redução de perdas econômicas: As doenças nas folhas da soja podem vir a causar danos relevantes à cultura, decorrendo em perdas econômicas para os agricultores. Ao identificar as doenças precocemente, é possível aplicar medidas de controle de maneira favoráveis, diminuindo os danos e preservando a produtividade;
- Manejo integrado de pragas: Quando a identificação de doenças em folhas de soja é rápida é possível aplicar diferentes estratégias combinadas para controlar com eficiência a doença. Como por exemplo, cultivares resistentes, rotação de culturas e controle de plantas daninhas e quando necessário o uso correto de produtos químicos;
- Prevenção da propagação: Certas doenças nas folhas da soja podem se alastrar rapidamente e afetar também outras áreas vizinhas. Com a identificação rápida, é possível a aplicação de medidas de quarentena e a escolha de práticas de biossegurança para barrar a propagação da doença para lavouras próximas;
- Monitoramento eficaz: Ao identificar e monitorar regularmente as doenças nas folhas da soja, os agricultores conseguem juntar dados indispensáveis sobre a ocorrência e a gravidade das doenças em diferentes regiões e períodos. Assim conseguem avaliar a eficácia das estratégias de manejo utilizadas.

Em resumo, é extremamente importante identificação rápida das doenças nas folhas da soja para que tenha uma boa produção agrícola. Permitindo que de medidas de controle

sejam aplicadas adequadamente, tendo uma redução de perdas econômicas e a prevenção do aumento das doenças. Por esse motivo, é importante que os agricultores tenham meios precisos e confiáveis de identificar os sinais e sintomas das doenças.

2.5.1 Utilização de CNN para Identificação de Doenças em Folhas de Soja

O reconhecimento de doenças nas folhas da soja utilizando redes neurais é uma ferramenta muito importante no manejo eficiente das culturas, trazendo vários benefícios como detecção precisa, velocidade e eficiência, automatização do processo, monitoramento contínuo e suporte à tomada de decisão. Essa abordagem inovadora tem o potencial de melhorar o manejo das culturas e auxiliar os agricultores na prevenção e controle de doenças, resultando em uma produção mais saudável e produtiva, não deixando o agricultor dependente cem por cento dos engenheiros agrônomos, podendo ter uma tomada de decisão mais rápida sem prejudicar sua plantação.

Um dos benefícios citados acima é a detecção precisa, as redes neurais têm a capacidade de aprender padrões complexos a partir de um grande conjunto de dados. Ao treinar uma rede neural com imagens de folhas saudáveis e infectadas por doenças, ela pode aprender a identificar características distintas e sutis que indicam a presença da doença, possibilitando uma detecção precisa e confiável das doenças nas folhas da soja. Outro ponto muito importando com a utilização de redes neurais é a velocidade e eficiência, as *CNNs* podem processar muitos dados de forma rápida e eficaz, ao utilizar técnicas de aprendizado profundo e processamento paralelo, o reconhecimento de doenças nas folhas da soja pode ser realizado em tempo real ou com uma baixa latência, permitindo uma identificação rápida e oportuna das doenças, possibilitando uma ação imediata de controle.

Vale lembrar que a automatização do processo com o uso de redes neurais, possibilita automatizar o processo de reconhecimento de doenças nas folhas da soja, uma vez que a rede neural tenha sido treinada e validada, ela pode ser aplicada para analisar imagens em larga escala de forma automatizada, sem a necessidade de intervenção manual, economizando tempo e recursos, além de permitir uma análise mais abrangente e consistente. Com o monitoramento contínuo, é possível estabelecer um monitoramento contínuo das doenças nas folhas da soja, as imagens das plantas podem ser capturadas regularmente e processadas pela rede neural para identificar a presença de doenças, possibilitando a detecção precoce de infecções e um acompanhamento em tempo real da evolução das doenças ao longo do tempo.

Por último, o suporte à tomada de decisão, a utilização de redes neurais no reconhecimento de doenças nas folhas da soja fornece informações valiosas para a tomada de decisão agrícola, os agricultores e especialistas podem receber alertas automáticos sobre a presença de doenças, avaliar a gravidade da infecção e adotar medidas de controle adequadas, contribuindo para a implementação de estratégias de manejo mais eficazes e redução das perdas na produção.

É visto em trabalhos semelhantes os benefícios citados acimas, como em (JADHAV; UDUPI; PATIL, 2021) que foi proposto uma abordagem de identificação de doenças em folhas de soja, utilizando folhas doentes e saudáveis, para identificar três doenças da soja, com base em abordagem de *transfer learning* usando redes neurais convolucionais AlexNet e GoogleNet pré-treinadas, alcançando uma precisão de 98,75% e 96,25%, respectivamente.

Em (APRENDIZADO. . . , 2019) apresenta um estudo sobre o uso da abordagem baseada em aprendizado profundo para identificar plantas doentes usando imagens de folhas por *transfer learning*, utilizando a arquitetura NASNet para as CNN. O modelo é treinado e testado usando um conjunto de dados do projeto PlantVillage disponível publicamente contendo imagens variadas de folhas de plantas com múltiplas variações, atingindo uma precisão de 93,82%.

Em (HASSAN *et al.*, 2021) modelos profundos de rede neural convolucional são implementados para identificar e diagnosticar doenças em plantas a partir de suas folhas, treinados utilizando *transfer learning* com um conjunto de dados aberto composto por 14 espécies diferentes de plantas e 38 classes categóricas diferentes de doenças e folhas de plantas saudáveis. O desempenho do modelo foi avaliando variando parâmetros como tamanho do lote e números de épocas, alcançando taxas de precisão de classificação de doenças de 98,42%, 99,11%, 97,02% e 99,56% usando InceptionV3, InceptionResNetV2, MobileNetV2 e EfficientNetB0, respectivamente.

3 METODOLOGIA E DESENVOLVIMENTO

No respectivo capítulo os métodos utilizados e o desenvolvimento realizado para a detecção de doenças em folhas de soja são apresentados. É comentado em forma sequencial as principais partes implementadas no presente trabalho, a fim de facilitar o entendimento dos processos usados no projeto e proporcionando que trabalhos futuros análogos a área de pesquisa consigam comparar e validar totalmente ou parcialmente os resultados alcançados pelo autor.

Como mencionado, o processo de detecção de doenças em folhas de soja é formado por quatro etapas fundamentais, a primeira etapa é a chamada de formação de *"Composição do conjunto de dados"* onde é feita toda a pesquisa e junção das imagens obtidos, a segunda etapa é chamada de "Pré-Processamento", onde são aplicadas técnicas nas imagens para entrarem na rede neural de forma mais eficiente para o treinamento. A terceira etapa é chamada de "Modelagem e treinamento de redes neurais e algoritmos tradicionais", onde serão utilizadas duas Redes Neurais Convolucionais, a qual uma é existente e a outra é uma Rede Neural desenvolvida pelo autor, também é utilizado o Descritor GIST, onde as *features* das imagens obtidas são treinados por algoritmos de classificação no software WEKA *(Waikato Environment for Knowledge Analysis)*. Por último, na quarta etapa é onde é realizada a comparação dos resultados das redes neurais e os algoritmos tradicionais. Em seguida serão discutidas em sequência e individualmente com uma maior descrição de detalhes todas as três partes metodológicas utilizadas no projeto.

Para a realização deste trabalho, foi construído um banco de imagens composto de quatro diferentes bancos de imagens de folhas de soja. O primeiro *dataset* utilizado foi do Digiphatos, sendo uma base de dados mantida pela Embrapa (Empresa Brasileira de Pesquisa Agropecuária) reconhecida como uma instituição de pesquisa agrícola de renome no Brasil (BARBEDO; KOENIGKAN; SANTOS, 2016). O respectivo *dataset* contém imagens da folha da soja com 5 lesões, sendo ferrugem da soja, mancha alvo, oídio, *bacterial blight* e míldio, onde as folhas continham lesões desde os índices mais baixos de severidade até os índices mais graves das doenças. No total foram contabilizadas 311 imagens de folhas contaminadas, na Tabela 1 é possível observar as classes contidas juntamente com a quantidade de imagens respectivamente.

O segundo *dataset* utilizado foi do banco de imagem da *Harvard (dataverse)* (HARVARD, 2022), frequentemente utilizado por pesquisadores de diversas áreas, incluindo ciências sociais, ciências naturais, saúde, humanidades e vários outros temas. Ele foi uma fonte valiosa de dados para a composição e análises deste trabalho, pois constituem de imagens de folhas de sojas com a doença *rust*, como mostrado na Tabela 1.

O terceiro *dataset* utilizado neste trabalho foi o banco de imagem da ipmimage (IPMIMAGE, 2022), contendo uma extensa coleção de imagens relacionadas ao MIP (Manejo Integrado de Pragas) e ao controle de pragas em diferentes culturas. O banco

de imagem é mantido pelo CIPMP (Centro de Informação de Pesquisa em Manejo de Pragas), um programa conjunto entre a Universidade da Califórnia e o Departamento de Agricultura dos Estados Unidos. As imagens disponíveis abrangem diferentes tipos de pragas, incluindo insetos, ácaros, nematoides, doenças causadas por fungos, vírus e bactérias, danos causados por herbicidas e outros fatores ambientais. As imagens são classificadas por cultura, tipo de praga e outros critérios relevantes, o que facilita a navegação e a localização de imagens específicas. Além disso, o banco de imagem fornece informações adicionais sobre cada imagem, como descrições, características das pragas e culturas afetadas, permitindo uma compreensão mais completa dos problemas enfrentados no manejo de pragas. Foi um *dataset* que ofereceu uma ampla variedade de imagens que foram úteis para esse trabalho, disponibilizando folhas contaminadas com *bacterial blight*, míldio e *rust*, na Tabela 1 são mostradas as quantidades.

O quarto e último *dataset* utilizado, foi o banco de imagem da *Mendeley Data*, com folhas saudáveis *(healthy* (MENDELEY, 2022), como observado na Tabela 1.

Tabela 1 – Tabela dos quatro bancos de imagens.

CLASSE	**DIGIPHATOS**	**HARVARD**	**IPMIMAGE**	**MENDELEY**
HEALTHY	-	-	-	240
RUST	65	139	36	-
BACTERIAL BLIGHT	57	-	43	-
MÍLDIO	51	-	45	-
OÍDIO	77	-	-	-
MANCHA ALVO	62	-	-	-

3.1 COMPOSIÇÃO DO CONJUNTO DE DADOS

As imagens que constituem o banco de imagem foram divididas em três categorias,sendo treinamento, teste e validação, totalizando 815 imagens, descritas na Tabela 2.

Tabela 2 – Tabela do banco de imagem do projeto.

CLASSE	**QUANTIDADE**
RUST	240
HEALTHY	240
BACTERIAL BLIGHT	100
MÍLDIO	96
OÍDIO	77
MANCHA ALVO	62

Para a constituição do grupo de treinamento 74,11% das imagens foram utilizadas, 8,71% das imagens usadas no grupo de teste, o restante 17,18% das imagens no grupo

de validação, como mostrado na Tabela 3. A maior parte das imagens que são de folhas no meio da lavoura, estão no grupo de validação e teste, a fim de ter um resultado real, pois são imagens parecidas com as que os cooperados irão tirar, quando o modelo for implementado na cooperativa.

Tabela 3 – Tabela do conjunto de imagens utilizadas no treinamento, teste e validação.

CONJUNTO DE IMAGENS	**QUANTIDADE**	**PORCENTAGEM**
TREINAMENTO	604	74,11%
VALIDAÇÃO	140	17,18%
TESTE	71	8,71%

A partir da divisão do banco de imagem, foi avaliado a capacidade das técnicas em identificar a existência das doenças nas folhas e a aptidão de classificar o tipo da classe entre 6 existentes.

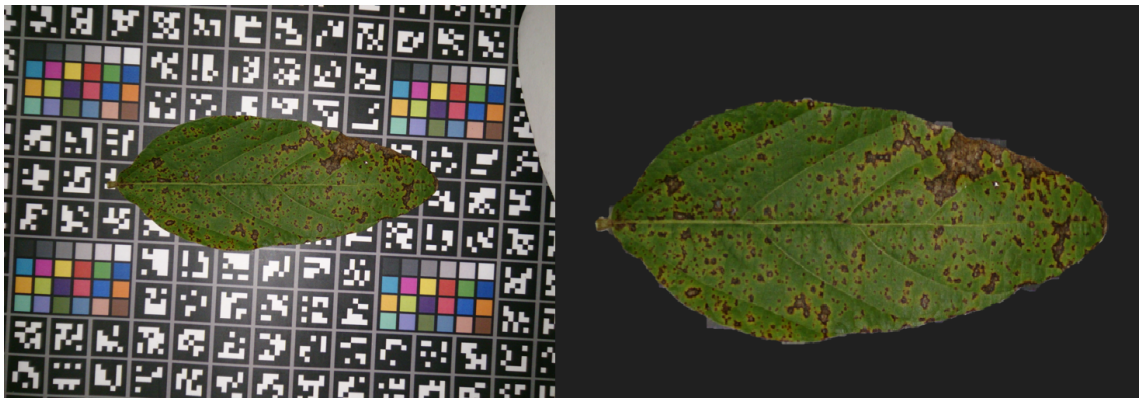
3.2 PRÉ-PROCESSAMENTO

A etapa de pré-processamento dos dados é de grande importância para o treinamento da rede neural, sendo possível aplicar diversas técnicas, como a função *ImageDataGenerator* da biblioteca do *Keras*. A seguir são exemplificadas as técnicas usadas nesta respectiva etapa do projeto.

1. Ajuste de dimensões: Como os diferentes bancos de dados utilizados nesse projeto têm diferentes tamanhos, necessitou-se realizar um redimensionamento das imagens. Por meio de experimentos realizados via treinamento da rede neural, com diversos tamanhos de imagens, o tamanho de 128 *pixels* de altura por 128 *pixels* de largura foi a dimensão de imagem escolhida, por ter o melhor resultado de acertos no treinamento.
2. Ajuste de escala: Como mencionado anteriormente o presente trabalho utiliza imagens no formato RGB, resultando em uma representação matemática de 3 valores de 0 a 255. Para o treinamento dos classificadores foi utilizado a opção *rescale*=1/255 para normalizar os valores de pixel das imagens, dividindo cada valor por 255 para que fiquem no intervalo de 0 a 1, para garantir que os valores das imagens estejam na mesma escala e facilitem o treinamento da rede neural, realizando esse processo para os 3 conjuntos de dados: Treinamento, teste e validação.
3. Ponto de interesse: No banco de imagem fornecido pela *dighphatos* as imagens continham uma espécie de *QRcode* no fundo da imagem, o que poderia prejudicar o resultado final da rede neural, podendo dar um valor falso nos parâmetros analisados, com base nesse problema, foi realizado o recorte da imagem de forma

manual, deixando apenas o ponto de real interesse na imagem, como mostrado na Figura 14.

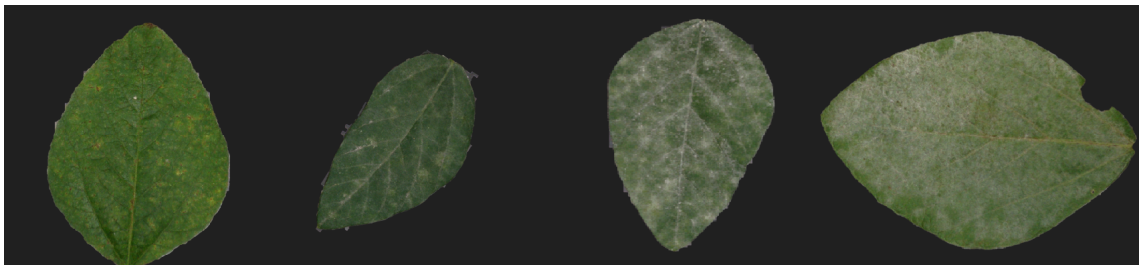
Figura 14 – Recorte realizado para obter o ponto de interesse.



Fonte: Modificada de Barbedo, Koenigkan e Santos (2016).

4. Rotação e zoom: Esses processos foram realizados manualmente, virando as folhas em diferentes posições e diferentes distâncias, a fim de diversificar as posições das folhas, com a finalidade de deixar o banco de imagem o mais próximo da realidade, deixando a CNN o mais precisa possível em diferentes tipos de imagens, com o intuito de deixar o banco de imagem o mais diversificado possível para conseguir se assemelhar as fotos tiradas pelos cooperados. Como mostrado na Figura 15.

Figura 15 – Rotações e Zoom.



Fonte: Modificada de Barbedo, Koenigkan e Santos (2016).

3.3 MODELAGEM E TREINAMENTO DE REDES NEURAIS E ALGORITMOS TRADICIONAIS

Nesta seção são apresentadas as duas redes neurais convolucionais utilizadas neste projeto, uma chamada de MobileNetV2 e a outra uma arquitetura própria, desenvolvida pelo autor. Foram utilizadas bibliotecas específicas e linguagem de programação *Python* para o treinamento das redes neurais, a aplicação das técnicas são exercidas na plataforma de desenvolvimento e execução online, *Google Collaboratory*, em paralelo com o *Google Drive*, a fim de facilitar o manuseio dos dados e treinamentos.

Com base nos treinamentos serão exercidas inferências aplicando o conjunto de teste para extração de métricas de acurácia, reproduzidas por matrizes de confusão e tabelas de precisão, sensibilidade *(recall)* e F1-score.

3.3.1 Métricas

As métricas são utilizadas para avaliar o desempenho e a qualidade de modelos de aprendizado de máquina, permitindo quantificar o quão bem o modelo está realizando determinada previsão.

3.3.1.1 Matriz de confusão

A matriz de confusão é uma tabela que mostra o desempenho de um classificador, apresentando os valores reais e preditos dos dados de teste, permitindo identificar padrões de classificação corretos e identificar classes que podem estar sendo confundidas com mais frequência, a Figura 16 é um exemplo de matriz de confusão multiclasse.

Figura 16 – Matriz de confusão.

Fonte: Joy (2020).

3.3.1.2 Acurácia

A acurácia de uma rede neural é uma métrica comumente usada para avaliar o desempenho de um modelo de aprendizado de máquina, medindo a proporção de exemplos que são classificados corretamente pelo modelo. Matematicamente, pode ser expresso da seguinte forma:

$$Acurácia = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Sendo, verdadeiros positivos*(true positive)* (TP), verdadeiros negativos *(true negative)* (TN), falsos positivos *(false positive)* (FP) e falsos negativos *(false negative)* (FN).

Vale ressaltar que, a acurácia pode ser influenciada por desequilíbrios de classes nos dados de teste, quando há uma distribuição desigual das classes, o modelo pode obter uma alta acurácia simplesmente classificando a classe majoritária corretamente, enquanto falha na classificação da classe minoritária(CHICCO; JURMAN, 2020). Devido a esse tipo de incerteza, foram avaliadas outras métricas para analisar o desempenho das redes neurais, como já citado a cima e descritas a baixo.

3.3.1.3 Precisão

É uma métrica utilizada para avaliar o desempenho de uma rede neural, medindo a proporção de exemplos positivos corretamente classificados em relação ao total de exemplos positivos previstos pelo modelo.

A precisão é calculada dividindo o número de verdadeiros positivos (TP) pelo somatório dos verdadeiros positivos e falsos positivos (FP). Em termos matemáticos, pode ser expressa como:

$$Precisão = \frac{TP}{TP + FP} \tag{2}$$

A precisão é uma métrica importante quando é crucial minimizar os falsos positivos, ou seja, evitar classificar erroneamente exemplos negativos como positivos. É particularmente relevante em casos em que os falsos positivos têm um impacto significativo ou prejudicial.

No entanto, a precisão não leva em consideração os casos em que um exemplo positivo é erroneamente classificado como negativo (falso negativo). Devido a esse motivo foi realizado a métrica de *recall*, para obter uma avaliação mais abrangente do desempenho da rede neural.

3.3.1.4 Sensibilidade (recall)

Também conhecido como taxa de verdadeiros positivos (TPR), é utilizado para avaliar o desempenho de um modelo de rede neural, medindo a proporção de exemplos positivos corretamente identificados em relação ao total de exemplos positivos existentes no conjunto de dados.

O *recall* é calculado dividindo o número de verdadeiros positivos (TP) pelo somatório dos verdadeiros positivos e falsos negativos (FN). Em termos matemáticos, pode ser expresso como:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

O *recall* é particularmente útil em situações em que é importante minimizar os falsos negativos, ou seja, evitar classificar erroneamente exemplos positivos como negativos, sendo

relevante em casos em que os falsos negativos têm um impacto significativo ou prejudicial, como na detecção de doenças graves.

No entanto, o *recall* não leva em consideração os casos em que um exemplo negativo é erroneamente classificado como positivo (falso positivo). Devido a esse motivo foi realizado a métrica de precisão, para obter uma avaliação mais abrangente do desempenho da rede neural.

3.3.1.5 F1-score

O F1-score é uma medida que combina a precisão e a sensibilidade de uma rede neural em uma única métrica, é apropriado quando se deseja ter uma avaliação balanceada entre a precisão e o *recall*.

O F1-score é calculado como a média harmônica entre a precisão e o *recall*. Matematicamente, pode ser expresso da seguinte forma:

$$F1 - score = \frac{2 * Precisão * Recall}{Precisão + Recall} \tag{4}$$

O F1-score varia de 0 a 1, onde 1 representa um desempenho perfeito e 0 indica um desempenho ruim, buscando encontrar um equilíbrio entre a capacidade de um modelo em identificar corretamente exemplos positivos *(recall)* e evitar classificar erroneamente exemplos negativos como positivos (precisão). É especialmente útil em problemas com desequilíbrio de classes (CHICCO; JURMAN, 2020).

3.3.2 Extração de *features* e utilização de algoritmos tradicionais

Para o descritor GIST, foi aproveitado algoritmos reimplementados do código utilizado em (OLIVA; TORRALBA, 2001), sendo o tamanho do vetor descritor se mantendo o mesmo. Com o valor de *features* estabelecido, os descritores das imagens foram extraídos em um arquivo no formato .csv, em que cada linha corresponde uma imagem e cada coluna apresenta um valor adquirido pelo descritor, com ressalva na última coluna, pois foi inserido o rótulo das imagens, sendo fundamental na etapa seguinte de treinamento dos dados.

A próxima etapa, foi realizar o treinamento dos dados alcançados pelo descritor no software WEKA *(Waikato Environment for Knowledge Analysis)*, desenvolvido na Nova Zelândia, pela Universidade de Waikato, contendo diversos algoritmos de aprendizado de máquina. Para este trabalho foram utilizados os seguintes algoritmos de classificação:

- J48: é um algoritmo de aprendizado de máquina baseado em árvore de decisão, desenvolvido a partir do algoritmo C4.5, sendo frequentemente utilizado para problemas de classificação em que se deseja construir um modelo que possa tomar decisões com base em características (BHARGAVA *et al.*, 2013).

- IBk: também conhecido como k-Nearest Neighbors (k-NN), é um algoritmo de aprendizado de máquina baseado em instâncias, usado principalmente para problemas de classificação, embora também possa ser aplicado a problemas de regressão (VIJAYARANI; MUTHULAKSHMI, 2013).
- Naive Bayes: O Naive Bayes é um algoritmo de classificação probabilístico que se baseia no teorema de Bayes e na suposição de independência condicional entre os atributos (RISH *et al.*, 2001).

3.3.3 MobileNetV2

Com o objetivo de avaliar a efetividade de uso das redes neurais artificiais como extratores de características, buscou-se no início utilizar o modelo de uma rede pré-treinada, neste caso a MobileNetV2.

Foi realizado *transfer learning* utilizando todas as classes do banco de imagem constituído pelo autor, mostrado na Tabela 2 . Logo foram adicionadas duas últimas camadas nessa CNN, a fim de ser possível treina-la para responder aos objetivos deste trabalho.

A primeira camada adicionada no final da rede MobileNetV2 foi a *GlobalAvarage-Pooling2D*, para reduzir a dimensionalidade dos mapas de características gerados pelas camadas convolucionais anteriores, fazendo com que essa camada resumisse as informações espaciais em cada mapa de características em um valor médio. A principal vantagem do *Global Average Pooling* é que ele fornece uma representação compacta das características extraídas pela rede, e ao resumir as informações em um valor médio por característica, o *Global Average Pooling* reduz drasticamente o número de parâmetros da rede e evita o *overfitting*, o que é especialmente útil em casos de limitações computacionais e quando há um número limitado de amostras de treinamento. Outra vantagem é que o *Global Average Pooling* torna a rede invariante a variações de escala e translação nas imagens de entrada, o que significa que a rede será capaz de reconhecer objetos ou padrões independentemente de sua posição exata na imagem, tornando o modelo mais robusto e generalizável.

A segunda e última camada adicionada na rede foi uma camada de previsão *(prediction layer)*, é a camada final em uma rede neural, responsável por produzir as previsões finais do modelo. Com isso, a camada densa foi configurada para ter 6 neurônios na saída, pois o banco de imagem do projeto é composto por 6 classes, e uma função de ativação do tipo *softmax*, comumente usada para problemas de classificação multiclasse, onde a rede neural precisa atribuir uma probabilidade para cada classe, normalizando as saídas para que a soma de todas as probabilidades seja igual a 1.

A escolha da utilização da rede neural MobileNetV2 teve alguns motivos, dentre eles, durante pesquisas foi notado que ela foi projetada para tarefas de visão computacional em dispositivos móveis, que é o objetivo futuro deste projeto. Ela é uma evolução do modelo MobileNet original, desenvolvida pelo Google e introduzida em 2018. A arquitetura

MobileNetV2 é conhecida por sua eficiência e baixo consumo de recursos computacionais, tornando-a adequada para dispositivos com restrições de poder de processamento, como *smartphones* e dispositivos embarcados. Utilizando uma combinação de técnicas, como camadas de convolução separáveis em profundidade, expansão linear e resíduos lineares, para alcançar um equilíbrio entre precisão e eficiência. A ideia por trás das camadas de convolução separáveis em profundidade é reduzir a complexidade computacional, dividindo as operações de convolução em duas etapas separadas: uma convolução em profundidade e uma convolução espacial separada, permitindo uma redução significativa no número de parâmetros e operações, tornando a rede mais leve (HOWARD *et al.*, 2017), com um total de 2,265,670 parâmetros.

A expansão linear é uma técnica utilizada para aumentar a dimensão das *features maps* nas camadas intermediárias da rede, antes de aplicar as convoluções espaciais, ajudando a capturar informações mais ricas e complexas, permitindo uma melhor representação dos dados. Os resíduos lineares *(linear bottlenecks)* são usados para conectar as camadas intermediárias da rede, permitindo que informações sejam transmitidas diretamente de uma camada para outra, facilitando o fluxo de gradientes durante o treinamento.

Figura 17 – Arquitetura Geral da Rede Neural MobileNetV2, t= fator de expansão, c= números de canais de saída, n= número de repetição, s= strides.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fonte: Sandler *et al.* (2018).

Figura 18 – Bloco residual *Bottlenecks* transformando k canais para k', com stride igual a "s" e fator de expansão "t".

Input	Operator	Output
$h \times w \times k$	1x1 conv2d , ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	3x3 dwise s=s, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	linear 1x1 conv2d	$\frac{h}{s} \times \frac{w}{s} \times k'$

Fonte: Sandler *et al.* (2018).

No geral, o MobileNetV2 é uma arquitetura de rede neural eficiente que combina técnicas inteligentes para reduzir o consumo de recursos computacionais, mantendo um excelente desempenho em tarefas de visão computacional, sendo amplamente utilizado em aplicativos móveis e projetos de aprendizado de máquina em dispositivos com recursos limitados.

3.3.4 CNN com arquitetura própria

A arquitetura própria foi desenvolvida operando as bibliotecas *Keras* e *TensorFlow*, conforme apresentado na Figura 19. Foi criado um modelo sequencial, onde as camadas são empilhadas sequencialmente, posteriormente foi adicionada uma camada de convolução 2D com 64 filtros, um tamanho de *kernel* de 5x5 e função de ativação ReLU e *softmax*. A camada tem um preenchimento *("padding"= "same")*, o que significa que a entrada é preenchida com zeros para manter a mesma dimensão de largura e altura, conta também com uma camada de *pooling* 2D, *(max pooling)* com um *pool size* de 2x2 e um passo *(strides)* de 1. Isso reduz a dimensionalidade espacial da saída da camada anterior, e adiciona uma camada de normalização em lote *(batch normalization)*, responsável por normalizar as ativações da camada anterior, ajudando na estabilização e aceleração do treinamento da rede. Esses passos são repetidos duas vezes adicionando mais camadas de convolução, *pooling* e normalização em lote.

Depois, é adicionada uma camada de *dropout* com uma taxa de *dropout* de 0,15. O *dropout* é uma técnica de regularização que desativa aleatoriamente unidades durante o treinamento, ajudando a reduzir o *overfitting*. Após isso, uma camada de achatamento *(flatten)* é adicionada para converter o tensor de saída 3D em um vetor 1D, preparando-o para as camadas totalmente conectadas. E é adicionada uma camada totalmente conectada com 128 neurônios e função de ativação ReLU. Por último, é adiciona a camada de saída com 6 neurônios, correspondendo às classes de destino. A função de ativação *softmax* é usada para obter probabilidades das diferentes classes, como explicado anteriormente.

Figura 19 – Arquitetura própria da CNN desenvolvida pelo autor.

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 128, 128, 64)
max_pooling2d (MaxPooling2D)	(None, 127, 127, 64)
batch_normalization (BatchNormalization)	(None, 127, 127, 64)
conv2d_1 (Conv2D)	(None, 127, 127, 64)
max_pooling2d_1 (MaxPooling2D)	(None, 126, 126, 64)
batch_normalization_1 (BatchNormalization)	(None, 126, 126, 64)
conv2d_2 (Conv2D)	(None, 126, 126, 64)
max_pooling2d_2 (MaxPooling2D)	(None, 125, 125, 64)
dropout (Dropout)	(None, 125, 125, 64)
flatten (Flatten)	(None, 1000000)
dense (Dense)	(None, 128)
dense_1 (Dense)	(None, 6)

Fonte: Elaborada pelo autor.

3.3.5 Treinamento e testes

Para a realização do treinamento e testes da rede neural MobileNetV2 e a rede neural desenvolvida pelo autor, foram utilizadas as ferramentas listadas a seguir:

- Python: Linguagem de programação de alto nível, é usado em uma ampla variedade de áreas, como por exemplo automação de tarefas e inteligência artificial;
- TensorFlow 2.8.2: Biblioteca usada para gerar os dois modelos de redes neurais utilizados no trabalho, realizando o treinamento e testes dos mesmos;
- Numpy: Biblioteca para a linguagem de programação Python, usada para efetuar operações de estruturas de dados, como matrizes e vetores;
- Keras: Biblioteca de alto nível para construção e treinamento de redes neurais, também oferecendo suporte para treinamento de redes neurais em GPUs, o que acelera significativamente o processo de aprendizado;
- Windows 11 Pro: Sistema operacional realizado para criação do ambiente de desenvolvimento e laboração dos testes;

- Acer Nitro 5: Notebook utilizado para o desenvolvimento do projeto, possuindo as configuração a seguir: Processador Intel(R) Core(TM) i5-9300H 2.40GHz, Memória RAM DDR4 de 8 GB e GPU GTX 1650 NVIDIA-SMI 460.32.03.

Para as etapas de treinamento e testes da rede neural mobilenetv2, foram divididas em 12 etapas, feitas progressões no *batch-size* (tamanho do lote) e nas *epochs*.

O *batch-size* é um parâmetro muito importante usado durante o treinamento de redes neurais, determinando o número de amostras de treinamento que serão processadas em cada passagem *(epoch)* durante o treinamento e a escolha do tamanho do lote afeta a eficiência computacional, a precisão do modelo e a capacidade de generalização.

Uma *epoch* é uma passagem completa por todo o conjunto de dados de treinamento, na qual, cada amostra de treinamento é apresentada à rede neural, os cálculos são realizados para propagar as ativações pela rede, os erros são calculados e, em seguida, os pesos são atualizados com base na retropropagação do erro, o número de *epochs* é muito importante, pois é um parâmetro que define a quantidade de vezes que o algoritmo percorrerá todo o conjunto de dados de treinamento. Um número insuficiente de *epochs* pode levar a um modelo subajustado, com baixa capacidade de generalização, enquanto um número excessivo de *epochs* pode levar ao *overfitting*, onde o modelo se ajusta demasiadamente aos dados de treinamento, perdendo a capacidade de generalizar para novos dados.

Com o resultado desses dois parâmetros foi avaliado em quais métricas a rede melhor se saia, na Tabela 4 é mostrada os valores usados dos respectivos parâmetros. O modelo também foi configurado para receber uma entrada de dados iguais a 128x128x3 *Pixels*, por fim uma saída é gerada com uma resposta apontando a probabilidade que varia de 0 a 1 mostrando quais são as chances da imagem que foi classificada corresponder a alguma das seis classes de saídas presentes na rede neural.

Tabela 4 – Tabela dos parâmetros usados de *batch-size* e *epoch* no treinamento do modelo MobileNeV2.

ETAPA	**EPOCH**	**BATCH-SIZE**
1	10	8
2	20	8
3	50	8
4	100	8
5	10	16
6	20	16
7	50	16
8	100	16
9	10	32
10	20	32
11	50	32
12	100	32

Para as etapas de treinamentos e testes da rede neural desenvolvida pelo autor, foram feitas em 5 etapas, também desenvolvendo progressões de *batch-size* e *epoch* nas 4 primeiras etapas, na última etapa foi adicionado um *dropout* com taxa de 0,15 e sem *batchnormalization* na última camada, como observado na Tabela 5 com entradas configuradas de 128x128x3, a fim de que se possa ser comparado as duas redes neurais e qual teria um melhor resultado para poder ser escolhida para o projeto.

Tabela 5 – Tabela dos parâmetros usados de *batch-size*, *epoch*, e alterações de outros parâmetros no treinamento do modelo desenvolvido pelo autor.

ETAPA	**EPOCH**	**BATCH-SIZE**	**DROPOUT**	**BATCHNORMALIZATION**
13	10	8	NÃO	SIM
14	20	8	NÃO	SIM
15	50	8	NÃO	SIM
16	100	8	NÃO	SIM
17	20	8	0,15	NÃO NA ÚLTIMA CAMADA

As Tabelas 4 e 5 ilustram todas as etapas de treinamento que foram efetuadas com os dois modelos de redes neurais convolucionais, totalizando dezessete treinamentos, em todas as etapas de treinamento foram utilizadas as mesmas quantidades de imagens contidas no banco de imagem.

4 RESULTADOS

Nesta seção são discorridos os resultados obtidos pelos modelos propostos no Capítulo 3 e experimentos realizados conforme as Tabelas 4 e 5.

Os resultados apresentados na próxima seção mostram a acurácia, precisão sensibilidade e f1-score, obtidas pelos conjuntos de dados de validação e teste após o treinamento. A fim de se obter resultados melhores das redes utilizadas, foram adicionadas mais três métricas para avaliação da CNN, a métrica *macro avg* fornece uma visão geral do desempenho médio do modelo, avaliando o modelo de forma equitativa em todas as classes. A outra métrica utilizada é a *weighted avg*, como há um desbalanceamento de quantidade de imagens em cada classe, é interessando analisarmos essa métrica, pois ela fornece uma média ponderada das métricas de avaliação, levando em consideração o número de amostras em cada classe, é uma maneira de avaliar o desempenho do modelo considerando o desequilíbrio entre as classes. A última métrica avaliada é o *Loss*, representando a perda *(loss)* calculada durante o treinamento de um modelo de aprendizado de máquina, indicando o quão bem o modelo está se ajustando aos dados de treinamento, quanto menor o loss melhor.

Nos dados de teste foram utilizadas matrizes de confusão para melhor identificar onde a rede estava obtendo maiores taxas de acertos e erros das classes.

4.1 TREINAMENTO DOS ALGORITMOS TRADICIONAIS ATRAVÉS DAS *FEATURES* OBTIDAS PELO DESCRITOR GIST

A partir dos treinamentos realizados dos algoritmos J48, IBk e Naive Bayes, com os dados das *features* obtidas através do descritor GIST, os resultados das métricas, acurácia, precisão, *recall* e *f1-score* são apresentados na Tabela 6.

Tabela 6 – Tabela de resultados das métricas utilizadas nos algoritmos tradicionais das extrações de *features* pelo descritor GIST.

Algoritmo	J48	**IBk**	Naive Bayes
Acurácia teste	71,8%	**74,6%**	57,7%
Macro avg precisão	0,73	**0,72**	0,36
Weighted avg precisão	0,79	**0,75**	0,50
Macro avg recall	0,67	**0,71**	045
Weighted avg recall	0,71	**0,74**	0,57
Macro avg F1-score	0,67	**0,70**	0,40
Weighted avg F1-score	0,72	**0,73**	0,53

É notável que dentre os três algoritmos utilizados, o que obteve o melhor desempenho, analisando as métricas utilizadas, foi o IBk, alcançando uma acurácia de 74,6%, tendo um desempenho de 2,8% melhor que o J48 e 16,9% que o Naive Bayes.

Com uma melhor performance do algoritmo IBk, foi gerada uma matriz de confusão, Tabela 7, sendo as 6 classes presentes no trabalhos representadas na matriz de confusão como:

- A: Bacterial blight
- B: Healthy
- C: Mancha Alvo
- D: Mildio
- E: Oidio
- F: Ferrugem da soja *(rust)*

Tabela 7 – Matriz de confusão do algoritmo IBk.

Classes	**A**	**B**	**C**	**D**	**E**	**F**
Bacterial blight = A	5	0	1	0	0	4
Healthy = B	0	19	0	0	0	1
Mancha Alvo = C	0	0	5	0	0	0
Mildio = D	2	0	0	3	2	2
Oidio = E	0	0	0	1	5	1
Rust = F	3	0	0	0	1	16

O algoritmo errou um total de 18 imagens, sendo um erro na classe *healthy*, quatro erros em *rust*, acertando todas as imagens na classe mancha alvo, errando cinco imagens em *bacterial blight* correspondendo em 50%, seis erros em míldio e dois erros na classe oídio.

4.2 TREINAMENTO DA CNN MOBILENETV2

A partir dos treinamentos realizados da rede MobileNetV2 com as variações de progressões de *epoch* e *batch-size* mostradas na Tabela 4. Primeiramente as doze etapas foram divididas em três grupos, contendo quatro etapas em cada grupo, sendo representadas por Grupo 1, Grupo 2 e Grupo 3, após isso gerou-se tabelas, gráficos e matrizes de confusão para melhor visualização dos resultados obtidos.

4.2.1 Resultados e análises do Grupo 1

Na Tabela 8 é mostrado as etapas e o conjunto de parâmetros utilizados no Grupo 1.

Tabela 8 – Tabela do Grupo 1 no treinamento do modelo MobileNeV2.

ETAPA	BATCH-SIZE	EPOCH	IMAGENS
1	8	10	815
2	8	20	815
3	8	50	815
4	8	100	815

Os resultados das métricas aplicadas com as etapas descritas na Tabela 8 são mostradas abaixo na Tabela 9.

Tabela 9 – Tabela de resultados das métricas utilizadas referente ao Grupo 1.

Etapa	1	**2**	3	**4**
Acurácia validação	93,5%	**92,1%**	91,4%	**91,4%**
Acurácia teste	88,7%	**92,9%**	91,5%	**92,9%**
Macro avg precisão	0,87	**0,93**	0,91	**0,92**
Weighted avg precisão	0,90	**0,94**	0,92	**0,93**
Macro avg recall	0,82	**0,89**	0,86	**0,89**
Weighted avg recall	0,89	**0,93**	0,92	**0,93**
Macro avg F1-score	0,83	**0,90**	0,87	**0,90**
Weighted avg F1-score	0,89	**0,93**	0,91	**0,93**

A partir dos resultados obtidos através da Tabela 9, é observado que a etapa 1 teve a maior porcentagem de acurácia no conjunto de dados de validação, com 93,5%, porém obteve a menor porcentagem de acurácia no conjunto de testes, métrica considerada mais importante, pois é nela que se encontra as imagens mais parecidas com as que irão ser tiradas pelos usuários. Com isso, as etapas 2 e 4 foram as que obtiveram os melhores resultados do grupo 1.

4.2.2 Resultados e análises do Grupo 2

Na Tabela 10 é mostrado as etapas e o conjunto de parâmetros utilizados no grupo 2.

Tabela 10 – Tabela do Grupo 2 no treinamento do modelo MobileNeV2.

ETAPA	BATCH-SIZE	EPOCH	IMAGENS
5	16	10	815
6	16	20	815
7	16	50	815
8	16	100	815

Os resultados das métricas aplicadas com as etapas descritas na Tabela 10 são mostradas abaixo na Tabela 11.

Tabela 11 – Tabela de resultados das métricas utilizadas referente ao Grupo 2.

ETAPA	5	6	**7**	**8**
Acurácia validação	92,1%	92,1%	**93,5%**	**90,7%**
Acurácia teste	90,1%	91,5%	**92,9%**	**92,9%**
Macro avg precisão	0,89	0,90	**0,92**	**0,92**
Weighted avg precisão	0,91	0,91	**0,93**	**0,93**
Macro avg recall	0,84	0,87	**0,89**	**0,89**
Weighted avg recall	0,90	0,92	**0,93**	**0,93**
Macro avg F1-score	0,85	0,88	**0,90**	**0,90**
Weighted avg F1-score	0,90	0,91	**0,93**	**0,93**

Com os resultados obtidos através da Tabela 11, é notado que as etapas 5 e 6 tiveram um resultado inferior quando comparado com as etapas 7 e 8, que obtiveram os melhores resultados, dando uma diferença de mais de 2% e 1,5% na métrica de acurácia de teste, obtendo uma ligeira diferença positiva nas demais métricas, sendo os destaques do grupo 2.

4.2.3 Resultados e análises do Grupo 3

Na Tabela 12 é mostrado as etapas e o conjunto de parâmetros utilizados no grupo 3.

Tabela 12 – Tabela do Grupo 3 no treinamento do modelo MobileNeV2.

ETAPA	**BATCH-SIZE**	**EPOCH**	**IMAGENS**
9	32	10	815
10	32	20	815
11	32	50	815
12	32	100	815

Os resultados das métricas aplicadas com as etapas descritas em na Tabela 12 são mostradas abaixo na Tabela 13.

Tabela 13 – Tabela de resultados das métricas utilizadas referente ao Grupo 3.

Etapa	9	10	**11**	**12**
Acurácia validação	92,8%	91,4%	**91,4%**	**92,1%**
Acurácia teste	88,7%	87,3%	**92,9%**	**90,1%**
Macro avg precisão	0,88	0,87	**0,92**	**0,88**
Weighted avg precisão	0,90	0,89	**0,93**	**0,90**
Macro avg recall	0,83	0,81	**0,89**	**0,85**
Weighted avg recall	0,89	0,87	**0,93**	**0,90**
Macro avg F1-score	0,84	0,82	**0,90**	**0,86**
Weighted avg F1-score	0,89	0,87	**0,93**	**0,90**

Como observado na Tabela 13, novamente a acurácia de validação na etapa 9 com a menor quantidade de *epoch* teve o melhor resultado, igualmente mostrado na etapa 1 9, porém quando é analisado as outras métricas, cujo tem uma maior relevância, ela tem um desempenho inferior das demais etapas desse grupo. A etapa 10 tem o mesmo valor de acurácia de validação que a etapa 11, porém, uma diferença de mais de 5% quando comparada com a acurácia de teste, o que é um valor considerado grande para uma rede neural, com isso, os melhores resultados desse grupo foram as etapas 11 e 12.

4.2.4 Resultados e análises das etapas que obtiveram os melhores desempenhos

Na Tabela 14 é mostrado as etapas que obtiveram os melhores resultados e o conjunto de parâmetros utilizados.

Tabela 14 – Tabela das etapas que apresentaram um melhor desempenho no treinamento do modelo MobileNeV2.

ETAPA	**BATCH-SIZE**	**EPOCH**	**IMAGENS**
2	8	20	815
4	8	100	815
7	16	50	815
8	16	100	815
11	32	50	815
12	32	100	815

A fim de facilitar a visualização e comparação dos resultados obtidos nas melhores etapas, é criada uma nova tabela com as respectivas etapas, Tabela 15.

Tabela 15 – Tabela de resultados das métricas utilizadas nas etapas que tiveram os melhores desempenhos.

Etapa	**2**	**4**	**7**	8	**11**	12
Acurácia validação	**92,1%**	**91,4%**	**93,5%**	92,1%	**91,4%**	92,1%
Acurácia teste	**92,9%**	**92,9%**	**92,9%**	90,1%	**92,9%**	90,1%
Macro avg precisão	**0,93**	**0,92**	**0,92**	0,88	**0,92**	0,88
Weighted avg precisão	**0,94**	**0,93**	**0,93**	0,90	**0,93**	0,90
Macro avg recall	**0,89**	**0,89**	**0,89**	0,85	**0,89**	0,85
Weighted avg recall	**0,93**	**0,93**	**0,93**	0,90	**0,93**	0,90
Macro avg F1-score	**0,90**	**0,90**	**0,90**	0,86	**0,90**	0,86
Weighted avg F1-score	**0,93**	**0,93**	**0,93**	0,90	**0,93**	0,90

Como observado, a melhor taxa de acurácia de teste atingida pela rede neural foi de 92,9% nas etapas 2, 4, 7 e 11, com uma diferença de quase 3% entre as etapas que tiveram um menor desempenho, etapas 8 e 12.

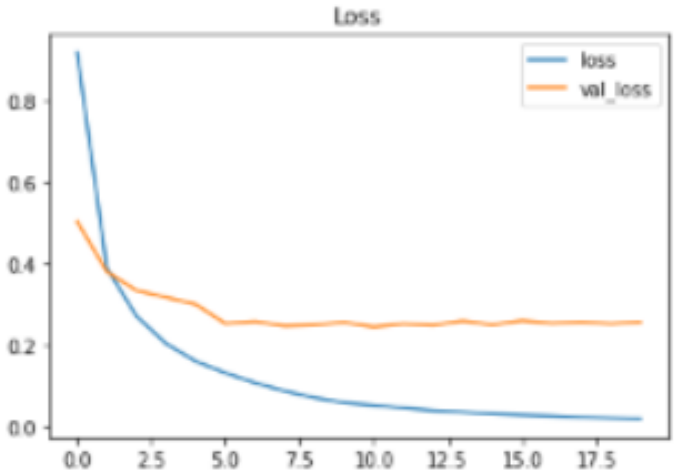
Nas etapas que alcançaram os melhores resultados, foram gerados gráficos de *loss* e matrizes de confusão.

As linhas dos gráficos de *loss* são representadas como:

- Azul: Treinamento
- Laranja: Validação

Os gráficos e tabelas citadas acima, serão apresentadas e analisadas por ordem crescente das etapas citadas na Tabela 15.

Figura 20 – Gráfico de *loss* da etapa 2.



Fonte: Elaborada pelo autor.

Tabela 16 – Matriz de confusão da etapa 2.

Classes	A	B	C	D	E	F
Bacterial blight = A	7	0	0	2	0	1
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	0	0	4	0	1	0
Mildio = D	0	0	0	9	0	0
Oidio = E	0	0	0	1	6	0
Rust = F	0	0	0	0	0	20

Tabela 17 – Tabela de precisão, recall e f1-score da etapa 2.

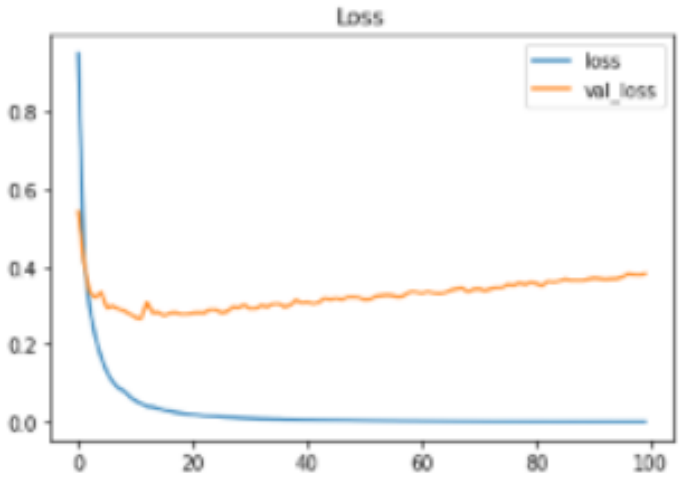
Classes	Precisão	Recall	F1-score	Quantidade
Bacterial blight = A	1.00	0.70	0.82	10
Healthy = B	1.00	1.00	1.00	20
Mancha Alvo = C	1.00	0.80	0.89	5
Mildio = D	0.75	1.00	0.86	9
Oidio = E	0.86	0.86	0.86	7
Rust = F	0.95	1.00	0.98	20

Na etapa 2, como é observado na Figura 20, a linha de validação tem uma queda significativa, porém, com o decorrer das *epochs* ela começa a se distanciar da linha de treinamento, o que pode significar que o modelo está se ajustando demais aos dados de treinamento e não está generalizando bem para os dados de validação, resultando em um *loss* de 0,25 para a validação.

Na matriz de confusão, Tabela 16, é visto que na primeira classe, *bacterial blight* de 10 imagens, 7 foram classificadas corretamente e 3 classificadas erradamente, sendo 2 destas classificadas como *mildio* e 1 como *rust*. Na segunda classe, *healthy* com 20 imagens, a rede apresentou 100% de assertividade. Na terceira classe, *mancha alvo*, com um toral de 5 imagens, a rede errou 1 imagem, sendo classificada como *oidio*. A quarta classe, *mildio* com 9 imagens, a rede neural também foi capaz de acertar 100% das imagens. A quinta classe, *oidio*, com 7 imagens, 1 imagem foi classificada erroneamente, sendo classificada como *mildio*. Na última classe, *rust* com 20 imagens, a rede também foi capaz de acertar 100% das imagens.

Quando a rede não tem erros nas classes, como nas classes B, D e F, é visto que o *recall* tem valores de 1.00, e nem sempre 1.00 na precisão, como visto na Tabela 17, resultando em um total de 5 erros na somatória das classes.

Figura 21 – Gráfico de *loss* da etapa 4.



Fonte: Elaborada pelo autor.

Tabela 18 – Matriz de confusão da etapa 4.

Classes	**A**	**B**	**C**	**D**	**E**	**F**
Bacterial blight = A	7	0	0	1	0	2
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	0	0	4	0	1	0
Mildio = D	0	0	0	9	0	0
Oidio = E	1	0	0	0	6	0
Rust = F	0	0	0	0	0	20

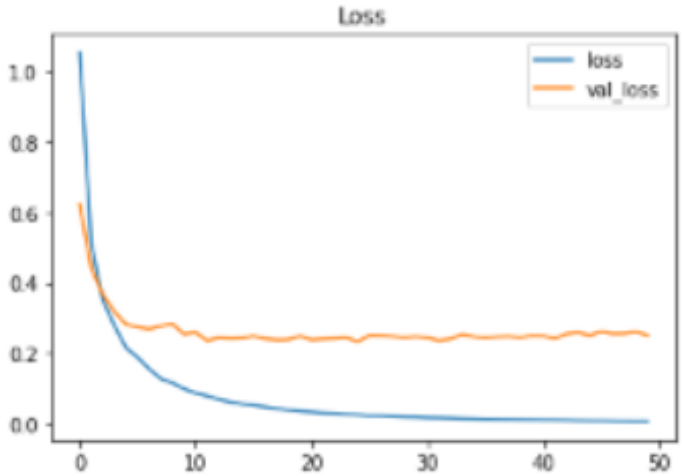
Tabela 19 – Tabela de precisão, recall e f1-score da etapa 4.

Classes	Precisão	Recall	F1-score	Quantidade
Bacterial blight = A	0.88	0.70	0.78	10
Healthy = B	1.00	1.00	1.00	20
Mancha Alvo = C	1.00	0.80	0.89	5
Mildio = D	0.90	1.00	0.95	9
Oidio = E	0.86	0.86	0.86	7
Rust = F	0.91	1.00	0.95	20

Na etapa 4, se observa na Figura 21, uma queda das linha e posteriormente um pequeno distanciamento entre as linhas e treinamento e validação, significando que pode ser um sinal de *overfitting*, com um valor de 0,38 para o *loss* de validação.

Na matriz de confusão e tabela de métricas, Tabela 18 e Tabela 19, é visto um acerto de 100% na segunda, quarta e sexta classe. Errando na primeira classe, classificando 1 imagem como *mildio* e 2 como *rust*, na terceira classe, foi classificada uma imagem incorretamente como *oidio*, e na quinta classe, classificando uma imagem incorretamente como *bacterial blight*, resultando em uma somatória total de 5 erros.

Figura 22 – Gráfico de *loss* da etapa 7.



Fonte: Elaborada pelo autor.

Tabela 20 – Matriz de confusão da etapa 7.

Classes	A	B	C	D	E	F
Bacterial blight = A	7	0	0	2	0	1
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	0	0	4	0	1	0
Mildio = D	0	0	0	9	0	0
Oidio = E	1	0	0	0	6	0
Rust = F	0	0	0	0	0	20

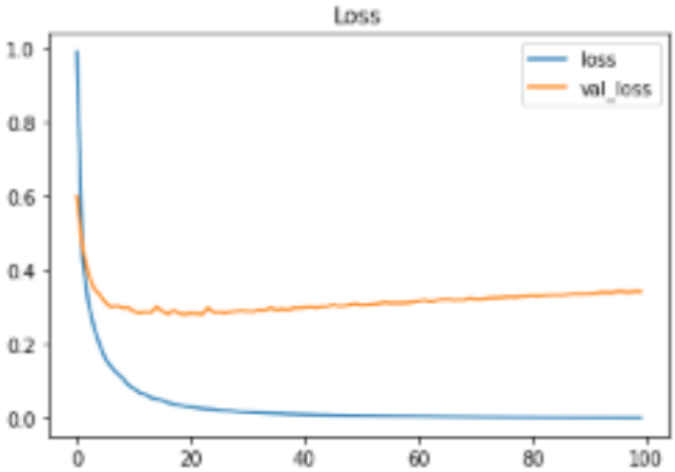
Tabela 21 – Tabela de precisão, recall e f1-score da etapa 7.

Classes	Precisão	Recall	F1-score	Quantidade
Bacterial blight = A	0.88	0.70	0.78	10
Healthy = B	1.00	1.00	1.00	20
Mancha Alvo = C	1.00	0.80	0.89	5
Mildio = D	0.82	1.00	0.90	9
Oidio = E	0.86	0.86	0.86	7
Rust = F	0.95	1.00	0.98	20

Na etapa 7, se observa na Figura 22, no gráfico que se tem um leve evolução de distanciamento entre as linhas de treinamento e validação, resultando em 0,25 para o *loss* de validação.

Na matriz de confusão e tabela de métricas, Tabela 20 e Tabela 21, é visto o acerto de 100% na segunda, quarta e sexta classe. Na primeira classe, três classificações erradas, sendo uma imagem classificada como *mildio* e duas imagens como *rust*. A terceira classe, teve uma imagem classificada errada, como *oidio*. Na quinta classe, também com um erro, sendo classificada como *bacterial blight*, resultando em 5 erros na somatório das classes.

Figura 23 – Gráfico de *loss* da etapa 8.



Fonte: Elaborada pelo autor.

Tabela 22 – Matriz de confusão da etapa 8.

Classes	A	B	C	D	E	F
Bacterial blight = A	7	0	0	1	0	2
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	0	0	4	0	1	0
Mildio = D	0	0	0	9	0	0
Oidio = E	1	0	0	0	6	0
Rust = F	0	0	0	0	0	20

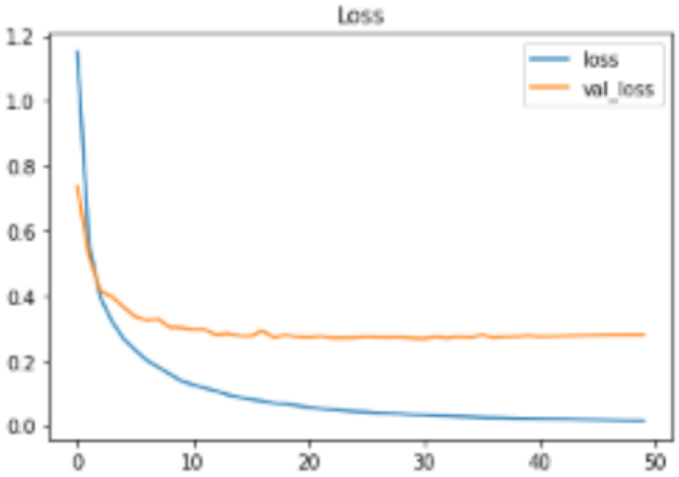
Tabela 23 – Tabela de precisão, recall e f1-score da etapa 8.

Classes	Precisão	Recall	F1-score	Quantidade
Bacterial blight = A	0.88	0.70	0.78	10
Healthy = B	1.00	1.00	1.00	20
Mancha Alvo = C	1.00	0.80	0.89	5
Mildio = D	0.90	1.00	0.95	9
Oidio = E	0.86	0.86	0.86	7
Rust = F	0.91	1.00	0.95	20

Na etapa 8, Figura 23, é possível analisar que a linha de treinamento se distancia muito da linha de treinamento, o que significa que pode estar ocorrendo *overfitting*, com um valor de 0,34 para o *loss* de validação.

Na matriz de confusão e tabela de métricas, Tabela 22 e Tabela 23, é visto o acerto de 100% na segunda, quarta e sexta classe. Na primeira classe, três classificações erradas, sendo uma imagem classificada como *míldio*, e duas imagens classificadas como *rust*. Na terceira classe, 1 imagem é classificada errada, como *oidio*. Na quinta classe, também é apresentado uma classificação incorreta, sendo como *bacterial blight*, resultando em 5 erros na somatória das classes.

Figura 24 – Gráfico de *loss* da etapa 11.



Fonte: Elaborada pelo autor.

Tabela 24 – Matriz de confusão da etapa 11.

Classes	A	B	C	D	E	F
Bacterial blight = A	7	0	0	1	0	2
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	0	0	4	0	1	0
Mildio = D	0	0	0	9	0	0
Oidio = E	1	0	0	0	6	0
Rust = F	0	0	0	0	0	20

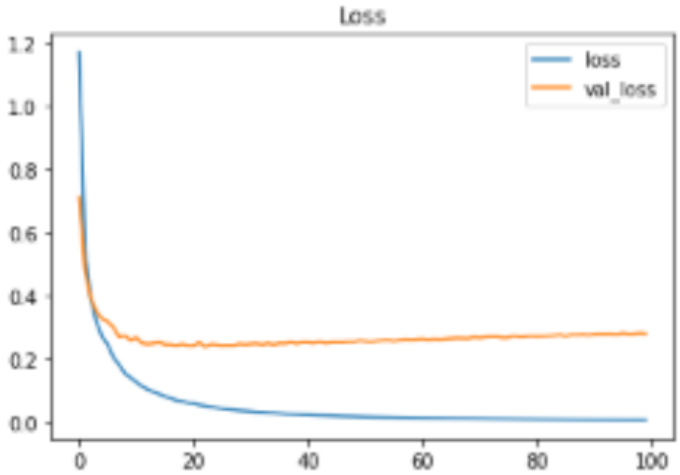
Tabela 25 – Tabela de precisão, recall e f1-score da etapa 11.

Classes	Precisão	Recall	F1-score	Quantidade
Bacterial blight = A	0.88	0.70	0.78	10
Healthy = B	1.00	1.00	1.00	20
Mancha Alvo = C	1.00	0.80	0.89	5
Mildio = D	0.90	1.00	0.95	9
Oidio = E	0.86	0.86	0.86	7
Rust = F	0.91	1.00	0.95	20

Na etapa 11, Figura 24, é possível observar que no final da *epoch* começa a ocorrer um leve distanciamento entre as duas linhas, resultando em um valor de 0,27 para o *loss* de validação.

Na matriz de confusão e tabela de métricas, Tabela 24 e Tabela 25, como observado, a segunda, quarta e sexta classe, tiveram 100% de assertividade. Com 3 erros na primeira classe, sendo uma imagem classificada como *míldio* e duas classificadas como *rust*. Na terceira classe, uma imagem classificada errada, como *oidio*, e na quinta classe, ocorrendo também um erro, sendo classificada como *bacterial blight*, totalizando 5 erros na somatória das classes.

Figura 25 – Gráfico de *loss* da etapa 12.



Fonte: Elaborada pelo autor.

Tabela 26 – Matriz de confusão da etapa 12.

Classes	A	B	C	D	E	F
Bacterial blight = A	7	0	0	2	0	1
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	0	0	4	0	1	0
Mildio = D	1	0	0	8	0	0
Oidio = E	1	0	0	1	5	0
Rust = F	0	0	0	0	0	20

Tabela 27 – Tabela de precisão, recall e f1-score da etapa 12.

Classes	**Precisão**	**Recall**	**F1-score**	**Quantidade**
Bacterial blight = A	0.78	0.70	0.74	10
Healthy = B	1.00	1.00	1.00	20
Mancha Alvo = C	1.00	0.80	0.89	5
Mildio = D	0.73	0.89	0.80	9
Oidio = E	0.83	0.71	0.77	7
Rust = F	0.95	1.00	0.98	20

Na etapa 12, Figura 25, é visto a ocorrência de um pequeno distanciamento entre as duas linhas conforme vão se passando as *epoch*, com um valor de 0,28 para o *loss* de validação.

Na matriz de confusão e tabela de métricas, Tabela 26 e Tabela 27, é observado o acerto de 100% de duas classes, a segunda e sexta, sendo as duas classes com os maiores números de imagens. Na primeira classe, é visto o erro de três imagens, sendo duas imagens classificadas como *míldio* e uma imagem classificada como *rust*. Na terceira classe, uma imagem é classificada errada, como *oidio*. Na quarta classe, uma imagem classificada errada, sendo classificada como *bacterial blight*. Já na quinta classe, duas imagens foram classificadas erradas, uma como *bacterial blight* e a outra como *healthy*, totalizando 7 erros na somatória das classes.

4.3 TREINAMENTO DA REDE DESENVOLVIDA PELO AUTOR

A partir dos treinamentos realizados na rede desenvolvida pelo autor com as variações de métricas apresentadas na Tabela 5, gerou-se as mesmas tabelas, gráficos e matrizes de confusão para melhor visualização dos resultados e comparação entre as etapas anteriores.

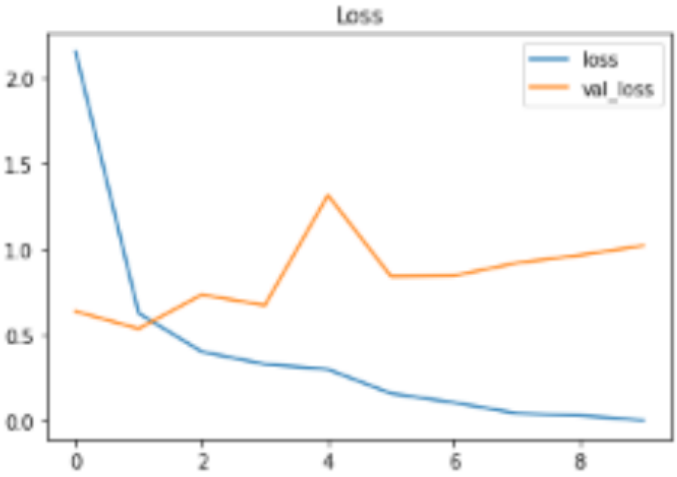
Tabela 28 – Tabela de resultados das métricas utilizadas referente a rede neural desenvolvida pelo autor

Etapa	**13**	14	15	16	**17**
Acurácia validação	**85%**	72,1%	72,1%	80,7%	**83,5%**
Acurácia teste	**81%**	69%	67,6%	77,4%	**81,6%**
Macro avg precisão	**0,76**	0,65	0,64	0,70	**0,75**
Weighted avg precisão	**0,82**	0,73	0,72	0,82	**0,83**
Macro avg recall	**0,75**	0,65	0,65	0,73	**0,77**
Weighted avg recall	**0,82**	0,69	0,68	0,77	**0,82**
Macro avg F1-score	**0,74**	0,63	0,63	0,71	**0,76**
Weighted avg F1-score	**0,81**	0,70	0,68	0,79	**0,82**

Com os resultados obtidos através da Tabela 28, é notado que as etapas 13 e 17 obtiveram um resultado superior quando comparado com as demais etapas desta tabela.

Com um destaque positivo nas etapas 13 e 17, mais métricas serão apresentadas para uma melhor análise dos resultados.

Figura 26 – Gráfico de *loss* da etapa 13.



Fonte: Elaborada pelo autor.

Tabela 29 – Matriz de confusão da etapa 13.

Classes	**A**	**B**	**C**	**D**	**E**	**F**
Bacterial blight = A	5	0	2	1	0	2
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	0	0	3	1	1	0
Mildio = D	0	0	0	7	2	0
Oidio = E	0	1	0	1	5	0
Rust = F	1	0	0	1	0	18

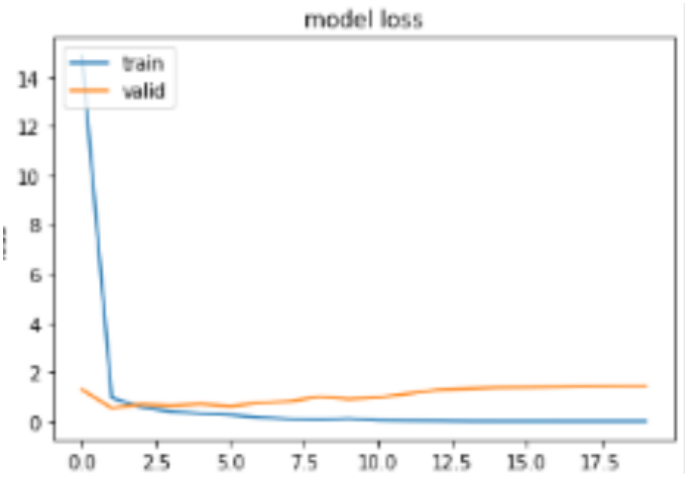
Tabela 30 – Tabela de precisão, recall e f1-score da etapa 13.

Classes	**Precisão**	**Recall**	**F1-score**	**Quantidade**
Bacterial blight = A	0.83	0.50	0.62	10
Healthy = B	0.95	1.00	0.98	20
Mancha Alvo = C	0.60	0.60	0.60	5
Mildio = D	0.64	0.78	0.70	9
Oidio = E	0.62	0.71	0.67	7
Rust = F	0.90	0.90	0.90	20

Na etapa 13, Figura 26, é observado um grande distanciamento entre as duas linhas, com um valor de 1,02 no *loss* de validação, podendo representar um *overfitting* na rede neural.

Na matriz de confusão e tabela de métricas, Tabela 29 e Tabela 30, é visto um acerto de 100% na segunda classe, já na primeira classe, cinco imagens são classificadas erradas, em três classes diferentes. Na terceira classe, duas imagens são classificadas erradas, uma em cada classe distintas. Na quarta classe, duas imagens são classificadas erradas, na mesma classe. A quinta classe, também apresenta duas imagens classificadas erradas, uma em cada classe. E na sexta classe, duas imagens também são classificadas erradas, uma em cada classe, resultando em 13 imagens classificadas erradas.

Figura 27 – Gráfico de *loss* da etapa 17.



Fonte: Elaborada pelo autor.

Tabela 31 – Matriz de confusão da etapa 17.

Classes	**A**	**B**	**C**	**D**	**E**	**F**
Bacterial blight = A	7	0	2	0	0	1
Healthy = B	0	20	0	0	0	0
Mancha Alvo = C	1	0	3	0	0	1
Mildio = D	0	0	0	7	2	0
Oidio = E	0	0	0	2	5	0
Rust = F	2	0	2	0	0	16

Tabela 32 – Tabela de precisão, recall e f1-score da etapa 17.

Classes	**Precisão**	**Recall**	**F1-score**	**Quantidade**
Bacterial blight = A	0.70	0.70	0.70	10
Healthy = B	1.00	1.00	1.00	20
Mancha Alvo = C	0.43	0.60	0.50	5
Mildio = D	0.78	0.78	0.78	9
Oidio = E	0.71	0.71	0.71	7
Rust = F	0.89	0.80	0.84	20

Na etapa 17, Figura 27, é visto um distanciamento com o decorrer das *epochs* entre as linhas e um valor de validação de *loss* igual a 1,43.

Na matriz de confusão e tabela de métricas, Tabela 31 e Tabela 32, é visto um acerto de 100% também apenas na segunda classe. Na primeira classe, três imagens sendo classificas erradas em duas classes distintas. Na terceira classe, duas imagens classificadas erradas em duas classes. Na quarta classe, duas imagens classificadas erradas em uma classe, o mesmo ocorre na quinta classe. Na sexta classe, quatro imagens classificadas erradas em duas classes, resultando em 13 imagens com erros de classificação.

4.4 COMPARAÇÕES ENTRE AS CNNS E ALGORITMOS TRADICIONAIS

Na Tabela33 é apresentado os melhores resultados de cada técnica utilizada nas etapas anteriores.

Tabela 33 – Tabela de resultados das métricas utilizadas nas etapas que tiveram os melhores desempenhos

Método	**M.N.V2**	**M.N.V2**	**M.N.V2**	CNN Des.	Algor. Trad.
Etapa	**2**	**7**	**11**	17	IBk
Acurácia validação	**92,1%**	**93,5%**	**91,4%**	83,5%	-
Acurácia teste	**92,9%**	**92,9%**	**92,9%**	81,6%	74,6%
Macro avg precisão	**0,93**	**0,92**	**0,92**	0,75	0,72
Weighted avg precisão	**0,94**	**0,93**	**0,93**	0,83	0,75
Macro avg recall	**0,89**	**0,89**	**0,89**	0,77	0,71
Weighted avg recall	**0,93**	**0,93**	**0,93**	0,82	0,74
Macro avg F1-score	**0,90**	**0,90**	**0,90**	0,76	0,70
Weighted avg F1-score	**0,93**	**0,93**	**0,93**	0,82	0,73

Como é visto, os melhores resultados foram obtidos pela rede neural MobileNetV2 (M.N.V2), nas etapas 2, 7 e 11, atingindo um desempenho de 11,3% superior do que a rede neural desenvolvida pelo autor e 18,3% melhor do que o algoritmo tradicional IBk.

5 CONCLUSÕES

Neste trabalho foi apresentado o contexto da aplicação, o processo de coleta de dados, pré-processamento de imagens e visão computacional para. A detecção de doenças em folhas de soja é realizada utilizando Rede Neural de Convolução. A base de dados utilizada continha imagens de 6 tipos de classes diferentes, sendo cinco classes de folhas de soja com doenças e uma classe com as folhas saudáveis, totalizando 815 imagens nas quais foram divididas em três conjuntos de dados, sendo imagens de treinamento, validação e teste.

Foram utilizadas duas CNNs distintas nos experimentos, uma chamada de MobileNetV2 e outra desenvolvida pelo autor, alterando o número de *batch-size* e *epoch* durante os treinamentos. Os melhores resultados foram obtidos utilizando a CNN MobileNetV2, alcançando uma acurácia de 92,9%, precisão entre 92% e 93%, sensibilidade de 89% e f1-score de 90%, resultados positivos e satisfatórios, pois mostram que a utilização de redes neurais convolucionais em imagens para detecção de doenças em folhas, apresentam resultados convenientes e podem gerar objetivos de estudos mais profundos para avançar o problema.

O método proposto no trabalho tem por objetivo encontrar uma CNN capaz de resolver com efetividade o problema de detecção de doenças em folhas se soja, com os resultados apresentados pela CNN MobileNetV2, conclui-se que os objetivos propostos na monografia foram alcançados.

Vale ressaltar que os resultados obtidos pela rede desenvolvida pelo autor, por ser uma rede menos profunda e menos complexa, alcançaram um resultado inferior quando comparado a CNN MobileNetV2, porém não foram resultados ruins, obtendo uma acurácia de 81,6%, precisão de 75%, sensibilidade de 77% e f1-score igual a 76%. Os resultados obtidos através dos algoritmos tradicionais alcançaram resultados inferiores quando comparados com as CNNs, ficando 18,3% abaixo no melhor resultado atingido (IBk).

Portando, conclui-se que a CNN MobileNetV2 é a melhor escolha para o tipo do problema apresentado no decorrer do trabalho, por apresentar bons resultados na identificação de doenças em folhas de soja e por ser uma rede com uma quantidade baixa de parâmetros, sendo uma rede neural considerada leve, com capacidade de ser implementada em aplicativos para serem utilizadas em *smartphones*.

5.1 TRABALHOS FUTUROS

Como é observado nas matrizes de confusão, as classes que obtiveram taxas de assertividade maiores, foram as que continham um maior número de imagens, então, para que a rede consiga melhores resultados, é viável o aumento do banco de imagem, principalmente nas classes que tem menos imagens.

Como a ideia final do projeto é que o trabalho seja implementado dentro da

cooperativa, para utilização dos cooperados, o aplicativo será desenvolvido junto ao time de tecnologia e inovação da cooperativa.

REFERÊNCIAS

ALBAWI, Saad; MOHAMMED, Tareq Abed; AL-ZAWI, Saad. Understanding of a convolutional neural network. *In*: IEEE. 2017 international conference on engineering and technology (ICET). [*S.l.*: *s.n.*], 2017. P. 1–6.

APRENDIZADO profundo baseado em nasnet para reconhecimento de doenças de plantas usando imagens de folhas. [*S.l.*]: IEEE, 2019. P. 1–5.

ARAÚJO, Flávio HD; CARNEIRO, AC; SILVA, Romuere RV; MEDEIROS, Fátima NS; USHIZIMA, Daniela M. Redes neurais convolucionais com tensorflow: Teoria e prática. **SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauı. Livro Anais-Artigos e Minicursos**, Sociedade Brasileira de Computação, v. 1, p. 382–406, 2017.

BARBEDO, Jayme Garcia Arnal; KOENIGKAN, Luciano Vieira; SANTOS, Thiago Teixeira. Identifying multiple plant diseases using digital image processing. **Biosystems engineering**, Elsevier, v. 147, p. 104–116, 2016.

BASU, PK; BUTLER, G. An evaluation of soybean bacterial blight assessment methods. **Canadian Journal of Plant Pathology**, Taylor & Francis, v. 8, n. 4, p. 459–463, 1986.

BHARGAVA, Neeraj; SHARMA, Girja; BHARGAVA, Ritu; MATHURIA, Manish. Decision tree analysis on j48 algorithm for data mining. **Proceedings of international journal of advanced research in computer science and software engineering**, v. 3, n. 6, 2013.

BUDUMA, Nithin; BUDUMA, Nikhil; PAPA, Joe. **Fundamentals of deep learning**. [*S.l.*]: "O'Reilly Media, Inc.", 2022.

CASANOVA, Marco Antonio; CÂMARA, Gilberto; DAVIS, Clodoveu; VINHAS, Lúbia; QUEIROZ, GR de. Banco de dados geográficos. MundoGEO Curitiba, 2005.

CHICCO, Davide; JURMAN, Giuseppe. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. **BMC genomics**, Springer, v. 21, p. 1–13, 2020.

CHOI, Seungwoo; SEO, Seokjun; SHIN, Beomjun; BYUN, Hyeongmin; KERSNER, Martin; KIM, Beomsu; KIM, Dongyoung; HA, Sungjoo. Temporal

convolution for real-time keyword spotting on mobile devices. **arXiv preprint arXiv:1904.03814**, 2019.

CHOLLET, Francois. **Deep learning with Python**. [*S.l.*]: Simon e Schuster, 2021.

DATA SCIENCE ACADEMY. **Data Science Academy. Deep Learning Book, 2022**. [*S.l.*: *s.n.*], 2022. Disponível em: https://www.deeplearningbook.com.br/. Acesso em: 10 Dezembro. 2022.

DATE, Christopher J. **Introdução a sistemas de bancos de dados**. [*S.l.*]: Elsevier Brasil, 2004.

DEVELOPERS, TensorFlow. TensorFlow. **Zenodo**, 2022.

DMATH. **How CNNs Work**. [*S.l.*: *s.n.*], 2023. Disponível em: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html. Acesso em: 09 Abril. 2023.

DOUZE, Matthijs; JÉGOU, Hervé; SANDHAWALIA, Harsimrat; AMSALEG, Laurent; SCHMID, Cordelia. Evaluation of gist descriptors for web-scale image search. *In*: PROCEEDINGS of the ACM International Conference on Image and Video Retrieval. [*S.l.*: *s.n.*], 2009. P. 1–8.

DUNN, Dennis; HIGGINS, William E. Optimal Gabor filters for texture segmentation. **IEEE Transactions on image processing**, IEEE, v. 4, n. 7, p. 947–964, 1995.

GHOTRA, Manpreet Singh; DUA, Rajdeep. **Neural Network Programming with TensorFlow: Unleash the power of TensorFlow to train efficient neural networks**. [*S.l.*]: Packt Publishing Ltd, 2017.

GU, Jiuxiang *et al.* Recent advances in convolutional neural networks. **Pattern recognition**, Elsevier, v. 77, p. 354–377, 2018.

GULLI, Antonio; KAPOOR, Amita; PAL, Sujit. **Deep learning with TensorFlow 2 and Keras: regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API**. [*S.l.*]: Packt Publishing Ltd, 2019.

GUO, Kaiyuan; SUI, Lingzhi; QIU, Jiantao; YU, Jincheng; WANG, Junbin; YAO, Song; HAN, Song; WANG, Yu; YANG, Huazhong. Angel-eye: A complete design flow for

mapping CNN onto embedded FPGA. **IEEE transactions on computer-aided design of integrated circuits and systems**, IEEE, v. 37, n. 1, p. 35–47, 2017.

GUTUB, Adnan Abdul-Aziz *et al.* Pixel indicator technique for RGB image steganography. **Journal of emerging technologies in web intelligence**, Academy Publisher, PO Box 40 Oulu 90571 Finland, v. 2, n. 1, p. 56–64, 2010.

HARVARD. **dataverse**. [*S.l.*: *s.n.*], 2022. Disponível em: https://dataverse.harvard.edu/. Acesso em: 20 Julho. 2022.

HASSAN, Sk Mahmudul; MAJI, Arnab Kumar; JASIŃSKI, Michał; LEONOWICZ, Zbigniew; JASIŃSKA, Elżbieta. Identification of plant-leaf diseases using CNN and transfer-learning approach. **Electronics**, MDPI, v. 10, n. 12, p. 1388, 2021.

HENNING, Ademir Assis *et al.* Manual de identificação de doenças de soja. Londrina: Embrapa Soja, 2014., 2014.

HOWARD, Andrew G; ZHU, Menglong; CHEN, Bo; KALENICHENKO, Dmitry; WANG, Weijun; WEYAND, Tobias; ANDREETTO, Marco; ADAM, Hartwig. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

IPMIMAGE. **ipmimage**. [*S.l.*: *s.n.*], 2022. Disponível em: https://www.ipmimages.org/. Acesso em: 20 Julho. 2022.

JADHAV, Sachin B; UDUPI, Vishwanath R; PATIL, Sanjay B. Identification of plant diseases using convolutional neural networks. **International Journal of Information Technology**, Springer, v. 13, n. 6, p. 2461–2470, 2021.

JOY. **Confusion Matrix**. [*S.l.*: *s.n.*], 2020. Disponível em: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826. Acesso em: 11 de Junho de. 2023.

KETKAR, Nikhil; KETKAR, Nikhil. Introduction to keras. **Deep learning with python: a hands-on introduction**, Springer, p. 97–111, 2017.

KOVÁCS, Zsolt László. **Redes neurais artificiais**. [*S.l.*]: Editora Livraria da Fisica, 2002.

KOWATA, Ligia Sayko; MAY-DE-MIO, Louise Larissa; DALLA PRIA, Maristella; SANTOS, Hellen Aparecida Arantes do. Escala diagramática para avaliar severidade de mıldio na soja, 2008.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **nature**, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LEE, Kai-Fu. **Inteligência artificial**. [*S.l.*]: Globo Livros, 2019.

LITVAK, Marina. Deep dive into authorship verification of email messages with convolutional neural network. *In*: SPRINGER. INFORMATION Management and Big Data: 5th International Conference, SIMBig 2018, Lima, Peru, September 3–5, 2018, Proceedings 5. [*S.l.*: *s.n.*], 2019. P. 129–136.

LUGER, George F. **Inteligência Artificial-: Estruturas e estratégias para a solução de problemas complexos**. [*S.l.*]: Bookman, 2004.

MANASWI, Navin Kumar; MANASWI, Navin Kumar. Understanding and working with Keras. **Deep learning with applications using Python: Chatbots and face, object, and speech recognition with TensorFlow and Keras**, Springer, p. 31–43, 2018.

MENDELEY. **mendeley**. [*S.l.*: *s.n.*], 2022. Disponível em: https://data.mendeley.com/datasets/tywbtsjrjv/1. Acesso em: 20 Julho. 2022.

MURILLO-ESCOBAR, Miguel Angel; CRUZ-HERNÁNDEZ, César; ABUNDIZ-PÉREZ, Fausto; LÓPEZ-GUTIÉRREZ, Rosa Martha; DEL CAMPO, OR Acosta. A RGB image encryption algorithm based on total plain image characteristics and chaos. **Signal Processing**, Elsevier, v. 109, p. 119–131, 2015.

O'SHEA, Keiron; NASH, Ryan. An introduction to convolutional neural networks. **arXiv preprint arXiv:1511.08458**, 2015.

OLIVA, Aude; TORRALBA, Antonio. Modeling the shape of the scene: A holistic representation of the spatial envelope. **International journal of computer vision**, Springer, v. 42, p. 145–175, 2001.

OUJAOURA, M; MINAOUI, B; FAKIR, M; EL AYACHI, R; BENCHAREF, O. Recognition of isolated printed tifinagh characters. **International Journal of Computer Applications**, Foundation of Computer Science, v. 85, n. 1, 2014.

RAWAT, Waseem; WANG, Zenghui. Deep convolutional neural networks for image classification: A comprehensive review. **Neural computation**, MIT Press, v. 29, n. 9, p. 2352–2449, 2017.

RICH, E.; KNIGHT, K.; CALERO, P.A.G. **Inteligencia artificial**. [*S.l.*]: McGraw-Hill, 1994. P. 7–250. ISBN 9788448118587.

RISH, Irina *et al.* An empirical study of the naive Bayes classifier. *In*: 22. IJCAI 2001 workshop on empirical methods in artificial intelligence. [*S.l.*: *s.n.*], 2001. v. 3, p. 41–46.

SACOMANO, José Benedito; GONÇALVES, Rodrigo Franco; BONILLA, Sılvia Helena; SILVA, Márcia Terra da; SÁTYRO, Walter Cardoso. **Indústria 4.0**. [*S.l.*]: Editora Blucher, 2018.

SANDLER, Mark; HOWARD, Andrew; ZHU, Menglong; ZHMOGINOV, Andrey; CHEN, Liang-Chieh. Mobilenetv2: Inverted residuals and linear bottlenecks. *In*: PROCEEDINGS of the IEEE conference on computer vision and pattern recognition. [*S.l.*: *s.n.*], 2018. P. 4510–4520.

SOARES, Rafael M; GODOY, Cláudia V; OLIVEIRA, Maria Cristina N de. Escala diagramática para avaliação da severidade da mancha alvo da soja. **Tropical Plant Pathology**, SciELO Brasil, v. 34, p. 333–338, 2009.

SOUSA, A de L; SALAME, Marcos Filipe Alves. Uma abordagem comparativa de algoritmos de aprendizado supervisionado para classificação dos cultivares da planta Paullinia cupana. In: ENCONTRO REGIONAL DE COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO, 6., 2017 . . ., 2017.

TEIXEIRA, João de Fernandes; GONZALES, Maria Eunice Quilici. Inteligência artificial e teoria de resolução de problemas. **Trans/Form/Ação**, SciELO Brasil, v. 6, p. 45–52, 1983.

TENSORFLOW. **Tensorflow,2018**. [*S.l.*: *s.n.*], 2018. Disponível em: https://www.tensorflow.org/?hl=pt-br/. Acesso em: 16 de Janeiro.2023.

VARGAS, Ana Caroline Gomes; PAES, Aline; VASCONCELOS, Cristina Nader. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. *In*: SN, 4. PROCEEDINGS of the xxix conference on graphics, patterns and images. [*S.l.*: *s.n.*], 2016. v. 1.

VAS. **machine learning**. [*S.l.*: *s.n.*], 2022. Disponível em: https://vas3k.com/blog/machine_learning/?fbclid=IwAR1lL7H43loT7wShhonf-1OkdHNvoPOV0jsm-r4Mkair6DRAqBBCFRHE76g. Acesso em: 21 Setembro. 2022.

VIJAYARANI, S; MUTHULAKSHMI, M. Comparative analysis of bayes and lazy classification algorithms. **International Journal of Advanced Research in Computer and Communication Engineering**, v. 2, n. 8, p. 3118–3124, 2013.

VOULODIMOS, Athanasios; DOULAMIS, Nikolaos; DOULAMIS, Anastasios; PROTOPAPADAKIS, Eftychios *et al.* Deep learning for computer vision: A brief review. **Computational intelligence and neuroscience**, Hindawi, v. 2018, 2018.

YALÇIN, Orhan Gazi. **Applied Neural Networks with TensorFlow 2: API Oriented Deep Learning with Python**. [*S.l.*]: Apress, 2021.

YORINORI, José Tadashi. Oıdio da soja. Londrina: EMBRAPA-CNPSo, 1997., 1997.

YORINORI, José Tadashi; NUNES JÚNIOR, José; LAZZAROTTO, Joelsio José. Ferrugem"asiática"da soja no Brasil: evolução, importância econômica e controle. Londrina: Embrapa Soja, 2004., 2004.

ZHANG, Aston; LIPTON, Zachary C; LI, Mu; SMOLA, Alexander J. Dive into deep learning. **arXiv preprint arXiv:2106.11342**, p. 38, 2021.