Uso de machine learning para detecção de pragas e doenças em plantas por meio de análise de imagens

David Francisco[1]*; Felipe Bastos dos Reis[2]

[1] Representante Comercial. Rua Antônio Cyrillo Pereira 208 – Jardim Primavera; 14708-326 Bebedouro, São Paulo, Brasil
[2] Faculdade de Economia, Administração, Contabilidade e Atuária (FEA - USP). Professor. Av. Prof. Luciano Gualberto, 908 – Butantã; 05508-010 São Paulo, São Paulo, Brasil
*autor correspondente: david.app@gmail.com

Uso de machine learning para detecção de pragas e doenças em plantas por meio de análise de imagens

Resumo

A detecção e classificação de imagens na agricultura é uma técnica usada em várias aplicações, como identificação de área sem plantio, área com infestação de pragas e doenças, análise de solo e outras. O objetivo do trabalho é entender como a IA e o machine learning podem ser usados para detectar pragas e doenças agrícolas. O algoritmo foi desenvolvido na linguagem Python utilizando Anaconda, Jupyter e Spyder. O modelo gerado é capaz de classificar as imagens com uma acurácia de 96% e especificidade de 97%. Esse trabalho oferece uma contribuição teórica ao explorar a eficácia das redes neurais convolucionais no reconhecimento de imagens, também oferece uma contribuição prática, pois com o modelo gerado é possível auxiliar na tomada de decisão indicando quais plantas estão doentes ou com infestação de pragas, para que possa ser tratado antes de espalhar e atingir toda a lavoura.
Palavras-chave: aprendizado profundo; classificação de imagens; fitossanidade; Python; redes neurais convolucionais.

Use of machine learning to detect pests and diseases in plants through image analysis

Abstract

Image detection and classification in agriculture is a technique used in various applications, such as identifying unplanted areas, areas with pest and disease infestation, soil analysis and many others. The objective of the work is to train a convolutional neural network, which classifies images of plants as sick or healthy. The algorithm was developed in the Python language using Anaconda, Jupiter and Spyder. The generated model is capable of classifying images with an accuracy of 96% and specificity of 97%. This work offers a theoretical contribution by exploring the effectiveness of convolutional neural networks in recognizing images, it also offers a practical contribution, as with the generated model it is possible to assist in decision making by indicating which plants are sick or have pest infestation, so that it can be treated before spreading and affecting the entire crop.
Keywords: deep learning; image classification; plant health; python; convolutional neural networks.

Introdução

Fitossanidade é um termo utilizado para o estudo da saúde das espécies vegetais, estima-se que mais de 30% da produção perde-se por problemas na lavoura, e a fitossanidade agrega informações técnicas e de várias ciências para combater pragas e doenças, os problemas se originam da interação entre ambiente, patógeno e a planta, as doenças são ocasionadas por mau funcionamento de células e tecidos da planta devido a irritação causada pelos patógenos ou fator ambiental, (Fonseca e Araújo, 2014; Stein e Coscolin, 2020). As doenças podem ser não infecciosas, causadas por fatores climáticos ou nutricionais, ou por insetos e ácaros que ao se alimentarem da planta causam manchas cloróticas ou necróticas, e infecciosas causadas por fungos, bactérias, vírus, nematoides, protozoários e plantas

parasitas superiores, que debilitam as plantas por absorção contínua dos nutrientes, nos dois casos a doença pode ser identificada pela alteração da cor nas folhas, (Fonseca e Araújo, 2014; Stein e Coscolin, 2020). O Brasil por ser um grande produtor agrícola, infelizmente é um grande consumidor de agrotóxicos, o desafio é minimizar o uso desses produtos mantendo a alta produção, é fundamental buscar conhecimentos técnicos e conscientização para a aplicação desses produtos, uma vez que mal administrados podem causar riscos à saúde humana e ao ambiente, (Fonseca e Araújo, 2014; Stein e Coscolin, 2020). O monitoramento de agentes nocivos é feito por amostragem, não é possível expressar o limite de dano causado apenas pela amostragem, antes que ocorra a injúria, uma forma de contornar essas dificuldades é por meio de práticas de coletas de dados na lavoura para identificar possíveis infecção antes que ocorra a injúria (Fonseca e Araújo, 2014).

O uso de tecnologia computacional na agricultura tem impactado de forma positiva a produção e previne a propagação de pragas e doenças, a agricultura de precisão conta com tecnologias de ponta para o aumento da produção e a minimização de gastos com insumos agrícolas. O uso de veículo aéreo não tripulado [VANT], um equipamento de suma importância no processo inicial da análise, ele é capaz de capturar uma grande quantidade de imagens e enviar por meio de telemetria para uma base de dados na nuvem e com esses dados é possível examinar em laboratório de dados e classificar as imagens, dessa forma em um curto período é possível investigar e tratar alguma anomalia na lavoura. O VANT pode percorrer grandes distâncias em curto período e alcançar locais de difícil acesso, em uma pesquisa 57,1% dos agricultores que utilizam o VANT tem como objetivo o monitoramento de pragas e doenças, Artioli e Beloni (2016). O uso do VANT na lavoura é considerado um importante instrumento para identificar manchas de doenças folhares e pragas, tornando possível os agricultores e especialistas tomarem melhores decisões de manejo, as tecnologias de precisão tem feito com que os produtores e técnicos tratem cada região da lavoura de forma diferenciada realocando os insumos de forma equilibrada, Tetila (2019). Após a coleta dos dados com o uso de algoritmos de machine learning pode-se identificar padrões e assim classificar os dados conforme a necessidade do negócio.

A inteligência artificial [IA] é uma área da ciência que atua com comportamento inteligente nos processos de automação, em diversas áreas e com diversos propósitos, desde o diagnóstico de doenças até a condução de veículos sem a intervenção humana. Na agricultura é possível combinar técnicas de detecção e aplicação de defensivos em loco, de forma automatizada com algoritmos de machine learning. Os algoritmos de machine learning permitem por meio de cálculos matemáticos que o computador aprenda de forma supervisionada ou não supervisionada. O deep learning é um campo do machine learning que permite que o computador imite as funções cerebrais como sinapses e neurônios

interconectados, onde existe uma camada de entrada e uma camada de saída e entre essas camadas há camadas ocultas que efetuam os cálculos (Elias e Manhiça, 2019; Silva et al., 2018).

O trabalho será desenvolvido com a linguagem de programação Python, a escolha da linguagem se deu pelo poder de processamento e as diversas bibliotecas disponibilizadas, sua implementação original começou em 1989 com o matemático e programador holandês Guido Van Rossum como ferramenta alternativa aos scripts bash e programas em C para administração de sistemas, (Behrman, 2023; Netto e Maciel, 2021; Silva et al., 2018). Devido a sua robustez a linguagem é amplamente utilizada em empresas globais como Google, 'National Aeronautics and Space Administration' [NASA], YouTube, Disney, Empresa Brasileira de Telecomunicações [Embratel], 'International Business Machines' [IBM] e Rede Globo e o mercado para a linguagem está em pleno crescimento, Silva et al., (2018). As estruturas de dados do Python são de fáceis manipulação, a curva de aprendizado é curta e tem vários frameworks de código aberto para utilizar, como PyCharm, Anaconda, Google Colab etc.

O trabalho tem como objetivo entender como a IA e o machine learning podem ser usados para detectar pragas e doenças agrícolas por análise de imagens, a partir desse objetivo o trabalho espera construir um algoritmo que alcance 95% de acurácia na classificação das plantas doentes e saudáveis.

Material e Métodos

Esse estudo analisou imagens extraídas do repositório do Google, as palavras-chaves usadas foram: manchas folhares, infestação de lagartas, doenças do milho, amendoim, soja, laranja para as classificações de doentes, e as palavras-chaves: culturas saudáveis de amendoim, laranja, soja, milho e outras para a classificação de saudáveis, e imagens coletadas na estação experimental da cidade de Bebedouro, São Paulo.

O método estatístico usado foi de análise preditiva, a coleta de dados será feita por meio de downloads de arquivos 'Joint Photographic Experts Groups' [JPEG], e no desenvolvimento do trabalho será usado a linguagem Python com o Framework Anaconda com as aplicações Jupyter e Spyder. Após a coleta será aplicado a técnica de PCA para reduzir e dimensionar a imagem, e transformá-la em um vetor, a base de dados será separada em 75% para treinamento e 25% teste, (Grus, 2021; Mueller, 2020; Netto e Maciel, 2021), que é o percentual indicado na maioria das literaturas estudadas. Antes de codificar o modelo de treinamento os dados serão normalizados para que o modelo possa receber as imagens em formato de vetor unidimensional, para ser conectado na camada densa do modelo, em

seguida é compilado e treinado. Para a elaboração do trabalho foi instalado previamente o Python 3 e a plataforma de desenvolver e implantar soluções em Python Anaconda, no desenvolvimento será usado a plataforma web Jupyter notebook e a 'Integrated Development Environment ' [IDE] Spyder, após esse processo é preciso instalar as bibliotecas necessárias.

Criação do algoritmo

O termo inteligência artificial [IA] representa um software que faz computadores realizarem tarefas que eram exclusivamente dos humanos, permite que o computador treine e aprenda por meio da matemática, estatística e computação. O primeiro projeto surgiu em 1950 no Dartmouth College, em Hanover, New Hampshire, Estados Unidos, mas por falta de poder computacional da época não foi possível evoluir como atualmente, dentro do campo de IA está o campo do machine learning (aprendizado de máquina) e do campo do machine learning está o deep lerning (aprendizado profundo), onde os modelos são redes neurais artificiais [RNA] baseados no cérebro humano, (Netto e Maciel, 2021; Silva et al., 2018). O machine learning é uma aplicação de IA que aprende e aprimora automaticamente sua performance ao analisar grandes conjuntos de dados. Os algoritmos básicos permanecem constantes, mas os pesos e vieses internos são ajustados para otimizar as respostas sem programação explícita para cada melhoria, Mueller (2020). Machine learning compreende à criação e uso de modelos que aprendem com dados, é usado para identificar se um e-mail é spam ou não, se há fraude em uma transação de cartão de crédito, qual time ganhará o campeonato e muitas outras aplicações, Grus (2021), no trabalho será usado o modelo supervisionado, essa técnica permite aprender com os dados de treinamento e testar com os dados de testes, geralmente são separados 75% para treinamento e 25% para testes, nesse processo será possível obter por meio dos dados as respostas sobre quais plantas estão doentes e quais estão saudáveis (Grus, 2021; Mueller, 2020; Netto e Maciel, 2021).

O machine learning é uma subárea da IA que é a principal técnica por trás da automação, seu grande objetivo é desenvolver sistemas que aprendam com experiências passadas, os modelos são: aprendizado não supervisionado, aprendizado supervisionado e aprendizado por reforço, para a implementação do machine learning é necessário usar diversas técnicas como: estatística, mineração de dados, processos de clustering, entre outros, (Netto e Maciel, 2021; Silva et al., 2018). No trabalho será usado a aprendizagem supervisionada, os dados e os hiper parâmetros serão ajustados até obter um modelo com boa performance. O deep learning é uma técnica de machine learning que utiliza grandes quantidades de dados não estruturados, permitindo a representação hierárquica das suas camadas, seus algoritmos permitem criar aplicações que envolvam processamento de

imagens, reconhecimento de voz, análise de comportamento e outras aplicações que exijam redes neurais artificiais, (Silva et al., 2018). A RNA é um modelo preditivo baseado no funcionamento do cérebro, que tem uma série de neurônios conectados, cada neurônio recebe o dado, faz o cálculo e retorna a saída para o neurônio ligado a ele, e é feito assim até o último neurônio que é a saída da informação. Como os seres humanos a rede tem uma fase de aprendizado, analisando características e detectando padrões, (Grus, 2021; Sicsú et al., 2023).

Foram criadas diversas técnicas de RNA para lidar com imagens, porém para o reconhecimento de imagens as RNA se mostraram lentas, o modelo além de consumir muito poder computacional, também se adapta ao modelo de treinamento causando o overfitting, por esse motivo foi criado a Convolutional Neural Network [CNN], uma rede neural bastante efetiva no reconhecimento de imagem, (Silva et al., 2018). No trabalho será utilizado a CNN, classificando as plantas em doentes ou saudáveis, devido ao melhor desempenho se comparado com a RNA . A CNN surgiu no final dos anos de 1980 como uma solução para problemas de reconhecimento de caracteres, desenvolvido por Yann LeCun, em uma rede chamada LeNet5, onde o objetivo era identificar os caracteres escritos em cheques, hoje essas redes viabilizam novas aplicações em diversas áreas como: robótica, medicina, agricultura, segurança etc. Mueller (2020). Em trabalho semelhante em que foi aplicado para identificação de pragas e doenças em soja, os resultados mostraram que as arquiteturas de CNN obtêm maiores taxas de classificação em comparação a outras abordagens, alcançando uma acurácia de até 93,82%, por outro lado, o sistema de visão computacional não identificou com eficiência os insetos nas imagens, Tetila (2019). Em outros dois estudos obtiveram resultados próximos aos 93,82% citados anteriormente, um deles obteve 98,22% de acurácia classificando frutas ou alimentos em geral, nesse trabalho usou 82072 imagens para treinamento, e o segundo trabalho foi classificar ervas daninhas em uma lavoura de soja, esse alcançou pouco mais de 99% de acurácia Ribeiro (2020). As classificações de imagens pela CNN são feitas de forma automática, que permite avaliar muitas imagens em um curto período, as CNN têm sido muito bem-sucedidas nas análises de imagens médicas, sensoriamento remoto, na agricultura superam os meios tradicionais (Gonçalves, 2020).

As CNN não se restringem a aplicação visual, tem bom desempenho em reconhecimento de voz e processamento de linguagem natural, também teve participação importante no desenvolvimento de carros autônomos, sua arquitetura foi baseada no estudo sobre o córtex visual dos cientistas David Hubel e Torsten Wiesel em 1959, trabalho que rendeu o prêmio Nobel de medicina em 1981. As CNN possuem camadas convolucionais, que por meio de deslizamento de pixel cria um mapa de características, cada pixel é um neurônio e todos os neurônios de um mapa usam os mesmos pesos, com isso ao treinar é aplicado

vários filtros de características e assim pode ser detectado qualquer característica na camada de entrada, e camadas de pooling tem a função de reduzir a imagem de entrada para diminuir o custo computacional, o uso de memória, e o número de parâmetros para limitar o risco de overfitting, Ferreira (2021). Na Figura 1 mostra a atuação da camada de pooling.

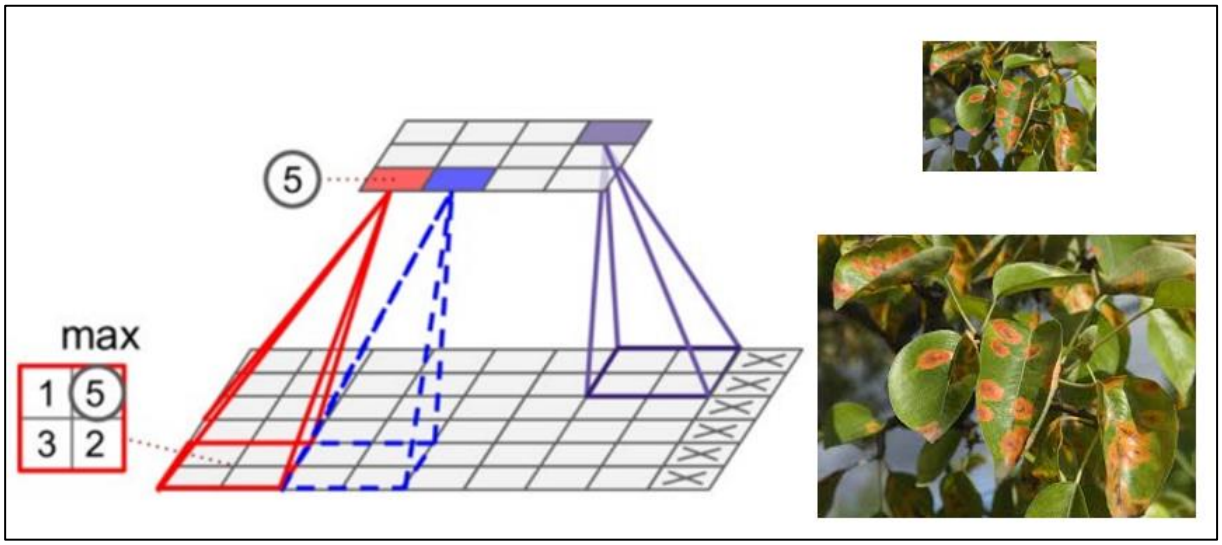

Figura 1 – Camada de pooling
Fonte: Dados originais da pesquisa

Importação de módulos e bibliotecas

O primeiro passo foi importar os módulos e bibliotecas necessários para o desenvolvimento do trabalho, para importar uma biblioteca ela precisa estar previamente instalada, caso contrário será emitido uma mensagem de erro, para instalar é simples, é preciso abrir o Anaconda prompt e digitar pip install e o nome da biblioteca desejada. As bibliotecas importadas foram: a numpy, utilizada para computação numérica, fornece funções matemáticas eficiente em grandes conjuntos de dados e suporte para arrays multidimensionais. A matplotlib.pyplot que é necessária para criação de gráficos e figuras, e CV2 que fornece ferramentas de visão computacional para ler, gravar e manipular vídeos, imagens, detecção de objetos e reconhecimento facial (Behrman, 2023; Netto e Maciel, 2021).

Da biblioteca tensorflow.keras foram importados alguns módulos que serão descritos abaixo:

Conv2D: realiza convoluções bidimensionais nas imagens, extrai características importantes de uma imagem, seus parâmetros são filters, kernel_size e activation.

MaxPooling2D: realiza operação de max pooling em dados bidimensionais, reduz a dimensão da camada mantendo as características mais importantes, possui um único parâmetro, o pool_size.

Flatten: seu uso é para transformar uma camada de entrada em um vetor unidimensional, não possui parâmetros, seu uso é após as camadas de convoluções para conectar as camadas densas do modelo.

Dense: é uma camada onde todos os neurônios são conectados da entrada até a saída, usado para aprendizado e classificação, possui os parâmetros units e activation.

Dropout: usado para prevenir o overfitting, desativa um percentual especificado de neurônios durante o treinamento, seu parâmetro é o rate, taxa de desativação de neurônios.

ImageDataGenerator: usada para aumentar o lote de imagens usada, redimensiona, altera o zoom, rotaciona e modifica, para que o modelo possa identificar as imagens em perspectivas diferentes da original, dentro de seus vários parâmetros os mais utilizados são rescale, rotation_range, horizontal_flip, zoom_range etc.

BatchNormalization: usada para ativação em lotes, normaliza a ativação de cada neurônio na camada, estabilizando e acelerando o treinamento.

Roc_curve e auc: usadas para calcular e plotar a área da curva roc os parâmetros da primeira são os dados reais e os dados previstos do modelo, e os parâmetros da auc são fpr e o tpr que são métricas obtidas após rodar o modelo. A Figura 2 mostra a importação das bibliotecas e módulos.

Os módulos podem sofrer alterações durante possível mudança de versão, pois por se tratar de codificações de terceiros disponibilizado de forma gratuita, o usuário dos módulos não tem controle sobre as mudanças, é importante consultar a documentação da biblioteca caso o framework não reconheça algum comando.

```
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

Figura 2 – Importação das bibliotecas em Python
Fonte: Dados originais da pesquisa

ETAPA DE EXTRAÇÃO, TRANSFORMAÇÃO E CARGA

Boa parte do tempo de trabalho de ciência de dados está na etapa de ‘Extract, Transform and Load’ [ETL], esse processo dura em torno de 50 a 80% do tempo de desenvolvimento do modelo, (Grus, 2021; Sicsú et al., 2023), isso devido a maioria dos dados

em formatos brutos conter erros de digitação, campos faltantes, tipos de dados em desacordo com o seu real formato, e sem essa etapa o resultado pode não ter o sucesso esperado. Para o trabalho de análise de classificação de plantas doentes ou saudáveis os dados utilizados serão não estruturados, imagens em formato JPEG, as imagens serão baixadas do repositório do Google e coletados na estação experimental de Bebedouro, São Paulo. As imagens são de plantas com infestação de pragas e doenças que mudam a coloração original da espécie. Após a coleta os dados serão tratados utilizando Python, Jupyter e Spyder.

O data mining é relevante em vários contextos, existem diversas técnicas que auxiliam profissionais e a procura por novas técnicas é constante, os dados estão disponíveis em vários formatos e é preciso aplicar técnicas diferentes, os dados podem ser minerados por associação, classificação, clustering, árvores de decisão e padrões sequenciais, o método de classificação os dados são classificados de acordo com seus atributos e divididos em classes ou categorias (Morais et al., 2018).

O data mining iniciou com o download de 1050 imagens do repositório do Google, sendo 525 de plantas com características saudáveis e 525 com características de doenças ou pragas, o diretório com as imagens foi organizado da seguinte forma, uma pasta de nome plantas e nessa pasta duas pastas nomeadas como treino e teste, e em cada pasta de treino e teste mais duas pastas de nome saudáveis e doente, essa organização foi para facilitar a manipulação das imagens com a linguagem Python. Nas pastas de doentes e saudáveis da pasta treino foram armazenadas respectivamente 394 imagens de plantas doentes e 394 imagens de plantas saudáveis e nas pastas doentes e saudáveis da pasta teste foram armazenadas respectivamente 131 imagens de plantas doentes e 131 imagens de plantas saudáveis. A Figura 3 mostra exemplos de imagens com pragas ou doenças utilizado como treinamento do modelo.

Figura 3 – Imagens de teste da classe doentes
Fonte: Dados originais da pesquisa

No processo de transformação as imagens obtidas foram redimensionadas em 64 x 64, 4096 pixels , e a forma de encontrar os padrões de identificação foi percorrendo essa matriz com um filtro, também chamado de Kernel três por três, com isso foi possível obter um mapa de características da imagem, em seguida obtendo os padrões foi gerada uma matriz de cinco por cinco ao qual passou por um processo de pooling, foi usado o Max_Pooling2D com o argumento pool_size (2,2), esse parâmetro indica que a matriz cinco por cinco será percorrida por uma matriz dois por dois da esquerda para a direita e de cima para baixo, capturando o maior valor do quadrante, e esses valores formam uma matriz de três por três, que passa pela camada de flatten, que é a transformação em uma camada unidimensional, para conectar na rede densa (Grus, 2021; Sicsú et al., 2023).

Após o processo realizado acima cada imagem foi transformada em um vetor com 128 valores de zero a 255 que é o padrão de cor RGB, é preciso normalizar esses valores em uma escala de zero a um, a normalização é necessária para evitar estouro ou subfluxo numérico durante o treinamento, o treinamento fica mais rápido, pois a descida do gradiente sobressai melhor em escala numérica mais baixa. As entradas de dados representada com valores muito amplo, dificilmente o modelo conseguirá aprender, pois os cálculos causam distorções nos valores dos pesos e na avaliação do erro (Grus, 2021; Silva et al., 2018).

No processo de carga as imagens transformadas foram armazenadas em duas variáveis base_treinamento e base_teste, para serem usados posteriormente no modelo de classificação. O processo de ETL é fundamental para que os dados estejam disponíveis no formato adequado para dar continuidade no processo de classificação, (Grus, 2021; Silva et al., 2018), sem esse processo é impossível ter um bom resultado e até mesmo dar sequência

no modelo de treinamento. Na Figura 4 consta o script com os parâmetros de tratamento da imagem e sua separação para base de treinamento e teste.

```
#Normalização das imagens de 1 a 255 para 0 a 1
gerador_treinamento = ImageDataGenerator(rescale = 1./255,
                                         rotation_range = 7,
                                         horizontal_flip = True,
                                         shear_range = 0.2,
                                         height_shift_range = 0.07,
                                         zoom_range = 0.2)
gerador_teste = ImageDataGenerator(rescale = 1./255)

#base de treinamento e teste
base_treinamento = gerador_treinamento.flow_from_directory('folhas/treino',
                                                           target_size = (64, 64),
                                                           batch_size = 5,
                                                           class_mode = 'binary')
base_teste = gerador_teste.flow_from_directory('folhas/teste',
                                               target_size = (64, 64),
                                               batch_size = 5,
                                               class_mode = 'binary')
```

Figura 4 – Transformação e carga em Python
Fonte: Dados originais da pesquisa

ARQUITETURA DA CNN

A arquitetura CNN foi projetada para processar dados em grades como imagens, ela possui camadas de convoluções e de pooling que ajudam a aprender características importantes das imagens, possui uma camada de entrada onde cada pixel é tratado como um nó de entrada, camadas de convoluções onde cada camada aplica filtros a regiões locais da imagem, capturando bordas, texturas e padrões da imagem Mueller (2020). Em seguida as camadas passam por um achatamento e são conectadas nas camadas densas onde cada nó está conectado como todos os nós da camada anterior, a camada de saída é uma camada densa com uma ativação, está apresenta uma probabilidade de pertencer a classe. Entre as camadas de convolução são usadas camadas de normalização e nas camadas densas são utilizadas camadas de dropouts para regularizar e normalizar os dados evitando o dropout (Grus, 2021; Sicsú et al., 2023; Mueller, 2020).

A arquitetura típica de uma CNN é o empilhamento de algumas camadas convolucionais seguida de uma camada com a função de ativação ReLU, seguida por uma camada de pooling, quanto maior o número de camadas menor é a imagem e mais profunda é a rede, ao final das camadas convolucionais aplica-se uma camada de flatten para conectar na camada densa, Ferreira (2021). A Figura 5 mostra a sequência de uma CNN simples.

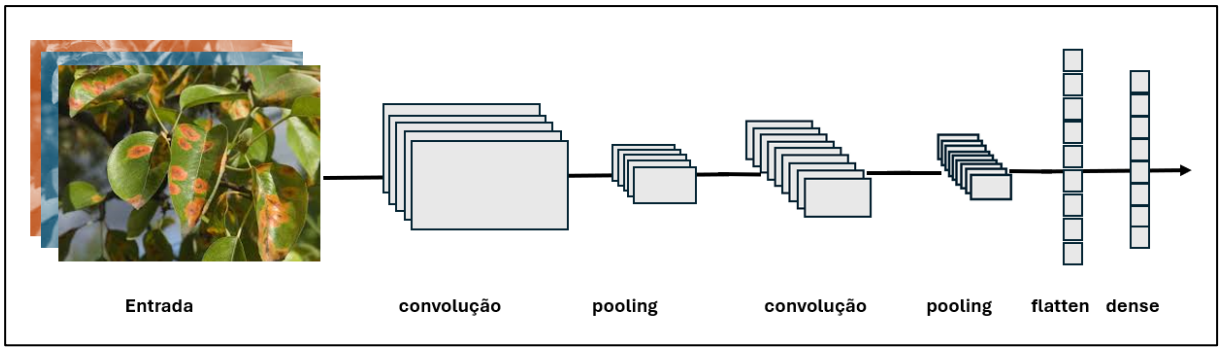

Figura 5 – Arquitetura de uma CNN simples
Fonte: Dados originais da pesquisa

Nas redes neurais valores dos números de camada e neurônios devem ser informados por pressuposição, esses valores são conhecidos como hiper parâmetros, e a definição desses valores é feita por tentativas e erros, é um processo moroso, pois o treinamento do modelo pode durar muitas horas dependendo dos hiper parâmetros escolhidos pelo analista, Sicsú et al., (2023), é preciso pesquisar qual o melhor conjunto de hiper parâmetros que otimizam o modelo. Encontrar uma CNN que atenda a necessidade do negócio não é simples, foi preciso pesquisar muito e testar o conjunto de hiper parâmetros que melhor atendesse o modelo, a escolha passou por um processo empírico com diversos conjuntos de treinos e testes com alterações de camadas e parâmetros até obter um resultado que atenda o objetivo, a (Tabela 1) mostra os parâmetros utilizados nas camadas convolucionais e densas e o tempo de duração de cada treinamento.

Tabela 1 – Arquitetura das camadas convolucionais e densas nos 4 treinamentos

(continua)

Camada		Treino 1	Treino 2	Treino 3	Treino 4
Convolucional 1	*Filtros*	*128*	128	128	128
	Kernell	3x3	3x3	3x3	3x3
Convolucional 2	*Filtros*	*128*	128	256	256
	Kernell	3x3	3x3	3x3	3x3
Convolucional 3	*Filtros*	-	64	512	512
	Kernell	-	3x3	3x3	3x3
Densa 1	Units	128	64	512	512
Densa 2	Units	128	64	256	256
Densa 3	Units	-	-	128	128
Densa 4	Units	-	-	64	64

Tabela 1 – Arquitetura das camadas convolucionais e densas nos 4 treinamentos

(conclusão)

Camada		Treino 1	Treino 2	Treino 3	Treino 4
Densa saída	Units	1	1	1	1
Épocas		10	100	100	200
Tempo		12min	45min	1h57min	2h39min

Fonte: Dados originais de pesquisa

No desenvolvimento inicial foi utilizado os seguintes parâmetros: 128 número de filtros na camada, com o tamanho de kernel três por três, as imagens foram redimensionadas em 64 x 64 e o canal de cor recebeu o parâmetro três, por se tratar de imagem colorida no padrão Red Green Blue [RGB]. Em seguida foi usado o método batchNormalization para adicionar uma camada de normalização em lote após a camada de convolução, e normalizar as ativações da camada anterior, melhorando a estabilidade e acelerando o treinamento. Após o batchNormalization foi utilizado o método MaxPooling2D com os parâmetros (2,2) no pool_size, indicando uma janela de dois por dois buscando o maior valor desse quadrante em uma matriz cinco por cinco que resulta um vetor unidimensional com nove valores, em seguida foi utilizado o método Flatten para adicionar uma camada de achatamento, para ser conectado na camada densa(Grus, 2021; Silva et al., 2018), conforme mostra a Figura 6.

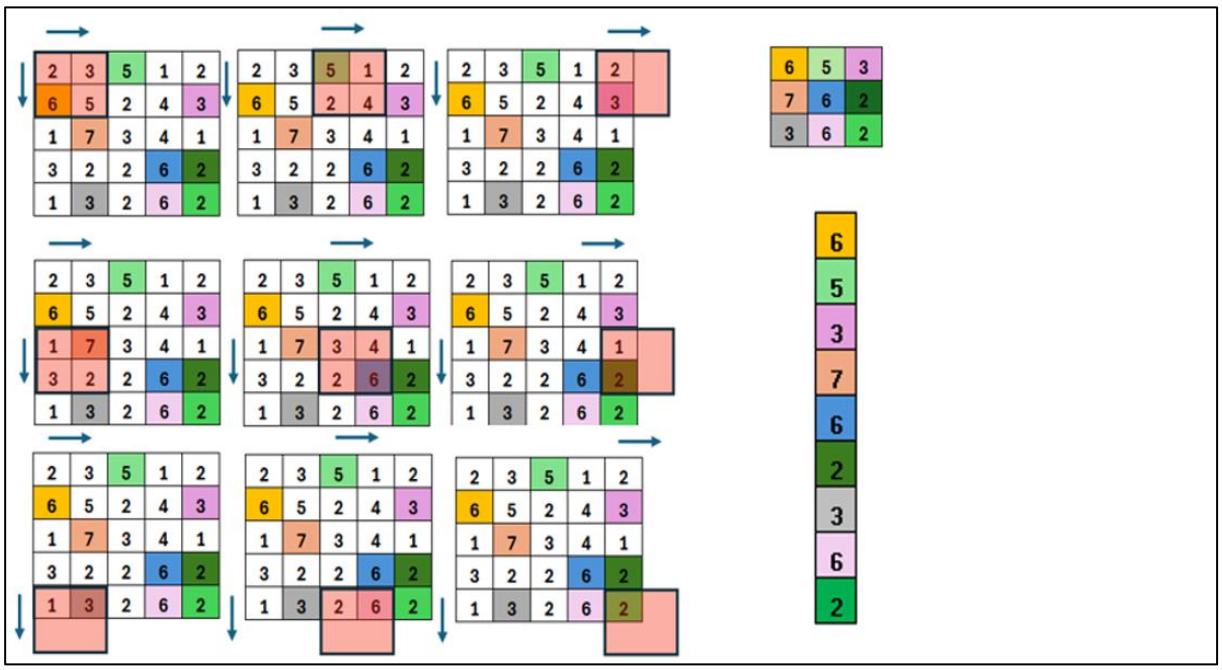

Figura 6 – Max pooling e flatten
Fonte: Dados originais da pesquisa

As camadas densas são compostas por duas camadas ocultas com a configuração de 128 neurônios e função de ativação Relu e uma camada de saída composta por um neurônio e função de ativação sigmoid para fornecer saída com probabilidades, entre as camadas foi adicionado um Dropout de 0.2 para indicar que durante o treinamento 20% dos neurônios serão desligados para evitar o overfitting, (Grus, 2021; Silva et al., 2018). Na compilação do modelo foi usado o otimizador adam para ajustar os pesos durante o treinamento, a função de perda usada foi binary_crossentropy por ser um problema de classificação binária e a métrica de avaliação durante o treinamento foi acurácia. Esses parâmetros utilizados podem ser alterados conforme o propósito do treinamento do modelo e eles interferem no resultado do modelo, existem diversos hiper parâmetros, testá-los para obter um melhor resultado foi um desafio, pois o treinamento do modelo gerava um tempo de processamento que muitas vezes passou de horas, todo o script da arquitetura inicial do modelo está na Figura 7.

```
#Arquitetura da CNN
#Etapas de 1 a 3 - dimensao da imagem 64 x 64 4096 pixels - 3 por ser padrao RGB
classificador = Sequential()
classificador.add(Conv2D(128, (3,3), input_shape = (64, 64, 3), activation = 'relu'))
classificador.add(BatchNormalization())
classificador.add(MaxPooling2D(pool_size = (2,2)))

classificador.add(Conv2D(128, (3,3), input_shape = (64, 64, 3), activation = 'relu'))
classificador.add(BatchNormalization())
classificador.add(MaxPooling2D(pool_size = (2,2)))

classificador.add(Flatten())

#Camadas densas
classificador.add(Dense(units = 128, activation = 'relu'))
classificador.add(Dropout(0.2))
classificador.add(Dense(units = 128, activation = 'relu'))
classificador.add(Dropout(0.2))
classificador.add(Dense(units = 1, activation = 'sigmoid')) #Sigmoid retorna probabilidade

classificador.compile(optimizer = 'adam', loss = 'binary_crossentropy',
                      metrics = ['accuracy'])
```

Figura 7 – Arquitetura CNN em Python
Fonte: Dados originais da pesquisa

O modelo citado acima não foi a arquitetura final do trabalho, conforme será mostrado a frente nos resultados, o modelo não obteve resultados de classificação satisfatório, o algoritmo foi sendo ajustado para melhorar a performance. A Primeira modificação para um novo teste foi adicionada uma camada densa com 64 neurônios, e o treinamento foi com 100 épocas, foi acrescentado mais dados para o treinamento, alterando de 400 para 600 e os dados de testes passaram de 176 para 262, a Figura 8 mostra que a adição da camada Densa. O modelo respondeu melhor, mas não foi atingido o objetivo do trabalho, foi necessário treinar com várias arquiteturas diferentes para alcançar um resultado que atingisse o objetivo do

trabalho, foi identificado um padrão de imagens que o modelo não estava classificando corretamente.

```
#Camadas densas
classificador.add(Dense(units = 128, activation = 'relu'))
classificador.add(Dropout(0.20))
classificador.add(Dense(units = 128, activation = 'relu'))
classificador.add(Dropout(0.20))
classificador.add(Dense(units = 64, activation = 'relu'))
classificador.add(Dropout(0.20))

classificador.add(Dense(units = 1, activation = 'sigmoid')) #Sigmoid retorna probabilidade

classificador.compile(optimizer = 'adam', loss = 'binary_crossentropy',
                      metrics = ['accuracy'])
```

Figura 8 – Adição da camada Densa
Fonte: Dados originais da pesquisa

Após vários testes de camadas, a arquitetura que apresentou melhor desempenho foi com três sequências de: camadas convolucionais, BatchNormalization e MaxPooling2D, onde o único parâmetro que foi alterado foi o conv2D na seguinte ordem 128 na primeira camada, 256 na segunda camada, 512 na terceira camada e quatro camadas densas com os seguintes parâmetros: units 512 na primeira camada, units 256 na segunda camada, units 128 na terceira camada, units 64 na quarta camada, em todas as camadas a função de ativação foi a ReLU, e em cada camada um dropout de 20% e por fim uma saída com a função sigmoid, o valor do parâmetro conv2D da última camada convolucional é o mesmo valor do parâmetro units da primeira camada densa, conforme ilustra a Figura 9.

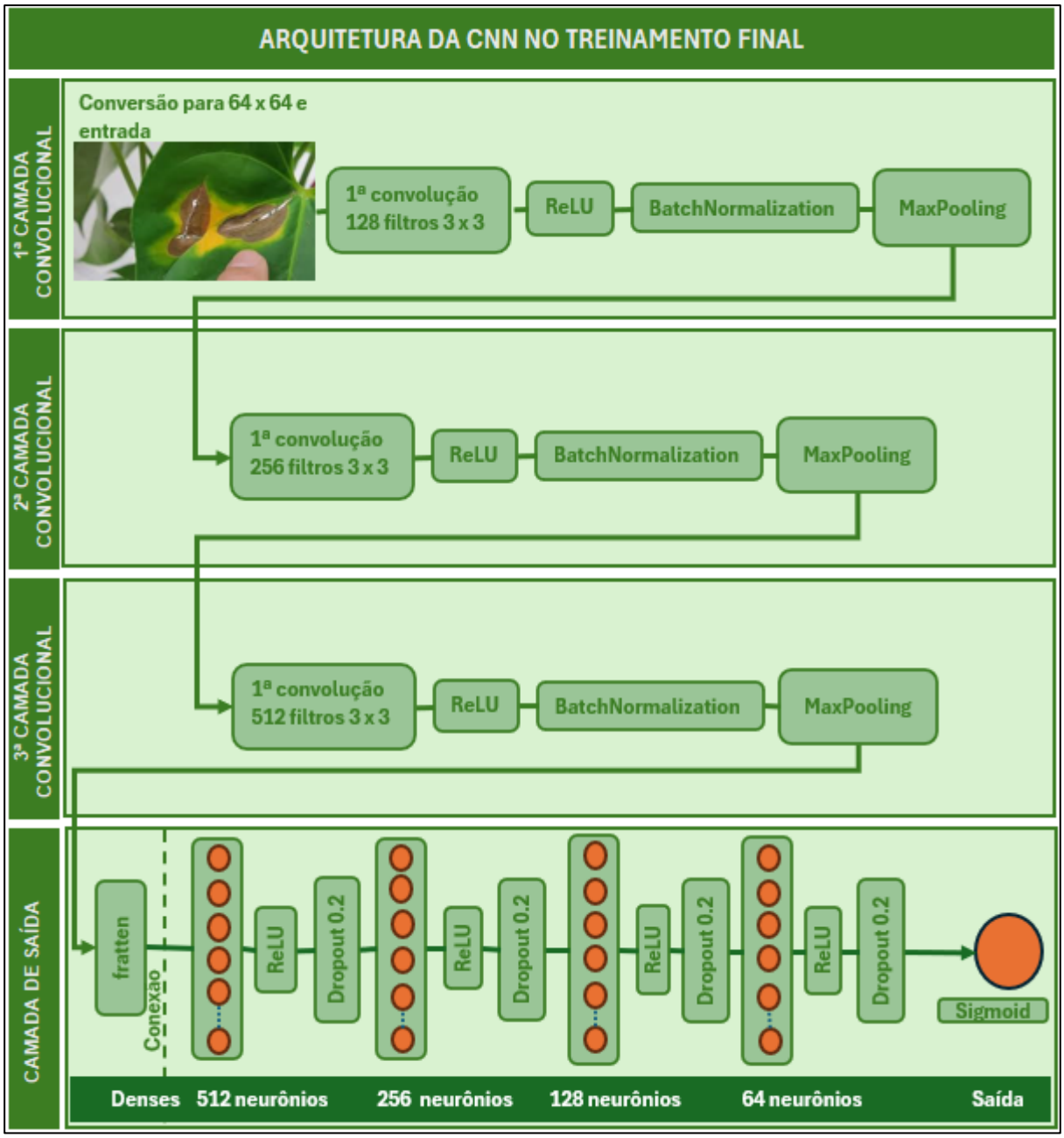

Figura 9 – Arquitetura da CNN
Fonte: Dados originais da pesquisa

TREINAMENTO DO MODELO

Após a criação da arquitetura foi criado o modelo de treinamento, que no primeiro ajuste recebeu os seguintes parâmetros step_per_epoch de 400 // 5 e epochs = 10, com isso cada época passa por 5 batchs de 80 imagens no conjunto de dados de treinamento, conforme mostra a Figura 10. Inicialmente o modelo foi treinado com 400 imagens, mas no decorrer do trabalho será aumentado a fim de buscar um melhor resultado.

```
classificador.fit(base_treinamento, steps_per_epoch=400 // 5,
                  epochs=10, validation_data=base_teste,
                  validation_steps=174 // 5)
```

Figura 10 – Treinamento do modelo em Python
Fonte: Dados originais da pesquisa

A rede em treinamento recebe em seus neurônios da camada de entrada os dados de treinamento, após calcular os valores e pesos ligados a cada neurônio, é feito uma somatória do resultado, e o resultado é submetido a uma função de ativação e esse valor é passado para o neurônio da camada seguinte, esse processo é repetido passando pelas camadas escondidas até ser transmitido o resultado final para a camada de saída, os modelos são padronizados com uma camada de entrada, uma camada de saída e de uma a n camadas escondidas, (Grus, 2021; Silva et al., 2018), conforme a Figura 11. No trabalho o valor de saída varia de zero a um, o valor até a 0,50 pertence a classe de plantas doentes e o valor acima de 0,50 pertence a classe de plantas saudáveis.

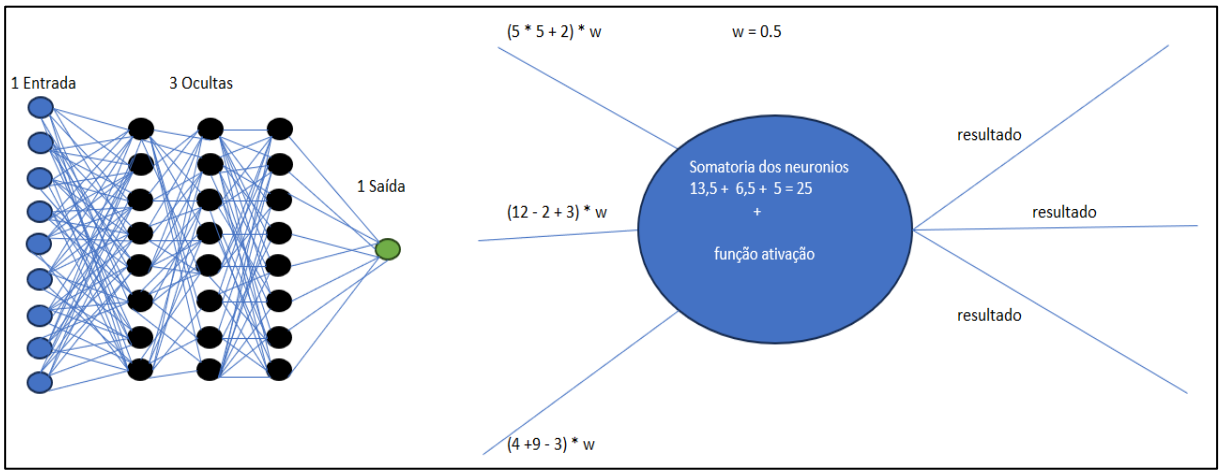

Figura 11 – Camadas densas e Neurônio
Fonte: Dados originais da pesquisa

Métricas para avaliação do modelo

Para avaliar a qualidade da predição do modelo foram criadas algumas métricas, obtidas a partir do resultado da matriz de confusão Figura 12, verdadeiro positivo, falso positivo, verdadeiro negativo e falso negativo, esses são os números reais de teste do modelo.

		Predito pelo modelo	
		Classe Negativa	**Classe Positiva**
Real	**Classe Negativa**	**VN** Verdadeiro Negativo	**FP** Falso Positivo
	Classe Positiva	**FN** Falso Negativo	**VP** Verdadeiro Positivo

Figura 12 – Matriz de confusão
Fonte: Dados originais da pesquisa

Com esses números é possível obter as taxas e as métricas do modelo, abaixo estão as equações referente a taxa de verdadeiro positivo conhecido como sensibilidade (1) que mede a proporção de resultados positivos classificados pelo modelo, taxa de verdadeiro negativo conhecido também como especificidade (2), proporção de resultados negativos que foram classificados como negativo pelo modelo, precisão (3), é a proporção de resultados positivos classificados pelo modelo e que são realmente positivos, a acurácia (4), mede a proporção de acertos do modelo, classificando positivos como positivos e negativos como negativos em relação ao total e F1-Score (5), que mede o equilíbrio entre as classes quanto mais próximo de um mais equilibrada estão as classes negativas e positivas, (Junior et al., 2022). A Figura 13 representa o código criado em Python para calcular as métricas.

$$Sensibilidade\ ou\ Recall = \frac{VP}{\mathrm{VP+FN}} \quad (1)$$

$$Especificidade = \frac{VN}{\mathrm{VN+FP}} \quad (2)$$

$$Precisão = \frac{VP}{\mathrm{VP+FP}} \quad (3)$$

$$Acurácia = \frac{VP+VN}{\mathrm{VP+FP+VN+FN}} \quad (4)$$

$$F1-Score = \frac{2*Precisão*Recall}{\mathrm{Precisão+Recall}} \quad (5)$$

onde, VP é o verdadeiro positivo; FN é o falso negativo; VN é o verdadeiro negativo; e FP é o falso positivo.

```
#Calculos das métricas
sensibilidade_Recall = steste_saude / (steste_saude + dteste_saude)
tx_FP = steste_doente / (steste_doente + dteste_doente)
especificidade = dteste_doente / (dteste_doente + steste_doente)
tx_FN = dteste_saude / (dteste_saude + steste_saude)
acuracia = (steste_saude + dteste_doente)/\
           (steste_saude + dteste_doente +
            dteste_saude + steste_doente)
precisao = steste_saude / (steste_saude + steste_doente)
f1score = 2*recall*precisao / (recall + precisao)
```

Figura 13 – Cálculo das métricas em Python
Fonte: Dados originais da pesquisa

Foi feito também uma codificação para obter a Curva ROC que mede a sensibilidade e especificidade do modelo, a curva quanto mais próximo do canto superior esquerdo do gráfico, indica um modelo com bom desempenho, indica taxa baixa de falsos positivos e taxa baixa de falsos negativos, a Figura 14 mostra a codificação e impressão da curva ROC. (Junior et al., 2022).

```
# Previstos (y_prob)  reais (y_true)
tx_FP, tx_VP, thresholds = roc_curve(Y_real, y_prob)

# Calcular a área sob a curva ROC (AUC-ROC)
roc_auc = auc(tx_FP, tx_VP)
print(f'Área sobre a curva ROC: {roc_auc:.2f}%')
# Plotar a curva ROC
plt.figure(figsize=(5, 5))
plt.plot(tx_FP, tx_VP, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('Taxa de Falsos Positivos (FPR)')
plt.ylabel('Taxa de Verdadeiros Positivos (TPR)')
plt.title('Curva ROC')
plt.legend(loc='lower right')
plt.show()
```

```
Área sobre a curva ROC: 0.95%
```

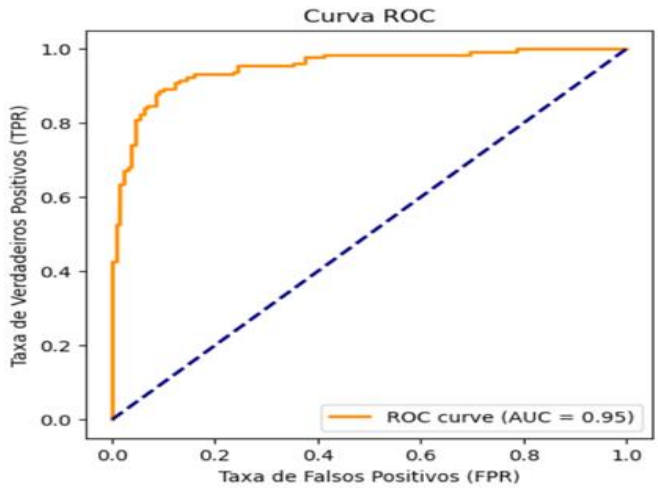

Figura 14 – Curva ROC em Python
Fonte: Dados originais da pesquisa

Resultados e Discussão

Após o ETL, a criação da arquitetura e o treinamento do modelo, a validação foi feita nos dados de teste, e os resultados serão apresentados abaixo com a métrica de avaliação do modelo. Inicialmente o resultado não foi satisfatório, o modelo foi treinado com poucas imagens e com camadas rasas, esses fatores contribuíram para que o modelo não performasse bem, o modelo precisou ser ajustado para obter um melhor desempenho.

Primeiro treinamento do modelo

Após executar o modelo de treinamento com dez épocas foi preciso criar uma lista para armazenar os dados reais e os indicados pelo modelo, para fazer a comparação dos resultados obtidos. O modelo foi efetivo com os dados referentes a plantas saudáveis acertando todos os dados previstos, mas na classe doentes não teve um bom desempenho acertou 23 e errou 65 dos 88. Na Figura 15 encontra as métricas do modelo.

```
           Acertos:                ERROS
Saudaveis:      88        -          0
Doentes:        23        -          65

Métricas do modelo

Taxa Verdadeiro positivo TPR: 0.58%
Taxa Falso positivo FPR: 0.00%
Falso positivo: 0
Taxa Verdadeiro negativo: 1.00%
NPV: 0.26%
Taxa Falso Negativo: 0.42%
Acurácia: 0.63%
Precisão: 1.00%
Recall: 0.58%
F1-Score: 0.73%
```

Figura 15 – Métricas do primeiro modelo
Fonte: Resultados originais da pesquisa

Ao analisar as métricas da Figura 15 foi observado que a taxa de verdadeiro positivo foi impactada pelo erro da classe doentes, que não classificou corretamente um percentual aceitável, o falso positivo e a taxa de falso positivo indicaram que a classe não classificou nenhum dado negativo como positivo de forma errada. A taxa de verdadeiro negativo indica que todas as instâncias que o modelo indicou como negativas eram realmente negativas, o NPV mostra que o modelo só previu 26% dos dados que eram realmente negativos, a taxa de falso negativo mostra que o modelo não encontrou anormalidade onde há anormalidade, a acurácia é o percentual de classificação correta no total testado pelo modelo, e 63% não é um valor aceitável, a precisão de 100% indica que todos os dados classificados como positivos pelo modelo são realmente positivos, o recall é a proporção dos verdadeiros positivos em relação ao total de positivos e o F1-score é o equilíbrio entre as classes, em resumo o modelo precisou ser melhorado. Esse primeiro modelo as camadas eram rasas, e o modelo com

poucas épocas e o número de imagens treinadas não contribuiu para um melhor resultado de treinamento.

Segundo treinamento do modelo

Após rodar o modelo com a modificação na arquitetura e com mais dados para treinamento foi possível obter um resultado melhor, com uma acurácia de 89%, o modelo está classificando bem os dados da classe saudáveis, mas a classe de plantas doentes ainda está com um percentual de erro alto 16.34%, e com o resultado é possível observar que ele pode classificar plantas que estão doentes como saudáveis e isso pode ser um fator grave, pois um foco de doença não identificado pode pôr a lavoura inteira em risco, o modelo precisa ser melhorado para errar menos nas classificações de plantas doentes. Na Figura 16 é mostrado a métrica do modelo.

```
          Acertos:                ERROS
Saudaveis:      128       -          3
Doentes:        106       -          25

Métricas do modelo

Taxa Verdadeiro positivo TPR: 0.8366    → O modelo identificou corretamente 83.66% dos casos positivos.
Taxa Falso Negativo: 0.1634             → Indica que perdeu 16.34% dos casos positivos.
Taxa Falso positivo FPR: 0.0275         → Indica que o modelo classificou incorretamente 2.75% dos casos negativos.
Taxa Verdadeiro negativo: 0.9725        → Indica que o modelo identificou corretamente 97.25% dos casos negativos.
Verdadeiro positivo: 128                → Modelo previu e acertou os dados como positivos.
Falso positivo: 3                       → Modelo previu como negativo, mas era positivo.
Verdadeiro negativo: 106                → Modelo previu e acertou os dados como negativo.
Falso negativo: 25                      → Modelo previu como positivo, mas era negativo.
NPV: 0.81                               → Indica que 80.92% das instâncias classificadas negativas eram realmente negativas
Acurácia: 0.8931                        → Indica que o modelo acertou 0.89% das previsões
Precisão: 0.9771                        → Indica que 97.71% das instâncias previstas como positivas eram realmente positivas.
Recall: 0.8366                          → Indicando que 83.66% dos casos positivos foram corretamente identificados pelo modelo
F1-Score: 0.9014                        → Indicando um equilíbrio entre precisão e recall de 90.14%
```

Figura 16 – Métricas do segundo modelo
Fonte: Resultados originais da pesquisa

Ao analisar as imagens que o modelo estava errando, foi identificado que eram imagens onde as pragas tinham coloração parecida com a da planta e o modelo não estava conseguindo classificar conforme mostra a Figura 17.

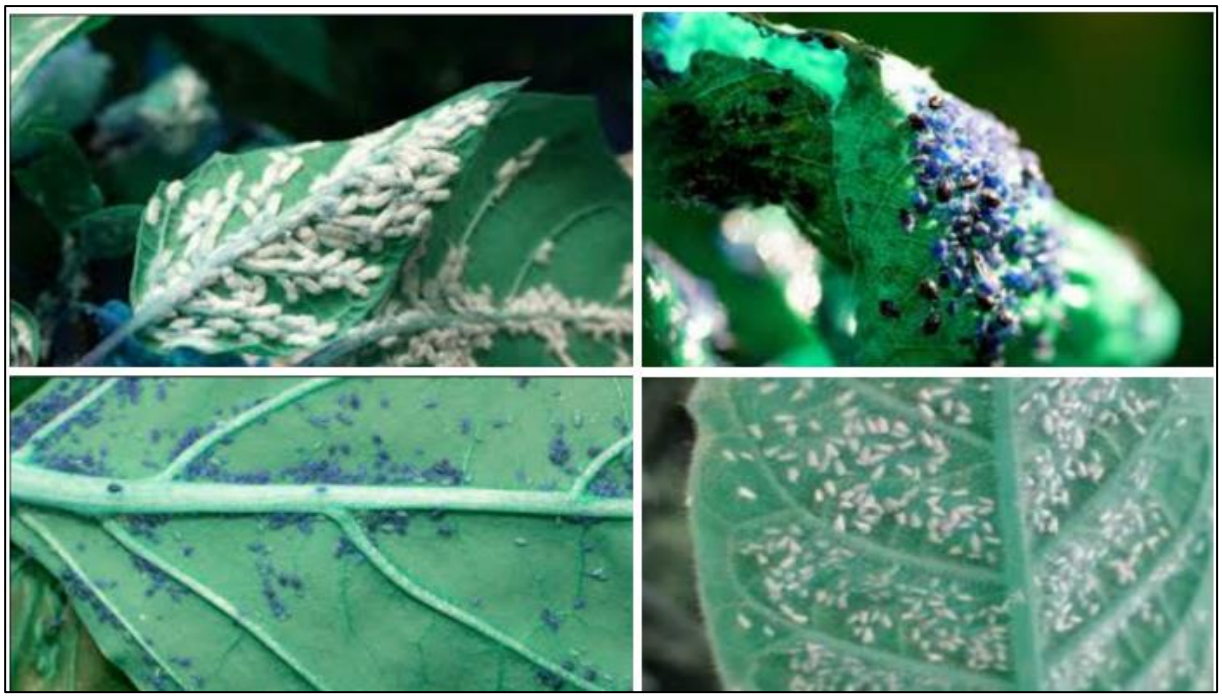

Figura 17 – Imagens classe doentes classificadas como saudáveis
Fonte: Resultados originais da pesquisa

Terceiro treinamento do modelo

Na arquitetura que obtive melhor resultado adicionei mais imagens semelhantes ao que o modelo não acertava para treinamento e adicionei camadas densas e convolucionais e o resultado atendeu o que eu tinha como objetivo, uma acurácia de 96,18%, a sensibilidade em 94,81%, a especificidade 97,64%, uma previsão de 97,91% e o F1Score 98,01%, conforme mostra a Figura 18, a métrica especificidade nesse trabalho é importante por ser a que classifica como doente a planta que realmente está doente, pois assim o produtor ou técnico pode agir e tratar o problema evitando uma possível infestação na lavoura, comparando com os modelos iniciais, e alguns trabalhos correlatos o modelo está classificando bem os dados de teste.

```
           Acertos:                ERROS
Saudaveis:      128       -          3
Doentes:        124       -          7

Verdadeiro positivo: 128              → Modelo previu e acertou os dados como positivos.
Falso positivo:      3                → Modelo previu como negativo, mas era positivo.
Verdadeiro negativo: 124              → Modelo previu e acertou os dados como negativo.
Falso negativo:      7                → Modelo previu como positivo, mas era negativo.


Métricas do modelo

Acurácia:               0.9618        → Indica que o modelo acertou 96.18% das previsões
Sensibilidade (Recall):  0.9481       → O modelo identificou corretamente 94.81% dos casos positivos.
Especificidade:         0.9764        → Indica que o modelo identificou corretamente 97.64% dos casos negativos.
Precisão:               0.9771        → Indica que 97.71% das instâncias previstas como positivas eram realmente positivas.
F1-Score:               0.9801        → Indicando um equilíbrio entre precisão e recall de 98.01%
```

Figura 18 – Métricas do terceiro modelo
Fonte: Resultados originais da pesquisa

Quarto treinamento do modelo

No quarto treinamento foi mantido a arquitetura do terceiro treinamento, com a alteração do número de épocas de 100 para 200, e o resultado não foi muito diferente do obtido no último treinamento, o erro na classe de doentes passou de 7 para 10. Abaixo na (Tabela 2) é apresentado as métricas dos treinamentos citados no trabalho e é possível analisar que o modelo melhorou muito se comparado com os treinamentos iniciais e que o modelo obtido é capaz de classificar as plantas com uma boa acurácia.

Tabela 2 – Métricas dos treinamentos

Métrica	Treino 1	Treino 2	Treino 3	Treino 4
Acurácia	63%	89%	96%	95%
Sensibilidade/Recall	58%	83%	94%	92%
Especificidade	100%	97%	97%	97%
Precisão	100%	97%	97%	97%
F1-Score	73%	90%	98%	98%
AUC	68%	82%	99%	98%

Fonte: Resultados originais de pesquisa

Testando o modelo com imagens reais

O modelo foi testado em dados coletados na estação experimental da cidade de Bebedouro, São Paulo, o teste foi feito com 30 imagens de plantas que devem ser classificados como doentes e 20 imagens que devem ser classificados como saudáveis. O resultado das classificações assemelha aos resultados das imagens de validação conforme mostra a (Tabela 3), o modelo foi efetivo tanto com dados de validação, quanto aos dados reais, Figura 19 métricas do modelo em dados reais, Figura 20 imagens coletadas na estação experimental de Bebedouro, São Paulo.

```
          Acertos:                ERROS
Saudaveis:      20        -          0
Doentes:        28        -          2

Verdadeiro positivo: 20         → Modelo previu e acertou os dados como positivos.
Falso positivo:      0                  → Modelo previu como negativo, mas era positivo.
Verdadeiro negativo: 28         → Modelo previu e acertou os dados como negativo.
Falso negativo:      2                  → Modelo previu como positivo, mas era negativo.


Métricas do modelo

Acurácia:               0.9600          → Indica que o modelo acertou 96.00% das previsões
Sensibilidade (Recall): 0.9091          → O modelo identificou corretamente 90.91% dos casos positivos.
Especificidade:         1.0000          → Indica que o modelo identificou corretamente 100.00% dos casos negativos.
Precisão:               1.0000          → Indica que 100.00% das instâncias previstas como positivas eram realmente positivas.
F1-Score:               0.9524          → Indicando um equilíbrio entre precisão e recall de 95.24%
```

Figura 19 – Métricas com imagens reais
Fonte: Resultados originais da pesquisa

Figura 20 – Imagens reais
Fonte: Resultados originais da pesquisa

Tabela 3 – Métricas das imagens de teste e imagens reais

Camada	Imagens de teste	Imagens reais
Acurácia	96%	96%
Sensibilidade/Recall	94%	90%
Especificidade	97%	100%
Precisão	97%	100%
F1-Score	98%	95%
AUC	99%	100%

Fonte: Resultados originais de pesquisa

Conclusão

O trabalho concluiu que a IA é uma ferramenta valiosa em diversas áreas, incluindo a agricultura, onde pode auxiliar na detecção de doenças e pragas em plantas, aplicação de defensivos com precisão, mapeamento e muitas outras aplicações. Foi ressaltado que os

resultados obtidos inicialmente pelo algoritmo foram abaixo do esperado, foi necessário testar diferentes parâmetros para atingir uma acurácia acima de 95%.

O modelo final alcançou uma acurácia de 96,18% nas imagens de teste e 96% nas imagens novas, demonstrando a eficácia da abordagem na classificação de plantas. Observou-se que um maior número de imagens de treinamento resulta em melhor desempenho do modelo, e que arquiteturas mais complexas, com mais camadas convolucionais, tendem a ter melhor performance, embora demandem mais recursos computacionais e levam muito mais tempo para ser treinado. O trabalho oferece uma contribuição teórica ao explorar a eficácia das redes neurais convolucionais no reconhecimento de imagens, as aplicações de deep learning contribuiu para a solução de um problema real de visão computacional, também oferece uma contribuição prática na agricultura 4.0, pois com o modelo gerado é possível auxiliar na tomada de decisão indicando quais plantas estão doentes ou com infestação de pragas, para que o problema possa ser tratado antes de espalhar por toda a lavoura, contribui na redução significante do uso de pesticidas, a informação gerada permite que o pesticida seja aplicado apenas no local afetado.

Como trabalho futuro, pretende-se dar continuidade ao treinamento do modelo, adaptando-o para classificar irregularidades em culturas específicas, como identificar doenças ou pragas específicas e fornecer informações sobre o percentual de incidência de cada uma das pragas e doenças da determinada cultura. Isso demonstra o potencial de aplicação da IA na agricultura, visando aumentar a eficiência e a produtividade das lavouras, além de contribuir para a sustentabilidade e segurança alimentar.

Agradecimento

Dedico este trabalho a minha esposa Fernanda, meus filhos Letícia, Ana Júlia e David Filho pela paciência na minha ausência, ao meu orientador Felipe Bastos dos Reis que sempre me apoiou e contribuiu para o desenvolvimento do trabalho, e aos professores e toda equipe da USP/ESALQ.

Referências

Artioli, Felipe; Beloni, Tatiane. 2016. Diagnóstico do perfil do usuário de Drones no Agronegócio Brasileiro. Revista iPecege, v. 2, n. 3, p. 40-56. Disponível em:
< https://ipecege.emnuvens.com.br/Revista/article/view/73> Acesso em: 30 janeiro 2024.

Behrman, K. R. (2023). Fundamentos de Python para ciência de dados. Grupo A. Disponível em:
< https://integrada.minhabiblioteca.com.br/books/9788582605974> Acesso em: 14 fevereiro 2024.

Elias, Richaldo do Lito; Manhiça, Engo Ruben Moisés. 2019. Detecção e Classificação de Doenças em Plantas Agrícolas com Recurso a Inteligência Artificial. Manografia em Engenharia de Informática. Universidade Eduardo Mondlane, Maputo, Moçambique. Disponível em:
<https://www.researchgate.net/profile/Ruben-Manhica/publication/372523269_Deteccao_e_Classificacao_de_Doencas_em_Plantas_Agricolas_com_Recurso_a_Inteligencia_Artificial/links/64bbffe495bbbe0c6e542345/Deteccao-e-Classificacao-de-Doencas-em-Plantas-Agricolas-com-Recurso-a-Inteligencia-Artificial.pdf> Acesso em: 9 fevereiro 2024.

Ferreira, R. (2021). Deep learning. Editora Saraiva. Disponível em:
< https://integrada.minhabiblioteca.com.br/books/9786589881520> Acesso em: 10 fevereiro 2024.

Fonseca, E.M.D. S., & Araújo, R.C. D. (2014). Fitossanidade princípios básicos e métodos de controle de doenças e pragas 1ª edição 2015. Editora Saraiva. Disponível em:
< https://integrada.minhabiblioteca.com.br/books/9788536530956> Acesso em: 20 janeiro 2024.

Gonçalves, Juliano de Paula. 2020. Aprendizado profundo aplicado à quantificação da severidade de sintomas de doenças e pragas foliares. Dissertação de Mestrado em Engenharia Agrícola. Universidade Federal de Viçosa, Viçosa, MG, Brasil. Disponível em:
<https://www.locus.ufv.br/handle/123456789/28373> Acesso em: 5 fevereiro 2024.

Grus, J. (2021). Data Science do Zero. Editora Alta Books. Disponível em:
< https://integrada.minhabiblioteca.com.br/books/9788550816463> Acesso em: 12 novembro 2023.

Junior, G. B. V., Lima, B. N., Pereira, A. A., Rodrigues, M. F., Oliveira, J. R. L., Silio, L. F., ... & Passos, R. P. (2022). Métricas utilizadas para avaliar a eficiência de classificadores em algoritmos inteligentes. Revista CPAQV–Centro de Pesquisas Avançadas em Qualidade de Vida| Vol, 14(2), 2. Disponivel em:
<https://www.researchgate.net/profile/Guanis-Vilela-Junior/publication/359541310_METRICAS_UTILIZADAS_PARA_AVALIAR_A_EFICIENCIA_DE_CLASSIFICADORES_EM_ALGORITMOS_INTELIGENTES/links/634ec60312cbac6a3ed73448/METRICAS-UTILIZADAS-PARA-AVALIAR-A-EFICIENCIA-DE-CLASSIFICADORES-EM-ALGORITMOS-INTELIGENTES.pdf> Acesso em:16 fevereiro 2024.

Maciel, F.M.D. B. (2020). Python e Django. Editora Alta Books. Disponível em:
<https://integrada.minhabiblioteca.com.br/books/9786555200973> Acesso em: 8 fevereiro 2024.

Morais, I.S. D., Gonçalves, P.D. F., & Ledur, C. L. et al. (2018). Introdução a Big Data e Internet das Coisas (IoT). Grupo A. Disponível em:
<https://integrada.minhabiblioteca.com.br/books/9788595027640> Acesso em: 23 outubro 2023.

Mueller, J. P. (2020). Aprendizado profundo para leigos. Editora Alta Books. Disponível em:
< https://integrada.minhabiblioteca.com.br/books/9788550816982> Acesso em: 17 novembro 2023.
Stein, R. T., & Coscolin, R.B. S. (2020). Agricultura climaticamente inteligente e sustentabilidade. Grupo A. Disponível em:
< https://integrada.minhabiblioteca.com.br/books/9786581492083> Acesso em: 30 janeiro 2023.

Netto, A., & Maciel, F. (2021). Python para Data Science e Machine Learning Descomplicado. Editora Alta Books. Disponível em:
<https://integrada.minhabiblioteca.com.br/books/9786555203172> Acesso em: 15 outubro 2023.

Ribeiro, Edivaine de Godoy. 2020. Rede neural convolucional aplicada ao reconhecimento de passagens de nível clandestinas em ferrovias. Manografia em Engenharia Elétrica. Instituto Federal do Espírito Santo, Vitória, ES, Brasil. Disponível em:
<https://repositorio.ifes.edu.br/handle/123456789/711> Acesso em 3 fevereiro 2024.

Sicsú, A. L., Samartini, A., & Barth, N. L. (2023). Técnicas de machine learning. Editora Blucher. Disponível em:
<https://integrada.minhabiblioteca.com.br/books/9786555063974> Acesso em: 10 dezembro 2023.

Silva, F. M., Lenz, M. L., & Freitas, P.H. C. et al. (2018). Inteligência artificial. Grupo A. Disponível em:
<https://integrada.minhabiblioteca.com.br/books/9788595029392> Acesso em: 1 fevereiro 2024.

Tetila, Everton, C. 2019. Detecção e classificação de doenças e pragas da soja usando imagens de veículos aéreos não tripulados e técnicas de visão computacional. Tese Doutorado em Desenvolvimento Local. Universidade Católica Dom Bosco, Campo Grande, MS, Brasil. Disponível em:
<https://repositorio.ufgd.edu.br/jspui/handle/prefix/2385> Acesso em: 2 fevereiro 2024.