IMPACTOS DA REMOÇÃO DO FUNDO NA CLASSIFICAÇÃO DE PATOLOGIAS VEGETAIS POR REDES NEURAIS CONVOLUCIONAIS

FERNANDO CEZAR RIBEIRO JUNIOR

TRABALHO DE CONCLUSÃO DE CURSO
EM ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

Universidade de Brasília
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

Impactos da remoção do fundo na classificação de patologias vegetais por Redes Neurais Convolucionais

Fernando Cezar Ribeiro Junior

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO(A) ELETRICISTA.

APROVADA POR:

Prof. Alexandre Ricardo Soares Romariz, Ph.D (ENE-UnB)
(Orientador)

Prof. Eduardo Peixoto Fernandes da Silva, Ph.D. (ENE-UnB)
(Examinador Interno)

Prof. Marcelo Lopes Pereira Junior, Ph.D. (ENE-UnB)
(Examinador Interno)

Brasília/DF, 10 de Outubro de 2024.

FICHA CATALOGRÁFICA

RIBEIRO JUNIOR, FERNANDO CEZAR

Impactos da remoção do fundo na classificação de patologias vegetais por Redes Neurais Convolucionais. [Brasília/DF] 2024.

xxx, nnnp., 210 x 297 mm (ENE/FT/UnB, Engenheiro(a) Eletricista, Trabalho de Conclusão de Curso, 2024).

Universidade de Brasília, Faculdade de Tecnologia, Departamento de Engenharia Elétrica.

Departamento de Engenharia Elétrica

1. Remoção do Fundo	2. Classificador de imagens
3. Patologias Vegetais	4. Rede Neural Convolucional
5. Rede Neural U^2-net	6. PyMatting
7. Rembg	8. RGB
I. ENE/FT/UnB	II. Impactos da remoção do fundo na classificação

REFERÊNCIA BIBLIOGRÁFICA

RIBEIRO JUNIOR, FERNANDO CEZAR (2024). Impactos da remoção do fundo na classificação de patologias vegetais por Redes Neurais Convolucionais. Trabalho de Conclusão de Curso, Publicação ENE.XXXXX/2024, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, xxxxp.

CESSÃO DE DIREITOS

AUTOR: Fernando Cezar Ribeiro Junior

TÍTULO: Impactos da remoção do fundo na classificação de patologias vegetais por Redes Neurais Convolucionais.

GRAU: Engenheiro(a) Eletricista ANO: 2024

Fernando Cezar Ribeiro Junior

Universidade de Brasília (UnB)

Campus Darcy Ribeiro

Faculdade de Tecnologia - FT

Departamento de Engenharia Elétrica (ENE)

Brasília - DF CEP 70919-970

Para minha família, Fernando, Daisy e Ana Beatriz.

AGRADECIMENTOS

À minha família, que sempre foi minha base, meu alicerce e meu maior orgulho.

Aos meus pais, Fernando e Daisy, por estarem sempre ao meu lado em todas as decisões, servindo como inspiração e força motriz para que eu pudesse seguir adiante.

À minha irmã, Ana Beatriz, por sua amizade e companheirismo ao longo desta caminhada, sempre me inspirando com sua dedicação.

À minha namorada, Laura Barreto, pelo carinho constante, amor, companheirismo e por ser meu porto seguro durante este processo.

Aos meus amigos, pela amizade verdadeira e pelas memórias compartilhadas que tornaram esta etapa da minha vida mais leve e feliz. Em especial, agradeço aos amigos que fiz durante a faculdade, a convivência com cada um de vocês foi essencial, e espero que a amizade se mantenha por toda a vida.

Ao meu professor e orientador, Alexandre Romariz, cuja dedicação, paciência e apoio irrestrito ao longo do desenvolvimento deste trabalho foram de grande valor.

Por fim, a todos que, de alguma forma, contribuíram para a realização deste trabalho, deixo aqui meu sincero agradecimento.

RESUMO

Este trabalho tem como foco a melhoria de um classificador de patologias vegetais, visando sua aplicação prática em ambientes reais, como o campo. O classificador tem como objetivo identificar a espécie de planta e, caso esteja contaminada, diagnosticar qual é a doença presente. O principal objetivo deste trabalho é investigar o impacto da remoção do fundo das imagens na acurácia de um classificador de doenças vegetais desenvolvido com redes neurais convolucionais. Para atingir esse objetivo, foi elaborado um algoritmo que faz uso da rede neural U^2-net, projetada para realizar a segmentação eficiente das imagens, removendo o fundo e destacando o objeto de interesse, neste caso a folha. Além de replicar e comparar o trabalho de referência com os modelos treinados neste estudo, foram criados novos conjuntos de dados com imagens reais para testar os modelos. Os testes realizados com o banco de dados composto por imagens reais demonstraram que a abordagem de retirada de fundo teve um melhor resultado em comparação ao modelo de referência, utilizando os mesmos parâmetros. Com a remoção do fundo, obtivemos uma acurácia de 47,02% para o problema com todas as 34 classes (contra 21,05% alcançada pelo modelo de referência), chegando a 60,34% em uma base com um menor número de classes.

Palavras-chave: Remoção do fundo, Classificador de imagens, Patologias vegetais, Rede Neural Convoluvional, Rede Neural U^2-net.

ABSTRACT

This work focuses on improving a classifier for plant pathologies, aiming for its practical application in real environments, such as the field. The main objective is to investigate the impact of background removal from images on the accuracy of a plant disease classifier developed with convolutional neural networks. To achieve this objective, an algorithm was developed using the U^2-net neural network, designed to perform efficient image segmentation by removing the background and highlighting the object of interest, in this case, the leaf. In addition to replicating and comparing the reference work with the models trained in this study, new datasets with real images were created to test the models. Tests conducted with the database composed of real images showed that the background removal approach achieved better results compared to the reference model, using the same parameters. With background removal, We achieved an accuracy of 47.02% for the problem with all 34 classes (compared to 21.05% reached by the reference model), reaching 60.34% in a dataset with a smaller number of classes.

Keywords: Background removal, Image classification, Plant pathologies, Convolutional Neural Network (CNN), U^2-net Neural Network.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSARY

PIB	Produto Interno Bruto
TCC	Trabalho de Conclusão de Curso
ML	*Machine Learning*
IA	Inteligência Artificial
ReLU	*Rectifed Linear Unit*
EQM	Erro Quadrático Médio
RGB	Red-Green-Blue
CNN	*Convolutional Neural Networks*
SOD	*Salient Object Detection*
RSU	*ReSidual U-block*
SSH	*Secure Shell*
API	*Application Programming Interface*
SDK	*Software Development Kit*
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*

CAPÍTULO 1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO DO TEMA

O agronegócio, cujo conceito abrange agropecuária, agroindústria, insumos, distribuição e outros serviços, é um dos setores com maior importância e influência para o Brasil, se não o mais importante. Estima-se que em 2023 o agronegócio foi responsável por, aproximadamente, 25% do Produto Interno Bruto (PIB) brasileiro, e por empregar diretamente 20% dos brasileiros. (PAULO, 2023)

Em 2022, foram destinados, aproximadamente, 41,141 milhões de hectares para o cultivo de soja, o que corresponde a aproximadamente 5% do território nacional. No entanto, a relação entre a área colhida e a área destinada ao cultivo de soja resultou em um déficit de mais de 240.000 hectares, evidenciando perdas significativas nessa cultura. Apesar dessas perdas, o Brasil liderou a produção mundial de soja na safra 2022/23, sendo responsável por 42% da produção mundial utilizando apenas 32% da área mundial destinada ao cultivo de soja. Essa eficiência coloca a produtividade da soja brasileira muito acima da média mundial. (PAULO, 2023) (EMBRAPA, 2023)

O agronegócio desempenha um papel crucial tanto no mercado interno quanto no comércio exterior brasileiro. Em 2023, segundo o Comex Stat, um software do Ministério do Desenvolvimento, Indústria e Comércio, o Brasil exportou 310 bilhões de dólares, sendo que, de acordo com o AgroStat, outro software do Ministério de Agricultura e Pecuária, o valor de exportação do agronegócio foi de 153 Bilhões de dólares. Praticamente 50% de todas exportações brasileiras foram de produtos do agronegócio, sendo a soja responsável por 41,96% de todas as exportações agrícolas brasileiras. A China foi o país com maior participação da exportações agropecuárias brasileiras com 36,5% de participação. (INDúSTRIA, 2023) (AGRICULTURA, 2023)

Com o crescimento da população mundial, que atualmente está em 8 bilhões de pessoas, e com a previsão de alcançar 10 bilhões até 2050, torna-se evidente a tendência e a necessidade de um agronegócio mais produtivo e sustentável. Tanto o setor agropecuário brasileiro quanto global precisam investir em novas tecnologias para atender à crescente demanda por alimentos e para mitigar os desafios impostos por pragas e mudanças climáticas. (AMBIENTE, 2023)

De acordo com uma pesquisa realizada pela Syngenta, os nematoides, um tipo de praga altamente destrutiva, foram encontrados em mais de 90% das amostras analisadas por laboratórios de nematologia em todo o Brasil. A pesquisa alerta que, se não forem implementadas medidas eficazes para controlar a disseminação desses organismos, os produtores brasileiros poderão enfrentar um prejuízo potencial de R$ 860 bilhões nos próximos 10 anos. (SYNGENTA, 2023)

Além disso, estima-se que cerca de 40% das plantações sejam perdidas anualmente devido a pragas e doenças, com uma tendência de aumento na disseminação para novas áreas, impulsionada pelas mudanças climáticas. Essas mudanças exacerbam os desafios enfrentados pelo agronegócio, reforçando a necessidade de estratégias adaptativas e tecnologias inovadoras para garantir a segurança alimentar e a sustentabilidade econômica. (BRASIL, 2023)

Dessa forma, uma parte significativa do custo de produção na lavoura é destinada à prevenção de pragas. Uma das estratégias de prevenção é a aplicação de defensivos agrícolas; outra é a utilização de sementes de alta qualidade, tratadas com fungicidas para proteção contra pragas e doenças. Estima-se que o custo de produção da soja hoje seja em torno de R$ 6.279,00 por hectare, sendo que aproximadamente 10% desses custos são destinados a sementes e 22% ao uso de defensivos. Assim, um terço do custo de produção da soja é alocado para a prevenção de pragas e doenças. (RURAL, 2023)

Além do investimento em defensivos e sementes, os produtores também investem em consultorias de técnicos especializados e experientes para monitorar a evolução da lavoura e, quando necessário, recomendar medidas para eliminar as pragas. No entanto, grande parte dessas consultorias ocorre em intervalos de tempo prolongados. Por serem realizadas presencialmente, muitas vezes tornam-se ineficientes, pois as pragas já podem ter se disseminado e afetado significativamente a lavoura até o momento da intervenção. Como resultado, o produtor pode sofrer grandes prejuízos, não devido à inexperiência ou falta de conhecimento do técnico, mas

sim pela dificuldade de conexão entre o campo e o profissional qualificado em tempo hábil.

Apesar dos grandes investimentos em tecnologias voltadas para o campo, a maioria dessas inovações são softwares de gestão e planejamento de atividades agrícolas. As tecnologias atualmente disponíveis, focadas na detecção e controle de pragas, ainda são limitadas, especialmente do ponto de vista financeiro, como é o caso dos softwares baseados em imagens de satélite e drones. Há, portanto, uma necessidade evidente de um software acessível, fácil de usar e que possa ser utilizado por qualquer produtor, independentemente de sua capacidade financeira ou conhecimento técnico.

1.2 MOTIVAÇÃO

A motivação para este Trabalho de Conclusão de Curso (TCC) surge da minha experiência como produtor rural, aliada à formação em Engenharia Elétrica e a segunda graduação em andamento em Gestão do Agronegócio. O desejo de unir essas áreas de conhecimento e aplicar as tecnologias desenvolvidas na engenharia ao setor agropecuário, um mercado de grande relevância global, repleto de oportunidades com o avanço das inovações no campo, sempre esteve presente na minha vida. Com esse interesse, aliado à necessidade de realizar o TCC, iniciaram-se pesquisas de temas que pudessem integrar essas disciplinas.

Durante a busca por um tema, encontrou-se o Trabalho de Conclusão de Curso de Murilo Pereira Botelho, que abordava uma questão crucial para o agronegócio: a identificação de doenças em plantas. Seu trabalho (BOTELHO, 2023) consistiu no desenvolvimento de um aplicativo de celular que utiliza redes neurais convolucionais para analisar imagens de folhas e identificar a presença de patologias. Esse projeto se alinhou perfeitamente com os objetivos mencionados anteriormente, ao possibilitar uma combinação prática entre os conhecimentos de engenharia e as necessidades do setor agropecuário.

1.3 OBJETIVOS

Ao optar por dar continuidade ao TCC mencionado, verificou-se que a melhor abordagem para aprimorar o trabalho anterior seria focar na retirada do fundo das imagens no conjunto

de dados utilizado. Portanto, não houve modificações da arquitetura, desenvolvida no trabalho prévio. Todo o código relacionado ao treinamento das redes neurais, feita para classificar patologias em plantas, é de autoria do trabalho anterior mencionado. Este trabalho tem os seguintes objetivos:

- Desenvolver um algoritmo eficiente para a retirada de fundo das imagens;
- Aplicar o algoritmo desenvolvido para processar todo o banco de dados;
- Realizar o treinamento do modelo com todas as base de dados;
- Avaliar o desempenho do modelo utilizando diferentes conjuntos de dados;
- Comparar os resultados obtidos com os de trabalhos anteriores, a fim de validar as melhorias implementadas e identificar possíveis áreas de avanço.

1.4 ESTRUTURA DO TEXTO

O trabalho está organizado por capítulo, o primeiro deles é este em questão, que trabalha uma introdução sobre o cenário do agronegócio brasileiro e mundial, motivação de realizar este trabalho e o propósito deste trabalho. Em sequência a esse capítulo teremos os seguintes:

- **Capítulo 2 - Referencial Teórico:** capítulo responsável por informar parte da teoria utilizada neste trabalho;
- **Capítulo 3 - Metodologia:** capítulo responsável por explicar com mais detalhes a metodologia utilizada durante o trabalho;
- **Capítulo 4 - Resultados:** capítulo responsável por apresentar os resultados encontrados e comparar com resultados de trabalhos anteriores;
- **Capítulo 5 - Conclusão:** capítulo responsável por informar a conclusão deste trabalho, baseando-se nos resultados encontrados.

CAPÍTULO 2

FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo explicar os principais conceitos teóricos utilizados para a realização deste trabalho.

2.1 APRENDIZADO DE MÁQUINA

A base deste trabalho está no aprendizado de máquina. Aprendizado de Máquina (ML, do inglês *Machine Learning*) é uma abordagem de programação que utiliza aprendizado estatístico e métodos de otimização para imitar a maneira como os humanos aprendem, visando resultados gradualmente mais precisos em análises estatísticas de bases de dados e na identificação de padrões. ML é considerado uma área da Inteligência Artificial (IA), e suas aplicações estão cada vez mais presentes no cotidiano. Algumas implementações de ML incluem: utilização em carros autônomos, classificação de imagens, recomendação de conteúdo em serviços de streaming, redes sociais e lojas virtuais, aplicação em diagnósticos médicos, análise de risco de crédito em empréstimos, otimização de rotas de entrega, entre diversas outras aplicações. (INFORMATION, 2024) (IBM, 2024)

Existem várias abordagens que permitem que a máquina realize seu aprendizado, cada uma adequada a diferentes propósitos. No entanto, podem-se destacar quatro principais abordagens (BURKOV, 2019):

- Aprendizado Supervisionado: o usuário faz uma rotulação e classificação dos dados inicialmente. Com isso o algoritmo é treinado, buscando entradas associadas com saídas já conhecidas. O algoritmo vai aprender a comparar novas entradas com os resultados já testados e validados pelo usuário, fazendo previsões sobre as novas entradas. Este tipo é mais utilizado para problemas de classificação e de regressão.

- Aprendizado Não Supervisionado: ao contrário do supervisionado, o usuário não faz ne-

nhuma rotulação ou classificação nos dados. Portanto, cabe o algoritmo tentar identificar por conta própria padrões e semelhanças nos dados apresentados. Este tipo é mais utilizado para problemas em que não se tem dados rotulados ou um resultado desejado específico.

- Aprendizado Semi-Supervisionado: é uma mistura dos dois anteriores (Supervisionado e Não supervisionado), nele há uma mescla de dados rotulados e dados não rotulados anteriormente. Logo, o modelo é treinado com dados rotulados e não rotulados, fazendo com que o algoritmo aprenda a rotular dados não rotulados previamente. Esse tipo é muito utilizado quando há uma dificuldade de extrair características relevantes dos dados, uteís para classificação de imagens médicas, como raio-x.

- Aprendizado por Reforço: o usuário determina um objetivo específico ou um melhoramento de desempenho de uma tarefa, dessa forma o algoritmo realiza diversas ações em direção ao objetivo proposto, por meio de tentativa e erro, e recebe feedback para essas ações (recompensas ou punições). Esse tipo de aprendizado é amplamente utilizado em jogos virtuais e sistemas de controle.

Neste trabalho, foi dado foco ao aprendizado supervisionado, seguindo a estratégia aplicada no trabalho prévio.

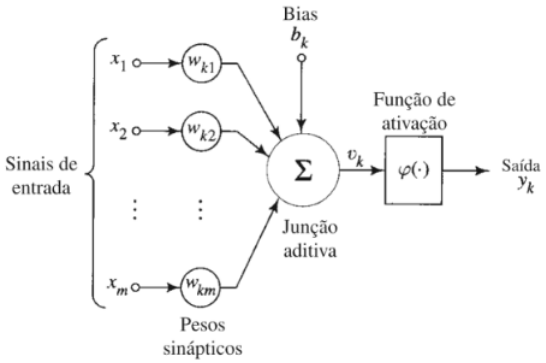
2.2 REDES NEURAIS

As redes neurais são um tipo de ML que emergiram da ideia de que o cérebro humano é um computador altamente complexo e eficiente, capaz de processar informações de maneira substancialmente diferente de um computador convencional. Inspiradas na estrutura e no funcionamento do cérebro humano, as redes neurais são modelos computacionais constituídos por unidades de processamento chamadas neurônios, que mimetizam o comportamento dos neurônios biológicos.

Esses neurônios em redes neurais atuam como as unidades fundamentais de processamento de informações, desempenhando um papel análogo ao dos neurônios no cérebro. A base para o desenvolvimento de redes neurais reside no modelo de um neurônio, conforme ilustrado na

Figura 2.1 abaixo (HAYKIN, 2001).

Figura 2.1: Modelo não linear de um neurônio



Fonte: (BOTELHO, 2023).

Como ilustrado no diagrama da Figura 2.1, o modelo neural é composto por três elementos básicos:

- Conjunto de Sinapses: Cada conjunto é caracterizado por um peso específico. O sinal X_j, que entra na sinapse j conectada ao neurônio k, é multiplicado pelo peso sináptico correspondente W_{kj};
- Somador: Este componente constitui um somador linear, que realiza a soma dos sinais de entrada, ponderados pelos respectivos pesos sinápticos do neurônio;
- Função de Ativação: É responsável por restringir a amplitude de saída de um neurônio, garantindo que a saída permaneça dentro de uma faixa específica de valores.

Além dos elementos mencionados anteriormente, o modelo inclui um viés B_k (em inglês *bias*) aplicado externamente, que tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo se o viés é positivo ou negativo. O modelo pode ser descrito por meio do seguinte par de equações:

$$U_k = \sum_{j=1}^{m} W_{kj} X_j \tag{2.1}$$

$$y_k = \varphi(U_k + B_k) \tag{2.2}$$

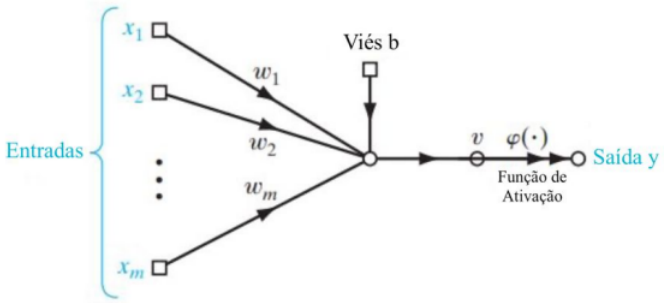
2.2.1 Perceptron

O Perceptron consiste em um único neurônio, sendo a forma mais simples de uma rede neural. Ele é construído em torno de um neurônio não linear. Conforme pode ser observado na Figura 2.2, que representa o diagrama do Perceptron, esta estrutura é semelhante à Figura 2.1. Portanto, o Perceptron é praticamente idêntico ao modelo mencionado anteriormente, com $k = 1$ (pois possui apenas um neurônio). Dessa forma, nas equações 2.1 e 2.2, podemos substituir $k = 1$ e combiná-las para obter a seguinte expressão:

$$y = \varphi\left(\sum_{j=1}^{m} W_j X_j + B\right) \tag{2.3}$$

Em que os parâmetros permanecem os mesmos: Xj representa as entradas aplicadas ao Perceptron, Wj os pesos sinápticos, B o viés, φ a função de ativação e y o valor de saída.

Figura 2.2: Perceptron de Rosenblatt. Sendo: X_m as entradas aplicadas ao perceptron, W_m os pesos sinápticos, B o viés, φ a função de ativação, u o argumento da função de ativação e y_m o valor de saída



Fonte: (BOTELHO, 2023).

Perceptrons são utilizados principalmente para a classificação de conjuntos de estímulos aplicados externamente às entradas Xm, distribuindo-os em uma de duas classes: φ_1 ou φ_2. A regra de decisão utilizada para essa classificação é bastante direta: se a saída y for $+1$, os pontos de entrada X_m são atribuídos à classe φ_1; caso contrário, se a saída do perceptron for $y = -1$, os pontos são atribuídos à classe φ_2. Essa simples regra de classificação permite ao perceptron aprender a separar dados em duas classes distintas, utilizando uma fronteira de decisão linear. (HAYKIN, 2001)

Pórem, por serem simples demais os perceptrons são eficazes apenas para resolver problemas

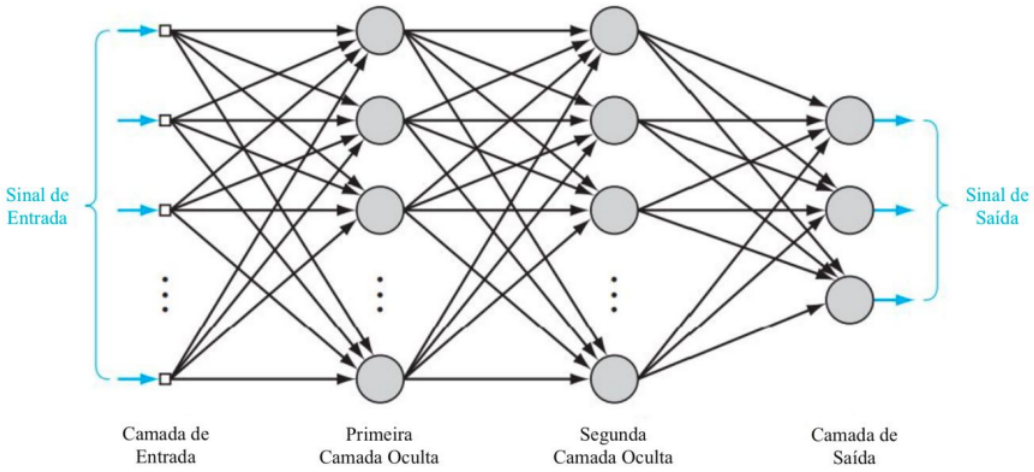
linearmente separáveis, ou seja, aqueles em que as duas classes podem ser divididas por uma linha reta (em duas dimensões) ou por um hiperplano (em dimensões superiores).

2.2.2 Perceptrons Multicamadas

Os perceptrons multicamadas foram desenvolvidos com o objetivo de superar as limitações dos perceptrons de camada única, que são restritos a problemas linearmente separáveis. Sua estrutura é formada por perceptrons simples combinados, introduzindo um novo nível de complexidade, permitindo que a rede neural resolva problemas não lineares. Os perceptrons simples são conectados de forma que cada neurônio de uma camada é ligado a todos os neurônios da camada subsequente.

A característica distintiva dos perceptrons multicamadas em relação aos perceptrons simples é a inclusão de uma ou mais camadas intermediárias, denominadas camadas ocultas. Estas camadas possuem esse nome por não terem contato direto com as entradas ou saídas da rede. Quanto mais camadas intermediárias são adicionadas, maior é o poder computacional de processamento não linear. Como pode ser observado na Figura 2.3, essa nova estrutura de camadas oculta é o que distingue o perceptron multicamadas do simples, além de aumentar a complexidade da rede. (BURKOV, 2019)

Figura 2.3: Perceptron Multicamadas



Fonte: (BOTELHO, 2023).

No perceptron multicamadas, o cálculo do valor de um neurônio em uma camada é realizado

através de uma combinação linear ponderada dos valores dos neurônios da camada anterior, somada a um termo de viés, seguida pela aplicação de uma função de ativação para introduzir não linearidades. A equação que descreve essa operação para um neurônio j na camada L é dada por:

$$y_j^{(L)} = \varphi\left(\sum_{i=1}^{m} w_{ji}^{(L-1)} y_i^{(L-1)} + b_j^{(L)}\right), \tag{2.4}$$

Em que:

- $\mathbf{y}_j^{(L)}$ representa o valor de saída do neurônio j na camada L;
- φ é a função de avaliação aplicada ao valor calculado pela soma ponderada dos valores da camada anterior, ela é reponsável por introduzir não linearidades no modelo. Um exemplo comum é a ReLU (do inglês Rectified Linear Unit), que transforma valores negativos em zero e mantém os valores positivos, o que permite ao modelo acelerar a convergência durante o treinamento;
- $\mathbf{w}_{ji}^{(L-1)}$ representa o peso da conexão entre o neurônio i na camada $L-1$ e o neurônio j na camada L;
- $\mathbf{y}_i^{(L-1)}$ representa o valor de saída do neurônio i na camada anterior $L-1$, servindo de entrada para o neurônio j na camada L;
- $\mathbf{b}_j^{(L)}$ é o viés associado ao neurônio j na camada L.

Para medir a qualidade da função calculada e avaliar o desempenho da rede, podemos calcular a diferença entre o valor esperado e o valor obtido pela rede para cada neurônio em sua camada de saída. Essa diferença é um reflexo do erro que a rede comete ao realizar suas previsões, e é realizada por meio de uma função de custo. Dentre diversas funções de custo, uma das mais utilizadas é o erro quadrático médio (EQM), que calcula a média das diferenças quadráticas entre os valores previstos e os valores reais como podemos verificar na equação abaixo: (BOTELHO, 2023)

$$C(w,b) = \frac{1}{2n}\sum_{x} \|y(x) - d(x)\|^2, \tag{2.5}$$

Em que:

- C representa a função de custo, sendo que w e b são, os já citados, peso da rede e o víes, respectivamente;
- n é o número total de amostras;
- $d(x)$ é o valor previsto da entrada x;
- $y(x)$ é o valor de saída (valor real) da rede de uma entrada x.

Idealmente, quanto menor for o valor da função de custo, melhor será o desempenho da rede, pois isso indica que o erro na previsão está significativamente reduzido. Uma função de custo com valor igual a 0 representaria um cenário ideal, onde as previsões da rede seriam perfeitamente alinhadas com os valores esperados. A partir da interpretação da função de custo, é possível realizar ajustes para promover melhorias, como a modificação dos pesos sinápticos e do viés. Assim, o objetivo é calcular os pesos e o viés da rede de forma a minimizar a função de custo ao máximo. No entanto, como se pode imaginar, calcular cada peso e viés de cada neurônio em uma rede grande e complexa seria inviável. Com isso, a solução veio a partir do algoritmo de descida por gradiente. (BOTELHO, 2023)

2.2.3 Retropropagação

A retropropagação (*backpropagation*) é uma etapa do algoritmo de descida por gradiente. Esta etapa é responsável por calcular o gradiente da função de custo, que é utilizado na descida por gradiente para ajustar os pesos da rede. Esse gradiente é fundamental para determinar a direção do movimento que reduz a função de custo, o que é o objetivo principal. Ele pode ser calculado da seguinte forma: (3BLUE1BROWN, 2024)

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \frac{\partial C}{\partial w^{(2)}} \\ \frac{\partial C}{\partial b^{(2)}} \\ \vdots \end{bmatrix} \tag{2.6}$$

Sendo que:

- $\frac{\partial C}{\partial w^{(L)}}$: é a derivada parcial da função custo de todos os pesos w de uma camada L;
- $\frac{\partial C}{\partial b^{(L)}}$: é a derivada parcial da função custo de todos os viéses b de uma camada L.

A partir do cálculo do gradiente, percebe-se que há uma interdependência entre as camadas, pois o cálculo do gradiente de uma camada está ligado aos gradientes calculados nas camadas subsequentes. O cálculo da função de custo depende da derivada parcial da função de ativação da camada anterior. Portanto, para o cálculo da função de custo em uma camada L, há a dependência da camada $L-1$, uma vez que o valor de um neurônio na camada L é a soma ponderada das ativações $a^{(L-1)}$. A camada L leva em consideração as ativações da camada $L-1$ e o erro propagado da camada L, que fornece o feedback necessário para ajustar os pesos da camada $L+1$. Os pesos $w^{(L+1)}$ são ajustados com base nos pesos $w^{(L)}$.

Para o aprendizado da rede, realiza-se um passo na direção oposta ao gradiente. Este processo, conhecido como descida por gradiente, ajusta os parâmetros continuamente em cada época até que a função de custo se estabilize, atinjindo um nível satisfatório de desempenho. Sendo que época é uma passagem completa por todo o conjunto de dados de treinamento durante o processo de treinamento de um modelo, este número total de épocas é determinado pelo usuário. O tamanho do passo realizado é diretamente proporcional à taxa de aprendizado, além de ser o oposto da direção do gradiente. A taxa de aprendizado é um hiperparâmetro que controla o tamanho dos passos que serão dados durante cada iteração. Quando a taxa de aprendizado é muito alta, o passo pode ultrapassar o ponto mínimo da função de custo, convergindo em um resultado insatisfatório, também conhecido como *overshooting*. Por outro lado, se a taxa de aprendizado for muito pequena, os passos serão dados de forma muito lenta, o que pode ser problemático, pois a rede pode não atingir um ponto favorável dentro do número de épocas disponível. Cada época é uma etapa do treinamento, quando todos os dados de entrada realizam o treinamento.

2.2.4 Descida por Gradiente

Com o objetivo de otimizar a rede, a descida por gradiente faz uso do gradiente previamente calculado para ajustar os pesos da rede de maneira eficiente. Entre os métodos de descida por gradiente, destaca-se o algoritmo de descida por gradiente em mini lotes (*mini batch gradient*

descent), considerado um dos mais eficazes pois combina as vantagens de duas abordagens distintas: a descida por gradiente em lote e a descida por gradiente estocástico.

No algoritmo de descida por gradiente em lote, o processo inicia-se com o cálculo de todos os gradientes da rede, abrangendo o conjunto completo de dados de treinamento. Somente após esse cálculo, são realizadas as atualizações dos parâmetros da rede. No entanto, esse método pode ser muito lento e inviável em termos de memória, especialmente em conjuntos de dados de grande escala. Por outro lado, a descida por gradiente estocástico adota uma abordagem diferente. Nesse método, os gradientes são recalculados antes de cada atualização de parâmetro, e essa atualização é realizada a cada iteração. Contudo, a convergência para o mínimo exato se torna menos estável pois, provavelmente, ultrapassará o ponto mínimo. Portanto, no método de descida por lotes os passos são mais precisos e no método estocástico é mais rápido. (CHOLLET, 2017)

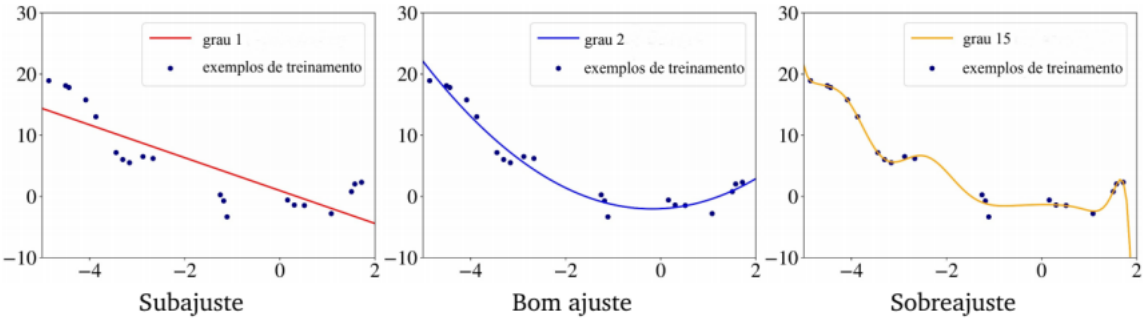
A descida por gradiente em mini lotes combina os aspectos positivos das abordagens anteriores. Nesse método, a atualização dos pesos ocorre a cada iteração, utilizando-se de mini-lotes, que são subconjuntos da base de dados de treinamento. Uma das vantagens desse método é a sua capacidade de proporcionar uma convergência mais estável. Isso ocorre porque o uso de mini lotes reduz a variância nas atualizações dos parâmetros. Além disso, a descida por gradiente em mini-lotes faz um uso mais eficiente da memória.

Um dos hiperparâmetros mais importante é a época, que corresponde a um ciclo completo de treinamento em que o modelo processa o conjunto de dados inteiro de uma vez. Cada época é responsável por realizar todo o processo de treinamento mencionado anteriormente, cálculo dos novos parâmetros, como pesos e vieses, e a avaliação dos erros gerados durante as previsões. Após a conclusão do número determinado de épocas, o modelo atinge o final do processo de treinamento. Idealmente, ao término do treinamento, o modelo deve ter aprendido os melhores pesos e vieses que minimizam a função de custo, resultando no melhor modelo possível. Com esses dados, é possível calcular a acurácia do modelo. A acurácia mede a proporção de previsões corretas em relação ao total de previsões, sendo uma métrica para avaliar a eficácia do treinamento.

Além da época, outro hiperparâmetro que influência na rede é a quantidade de camadas ocultas. Se houver o aumento desnecessário da quantidade de camadas ocultas o tempo de

de treinamento será consideravelmente grande e pode ocorrer um sobreajuste (*overfitting*). Já se o número de camadas ocultas for muito baixo, pode ocorrer um ajuste insuficiente da rede (*underfitting*). Os tipos de ajustes podem ser vistos na Figura 2.4. Quando uma rede utiliza uma grande quantidade de camadas ocultas para seu aprendizado, ele é chamado de aprendizado profundo (*deep learning*). Portanto, o aprendizado profundo refere-se ao uso de redes neurais com múltiplas camadas ocultas, que permitem a modelagem de padrões complexos. Essas redes são chamdas de Redes Neurais Profundas (DNN, do inglês *Deep Neural Networks*).

Figura 2.4: Tipos de ajustes



Fonte: (BOTELHO, 2023).

2.3 REDES NEURAIS CONVOLUCIONAIS

Para processamentos mais complexos, como imagens coloridas de alta resolução com uma grande quantidade de pixels, o número de parâmetros aumenta significativamente, tornando o processo ineficiente e inviável. Isso ocorre porque a entrada é o resultado da multiplicação da resolução pelo número de três canais (*RGB*) e pela quantidade de neurônios. Para resolver esse problema e os problemas de invariâncias no processamento, foram desenvolvidas as redes neurais convolucionais (*Convolutional Neural Networks* - CNNs). Nessas redes, o cálculo realizado é uma convolução com alguns filtros (*kernels*) aplicados aos dados de entrada da rede, neste caso, uma imagem.

A convolução é um operador linear que atua sobre duas funções, resultando em uma terceira função que mede a soma ponderada do produto dessas funções ao longo de uma região de sobreposição. A expressão geral da convolução é dada por:

$$H(i,j) = f * G(i,j) = \sum_a \sum_b f(a,b)G(i-a, j-b) \tag{2.7}$$

Sendo que para esta fórmula, $G(i,j)$ representa a imagem original, $H(i,j)$ representa o valor do novo pixel na imagem resultante e o f é o filtro de convolução (*kernel*).

Para ilustrar melhor o método de convolução, será apresentado um exemplo utilizando uma imagem em preto e branco com baixa resolução para facilitar a compreensão dos conceitos envolvidos. Neste exemplo, um filtro de 3×3 é aplicado, cujos valores são os seguintes:

$$\mathbf{f} = \begin{bmatrix} \frac{1}{9} & \frac{1}{6} & \frac{1}{9} \\ \frac{1}{6} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{9} & \frac{1}{6} & \frac{1}{9} \end{bmatrix} \tag{2.8}$$

A operação de convolução avança de pixel a pixel, aplicando o filtro sobre cada submatriz da imagem original. O cálculo para o primeiro pixel é realizado utilizando a submatriz correspondente:

$$\mathbf{G} = \begin{bmatrix} 180 & 210 & 255 \\ 230 & 140 & 120 \\ 170 & 160 & 80 \end{bmatrix}. \tag{2.9}$$

O cálculo do novo pixel é feito a partir da aplicação do filtro, conforme mostrado na equação 2.8, multiplicado pelos valores do primeiro pixel, como indicado na equação 2.9. O resultado para o novo pixel H(0,0) é:

$$\begin{aligned} \mathbf{H(0,0)} = 180 \cdot \frac{1}{9} + 210 \cdot \frac{1}{6} + 255 \cdot \frac{1}{9} \\ +230 \cdot \frac{1}{6} + 140 \cdot \frac{1}{3} + 120 \cdot \frac{1}{6} \\ +170 \cdot \frac{1}{9} + 160 \cdot \frac{1}{6} + 80 \cdot \frac{1}{9} = 190. \end{aligned}$$

Esse procedimento é iterado para cada pixel da imagem, deslocando o filtro por toda a extensão da matriz original. O cálculo realizado acima pode ser visualizado na Figura 2.5, que demonstra o passo a passo da aplicação do filtro. Após calcular o primeiro pixel, o filtro é então aplicado aos pixels subsequentes, recalculando com base nos valores adjacentes. Abaixo é

apresentado o cálculo do segundo pixel H(1,0), com o processo detalhado na Figura 2.6. Após todas iterações o resultado final está representado na Figura 2.7.

$$\begin{aligned}\mathbf{H(1,0)} = 210 \cdot \frac{1}{9} + 255 \cdot \frac{1}{6} + 250 \cdot \frac{1}{9} \\ +140 \cdot \frac{1}{6} + 120 \cdot \frac{1}{3} + 230 \cdot \frac{1}{6} \\ +160 \cdot \frac{1}{9} + 80 \cdot \frac{1}{6} + 190 \cdot \frac{1}{9} = 182.\end{aligned}$$

Figura 2.5: Processo de convolução da imagem para o primeiro pixel

(a) (b)

Fonte: (BOTELHO, 2023).

Figura 2.6: Processo de convolução da imagem para o segundo pixel

(a) (b)

Fonte: (BOTELHO, 2023).

Figura 2.7: Processo de convolução da imagem para o segundo pixel

(a) (b)

Fonte: (BOTELHO, 2023).

O resultado final da convolução, como ilustrado na Figura 2.7(b), resulta em uma imagem desfocada, o que se deve ao tipo de filtro utilizado, neste caso, um filtro *Gaussian blur*. Esse filtro aplica uma média ponderada dos pixels vizinhos, reduzindo as variações bruscas de intensidade, mas também resulta em uma perda de nitidez geral na imagem. Além do desfoque, a imagem final apresenta uma borda preta e uma ligeira redução em suas dimensões. Essas características decorrem de duas causas principais:

- *Padding*: é uma técnica responsável por fazer o preenchimento de pixels adicionais ao redor da imagem, faz com que as regiões próximas às bordas sejam computadas com valores nulos, frequentemente representados como preto, o que causa a borda preta observada;

- Passos de movimentação: com o objetivo de diminuir a carga de treinamento, as movimentações para os cálculos dos próximos pixels podem ocorrer de maneira mais ampla. No exemplo apresentado, os pixels foram deslocados um a um para a direita, mas, visando otimizar o treinamento, a movimentação pode ocorrer com um avanço (*stride*) maior, pulando mais pixels, em vez de avançar um por um.

Outra técnica utilizada para reduzir o tamanho da imagem e, consequentemente, diminuir o número de parâmetros na rede neural, além de prover alguma invariância ao deslocamento, é o *Pooling*. Essa técnica funciona aplicando uma operação de redução sobre regiões específicas da imagem, o que resulta em uma nova matriz de características reduzida em comparação com a original, auxiliando também no processo de generalização da rede.

Além do *Pooling*, um outro conceito fundamental em redes neurais é o *data augmentation*. A técnica de *data augmentation* visa expandir o banco de dados artificialmente, criando variações das imagens originais através de rotações, espelhamentos, alterações de escala, entre outras transformações. Isso aumenta a quantidade de dados disponíveis para treinamento, ajudando o modelo a generalizar melhor ao expor a rede a uma maior diversidade de exemplos.

2.3.1 Xception

Com o objetivo de aumentar a eficiência das CNNs, foi criado inicialmente o modelo Inception. O grande diferencial desse modelo foi a criação de blocos de camadas com diferentes tamanhos de filtros operando em paralelo, formando diversas ramificações convolucionais que são concatenadas para compor um único conjunto de características, posteriormente passado para a próxima camada. Esses blocos são chamados de módulos Inception. Há duas principais implementações para o modelo Inception: a implementação ingênua e a implementação com redução de dimensionalidade.(IOFFE; SZEGEDY, 2015)

A principal diferença entre as duas abordagens é que na implementação com redução de dimensionalidade há a inclusão de camadas de convolução $1x1$ antes das camadas convolucionais com filtros maiores. Essa adição é utilizada para reduzir a dimensionalidade antes da aplicação de convoluções maiores, tornando o processo mais eficiente e reduzindo o custo computacional.

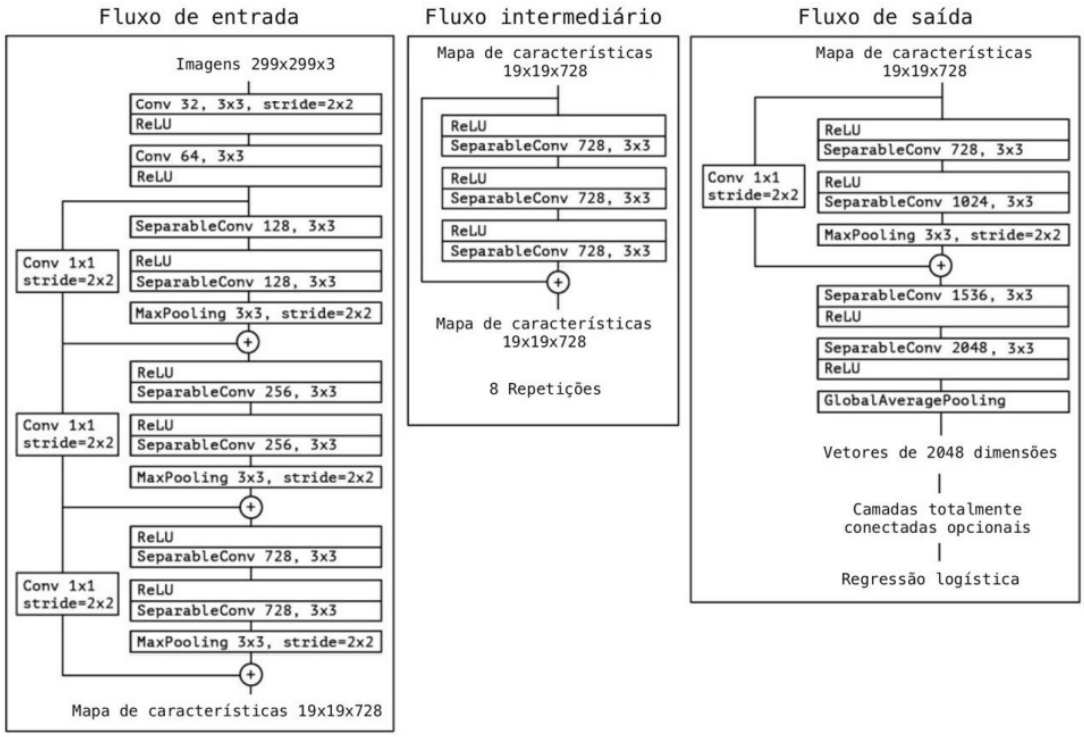
Com a finalidade de melhorar a abordagem do modelo Inception, foi desenvolvido o modelo Xception (*Extreme Inception*). A principal vantagem do Xception em relação ao Inception é a utilização de uma quantidade menor de parâmetros. Diferentemente da implementação do Inception com redução de dimensionalidade, o Xception aplica convoluções de filtros $1x1$ em todas as entradas e, em seguida, aplica convoluções de tamanhos maiores sem sobreposição direta.

A estrutura do modelo Xception, ilustrada na Figura 2.8, é composta por três partes principais: o fluxo de entrada, o fluxo intermediário e o fluxo de saída. O fluxo de entrada prepara os dados para processamento inicial, enquanto o fluxo intermediário realiza a maior parte do trabalho de extração de características com a aplicação sequencial de camadas de *Separable Convolution*. O fluxo de saída, por sua vez, condensa as características extraídas para gerar

a predição final. No modelo há o uso de convoluções separáveis em profundidade (*depthwise separable convolutions*), uma técnica que separa a convolução espacial da convolução por canais, otimizando o processo de aprendizado ao isolar a análise espacial da imagem do ajuste dos filtros por canal.

Além disso, é importante destacar que o Xception faz uso de conexões residuais e da normalização de lotes (*batch normalization*) em sua arquitetura. As conexões residuais permitem que a informação flua através das camadas sem ser excessivamente transformada, mitigando problemas comuns de redes profundas, como o gradiente evanescente. Já a normalização de lotes é aplicada após cada camada de *Separable Convolution* para estabilizar o treinamento, acelerando a convergência do modelo. Além disso, a normalização de lotes ajusta e escala a saída de uma camada de forma a manter as entradas com médias próximas de 0 e o devio padrão próximas de 1 durante o treinamento, ajudando a normalizar as entradas das camadas.

Figura 2.8: Modelo *Xception*



Fonte: (KINGMA; BA, 2017)

2.3.2 Avaliação

Após todo o processo de treinamento da rede descrito acima, é necessário realizar uma avaliação para verificar a qualidade do modelo. Um dos métodos de avaliação é a validação cruzada *K-fold*. Na validação cruzada *K-fold*, todo o banco de dados é dividido em K subconjuntos ou *folds*, mantendo a proporção original das classes para garantir a representatividade de cada subconjunto. O modelo é treinado K vezes, utilizando em cada iteração $k-1$ subconjuntos para o treinamento e o subconjunto restante para a validação. Esse processo é repetido até que cada subconjunto tenha sido utilizado como conjunto de validação exatamente uma vez. A acurácia final do modelo é calculada como a média das acurácias obtidas em cada iteração. Um exemplo de uma validação cruzada *k-fold* com $k = 4$ pode ser vista na Figura 2.9. Neste exemplo, para cada iteração o modelo é treinado com três folds e validado com o fold restante. (HAYKIN, 2009)

Figura 2.9: Exemplo da validação cruzada *k-fold* para $k = 4$

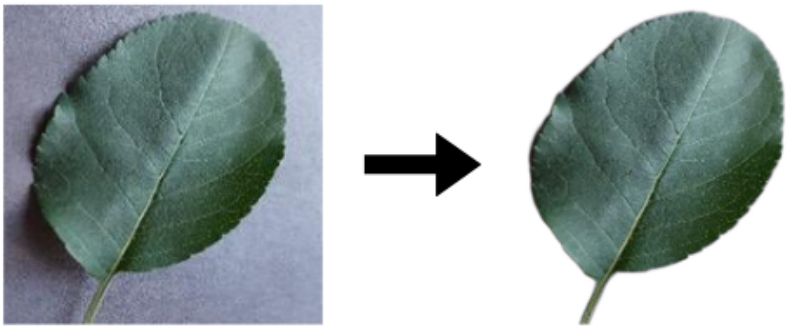
Fonte: Autoria própria.

2.4 REMOÇÃO DE FUNDO

O processo de retirada do fundo de uma imagem pode ser complexo e exigente, especialmente quando se busca alcançar alta eficiência e qualidade no resultado. Esse processo envolve a separação do primeiro plano (geralmente onde está o objeto principal que se deseja destacar) do fundo da imagem. A substituição do fundo por um plano transparente permite que o elemento do primeiro plano se sobressaia, possibilitando sua utilização em novos contextos ou sobreposição a outros fundos.

Com o avanço das redes neurais e a aplicação de aprendizado de máquina na visão computacional, surgiram métodos inovadores para a segmentação automática de imagens. Esses modelos são capazes de aprender características detalhadas das imagens, permitindo uma separação precisa entre o primeiro plano e o fundo. Neste contexto, a biblioteca rembg surge como uma ferramenta eficiente, projetada especificamente para facilitar a remoção de fundos em imagens. O algoritmo da biblioteca utiliza a arquitetura de rede neural profunda U^2-net, desenvolvida com foco na detecção de objetos salientes. Além disso, ela integra a técnica do pacote PyMatting para resolver o problema de *alpha matting*, proporcionando uma separação precisa e eficiente entre o primeiro plano e o fundo. Na imagem 2.10 podemos verificar um exemplo da retirada do fundo de uma das imagens do banco de dados utilizado neste trabalho. (GATIS, 2024)

Figura 2.10: Exemplo da retirada de fundo realizada neste trabalho



Fonte: Autoria própria.

2.4.1 U^2-net

A Detecção de Objetos Salientes (SOD, do inglês *Salient Object Detection*) é uma tarefa do campo da visão computacional e do processamento de imagens, que visa a identificar e destacar objetos mais salientes ou atraentes em uma imagem. Simulando o sistema visual humano, o modelo é capaz de identificar um objeto que se destaca do restante da imagem, de maneira semelhante a como o olho humano "captura"naturalmente uma área ou objeto em uma imagem que chama sua atenção. Dessa forma, a SOD se concentra em encontrar as partes da imagem que são visualmente distintas ou interessantes, como um objeto em primeiro plano, que contrasta com o fundo. Essa técnica tem ampla aplicação em diversos campos, incluindo

rastreamento visual, segmentação de imagens e outras áreas que exigem uma compreensão detalhada da composição visual.

Com a evolução das CNNs, a SOD tem apresentado melhorias significativas. A maioria das arquiteturas de redes voltadas para SOD é extremamente complexa e concentra-se em otimizar o uso dos recursos extraídos por *backbones* existentes. Um *backbone* é uma arquitetura principal de rede neural que serve como base para tarefas mais específicas, como a detecção de objetos. No entanto, ao realizar a extração de características, esses *backbones* muitas vezes não focam em detalhes locais e em informações de contraste global, que são características essenciais para a detecção de objetos salientes.

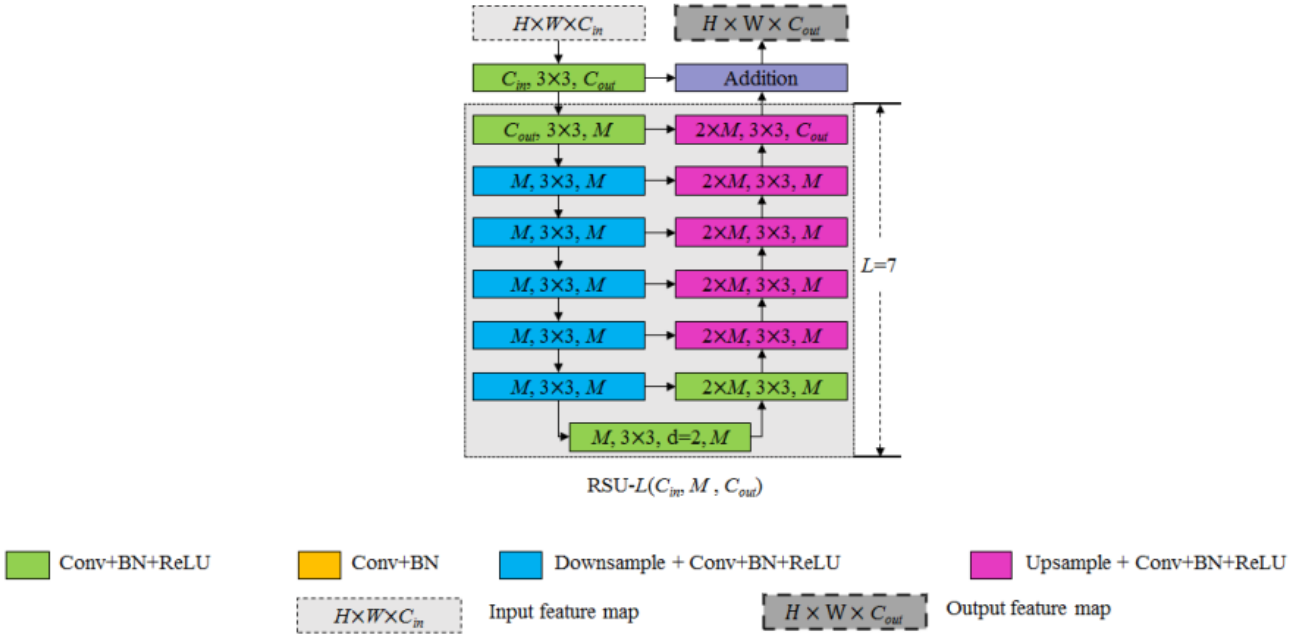
Além disso, esses modelos precisam ser pré-treinados em conjuntos de dados específicos, o que os torna ineficientes quando aplicados a diferentes tipos de dados em relação ao treinamento original. Um desafio comum é que os mapas de características sofrem uma redução de qualidade nas fases iniciais de processamento para permitir a execução do modelo. Os mapas de características são representações intermediárias dentro de uma rede neural que capturam as propriedades extraídas de uma entrada, como uma imagem, enquanto ela é processada pelas camadas da rede. No entanto, uma boa resolução nessas fases iniciais é crucial na segmentação. (BOCHKOVSKIY *et al.*, 2020)

A arquitetura de rede U^2-net foi desenvolvida para abordar as questões descritas anteriormente, sendo projetada especificamente para a SOD sem depender de *backbones* pré-treinados, o que permite seu treinamento desde o início para alcançar um desempenho superior. O U^2-net é estruturado em dois níveis, formando uma configuração U duplamente aninhada. Para essa arquitetura, foi projetado um novo bloco residual U (RSU, don inglês *ReSidual U-block*), capaz de extrair características em uma única etapa sem comprometer a resolução dos mapas de características, especialmente no nível inferior. No nível superior, a estrutura é complementada por um arranjo em que cada etapa é preenchida por um bloco RSU. Essa configuração em dois níveis resulta em uma estrutura U alinhada, que combina as vantagens de ambos os níveis sem aumentar significativamente os custos de memória e computação.

O novo RSU foi proposto com o objetivo de capturar características multiescala em um único estágio, preservando a resolução dos mapas de características. Isso permite que a rede aprofunde suas camadas sem elevar significativamente o custo computacional. A estrutura

do RSU-L (C_{in}, M, C_{out}) pode ser visualizada na Figura 2.11, onde L representa o número de camadas no codificador, C_{in} e C_{out} correspondem aos canais de entrada e saída, respectivamente, e M indica o número de canais nas camadas internas do bloco.

Figura 2.11: Estrutura do RSU utilizado



Fonte: (BOCHKOVSKIY *et al.*, 2020), com modificações.

Assim, o RSU consiste principalmente em três componentes:

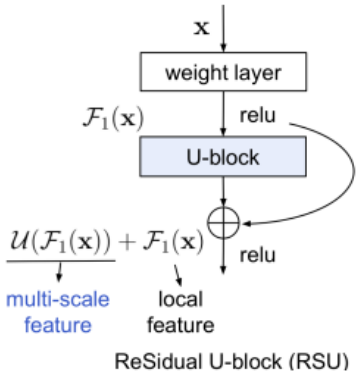
1. Uma camada convolucional simples de entrada, responsável pela extração eficiente de características locais. Ela transforma o mapa de características de entrada, com dimensões (H x W x C_{in}), em um mapa intermediário $F_1(x)$ com canal C_{out};

2. Uma estrutura codificadora-decodificadora simétrica, com altura L. Esta estrutura utiliza o mapa de características intermediário $F_1(x)$ como entrada, aprendendo a exrair e codificar a informação multiescala $U(F_1(x))$. A configuração de um L maior leva a um RSU mais profundo, com mais operações de *pooling*, o que, por sua vez, aumenta o alcance dos campos receptivos e enriquece as características locais e globais, mitigando a perda de detalhes.

3. Uma conexão residual, responsável por fundir as características locais e as características multiescala pela soma: $F_1(x) + U(F_1(x))$.

O principal diferencial de design do RSU, em comparação a outros blocos, é que ele substitui a convolução simples e de fluxo único por uma estrutura semelhante à U-net, substituindo a característica original pela característica local transformada por uma camada de peso. Isso é expresso pela equação $H_{RSU}(\mathrm{x}) = F_1(x) + \mathrm{U}(F_1(x))$, onde U representa a estrutura ilustrada na Figura 2.11. Essas mudanças de design fazem com que a rede extraia características de múltiplas escalas diretamente de cada bloco residual. O design do bloco residual U (RSU) pode ser visto na Figura 2.12.

Figura 2.12: RSU utilizado



Fonte: (BOCHKOVSKIY *et al.*, 2020).

Grande parte das arquiteturas do tipo U-net adota uma estrutura de empilhamento em cascata, conhecida como U×n-net, onde n representa o número de U-nets repetidos. No entanto, essa abordagem em cascata apresenta um aumento exponencial nos custos computacionais e de memória, proporcionais ao valor de n.

A estrutura da U^2-net, por outro lado, utiliza uma formulação distinta: U^n-net, onde a notação exponencial indica uma estrutura em U aninhada em vez de um empilhamento em cascata. Nesta configuração específica, $n = 2$, o que significa que a arquitetura possui duas camadas de aninhamento em U, como pode ser observado na Figura 2.13. O nível superior consiste em uma grande estrutura em U composta por 11 estágios, representados por blocos na Figura 2.13. Cada um desses estágios é preenchido por um bloco residual em U (RSU), configurado de maneira precisa para formar a estrutura em U do nível inferior.

Como destacado na Figura 2.13, a U^2-net consiste em três componentes principais:

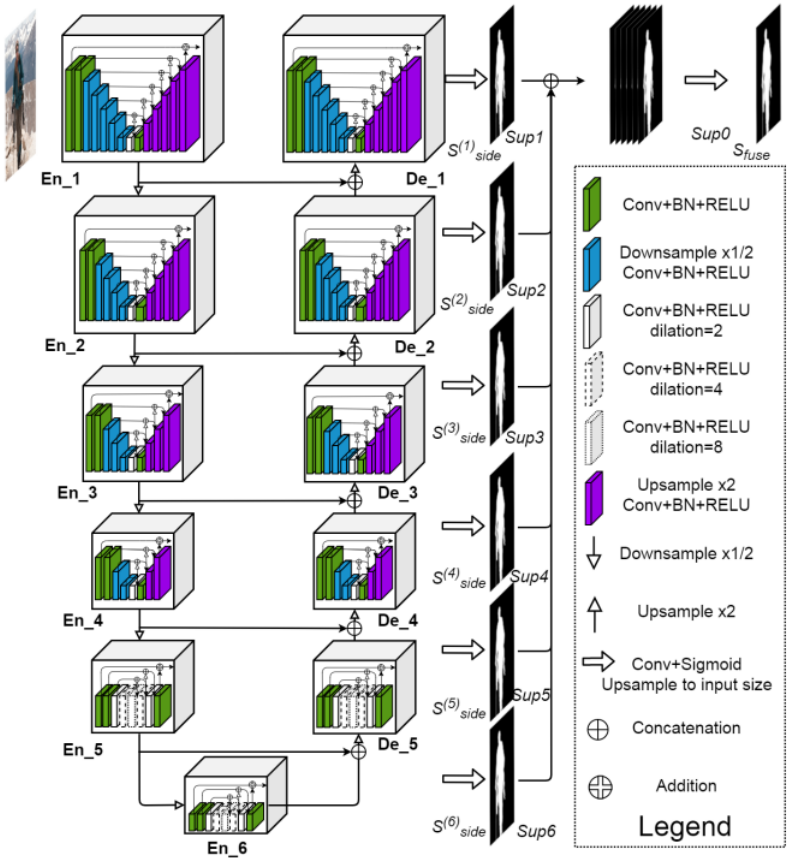
1. Um codificador de seis estágios;

2. Um decodificador de cinco estágios;

3. Um módulo de fusão de mapa de saliência anexado aos estágios do decodificador e ao último estágio do codificador, usando configurações específicas para o codificador e para o decodificador. No módulo de fusão de mapa de saliência são usados para gerar mapas de probabilidade de saliência.

Portanto, a U^2-net é construída exclusivamente com blocos RSU, sem a necessidade de *backbones* pré-treinados, o que a torna altamente flexível e facilmente adaptável a diferentes tarefas, mantendo um desempenho excepcional. A configuração da U^2-net permite uma arquitetura rica em detalhes, com custos de memória e computacionais relativamente baixos, o que a distingue das estruturas tradicionais U×n-net, que requerem maior quantidade de recursos computacionais. Esse design inovador torna a U^2-net uma grande ferramenta para aplicações que exigem alta precisão e eficiência, sem comprometer a flexibilidade ou a capacidade de adaptação a novos desafios.

Figura 2.13: Ilustração da U^2-net utilizado



Fonte: (BOCHKOVSKIY *et al.*, 2020).

2.4.2 PyMatting

Como mencionado anteriormente, o pacote PyMatting tem como objetivo solucionar um problema muito importante para nosso trabalho e para edições de imagens: a separação precisa dos objetos em primeiro plano do fundo da imagem. O *Alpha matting* refere-se ao problema de extrair a máscara de opacidade (*alpha matte*) de um objeto em uma imagem. Essa técnica é essencial para criar transições suaves entre o objeto e o fundo, especialmente em áreas onde as bordas são complexas ou possuem texturas finas. O modelo de *Aplha Matting* é formalizado pela equação de Composição, apresentada abaixo:

$$C_i = \alpha_i F_i + (1 - \alpha i) B_i \tag{2.10}$$

Nesta equação, C_i representa a cor observada no pixel i; F_i refere-se à camada do primeiro plano do pixel i e B_i a camada do segundo plano do mesmo pixel. A variável α_i, que o modelo busca determinar, é a opacidade de cada pixel, representando o nível de mistura entre as duas camadas. Um α_i com valor 1, indica um pixel completamente pertencente ao primeiro plano, enquanto o α_i com valor 0, corresponde a um pixel inteiramente do fundo. Estimar o *alpha matte* é um problema subdeterminado, pois, para cada pixel, há três equações (uma para cada canal de cor) e sete variáveis desconhecidas (FORTE; PITIé, 2020)

A solução oferecida pelo PyMatting baseia-se em métodos que definem um *trimap*, que classifica os pixels em três categorias: primeiro plano ($\alpha = 1$), fundo ($\alpha = 0$) e desconhecidos ($\alpha = ?$). Para a estimativa do alfa, o pacote utiliza 5 métodos:

- Matting de forma fechada: Este método assume a suavidade local das cores dos pixels, obtendo uma solução fechada para o *Alpha matting*; (KRIZHEVSKY, 2010)
- Matting KNN: Baseia-se em informações dos pixels vizinhos mais próximos para derivar soluções de forma fechadas (SZEGEDY *et al.*, 2013) (ZEILER; FERGUS, 2013)
- Matting de Large Kernel: Emprega um algoritmo eficiente baseado em um grande núcleo (kernel) de matrizes de Laplacianas para realizar o Matting; (KRIZHEVSKY, 2010)
- Matting por caminhos aleatórios: Utiliza caminhos aleatórios sobre os pixels para estimar o α. Neste método, o α calculado de um pixel é a probabilidade de que um caminho ale-

atório iniciado a partir desse pixel alcance um pixel do primeiro plano antes de encontrar um pixel do fundo;

- Matting Baseado em Aprendizado: Este método utiliza aprendizado local semi-supervisionado para estimar o α, partindo do princípio de que o valor de α de um pixel pode ser aprendido por meio de uma combinação linear dos pixels vizinhos.

Além das estimativas de alfa, o pacote PyMatting realiza a estimativa do primeiro plano utilizando os seguintes métodos:

Para um determinado α_i, os pixels do primeiro plano F_i podem ser estimados aplicando suavidade adicional em F_i e B_i.

- Estimativa de primeiro plano de Forma Fechada: Para um determinado α_i, os pixels do primeiro plano F_i podem ser estimados aplicando subposições adicionais de suavidade em F_i e B_i. O PyMatting emprega essa abordagem para determinar a fonte do primeiro plano, conforme descrito na literatura

- Estimativa de primeiro plano em Mútiplos Níveis: Além disso, a ferramenta PyMatting implementa uma abordagem inovadora de múltiplos níveis para a estimativa de primeiro plano. Para esse método, o pacote também proporciona implementações otimizadas em GPU para OpenCL e CUDA, aumentando a eficiência computacional do processo.

Neste capítulo, foram apresentados os conceitos que fundamentam o presente estudo, incluindo Machine Learning, Redes Neurais e Redes Neurais Convolucionais, além das abordagens para Remoção de Fundo. A teoria por trás de modelos como o Xception e as técnicas para remoção de fundo, como U^2-net e PyMatting, foram discutidas para demonstrar como essas ferramentas podem ser aplicadas de maneira eficiente no contexto da detecção de doenças em plantas e de remoção de fundo.

Esses fundamentos teóricos fornecem a base necessária para a aplicação das técnicas e algoritmos utilizados no desenvolvimento da metodologia proposta. A partir dessa base conceitual, é possível seguir para a implementação prática e avaliação dos resultados, que serão abordados no próximo capítulo.

CAPÍTULO 3

METODOLOGIA

Este capítulo tem como objetivo descrever os procedimentos utilizados para a realização deste trabalho. Os procedimentos serão descritos em subseções, abrangendo desde a obtenção de dados, processamento dos dados, ferramentas e tenconlogias utilizadas até as dificuldades enfrentadas.

3.1 DADOS DE TREINAMENTO

Os dados de treinamento podem ser divididos em:

1. Dados de treinameto originais: sendo o banco de dados utilizado no Trabalho Original (BOTELHO, 2023)

2. Dados Autorais: sendo o banco de dados utilizado neste trabalho

3.1.1 Dados Originais

O banco de dados original utlizado no primeiro trabalho foi adiquirido através de um artigo publicado na plataforma Mendeley (G.; J., 2017), inicialmente disponibilizado pela organização PlantVillage (HUGHES; SALATHE, 2016). No entanto, atualmente este banco de dados está disponível apenas no artigo mencionado anteriormente. Este conjunto de dados está divido em duas versões: um com *data augmentation* e outro sem.

Ambas classificações contêm 39 diferentes classes de folhas, incluindo uma classe de imagens de fundo aleatórias sem plantas, que é utilizada para aprimorar a capacidade da rede em diferenciar entre um fundo e uma folha. As classes são divididas entre 15 diferentes espécies de plantas e o fundo sem folha, conforme descrito na Tabela 3.1. Como evidenciado no trabalho

prévio, as classes com os nomes originais e as respectivas traduções podem ser vistas na Tabela 3.2, e a quantidade de fotos para cada classe, juntamente com o agente causador da doença, está na Tabela 3.3. A tradução das classes originais para o português foi realizada, principalmente, utilizando o site (AGROLINK, 2023). Os nomes biológicos dos causadores das doenças de cada classe, disponibilizados no artigo da organização PlantVillage (HUGHES; SALATHE, 2016), foram inseridos na ferramenta de pesquisa do site para obter as traduções adequadas.

Tabela 3.1: As 15 diferentes classes de espécies presentes no Banco de imagens

Classe original
Macieira
Cerejeira
Videira
Milho
Videira
Laranjeira
Pessegueiro
Pimentão
Batata
Framboesa
Soja
Abobrinha
Morangueiro
Tomateiro
Fundo sem folha

Fonte: Autoria própria.

Na amostra com *data augmentation* do banco de dados original, são utilizadas seis técnicas diferentes para aumentar o conjunto de dados. As técnicas empregadas são inversão de imagem, correção gama, injeção de ruído, aumento da cor PCA, rotação e dimensionamento das imagens. O banco com *data augmentation* contém 61.486 imagens, enquanto o conjunto sem *data augmentation* possui 55.448 imagens. Todas as imagens têm o tamanho de 256x256 pixels.

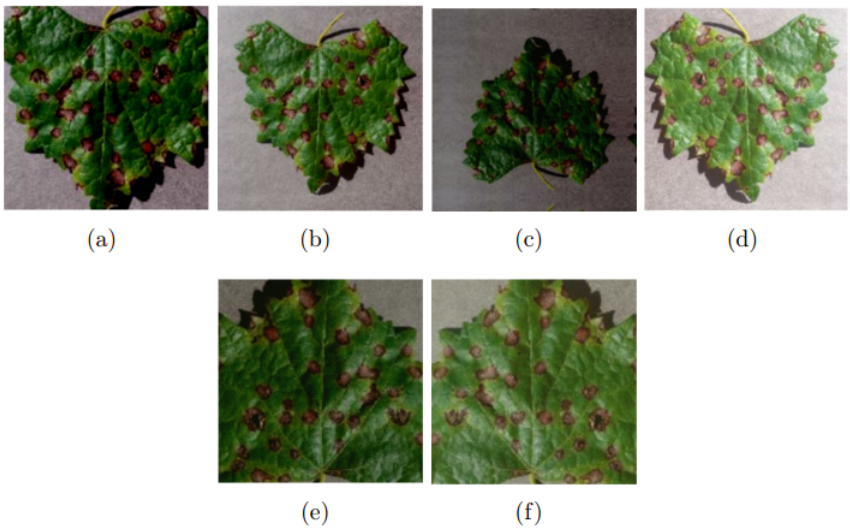
Para o trabalho original, foi utilizado o banco sem *data augmentation*, pois foi decidido

realizar o próprio *data augmentation* durante o treinamento. Isso foi feito mediante a introdução de uma camada de processamento que utiliza a classe Sequential do Keras. Nesta abordagem, a imagem original é a entrada, e a saída é a imagem original transformada pelas técnicas de *data augmentation*, resultando em novas imagens derivadas. As técnicas aplicadas para a transformação no *data augmentation* próprio são descritas a seguir:

- Rotações de 90°, 180° e 270°;
- Ampliações;
- Alteração de brilho;
- Alteração no contraste.

O autor do trabalho anterior escolheu essas transformações e seus respectivos valores por considerar que estas alterações são mais realistas e possuem uma maior probabilidade de ocorrer em imagens reais. A escolha criteriosa das técnicas de *data augmentation* visa aumentar a robustez do modelo, tornando-o capaz de lidar com variações comuns nas imagens do mundo real. Um exemplo de como essas imagens ficaram após a aplicação das transformações é apresentado na Figura 3.1.

Figura 3.1: Exemplos de imagens do banco de dados com o próprio *data augmentation* aplicado



Fonte: (BOTELHO, 2023)

3.1.2 Dados Autorais Utilizados

O banco de dados utilizado para este trabalho teve como base o banco de dados do trabalho anterior. Assim como no trabalho prévio, neste estudo também foi escolhido o conjunto de dados sem *data augmentation* para realizar os treinamentos. A abordagem selecionada envolveu a retirada do fundo das imagens, o que exigiu a remoção do fundo de todas as 55.448 imagens presentes no banco de dados original. Para isso, foi desenvolvido um algoritmo específico para essa finalidade, o qual será detalhado na subseção seguinte. Após a conclusão do processo de retirada do fundo, verificou-se que, apesar do algoritmo ter apresentado um desempenho muito bom, algumas classes não obtiveram resultados satisfatórios. Esses resultados insatisfatórios serão evidenciados e detalhados na seção de Resultados.

A maioria das classes apresentaram resultados extremamente positivos, contudo, optou-se por não utilizar certas classes devido à baixa qualidade dos resultados. Como resultado, a nova base de dados sem fundo passou a ter um total de 49.668 fotos distribuídas em 34 classes, representando uma redução de 5.779 fotos e a retirada de 5 classes. As classes utilizadas e a quantidade respectiva de imagens de cada classe estão listadas na Tabela 3.4.

Tabela 3.4: Classes do banco de dados utilizado e quantidade de imagens.

Nome da Classe	Imagens
Apple_Apple_scab	625
Apple_Black_rot	621
Apple_Cedar_apple_rust	275
Apple_healthy	1645
Background_without_leaves	1143
Blueberry_healthy	1502
Cherry_healthy	853
Cherry_Powdery_mildew	1052
Grape_Black_rot	1173
Grape_Esca_(Black_Measles)	1383
Grape_healthy	423
Grape_Leaf_blight_(Isariopsis_Leaf_Spot)	1076
Orange_Haunglongbing_(Citrus_greening)	5507
Peach_Bacterial_spot	2277
Peach_healthy	359
Pepper_bell_Bacterial_spot	996
Pepper_bell_healthy	1476
Potato_Early_blight	1000
Potato_healthy	152
Potato_Late_blight	1000
Raspberry_healthy	371
Soybean_healthy	5087
Strawberry_healthy	456
Strawberry_Leaf_scorch	1093
Tomato_Bacterial_spot	2127
Tomato_Early_blight	993
Tomato_healthy	1590
Tomato_Late_blight	1885
Tomato_Leaf_Mold	951
Tomato_Septoria_leaf_spot	1771
Tomato_Spider_mites_Two-spotted_spider_mite	1676
Tomato_Target_Spot	1404
Tomato_Tomato_mosaic_virus	373
Tomato_Tomato_Yellow_Leaf_Curl_Virus	5353
34	49.668

No decorrer do desenvolvimento deste trabalho, decidiu-se adotar duas abordagens adicionais, resultando na criação de mais dois bancos de dados oriundos do banco principal sem fundo. Essa decisão visou comparar os resultados utilizando três tipos distintos de bases de dados. A adoção dessas três abordagens diferentes permite uma análise comparativa abrangente dos resultados, possibilitando identificar qual estratégia de construção do conjunto de dados proporciona a melhor desempenho do modelo.

O primeiro banco de dados consistiu em todas as imagens sem fundo de todas as classes, ou seja, o banco mencionado anteriormente nessa seção. Conforme observado na Tabela 3.4, há uma discrepância significativa na quantidade de imagens por classe, com algumas classes possuindo mais de 5.000 imagens, enquanto outras possuem menos de 10% desse valor. Essa variação na quantidade de imagens por classe pode influenciar negativamente a acurácia do modelo devido ao desbalanceamento dos dados.

Para mitigar o problema de desbalanceamento, o segundo conjunto de dados foi elaborado incluindo todas as classes que possuíam pelo menos 1.000 imagens. Dessa forma, a diferença no número de imagens entre as classes foi reduzida, proporcionando um conjunto de dados mais equilibrado. Todas as classes com menos de 1.000 imagens foram eliminadas neste processo. Essa abordagem tem como objetivo verificar se a uniformidade na quantidade de imagens por classe pode melhorar o desempenho do modelo, verificando se o equilibrio na quantidade de dados realmente é crucial e afeta o modelo de aprendizado de máquina. As classes que foram utilizadas nessa base de dados e a quantidade de imagem de cada classe pode ser vista na tabela 3.5. Percebe-se que apesar da diminuição de 13 classes do primeiro banco, houve uma diminuição de apenas 7.448 imagens no total.

O terceiro conjunto de dados foi composto por todas as classes saudáveis do banco de dados utilizado, independentemente da quantidade de imagens por classe. Nesta abordagem, o objetivo foi avaliar a acurácia do algoritmo na diferenciação entre as classes saudáveis, sem a necessidade de distinguir entre classes e as doenças de cada classe. Isso permitiu focar exclusivamente na capacidade do modelo de reconhecer as diferentes classes de plantas sem considerar a presença de doenças. Portanto, o objetivo da criação deste terceiro conjunto de dadeos é testar a habilidade do modelo em classificar plantas sem a complexidade adicional de distinguir doenças. As classes que foram utilizadas nessa base e a quantidade de imagem de

cada classe pode ser vista na tabela 3.6. A diferença de classes deste banco de dados para o primeiro foi maior que a relação do segundo com o primeiro, neste houve uma eliminação de 22 classes e 34.611 imagens.

Tabela 3.5: Classes do 2° Dataset de treinamento e a quantidade de imagens.

Nome da Classe	Quantidade
Apple_healthy	1645
Background_without_leaves	1143
Blueberry_healthy	1502
Cherry_Powdery_mildew	1052
Grape_Black_rot	1173
Grape_Esca_(Black_Measles)	1383
Grape_Leaf_blight_(Isariopsis_Leaf_Spot)	1076
Orange_Haunglongbing_(Citrus_greening)	5507
Peach_Bacterial_spot	2277
Pepper_bell_healthy	1476
Potato_Early_blight	1000
Potato_Late_blight	1000
Soybean_healthy	5087
Strawberry_Leaf_scorch	1093
Tomato_Bacterial_spot	2127
Tomato_healthy	1590
Tomato_Late_blight	1885
Tomato_Septoria_leaf_spot	1771
Tomato_Spider_mites_Two-spotted_spider_mite	1676
Tomato_Target_Spot	1404
Tomato_Tomato_Yellow_Leaf_Curl_Virus	5353
21	42220

Fonte: Autoria própria.

Tabela 3.6: Classes do 3° dataset de treinamento e a quantidade de imagens.

Nome da Classe	Quantidade
Apple_healthy	1645
Background_without_leaves	1143
Blueberry_healthy	1502
Cherry_healthy	853
Grape_healthy	423
Peach_healthy	359
Pepper_bell_healthy	1476
Potato_healthy	152
Raspberry_healthy	371
Soybean_healthy	5087
Strawberry_healthy	456
Tomato_healthy	1590
12	15057

Fonte: Autoria própria.

3.2 PROCESSAMENTO DOS DADOS

Como mencionado anteriormente, a abordagem escolhida foi a retirada do fundo de todas as imagens do banco de dados sem *augmentation*. Para alcançar este objetivo, foram pesquisadas diversas técnicas para a remoção do fundo. O processo de validação da melhor técnica envolveu um método iterativo de tentativa e erro, onde cada algoritmo desenvolvido era testado e comparado com os anteriores para verificar sua eficiência. Sempre que um novo algoritmo demonstrava maior eficiência, ele substituía o anterior.

3.2.1 Algoritmo de retirada do fundo

Optou-se por criar o algoritmo de retirada de fundo separadamente do modelo de treinamento para garantir que quaisquer resultados insatisfatórios ou erros na remoção do fundo fossem detectados e corrigidos antes do início do treinamento. Isso evita que um modelo incor-

retamente treinado comprometa os resultados finais. Portanto, desenvolveu-se um algoritmo em Python para realizar a retirada do fundo de todas as imagens do *dataset*.

O algoritmo utilizado é uma ferramenta avançada para a retirada de fundos em imagens, empregando modelos de aprendizado de máquina baseados em redes neurais para detectar e separar o primeiro plano do fundo. No processo de separação, o modelo recebe a imagem e classifica cada pixel como relevante (1) ou irrelevante (0). Os pixels relevantes são mantidos, enquanto os pixels irrelevantes são alocados ao canal α, transformando-se em transparente. Após esta separação, o fundo original é substituído pelo fundo transparente, composto por todos os pixels irrelevante transformados em transparente, resultando em uma imagem onde o primeiro plano é claramente destacado.

O Algoritmo baseia-se na rede neural U^2-net e no pacote "PyMatting". A U^2-net é uma arquitetura de rede neural profunda projetada especificamente para a detecção de objetos salientes, enquanto a PyMatting fornece ferramentas adicionais para o processamento de imagens. A combinação dessas tecnologias resulta em um método altamente eficiente e preciso para a remoção de fundos em imagens, otimizando o processo de preparação de dados para modelos de aprendizado de máquina. Esta abordagem não só melhora a qualidade das imagens processadas, mas também facilita a análise subsequente, proporcionando resultados mais claros e consistentes. Como pode ser visto na imagem 2.10 O funcionamento detalhado de cada uma dessas tecnologias pode ser visto no Capítulo 2, já os resultados podem ser vistos no Capítulo 4. O algoritmo conta também com o auxílio das seguintes bibliotecas, com suas respectivas funções:

- Da função *remove* da biblioteca *rembg*: auxilia a remoção do fundo
- Da função *image* da biblioteca *PIL*: responsável para manipulação de imagens;
- Biblioteca *easygui*: para exibir diálogos de seleção de pastas;
- Bibliteca *os*: manipulação de caminhos de arquivos e diretórios

3.3 ALGORITMO DE TREINAMENTO

O algoritmo utilizado para o treinamento da CNN foi inicialmente desenvolvido no trabalho anterior, com algumas modificações introduzidas neste trabalho para aprimorar o desempenho e a eficiência do modelo. A maioria do código é de autoria de (BOTELHO, 2023), sendo as alterações realizadas focadas em otimizações específicas.

3.3.1 Estrutura do Modelo

O algoritmo original foi desenvolvido no trabalho anterior, com base no código apresentado no trabalho de referência. Este código utiliza classes da biblioteca Keras e uma versão reduzida do modelo Xception, discutido no Capítulo 2. A versão reduzida do modelo Xception inclui apenas os fluxos de entrada e saída, com a remoção do fluxo intermediário. Esta modificação visa reduzir os custos computacionais associados ao treinamento do modelo, mantendo um desempenho adequado.

Toda esta estrutura do modelo foi desenvolvida no trabalho anterior, e seu texto foi utilizado como base para esta subseção. O fluxo de entrada foi dividido em duas partes. Para a primeira parte do fluxo de entrada, representada na imagem 3.2, as camadas são organizadas na seguinte ordem:

1. Camada de Entrada (*input_ 1*): Esta camada representa os dados das imagens que serão entregues à rede de treinamento. Utiliza a classe "InputLayer"do Keras e possui tanto uma entrada quanto uma saída no formato (em **tuple** do Python) de (256, 256, 3), devido a cada imagem possuir 256x256 de tamanho e três canais (RGB). A escolha deste formato é fundamental para garantir que todas as imagens sejam processadas de maneira uniforme, mantendo a integridade dos dados visuais;

2. Camada de Escalonamento (*rescaling*): Responsável pelo escalonamento dos valores dos pixels de cada canal para a faixa [0, 1]. A normalização dos pixels é crucial para evitar que valores extremos influenciem desproporcionalmente o treinamento do modelo. A saída mantém o mesmo formato da entrada desta camada, utilizando a classe "*rescaling*"para realizar esta operação;

3. Camada de Convolução (*conv2d*): Esta camada realiza a convolução em duas dimensões das imagens, aplicando 128 filtros $3x3$ com um *stride* de valor 2 e um *padding* de valor "same". O *padding* "same"é utilizado para assegurar que todos os pixels da imagem original, com exceção dos pixels pulados pelo *stride*, sejam processados, preenchendo regiões inexistentes conforme necessário. O formato das imagens de saída é calculado pela fórmula:

$$formato\_saida = \frac{formato\_entrada - 1}{stride} + 1 \tag{3.1}$$

Neste caso, o tamanho da imagem de entrada é (256, 256), resultando em um formato de saída de (128, 128) devido ao *stride* de 2, conforme mencionado no Capítulo 2. A camada utiliza 128 filtros, resultando em uma saída com formato (128, 128, 128);

4. Camada de Normalização em Lote (*batch_ normalization*): Esta camada estabiliza a rede aplicando uma transformação que mantém a média de saída próxima a 0 e o desvio padrão próximo a 1. A normalização em lote é essencial para acelerar o treinamento e reduzir a sensibilidade do modelo às inicializações de parâmetros. A saída mantém o formato de entrada: (128, 128, 128).

5. Camada de Ativação (*activation*): Responsável pela aplicação da função de ativação ReLU, que é definida como $ReLU(x) = \max(0,x)$. Esta função retorna zero para qualquer valor de entrada menor que zero e retorna o valor da entrada para valores positivos, permitindo que a rede aprenda representações não lineares. A escolha da função ReLU é motivada por sua capacidade de introduzir não-linearidade ao modelo sem sofrer de problemas como o desaparecimento do gradiente.

Na segunda parte do fluxo de entrada, foi utilizada uma estrutura repetida cinco vezes, variando a quantidade de filtros (64, 128, 256, 512, 1024). As camadas que compõem essa estrutura são detalhadas a seguir, conforme ilustrado na figura 3.3:

1. Camada de Ativação (*activation*): Responsável pela aplicação da função de ativação ReLU. Esta função retorna zero para qualquer valor de entrada menor que zero e retorna o valor da entrada para valores positivos, permitindo que a rede aprenda representações não lineares. A função ReLU é amplamente utilizada devido à sua capacidade de introduzir não-linearidade ao modelo sem sofrer de problemas como o desaparecimento do gradiente;

2. Camada de Convolução Separável (*separable_conv2d*): Realiza a convolução separável por profundidade (*depthwise separable convolution*), uma característica diferencial do modelo Xception. Esta camada de convolução é eficiente, pois divide a convolução em duas partes: a convolução por profundidade e a convolução ponto a ponto, reduzindo significativamente o número de parâmetros e operações computacionais, o que aumenta a eficiência do modelo sem sacrificar a precisão;

3. Camada de Convolução Tradicional (*conv2d_1*): A convolução deve ser configurada de modo que a saída tenha o mesmo formato da camada que será somada a ela no final, servindo como um residual. Esta etapa é essencial para manter a integridade da informação ao longo das camadas, permitindo a soma das características extraídas nas camadas subsequentes;

4. Camada de Pooling Máximo (*max_pooling2d*): Reduz a dimensão dos mapas de características, abstraindo as características mais importantes. O *max pooling* ajuda a reduzir a dimensionalidade dos dados e a controlar o overfitting, selecionando as características mais relevantes e descartando as menos significativas;

5. Camada de Soma (*add*): Realiza a soma das abstrações obtidas na camada de convolução separável com o residual. Esta operação é crucial para implementar a técnica de redes residuais, que facilita a passagem de gradientes e melhora o treinamento de redes muito profundas, mitigando problemas de degradação.

Por fim, no fluxo de saída, os mapas de características são preparados para serem avaliados por meio das seguintes camadas, conforme ilustrado na figura 3.4:

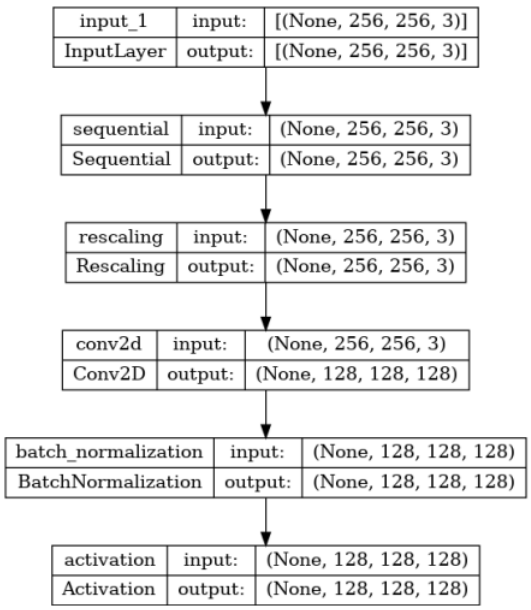
1. Camada de Pooling (*global_average_pooling2d*): Esta camada reduz a dimensão da entrada para um vetor unidimensional, realizando uma média global de cada mapa de característica, retornando apenas um valor para cada. Esta técnica simplifica a representação das características extraídas, facilitando a etapa de classificação final e reduzindo a complexidade computacional;

2. Camada de Regularização (*dropout*): Responsável por prevenir o sobreajuste (overfitting) e melhorar a capacidade de generalização da rede. A camada *dropout* aplica a

regularização, desativando aleatoriamente uma fração das unidades na camada durante o treinamento. Este processo força a rede a aprender características redundantes e robustas, aumentando sua capacidade de generalização;

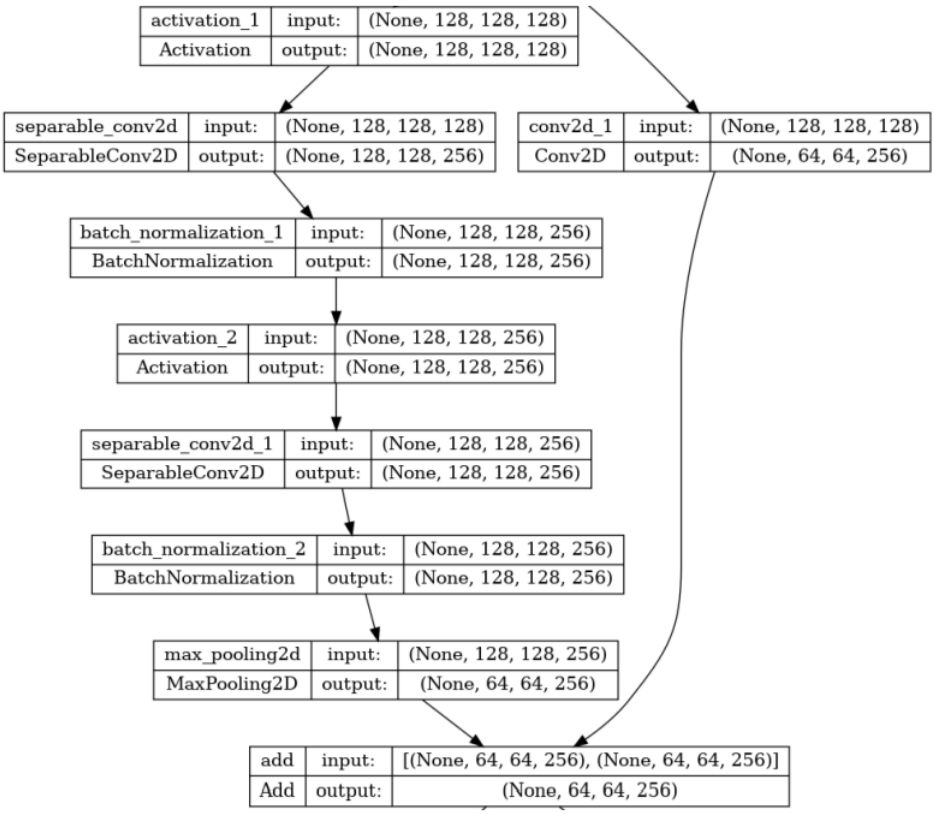
3. Camada Densa (*dense*): Esta camada reduz a entrada para um vetor do mesmo tamanho da quantidade total de classes, em que cada valor do vetor corresponde à probabilidade daquela classe ser a correta. Utiliza a função de ativação *softmax*, que normaliza os valores de saída para que a soma das probabilidades seja igual a 1. Esta camada é crucial para a etapa final de classificação, atribuindo uma probabilidade a cada classe possível.

Figura 3.2: Primeira parte do fluxo de entrada do modelo utilizado



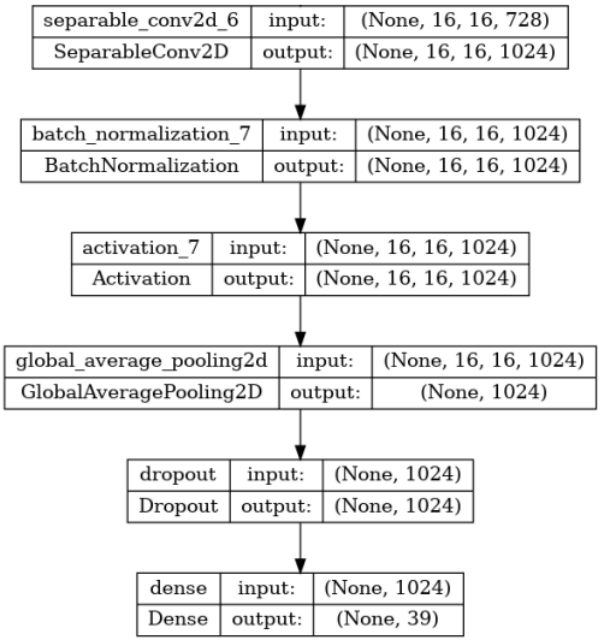
Fonte: (BOTELHO, 2023)

Figura 3.3: Segunda parte do fluxo de entrada do modelo utilizado



Fonte: (BOTELHO, 2023)

Figura 3.4: Fluxo de saída do modelo utilizado



Fonte: (BOTELHO, 2023)

3.3.2 Estrutura do Algoritmo

Grande parte da estrutura total do algoritmo foi desenvolvida no trabalho original, o que não for mencionado como alteração deste trabalho é do trabalho prévio. Inicialmente, o algoritmo envolve a importação de várias bibliotecas essenciais para o seu funcionamento. As seguintes bibliotecas foram importadas, com suas respectivas funções:

- pickle: Utilizada para serialização (processo de converter um objeto em uma sequência de bytes) e desserialização (processo de reverter essa sequência de bytes em objeto) de objetos Python;
- matplotlib.pyplot: Utilizada para verificação gráfica, permitindo a visualização de métricas de desempenho, como curvas de aprendizado e matrizes de confusão;
- tensoflow e keras: Utilizadas para a construção, treinamento e avaliação do modelo;
- numpy: Utilizada para operações matemáticas e manipulação de arrays.

O código original tem como objetivo treinar e avaliar o modelo citado anteriormente, implementando uma abordagem de validação cruzada *K-fold*, que é é uma técnica utilizada para avaliar a capacidade de generalização de um modelo de aprendizado de máquina. Este processo é detalhado no capítulo 2. Para esta abordagem *K-fold*, foi utilizado um valor de k igual a 4, significando que o banco de dados foi dividido em quatro partes iguais. Três partes foram usadas para treinamento e uma para validação, em cada uma das iterações. O embaralhamento dos dados foi realizado para garantir a aleatoriedade das divisões, prevenindo vieses e assegurando que cada subconjunto seja representativo da base de dados total.

Além da validação *K-fold*, o *dataset* foi dividido em três conjuntos: treinamento, validação e teste. Esta divisão seguiu a proporção de 70% dos dados para treinamento, 20% para validação e 10% para teste. O embaralhamento dos dados foi assegurado pela utilização do parâmetro *seed* das funções Keras, garantindo consistência no embaralhamento em todos os casos, o que é crucial para a reprodutibilidade dos resultados.

Para otimizar o treinamento do modelo, foi utilizado o algoritmo Adam (KINGMA; BA, 2017). O Adam combina as vantagens dos algoritmos AdaGrad e RMSProp, ajustando dinamicamente a taxa de aprendizado de cada parâmetro com base em estimativas dos primeiros e

segundos momentos dos gradientes. Este algoritmo utiliza o conceito de momento, permitindo que gradientes passados influenciem os gradientes futuros, o que contribui para uma convergência mais rápida e eficiente. Além disso, adota a ideia do RMSProp de possuir uma taxa de aprendizado distinta para cada peso da rede neural, adaptando-se aos diferentes cenários. Adicionalmente, foi utilizada a entropia cruzada categórica como função de perda. A entropia cruzada categórica mede a diferença entre a distribuição de probabilidade prevista (saída do modelo) e a distribuição real (rótulos verdadeiros). A utilização da entropia cruzada categórica garante que o modelo seja penalizado mais severamente por previsões incorretas, incentivando ajustes que melhorem a precisão geral.

Na abordagem original, foram testados oito modelos diferentes, cada um com hiperparâmetros distintos e todos com uma taxa de *dropout* de 0,5, com o objetivo de identificar qual modelo e quais hiperparâmetros apresentavam o melhor desempenho. Este projeto utilizou o modelo 6 da abordagem original, que foi identificado como o mais eficiente em termos de desempenho. No trabalho anterior, a diferença entre os modelos era apenas os hiperparêmtros utilizados para cada treinamento. Portanto, os hiperparâmetros utilizados para o treinamento do modelo 6 foram:

- Taxa de aprendizado: 0,001;
- Épocas: 100;
- Tamanho do mini lote: 32;
- Utilização do próprio *data augmentation*, citado anteriormente;
- Filtros: [64, 128, 256, 512, 1024].

As alterações feitas neste trabalho no algoritmo original focaram na melhoria da avaliação do modelo. No algoritmo original, era salva apenas a acurácia do último *fold* (3) e somente o modelo treinado correspondente à última acurácia. Para uma verificação mais abrangente de todas as acurácias e para identificar qual *fold* apresentava o melhor desempenho, este trabalho implementou a salvaguarda das acurácias de todos os *folds* e dos respectivos modelos treinados. Agora, são salvas todas as acurácias dos 4 *folds* e os respectivos modelos treinados.

Após a conclusão de todos os *folds*, é gerado um relatório detalhado mostrando todas as acurácias e perdas dos *folds*, indicando qual *fold* obteve o melhor desempenho. Além disso, o relatório calcula a média das acurácias dos *folds* e o desvio padrão dessas acurácias, proporcionando uma análise estatística mais robusta e completa. Este relatório será visto no Capítulo 4.

3.4 CONFIGURAÇÃO COMPUTACIONAL

Para realizar todas as ações deste trabalho, como execução do algoritmo de retirada de fundo e execução dos algoritmos de treinamento, foi utilizado, primeiramente, um computador pessoal com as seguintes especificações:

- CPU: 10th Gen Intel(R) Core(TM) i7-10510U 1.80 GHz com 4 núcleos;
- Memória RAM: 8 GB;
- Memória SSD: 239 GB;
- GPU: Intel(R) UHD Graphics com 3,9 GB de memória

Ao decorrer da realização deste trabalho no computador pessoal, este começou a travar e o tempo de execução tornou-se muito grande. Além disso, a memória SSD estava 90% consumida. Portanto, tornou-se inviável a utilização deste computador para todos os processos.

Com isso, foi utilizado um outro computador com função exclusiva de executar os algoritmos necessários. Este novo computador da marca LeNovo havia sido adquirido recentemente e nunca tinha sido utilizado, um grande benefício pois estava com todas as especificações em níveis de fábrica. Segue as especificações do computador LeNovo utilizado:

- CPU: 13th Gen Intel(R) Core(TM) i7-1355U 1.70 GHz com 10 núcleos
- Memória RAM 16 GB;
- Memória SSD 477 GB;
- GPU: Intel(R) Iris(R) Xe Graphics com 7,8 GB de memória

A melhoria computacional e a funcionilidade exclusiva fizeram com que o tempo de execução dos algoritmos tivesse uma melhoria significativa. Toda a remoção do fundo foi feita neste computador e foi finalizada em alguns dias. Apesar da melhoria, a execução do algoritmo de treinamento ainda era inviável, demorando semanas para executar partes do algortimo. Por isso, a solução foi tentar utilizar estruturas mais robustas.

Conseguimos utilizar a estrutura do Laboratório de Materiais Nanoestruturados Avançados (LaMNA) para usarmos. O LaMNA é um laboratório multiusuário que visa o avanço da área de materiais e microeletrônica em frentes téoricas e experiementais. Trabalhando, também, com temas diversos para o avanço da tencologia. As áreas de Engenharia Biomédica, Sensores, Inteligência Artificial, Células Solares e estudos de tecnologias em geral. Possui um estrutura computacional robusta que conta com 352 threads de processamento em 11 processadores AMD Ryzen9 7950x3D, 704 GB de Memória RAM DDR5, 30 TB de armazenamento, e GPU com uma GeForce RTX 4080 16GB.

Escolheu-se trabalhar com a GPU RTX 4080 16 GB, a qual faz parte do LaMNA-cluster. A escolha de utilizar essa GPU foi baseada na necessidade de alto desempenho gráfico para processamento paralelo, essencial para o treinamento de modelos de aprendizado de máquina. Com o uso da GPU do laboratório, a expectativa é de que tempo de execução do algoritmo de treinamento caia drasticamente.

3.5 DADOS DE TESTE

Apesar de no modelo de treinamento já ter sido realizado um teste e uma validação, decidiu-se realizar uma nova avaliação do modelo, desta vez com imagens diferentes das utilizadas na base de dados de treinamento. Todas as imagens do banco de dados de treinamento foram tiradas em ambientes controlados, com as mesmas iluminações e parâmetros. Nessas imagens, as folhas estavam separadas de outras partes das plantas, estando sozinhas com um fundo controlado. Portanto, realizar uma nova avaliação com imagens de campo é crucial para testar a robustez e a capacidade de generalização do modelo em situações reais. Enquanto o ambiente controlado da base de dados de treinamento proporciona um bom ponto de partida, a validação com imagens de campo é essencial para verificar se o modelo pode ser aplicado eficazmente

em condições variáveis, como diferentes tipos de iluminação, ângulos de captura, presença de outras partes da planta e de fundos diferentes.

A abordagem realizada foi criar uma nova base de dados com as classes utilizadas no treinamento, pesquisando as classes na internet e verificando a fonte de cada imagem para garantir sua confiabilidade. A pesquisa na internet envolveu buscar imagens representativas de cada classe de planta presente no banco de dados original. Foi fundamental assegurar que as imagens fossem provenientes de fontes confiáveis, como bases de dados acadêmicas, sites de instituições agrícolas e outras fontes verificadas. Este processo garantiu que a nova validação fosse realizada com imagens que realmente refletissem condições reais de campo. Embora tenhamos conseguido obter imagens de todas as classes, enfrentamos um desafio significativo: uma quantidade muito pequena de imagens. Essa pequena quantidade de imagens ocorreu pois algumas classes realmente não tinham mais exemplos confiáveis na internet ou por conta de um resultado ruim na retirada de fundo. Na tabela 3.7 estão a quantidade de imagens por classe para a base de dados de teste, que adiquirimos via internet. A classe "*Background_without_leaves*"é nossa classe nula. Todas fontes de cada imagem poderão ser vistas no Apêndice A .

3.6 TESTE

Com o objetivo de avaliar o modelo de treinamento, foi desenvolvido um algoritmo desde o início que pudesse carregar, avaliar e gerar relatórios do modelo previamente treinado. Essa avaliação do modelo com imagens novas é necessária para a teste, pois permite verificar como o modelo se comporta em dados que não foram utilizados durante o treinamento. Além de verificar a precisão do modelo na tarefa para a qual foi treinado, é possível identificar possíveis erros e melhorias cabíveis no código por meio das métricas de avaliação. Algumas métricas são essenciais para fornecer uma visão detalhada do desempenho do modelo, como a acurácia, recall e F1-score. Outras métricas, como a matriz de confusão, são úteis para identificar erros específicos do modelo.

O algoritmo é uma implementação em Python que utiliza as seguintes bibliotecas para suas respectivas funções:

- tensorflow e keras: utilizadas para carregar e avaliar o modelo;

- sklearn.metrics: utilizada para calcular a matriz de confusão e realizar o relatório de classificação, com a acurácia, precisão, recall e F1-score de cada classe e a média total.
- numpy: utilizada para operações numéricas.

Após a importação das bibliotecas necessárias, o modelo treinado é carregado e verificado para garantir que está no caminho especificado. Utilizando os mesmos parâmetros de dados do treinamento, como o redimensionamento das imagens para $256x256$ pixels e o tamanho do lote definido como 32, o novo conjunto de dados é carregado. Esse redimensionamento é crucial para garantir que as imagens estejam no formato esperado pelo modelo, permitindo uma avaliação precisa.

O Banco de dados utilizado para esta avaliação é o mencionado na subseção anterior, contendo imagens que o modelo não viu durante o treinamento. Dessa forma, o algoritmo compara as previsões do modelo com os rótulos reais, calculando as métricas mencionadas anteriormente. Para uma verificação completa do modelo, são realizados dois cálculos de acurácia diferentes neste algoritmo de avaliação:

1. Cálculo da Matriz de Confusão e relatório de classificação por meio da biblioteca "sklearn.metrics":

   A matriz de confusão é uma ferramenta essencial para avaliar a precisão de um modelo de classificação. Utilizando a função "confusion_matrix(y_true, y_pred)"da biblioteca "sklearn.metrics", a matriz de confusão C é calculada de forma que a entrada C_{ij} representa o número de observações conhecidas por pertencerem ao grupo $_i$ e que foram previstas como pertencentes ao grupo $_j$. Em uma classificação binária, a contagem de verdadeiros negativos é C_{00}, falsos negativos é C_{10}, verdadeiros positivos é C_{11} e falsos positivos é C_{01}. Os parâmetros utilizados nesta função são:

   - y_true: Classes verdadeiras (corretas);
   - y_pred: Classes estimados retornados pelo modelo.

   O relatório de classificação é gerado pela função "classification_report(y_true, y_pred, target_names=target_names", responsável por criar um relatório detalhado que exibe as principais métricas de avaliação. Os parâmetros desta função são:

- y_true: Classes verdadeiras (corretas);
- y_pred: Classes estimados retornados pelo modelo;
- target_names: Nomes de exibição correspondentes aos rótulos, na mesma ordem.

Após a entrada dos parâmetros, a função retorna a precisão, o recall, F1-score e "support"de cada classe. Além dessas métricas, ela também fornece a acurácia total, a média macro (média não ponderada por rótulo) e a média ponderada (média ponderada pelo "support"de cada rótulo). As métricas são definidas da seguinte forma:

- Precisão: É a razão $\frac{TP}{TP+FP}$, onde TP é o número de verdadeiros positivos e FP é o número de falsos positivos. A precisão é intuitivamente a capacidade do classificador de não rotular uma amostra negativa como positiva;
- Recall: É a razão $\frac{TP}{TP+FN}$, onde TP é o número de verdadeiros positivos e FN é o número de falsos negativos. O recall é intuitivamente a capacidade do classificador de encontrar todas as amostras positivas;
- F1-score: Pode ser interpretada como uma média harmônica ponderada da precisão e do recall, onde uma pontuação atinge seu melhor valor em 1 e o pior valor em 0;
- Support: É o número de ocorrências de cada classe em y_true (Classes verdadeiras).

2. Calculo da acurácia, através do método "evaluate"da biblioteca "keras":

   Com a função "model.evaluate(ds)", avalia-se o modelo a partir do conjunto de dados "ds". A função recebe como entrada o conjunto de dados de teste "ds", o conjunto de dados da internet visto na subseção anterior. Com isso, o modelo faz previsões sobre o conjunto de dados de teste e compara essas previsões com os rótulos verdadeiros para calcular as métricas definidas. Retornando as métricas: acurácia e perda totais da classificação. Durante a avaliação, o modelo faz previsões sobre o conjunto de dados de teste e compara essas previsões com os rótulos verdadeiros.

A acurácia é calculada contando o número de previsões corretas (onde a previsão do modelo coincide com o rótulo verdadeiro) e dividindo pelo número total de amostras. A perda é uma medida que quantifica a diferença entre as previsões do modelo e os valores reais (verdadeiros).A

função de perda calcula um valor de erro para cada previsão feita pelo modelo. Esses valores de erro são então agregados para produzir um valor final de perda.

Essa redundância no cálculo da acurácia foi feita justamente para validar a acurácia de avaliação do modelo treinado, garantindo que os resultados sejam consistentes e confiáveis. Além disso, cada método de cálculo pode revelar diferentes aspectos do desempenho do modelo, permitindo uma análise mais completa e detalhada. Essa abordagem não só valida a precisão do modelo, mas também ajuda a identificar áreas onde o modelo pode ser melhorado, seja ajustando hiperparâmetros, alterando a arquitetura da rede, ou melhorando o pré-processamento dos dados.

3.7 API

Para a construção da API, foi utilizada a linguagem Python em conjunto com o framework Flask, o que permite criar uma aplicação web capaz de receber requisições HTTP e responder de forma adequada. A API foi desenvolvida com o objetivo de ser um programa que permita ao usuário enviar uma imagem de uma planta possivelmente doente e obter uma resposta sobre a saúde da planta. O sistema identifica e classifica a imagem, determinando se a planta está saudável ou doente e, em caso positivo, qual doença a afeta, com base em um modelo previamente treinado. Além disso, a API fornece ao usuário o nome da classe identificada junto com o nível de confiança da previsão.

A estrutura e o funcionamento da API podem ser descritos nas seguintes etapas. Inicialmente, a API aceita requisições do tipo POST, nas quais o usuário envia uma imagem da planta. Ao receber a imagem, o sistema realiza uma série de processamentos essenciais para garantir a precisão da análise. Primeiramente, a imagem passa por um algoritmo de retirada de fundo, que é o mesmo utilizado durante o treinamento do modelo. Após essa etapa, a imagem é convertida para o formato RGB, assegurando que todas as entradas estejam no mesmo padrão. Em seguida, a imagem é redimensionada para 256x256 pixels, o formato exato esperado pelo modelo utilizado.

Com a imagem devidamente pré-processada, a API utiliza a biblioteca "keras"para carregar o modelo previamente treinado. A imagem é então transformada em um array numérico, que

serve como entrada para o modelo. O modelo processa essa entrada e gera probabilidades associadas a cada classe possível de doença ou saudável. A classe com a maior probabilidade é selecionada como o resultado da classificação, e a confiança dessa previsão é calculada e expressa em porcentagem, proporcionando ao usuário uma noção clara da precisão do resultado obtido.

Finalmente, a API retorna uma resposta no formato JSON, contendo duas informações principais: a classe identificada, que indica qual doença foi detectada na planta, e o nível de confiança, que representa a certeza do modelo em relação à sua previsão. Embora a aplicação, em sua forma atual, não esteja otimizada para cenários de alta demanda ou funcionalidades avançadas, ela serve como uma prova de conceito eficaz e oferece uma base sólida para futuras expansões e aprimoramentos.

3.8 DIFICULDADES ENFRENTADAS

Durante o desenvolvimento deste trabalho, enfrentamos diversas dificuldades que impactaram o andamento do projeto. Primeiramente, tivemos problemas para executar o código no meu computador pessoal. O tempo de execução era excessivo, o que nos levou a migrar para um computador Lenovo, que apresentou uma melhoria no tempo de processamento, mas ainda assim era insuficiente. Foi então que decidimos utilizar o cluster da UnB, processo que consumiu uma quantidade significativa de tempo devido às etapas de migração.

Outra dificuldade foi o acesso ao cluster da UnB, uma vez que foi a primeira vez que utilizamos esse tipo de recurso. Enfrentamos problemas específicos relacionados a permissões de arquivos e à transferência de arquivos para os nós computacionais do cluster. Este processo exigiu um aprendizado e ajustes nos comandos utilizados, e demandaram um tempo considerável. Motivo pelo qual, construi-se um manual de acesso ao cluster.

Além disso, tivemos que verificar manualmente todo o *dataset* sem fundo e remover as imagens que não apresentavam resultados satisfatórios, um processo demorado e minucioso. A dificuldade relacionada ao *dataset* de teste, composto por imagens da internet, também foi significativa. Foi necessário um esforço considerável para encontrar fontes confiáveis e assegurar que as imagens pertenciam às classes corretas, o que demandou um tempo significativo. A maior parte das dificuldades enfrentadas foram em relação a perda de um tempo considerável.

Essas dificuldades não apenas atrasaram o progresso do trabalho, mas também proporcionaram aprendizados e melhorias no processo de desenvolvimento.

Tabela 3.2: Classes do banco de dados, com o nome original e traduzido.

Classe original	Classe traduzida
Apple with scab	Sarna da macieira
Apple with black rot	Macieira com podridão negra
Apple with cedar apple rust	Macieira com ferrugem do cedro
Healthy apple	Macieira saudável
Healthy blueberry	Mirtilo saudável
Cherry with powdery mildew	Cerejeira com oídio
Healthy cherry	Cerejeira saudável
Grape with black rot	Videira com podridão negra
Corn with grey leaf spot	Milho com mancha cinzenta
Corn with common rust	Milho com ferrugem comum
Corn with northern leaf blight	Milho com mancha foliar do norte
Healthy corn	Milho saudável
Grape with black measles	Videira com sarampo negro
Grape with leaf blight	Videira com mancha foliar
Healthy grape	Videira saudável
Orange with Huanglongbing	Laranjeira com Huanglongbing
Peach with bacterial spot	Pessegueiro com cancro bacteriano
Healthy peach	Pessegueiro saudável
Pepper with bacterial spot	Pimentão com cancro bacteriano
Healthy pepper	Pimentão saudável
Potato with early blight	Batata com requeima precoce
Healthy potato	Batata saudável
Potato with late blight	Batata com requeima tardia
Healthy raspberry	Framboesa saudável
Healthy soybean	Soja saudável
Squash with powdery mildew	Abobrinha com oídio
Healthy strawberry	Morangueiro
Strawberry with leaf scorch	Morangueiro com queima de folhas
Tomato with bacterial spot	Tomateiro com mancha bacteriana
Tomato with early blight	Tomateiro com requeima precoce
Healthy tomato	Tomateiro saudável
Tomato with late blight	Tomateiro com requeima tardia
Tomato with leaf mold	Tomateiro com mofo das folhas

Tabela 3.3: Classes do banco de dados traduzidas, o agente causador da doença e a quantidade de imagens.

Classe	Agente	Imagens
Sarna da macieira	Fungo (*Venturia inaequalis*)	630
Macieira com podridão negra	Fungo (*Botryosphaeria obtusa*)	621
Macieira com ferrugem do cedro	Fungo (*Gymnosporangium juniperi-virginianae*)	275
Macieira saudável	-	1645
Mirtilo saudável	-	1502
Cerejeira com oídio	Fungo (*Podosphaera spp*)	1052
Cerejeira saudável	-	854
Milho com mancha cinzenta	Fungo (*Cercospora zeae-maydis*)	513
Milho com ferrugem comum	Fungo (*Puccinia sorghi*)	1192
Milho com mancha foliar do norte	Fungo (*Exserohilum turcicum*)	985
Milho saudável	-	1162
Videira com podridão negra	Fungo (*Guignardia bidwellii.*)	1180
Videira com sarampo negro	Fungo (*Phaeomoniella spp.*)	1383
Videira com mancha foliar	Fungo (*Pseudocercospor a vitis*)	1076
Videira saudável	-	423
Laranjeira com Huanglongbing	Bactéria (*Candidatus Liber ibacter*)	5507
Pessegueiro com cancro bacteriano (mancha bacteriana)	Bactéria (*Xanthomonas campestris*)	1197
Pessegueiro sauável	-	360
Pimentão com cancro bacteriano (mancha bacteriana)	Bactéria (*Xanthomonas campestris*)	997
Pimentão saudável	-	1478
Batata com requeima precoce	Fungo (*Alternaria solani*)	1000
Batata saudável	-	152
Batata com requeima tardia	Mofo (*Phytophthora infestans*)	1000
Framboesa saudável	-	371
Soja saudável	-	5090
Abobrinha com oídio	Fungo (*Erysiphe cichoracearum / Sphaerotheca fuliginea*)	1835
Morangueiro saudável	-	456
Morangueiro com queima de folhas	Fungo (*Diplocarpon earlianum*)	1109
Tomateiro com mancha bacteriana	Bactéria (*Xanthomonas campestris pv. Vesicatoria*)	2127
Tomateiro com requeima precoce	Fungo (*Alternaria solani*)	1000
Tomateiro saudável	-	1591

Tabela 3.7: Classes do banco de dados e a quantidade de imagens de teste.

Nome da Classe	Quantidade
Apple_Apple_scab	6
Apple_Black_rot	3
Apple_Cedar_apple_rust	4
Apple_healthy	4
Background_without_leaves	10
Blueberry_healthy	8
Cherry_healthy	5
Cherry_Powdery_mildew	3
Grape_Black_rot	5
Grape_Esca_(Black_Measles)	4
Grape_healthy	2
Grape_Leaf_blight_(Isariopsis_Leaf_Spot)	3
Orange_Haunglongbing_(Citrus_greening)	6
Peach_Bacterial_spot	4
Peach_healthy	7
Pepper_bell_Bacterial_spot	5
Pepper_bell_healthy	5
Potato_Early_blight	3
Potato_healthy	3
Potato_Late_blight	3
Raspberry_healthy	4
Soybean_healthy	5
Strawberry_healthy	4
Strawberry_Leaf_scorch	4
Tomato_Bacterial_spot	2
Tomato_Early_blight	3
Tomato_healthy	4
Tomato_Late_blight	5
Tomato_Leaf_Mold	5
Tomato_Septoria_leaf_spot	5
Tomato_Spider_mites_Two-spotted_spider_mite	5
Tomato_Target_Spot	4
Tomato_Tomato_mosaic_virus	5
Tomato_Tomato_Yellow_Leaf_Curl_Virus	3
34	151

CAPÍTULO 4

RESULTADOS

Neste capítulo, serão apresentados os resultados deste trabalho, acompanhados de uma análise que contextualiza e explica os dados obtidos. Para uma melhor organização, os modelos foram nomeados de acordo com a utilização de cada conjunto de dados.

Modelo 1: Este modelo foi treinado utilizando o banco de dados que contém todas as 34 classes resultantes da remoção de fundo, conforme descrito na Tabela 3.4;

Modelo 2: Este modelo foi treinado utilizando o banco de dados que contém apenas as classes com mais de 1.000 imagens, conforme descrito na Tabela 3.5;

Modelo 3: Este modelo foi treinado utilizando o banco de dados que contém apenas as classes saudáveis, conforme descrito na Tabela 3.6.

Neste capítulo, todos os modelos, incluindo aqueles desenvolvidos neste trabalho e o modelo original utilizado como referência, foram treinados utilizando os seguintes hiperparâmetros:

Taxa de aprendizado: 0,001;

Épocas: 100;

Tamanho do mini lote: 32;

***Data augmentation* próprio:** Rotações, Ampliações, Alterração de brilho e Alterações de Contraste;

Filtros: 64, 128, 256, 512, 1024;

4.1 RETIRADA DE FUNDO

A remoção do fundo das imagens do banco de dados constitui a principal abordagem deste trabalho, fundamentada na hipótese de que a retirada do fundo pode melhorar a acurácia do modelo, uma vez que elimina elementos que não são relevantes para o processo de classificação e o modelo consegue visualizar apenas o que importa.

Na Figura 4.1, é possível observar o resultado da aplicação do algoritmo de retirada de fundo na imagem 4.1a, com o resultado final evidenciado na imagem 4.1b. Esse exemplo ilustra a eficácia do processo, destacando a capacidade do algoritmo de isolar o objeto principal ao remover o fundo de maneira precisa. Adicionalmente, na Figura 4.2, verifica-se que o algoritmo distinguiu de forma eficaz as camadas RGB, enquanto a camada α, apresentada na Figura 4.2d, demonstrou um reconhecimento eficiente dos pixels de fundo em comparação com os pixels do objeto principal, garantindo uma separação adequada.

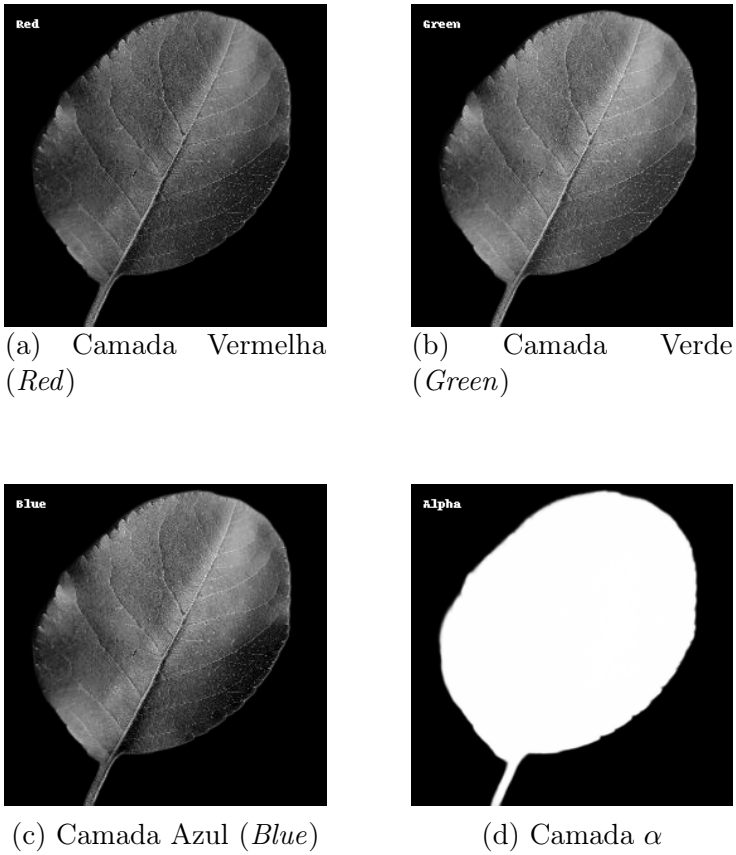
A nova base de dados resultante desse processo de remoção de fundo mantém aproximadamente 90% do total de imagens do banco de dados original. A Tabela 4.1 fornece uma comparação detalhada entre o banco de dados original, utilizado no trabalho anterior, e o banco de dados ajustado com a remoção de fundo, evidenciando as modificações em termos de quantidade das imagens.

Figura 4.1: Exemplo do resultado da retirada do fundo

(a) Imagem Original (b) Imagem sem fundo

Fonte: Autoria própria.

Figura 4.2: Camadas da imagem sem fundo 4.1b



(a) Camada Vermelha (*Red*)

(b) Camada Verde (*Green*)

(c) Camada Azul (*Blue*)

(d) Camada α

Fonte: Autoria própria.

Como destacado anteriormente e detalhado na Tabela 4.1, houve uma redução de aproximadamente 10% no volume da base de dados em comparação com a utilizada no trabalho anterior. Essa redução ocorreu devido a resultados insatisfatórios após a remoção do fundo. Os resultados ruins foram, em sua maioria, decorrentes da forma como as imagens foram capturadas. Nestes casos, os fotógrafos focaram exclusivamente nas áreas afetadas pelas doenças, cortando parte da folha ou enquadrando apenas o foco da patologia, ao invés de capturar a folha inteira. Essa abordagem gerou imagens parciais que comprometeram a capacidade do algoritmo de distinguir adequadamente o fundo do objeto de interesse.

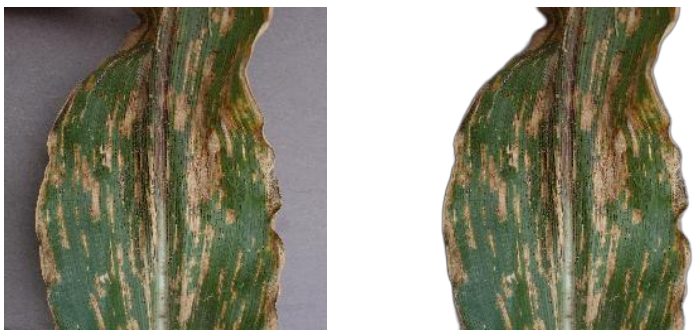
Embora algumas classes que exibiram resultados insatisfatórios tenham sido removidas, isso não implica que todas as imagens dessas classes sem o fundo apresentaram desempenho ruim. De fato, cerca de 60% das imagens dessas classes resultaram em segmentações de boa qualidade, enquanto 40% mostraram falhas consideráveis. A decisão de retirar essas classes foi orientada por duas principais justificativas:

- Primeiramente, visou-se assegurar que o modelo fosse treinado exclusivamente com dados consistentes e de alta qualidade, eliminando a possibilidade de comprometimento da acurácia geral devido a imagens que apresentavam segmentações inadequadas;

- Em segundo lugar, o processo de análise e filtragem manual de cada imagem para decidir quais seriam mantidas no banco de dados seria extremamente demorado e ineficiente. Esse procedimento exigiria uma revisão minuciosa de cada imagem individualmente, o que não só consumiria uma quantidade significativa de tempo, mas também implicaria em um esforço desproporcional para um benefício não tão evidente.

A seguir podem ser vistos alguns exemplo do que foi mencionado anteriormente

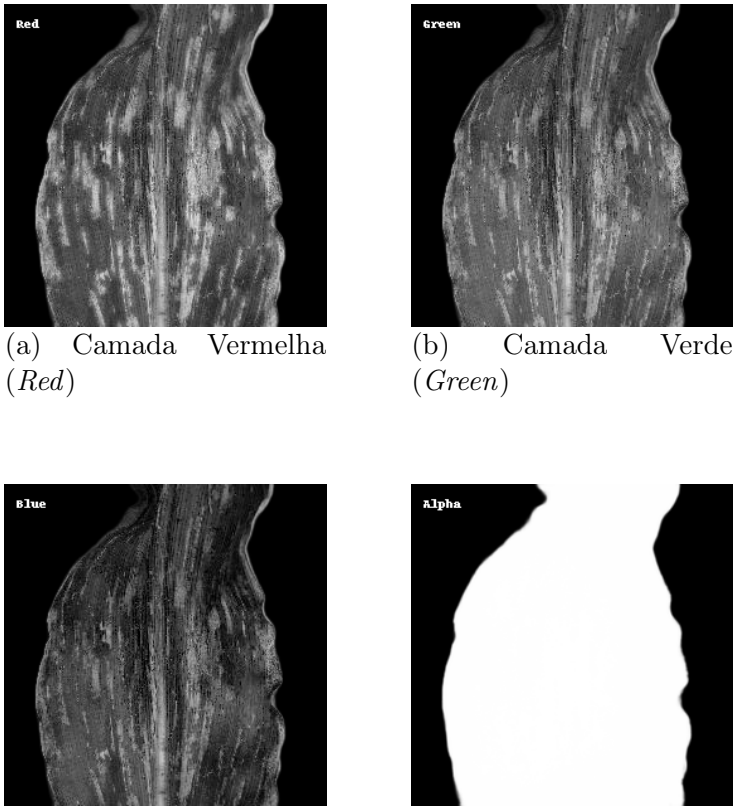
Figura 4.3: Exemplo de resultado satisfatório da classe Corn_Cercospora



(a) Imagem Original (b) Imagem sem fundo

Fonte: Autoria própria.

Figura 4.4: Camadas da imagem sem fundo 4.3b



(a) Camada Vermelha (*Red*)

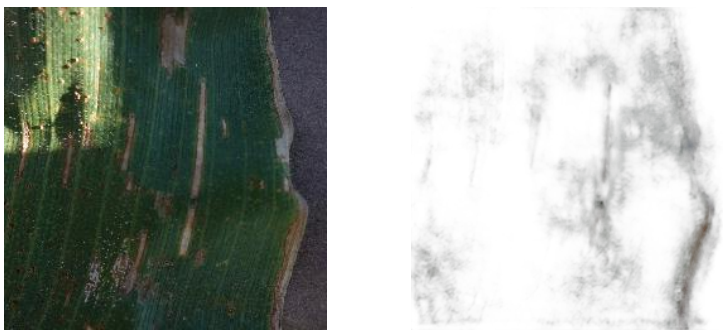
(b) Camada Verde (*Green*)

(c) Camada Azul (*Blue*)

(d) Camada α

Fonte: Autoria própria.

Figura 4.5: Exemplo de resultado insatisfatório da classe Corn_Cercospora

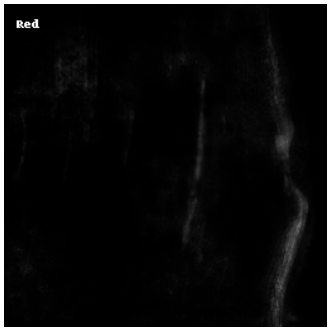


(a) Imagem Original

(b) Imagem sem fundo

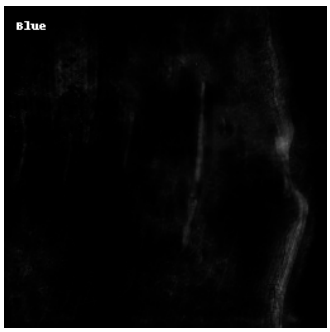
Fonte: Autoria própria.

Figura 4.6: Camadas da imagem sem fundo 4.5b

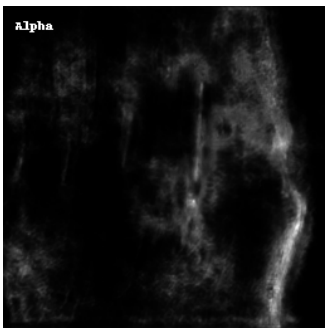


(a) Camada Vermelha (*Red*)

(b) Camada Verde (*Green*)



(c) Camada Azul (*Blue*)



(d) Camada α

Fonte: Autoria própria.

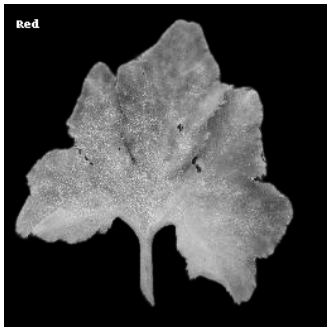
Figura 4.7: Exemplo de resultado satisfatório da classe Squash_Powdery_mildew

(a) Imagem Original

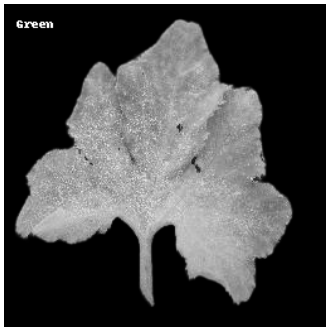
(b) Imagem sem fundo

Fonte: Autoria própria.

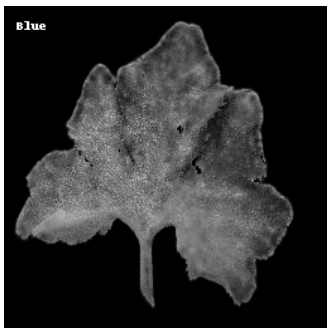
Figura 4.8: Camadas da imagem sem fundo 4.7b



(a) Camada Vermelha (*Red*)



(b) Camada Verde (*Green*)



(c) Camada Azul (*Blue*)

(d) Camada α

Fonte: Autoria própria.

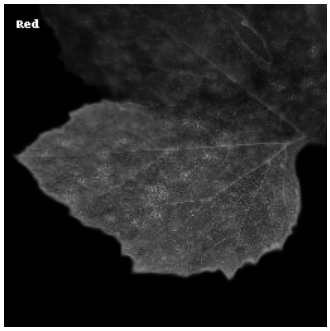
Figura 4.9: Exemplo de resultado insatisfatório da classe Squash_Powdery_mildew

(a) Imagem Original

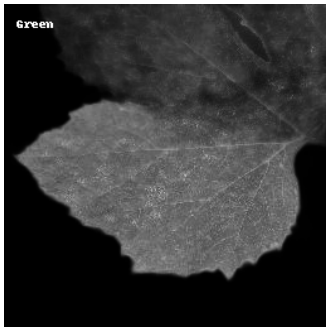
(b) Imagem sem fundo

Fonte: Autoria própria.

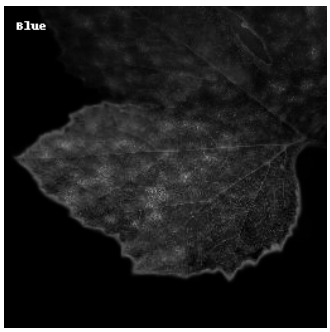
Figura 4.10: Camadas da imagem sem fundo 4.9b



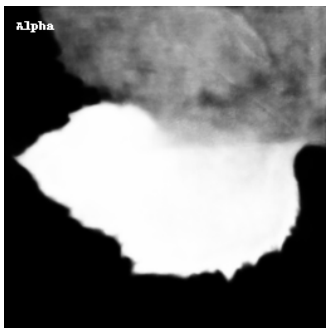
(a) Camada Vermelha (*Red*)



(b) Camada Verde (*Green*)



(c) Camada Azul (*Blue*)



(d) Camada α

Fonte: Autoria própria.

O funcionamento do algoritmo de remoção de fundo baseia-se na separação do primeiro plano do fundo da imagem, com o objetivo de destacar os objetos mais relevantes. Quando a folha é capturada com cortes que interrompem a continuidade visual, especialmente nas bordas, o desempenho do algoritmo é comprometido. Outra dificuldade observada ocorre quando o fundo apresenta uma cor similar à da folha. O modelo depende de padrões visuais consistentes para identificar e segmentar o objeto; cortes abruptos e a semelhança de cores entre o fundo e o objeto principal dificultam essa tarefa, resultando em uma segmentação imprecisa.

Os exemplos apresentados mostram que os problemas mais frequentes ocorrem quando a folha está mal enquadrada, com partes da folha cortando os limites da imagem. Nesses casos, o algoritmo encontra dificuldades em definir claramente onde termina o objeto de interesse e onde começa o fundo, comprometendo a precisão da segmentação. Por outro lado, quando o enquadramento da folha é adequado, mesmo que haja cortes ou a folha não esteja completamente visível, o algoritmo tende a apresentar resultados significativamente melhores.

4.2 TREINAMENTO

O principal objetivo deste trabalho é investigar se a remoção do fundo das imagens contribui para uma melhoria na acurácia da classificação de patologias em plantas. Para isso, foi necessário replicar o algoritmo desenvolvido no trabalho anterior, garantindo uma base de comparação direta entre os modelos com e sem a remoção do fundo. A comparação detalhada das acurácias de treinamento de todos os modelos desenvolvidos neste estudo está apresentada na Figura **??**.

4.2.1 Trabalho Prévio

A replicação do treinamento do modelo foi um passo essencial para garantir a fidelidade na comparação entre os resultados obtidos neste trabalho e os do estudo original. Para isso, seguimos rigorosamente todos os parâmetros e configurações estabelecidos no trabalho anterior, assegurando que as condições experimentais permanecessem iguais. Este treinamento utilizou todas as 39 classes do banco de dados original, sem qualquer alteração nos fundos das imagens.

O modelo utilizado manteve os mesmos hiperparâmetros previamente discutidos no capítulo anterior, preservando as especificações que influenciam diretamente no aprendizado da rede, como a taxa de aprendizado, o número de épocas e o tamanho do lote. A avaliação do modelo foi conduzida por meio de uma validação cruzada *k-fold*, com $k = 4$, uma abordagem que divide o banco de dados em quatro partes, onde, a cada iteração, uma parte é reservada para validação enquanto as outras três são usadas para treinamento. A acurácia final do modelo é a média da acurácia dos folds de treinamento. Neste caso, a acurácia final do modelo de referência foi de 98,04%.

Tabela 4.2: Acurácia do Modelo Original

Modelo Original	Acurácia
Fold 0	98,88%
Fold 1	97,96%
Fold 2	99,06%
Fold 3	96,24%
Final	98,04%

Fonte: Autoria própria.

4.2.2 Modelo 1

Após a replicação do modelo de referência, deu-se início à fase principal deste trabalho, onde foi realizado o treinamento do Modelo 1, que utiliza o banco de dados contendo todas as 34 classes sem o fundo. O treinamento do Modelo 1 manteve os mesmos hiperparâmetros utilizados no modelo de referência e seguiu a mesma abordagem de validação cruzada *k-fold*, garantindo que as condições experimentais fossem equivalentes, para que houvesse uma comparação entre os modelos. A acurácia final do modelo 1, apresentada na Tabela 4.3, foi de 98,17%.

Tabela 4.3: Acurácia do Modelo 1

Modelo 1	Acurácia
Fold 0	98,26%
Fold 1	97,82%
Fold 2	97,62%
Fold 3	98,98%
Final	98,17%

Fonte: Autoria própria.

4.2.3 Modelo 2

Durante a análise do Modelo 1, constatou-se um significativo desbalanceamento na quantidade de dados entre as diferentes classes, o que impactava negativamente o desempenho do

modelo, Para mitigar esse problema, foi adotada uma nova abordagem que consistiu na criação de um novo banco de dados, incluindo apenas as classes com pelo menos 1000 imagens. O Modelo 2 foi treinado com este novo banco de dados, mantendo-se os mesmos hiperparâmetros utilizados anteriormente e empregando a mesma abordagem de validação cruzada *k-fold*. Os resultados, apresentados na Tabela 4.4, indicam que uma acurácia final alcançada pelo Modelo 2 de 98,58%.

Tabela 4.4: Acurácia do Modelo 2

Modelo 2	Acurácia
Fold 0	98,29%
Fold 1	98,07%
Fold 2	98,91%
Fold 3	99,06%
Final	98,58%

Fonte: Autoria própria.

4.2.4 Modelo 3

Diante do sucesso observado na abordagem anterior, que consistiu em selecionar classes com pelo menos 1000 imagens para mitigar problemas de desbalanceamento, decidiu-se explorar uma nova estratégia de simplificação: a utilização exclusiva de classes saudáveis no conjunto de dados. Esta abordagem foi motivada pela hipótese de que, ao focar apenas na distinção entre espécies, sem a necessidade de identificar doenças específicas, o modelo poderia apresentar um desempenho ainda melhor. O novo banco de dados resultante dessa abordagem contém 12 classes, cada uma com diferentes quantidades de dados. Como apresentado na Tabela 4.5, o Modelo 3 alcançou uma acurácia de 98,77%, ligeiramente superior à acurácia do Modelo 2, que era com a maior acurácia, confimando a hipótese desta abordagem.

Tabela 4.5: Acurácia do Modelo 3

Modelo 3	Acurácia
Fold 0	98,29%
Fold 1	98,69%
Fold 2	98,72%
Fold 3	99,39%
Final	98,77%

Fonte: Autoria própria.

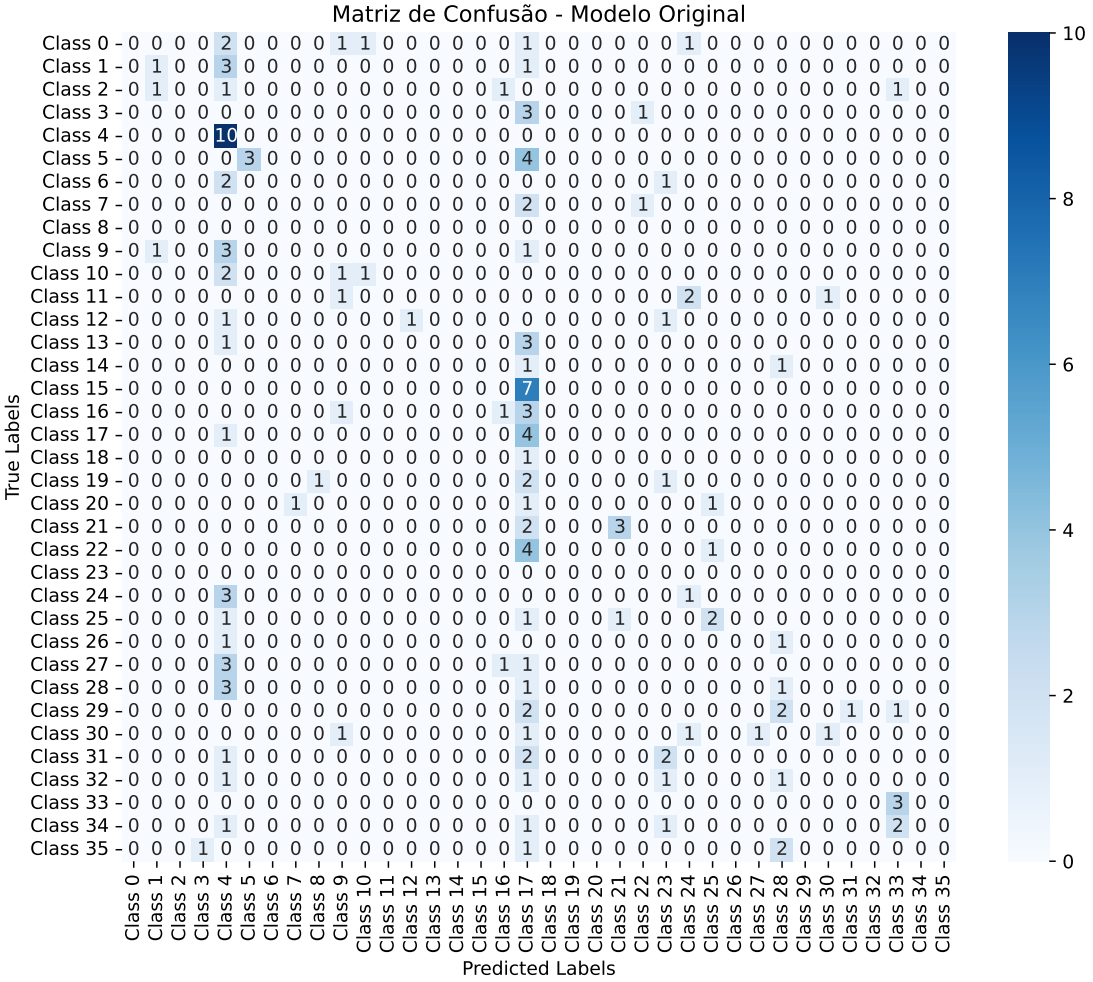
4.3 TESTE

Além de realizar o treinamento de todos os modelos previamente citados, este estudo incluiu uma etapa de teste adicional utilizando um conjunto de dados formado por imagens reais coletadas da internet. Foram coletadas mais de 150 imagens de fontes confiáveis, abrangendo todas as classes utilizadas neste trabalho. A validação com imagens da internet visa avaliar o desempenho dos modelos em condições que se aproximam mais das situações encontradas no campo, onde as variáveis de iluminação, ângulo de captura e variação de fundo não são controladas. As imagens do banco de dados original foram capturadas em ambientes controlados, com iluminação padronizada e condições uniformes, o que pode não refletir com precisão os desafios visuais encontrados em cenários reais. Vale ressaltar que, nesses testes, foi utilizada uma classe nula em cada banco de dados de testes. A classe nula consiste em imagens de fundos sem a presença de plantas.

4.3.1 Trabalho Prévio

No trabalho original, tentou-se avaliar o modelo utilizando imagens reais capturadas na fazenda da UnB. No entanto, os resultados não foram satisfatórios, pois o modelo não conseguiu identificar corretamente nenhuma das imagens testadas. Neste trabalho, foi realizada uma nova avaliação utilizando imagens coletadas da internet, todas mantidas com seus fundos originais, conforme encontradas nas fontes. Essas imagens foram classificadas e utilizadas para a avaliação

Figura 4.11: Matriz de confusão do Modelo Original



Fonte: Autoria Própria

do modelo desenvolvido no trabalho anterior. A acurácia do modelo original, quando avaliado com as imagens da internet, foi de 21,05%, conforme mostrado na Tabela 4.6. Percebe-se, pela matriz de confusão, que este modelo não conseguiu obter um resultado satisfatório, apresentando confusões frequentes entre as classes. A matriz de confusão deste modelo pode ser vista na Figura 4.11

Tabela 4.6: Acurácia de Teste do Modelo Original

Modelo Original	Acurácia
Teste	21,05%

Fonte: Autoria própria.

4.3.2 Modelo 1

Nesta avaliação, as imagens coletadas da internet foram processadas utilizando o algoritmo de retirada de fundo desenvolvido neste trabalho. Após a remoção do fundo de todas as imagens, cada uma delas foi classificada de acordo com as 34 classes do Modelo 1. Em seguida, utilizou-se essas imagens sem fundo para avaliar o desempenho do Modelo 1. Os resultados demonstraram que o Modelo 1, treinado com as imagens sem fundo do banco de dados original, apresentou uma acurácia mais que o dobro em relação ao trabalho original. A acurácia do Modelo 1 na validação com as imagens da internet foi de 47,02%, como mostrado na Tabela 4.7. A partir da análise da matriz de confusão deste modelo, apresentada na Figura 4.12, observa-se que, ao longo da diagonal principal, o modelo apresentou um desempenho relativamente melhor.

Tabela 4.7: Acurácia de Teste do Modelo 1

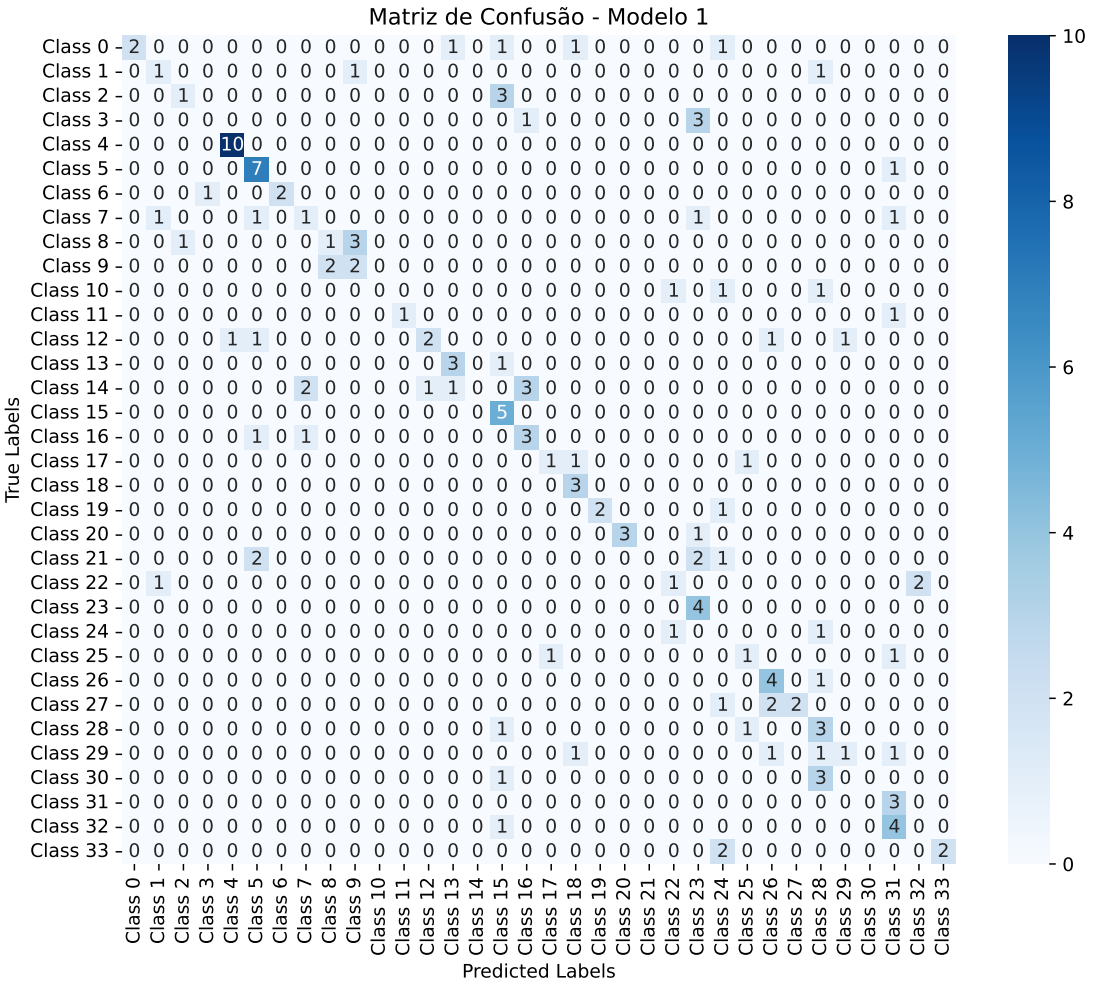
Modelo 1	Acurácia
Teste	47,02%

Fonte: Autoria própria.

4.3.3 Modelo 2

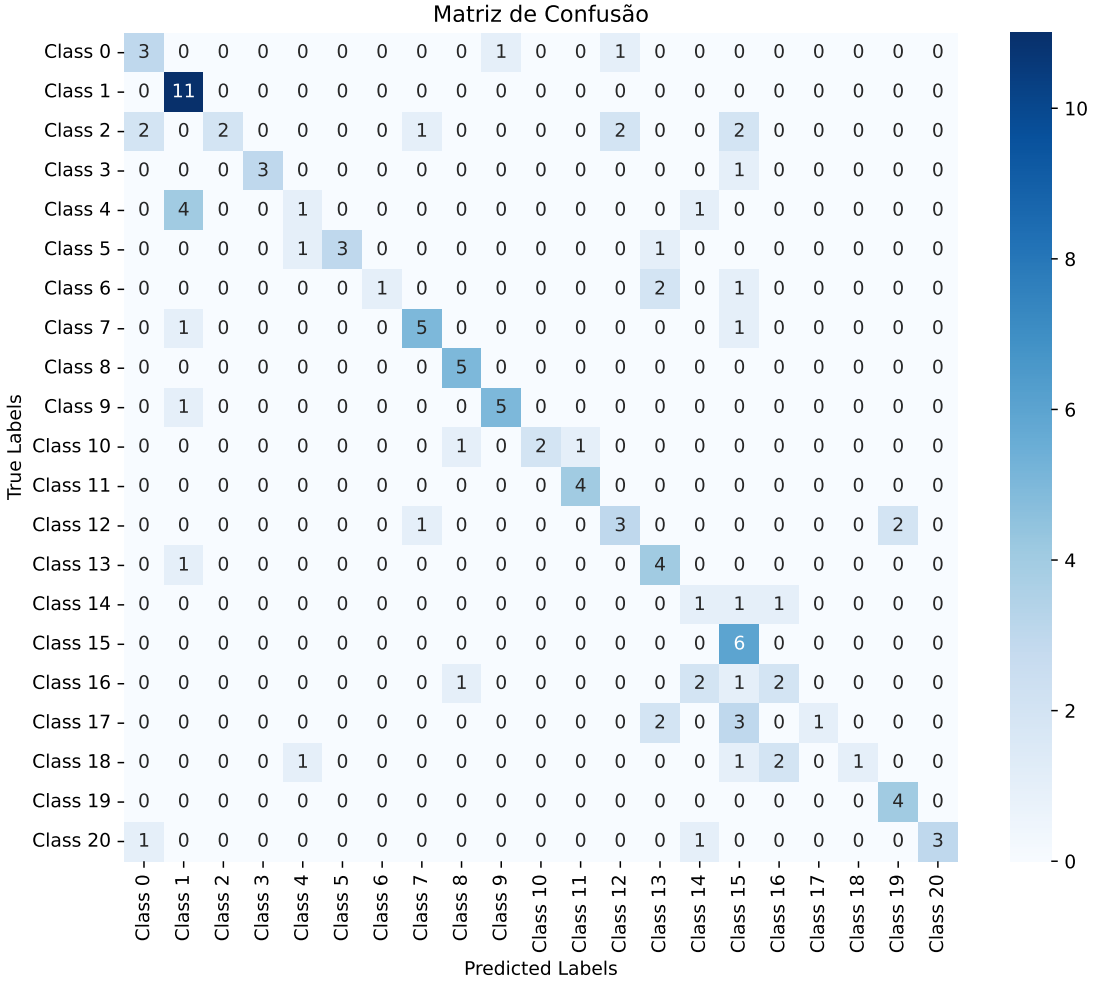
Para esta abordagem, utilizou-se primeiramente banco de dados construído a partir das 34 classes sem fundo retiradas da internet, conforme a avaliação realizada anteriormente, e com isso, foram selecionadas apenas as 12 classes que participaram do treinamento do Modelo 2. Essas 12 classes, derivadas das imagens coletadas da internet, resultaram em um conjunto de 116 imagens para a avaliação. O resultado desta avaliação apresentou um desempenho bastante superior ao alcançado pelo modelo no trabalho original. Especificamente, a acurácia observada foi de 60,34%, evidenciando uma melhoria de aproximadamente três vezes em relação aos resultados originais. A matriz de confusão deste modelo pode ser vista na Figura 4.13.

Figura 4.12: Matriz de confusão do Modelo 1



Fonte: Autoria Própria

Figura 4.13: Matriz de confusão do Modelo 2



Fonte: Autoria Própria

Tabela 4.8: Acurácia de Testeo do Modelo 2

Modelo 2	Acurácia
Teste	60,34%

Fonte: Autoria própria.

4.3.4 Modelo 3

Seguindo a mesma metodologia da avaliação anterior, esta etapa utilizou um banco de dados formado a partir da seleção específica das 12 classes saudáveis dentre as 34 classes totais

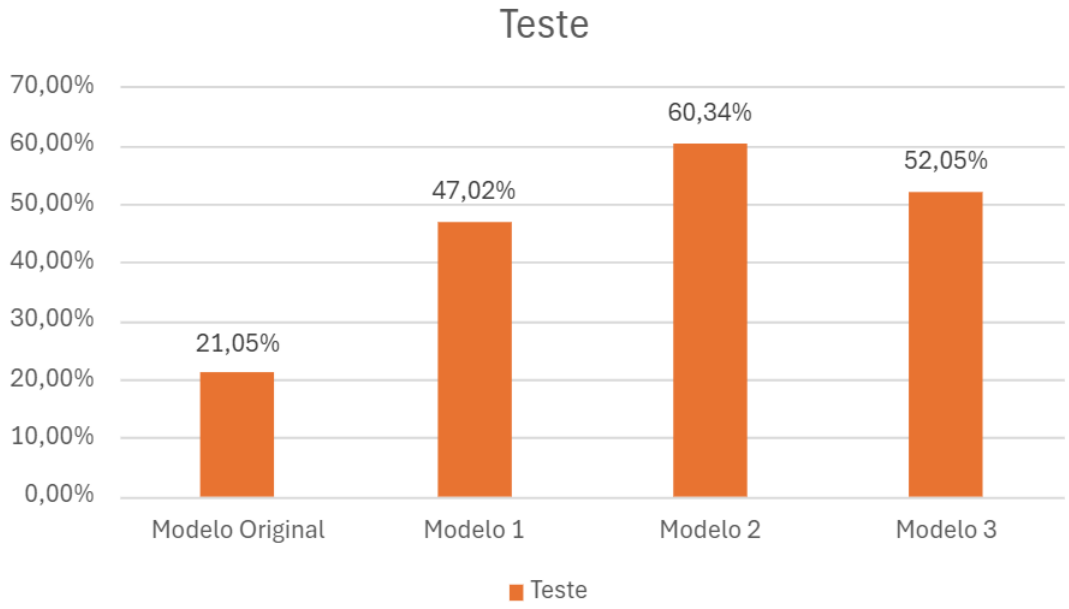
previamente analisadas. Este conjunto de dados das 12 classes saudáveis acumulou um total de 73 imagens. Os resultados desta avaliação evidenciam uma acurácia de 52,05%, conforme apresentado na Tabela 4.9. Este desempenho representa um avanço significativo, sendo quase 2,5 vezes superior ao obtido no trabalho original. A análise da matriz de confusão do modelo, conforme ilustrada na Figura 4.14, revela que os erros de classificação se concentraram em classes que apresentam similaridades físicas nas folhas. Essa semelhança visual entre as classes dificultou a distinção pelo modelo.

Tabela 4.9: Acurácia de Teste do Modelo 3

Modelo 3	Acurácia
Teste	52,05%

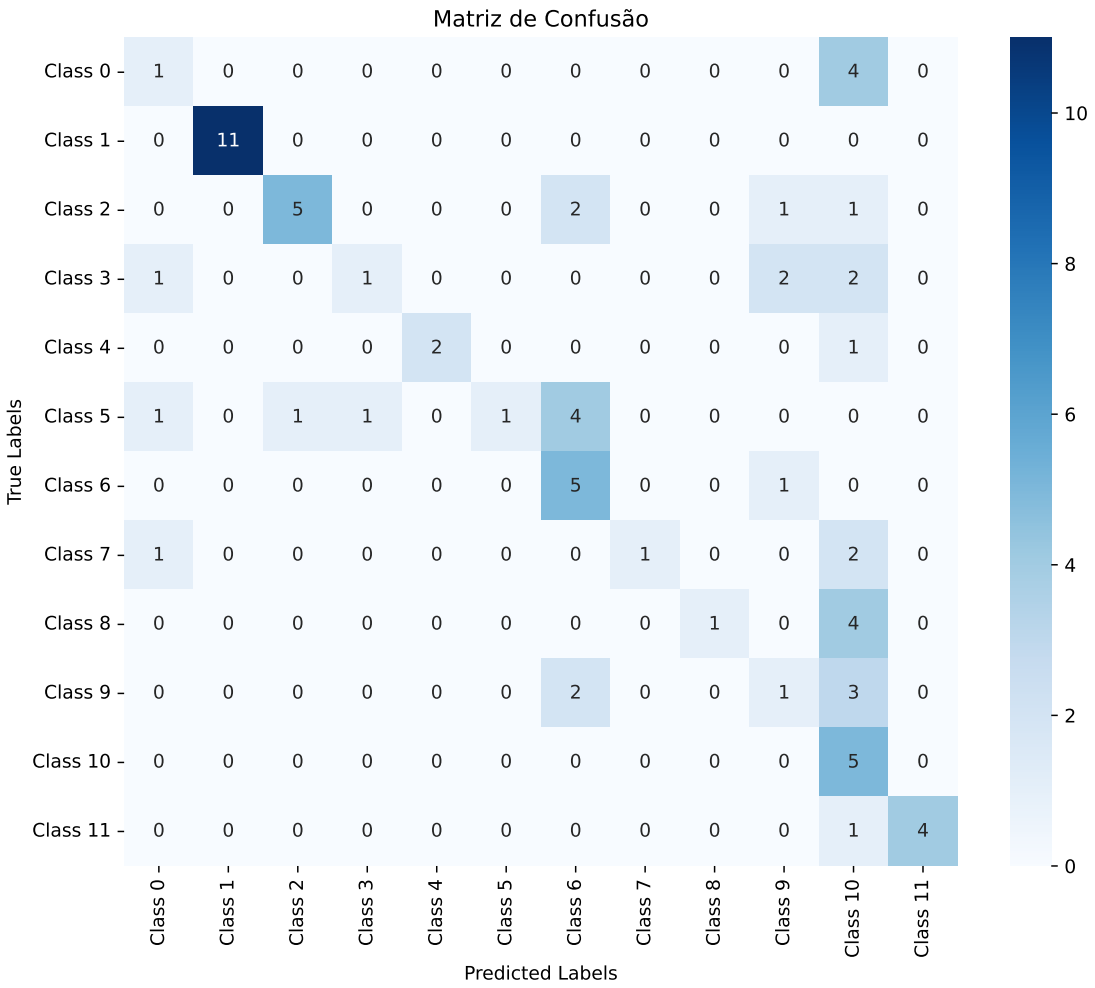
Fonte: Autoria própria.

Figura 4.15: Acurácia de teste de cada Modelo



Fonte: Autoria Própria

Figura 4.14: Matriz de confusão do Modelo 3



Fonte: Autoria Própria

Tabela 4.1: Diferença de quantidade de imagens da Banco de dados original e do novo Banco de dados sem Fundo.

Nome da Classe	Imagens Originais	Imagens Utilizadas	Diferença
Apple_Apple_scab	630	625	5
Apple_Black_rot	621	621	0
Apple_Cedar_apple_rust	275	275	0
Apple_healthy	1645	1645	0
Background_without_leaves	1143	1143	0
Blueberry_healthy	1502	1502	0
Cherry_healthy	854	853	1
Cherry_Powdery_mildew	1052	1052	0
Corn_Cercospora_leaf_spot_Gray_leaf_spot	513	0	513
Corn_Common_rust	1192	0	1192
Corn_healthy	1162	0	1162
Corn_Northern_Leaf_Blight	985	0	985
Grape_Black_rot	1180	1173	7
Grape_Esca_(Black_Measles)	1383	1383	0
Grape_healthy	423	423	0
Grape_Leaf_blight_(Isariopsis_Leaf_Spot)	1076	1076	0
Orange_Haunglongbing_(Citrus_greening)	5507	5507	0
Peach_Bacterial_spot	2297	2277	20
Peach_healthy	360	359	1
Pepper_bell_Bacterial_spot	997	996	1
Pepper_bell_healthy	1477	1476	1
Potato_Early_blight	1000	1000	0
Potato_healthy	152	152	0
Potato_Late_blight	1000	1000	0
Raspberry_healthy	371	371	0
Soybean_healthy	5090	5087	3
Squash_Powdery_mildew	1835	0	1835
Strawberry_healthy	456	456	0
Strawberry_Leaf_scorch	1109	1093	16
Tomato_Bacterial_spot	2127	2127	0
Tomato_Early_blight	1000	993	7
Tomato_healthy	1591	1590	1
Tomato_Late_blight	1909	1885	24

CAPÍTULO 5

CONCLUSÕES

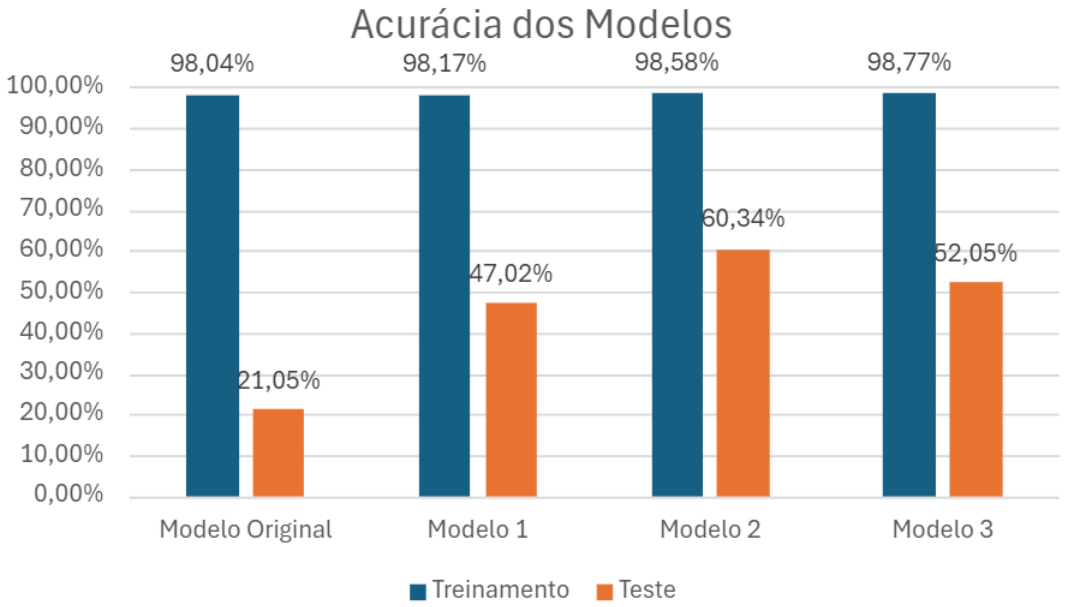
Este trabalho teve como objetivo investigar se a ausência de fundo nas imagens utilizadas no treinamento de redes neurais convolucionais poderia ser benéfica para a classificação de patologias vegetais. A motivação para a remoção do fundo surgiu após a análise dos resultados do trabalho anterior, que demonstrou que a presença de elementos de fundo nas imagens reais prejudicava o desempenho do modelo, especialmente em condições menos controladas. O estudo incluiu o desenvolvimento de um algoritmo eficiente para a remoção de fundo das imagens, além da implementação de algoritmos de validação com imagens reais e a criação de uma API para facilitar o uso prático do modelo.

Para comparar a abordagem proposta com o trabalho de referência, foram realizados vários treinamentos seguindo uma metodologia de validação cruzada. Primeiramente, reproduziu-se o algoritmo do trabalho anterior, o qual obteve uma acurácia de treinamento de 98,04%. Posteriormente, realizou-se os treinamentos com os modelos deste trabalho. Os resultados de treinamento dos modelos 1, 2, 3 obtidos neste trabalho foram, respectivamente, 98,17%, 98,58% e 98,77%.

Além das análises realizadas em ambiente controlado, foram incluídos testes com imagens reais coletadas da internet, com o objetivo de simular situações práticas, como as que um usuário enfrentaria no campo. Para isso, criou-se um novo banco de dados. Nestas condições, os modelos propostos neste trabalho apresentaram um desempenho significativamente superior, com a acurácia aumentando em até 3 vezes em comparação com o trabalho original. Embora os níveis de acurácia ainda não tenham atingido um patamar ideal para lançamento direto ao mercado ou para uso por usuários finais, os resultados alcançados são promissores e demonstram a viabilidade da metodologia empregada. A acurácia de teste, que reflete o desempenho em condições mais próximas da realidade, cresceu de 21,05% para 60,34%,em relação ao trabalho original e ao melhor modelo na validação (Modelo 2).

Portanto, o estudo reforça que a remoção de fundo é uma estratégia válida e eficaz para aprimorar modelos de classificação de doenças em plantas. Além disso, as estratégias adotadas de balanceamento das classes e a simplificação da tarefa de classificação, focando apenas na identificação da espécie, também foram validadas. Pode-se verificar uma comparação entre os resultados de treinamento e de teste para cada modelo na Figura 5.1

Figura 5.1: Acurácia de treinamento e de teste para cada Modelo



Fonte: Autoria Própria

A partir das análises realizadas neste trabalho, é possível sugerir as seguintes direções para estudos futuros:

- Desenvolvimento de um banco de dados de treinamento com classes balanceadas, garantindo maior uniformidade dos dados utilizados;
- Criação de um banco de dados de validação com uma quantidade maior de imagens por classe, incluindo fotos reais capturadas em condições de campo, para uma melhor simulação do ambiente prático;
- Aperfeiçoamento do algoritmo de retirada de fundo para aplicações mais específicas, visando aumentar a precisão na segmentação de diferentes tipos de folhas e ambientes;
- Realização de treinamentos com diferentes hiperparâmetros, para verificar uma possível melhoria.

- Divisão da tarefa de classificação em duas etapas: primeiramente identificar a espécie de planta e, em seguida, determinar a doença presente, caso haja;
- Desenvolvimento de uma API mais robusta e de um aplicativo funcional para uso em campo;

REFERÊNCIAS

3BLUE1BROWN. *Backpropagation calculus*. 2024. Acesso em: 06 ago. 2024. Disponível em: <https://www.3blue1brown.com/lessons/backpropagation-calculus>. Citado na página 11.

AGRICULTURA, P. e. A. Ministério da. *Agrostat: Indicadores do Agronegócio*. 2023. Acesso em: 6 nov. 2023. Disponível em: <https://indicadores.agricultura.gov.br/agrostat/index.htm>. Citado na página 1.

AGROLINK. *Problemas*. 2023. Disponível em: <https://www.agrolink.com.br/problemas> – acesso em 25 jan. 2023. Citado na página 29.

AMBIENTE, P. das Nações Unidas para o M. *Como alimentar 10 bilhões de pessoas até 2050*. 2023. Acesso em: 8 nov. 2023. Disponível em: <https://www.unep.org/pt-br/noticias-e-reportagens/reportagem/como-alimentar-10-bilhoes-de-pessoas-ate-2050>. Citado na página 2.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2005.09007*, 2020. Disponível em: <https://arxiv.org/pdf/2005.09007>. Citado 4 vezes nas páginas 22, 23, 24, and 25.

BOTELHO, M. P. *Classificador de Patologias Vegetais a partir de Imagens utilizando Redes Neurais Convolucionais*. Trabalho de Conclusão de Curso (Engenharia Elétrica) — Universidade de Brasília, Brasília, DF, jul 2023. Departamento de Engenharia Elétrica, Faculdade de Tecnologia. Citado 16 vezes nas páginas 3, 7, 8, 9, 10, 11, 14, 16, 17, 28, 30, 37, 40, 41, 52, and 53.

BRASIL, O. *FAO lista 5 doenças de plantas que a crise climática está agravando*. 2023. Acesso em: 17 nov. 2023. Disponível em: <https://brasil.un.org/pt-br/184058-fao-lista-5-doen%C3%A7as-de-plantas-que-crise-clim%C3%A1tica-est%C3%A1-agravando>. Citado na página 2.

BURKOV, A. *The Hundred-Page Machine Learning Book*. Quebec City, Canada: Andriy Burkov, 2019. Citado 2 vezes nas páginas 5 and 9.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1609.04747*, 2017. Disponível em: <https://arxiv.org/pdf/1609.04747>. Citado na página 13.

EMBRAPA. *Dados econômicos da soja*. 2023. Acesso em: 10 nov. 2023. Disponível em: <https://www.embrapa.br/soja/cultivos/soja1/dados-economicos>. Citado na página 1.

FORTE, M.; PITIé, F. F, b, alpha matting. *arXiv preprint arXiv:2003.07711*, 2020. Acesso em: 05 abr. 2024. Disponível em: <https://arxiv.org/pdf/2003.07711>. Citado na página 26.

G., G.; J., A. P. *Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network*. 2017. Mendeley Data, V1, doi: 10.17632/tywbtsjrjv.1. Citado na página 28.

GATIS, D. *Rembg: Remove Image Background*. 2024. Disponível em: <https://github.com/danielgatis/rembg>. Citado na página 21.

HAYKIN, S. *Redes Neurais: Princípios e Prática*. 2. ed. Porto Alegre: Bookman Editora, 2001. Citado 2 vezes nas páginas 7 and 8.

HAYKIN, S. *Neural Networks and Learning Machines*. Hamilton, Ontario, Canada: Pearson, 2009. Citado na página 20.

HUGHES, D. P.; SALATHE, M. *An open access repository of images on plant health to enable the development of mobile disease diagnostics*. 2016. Disponível em: <https://arxiv.org/abs/1511.08060>. Citado 2 vezes nas páginas 28 and 29.

IBM. *Aprendizado de Máquina*. 2024. Acesso em: 06 jun. 2024. Disponível em: <https://www.ibm.com/br-pt/topics/machine-learning>. Citado na página 5.

INDúSTRIA, C. E. e. S. Ministério da. *Comex Vis: Visão Integrada do Comércio Exterior*. 2023. Acesso em: 5 nov. 2023. Disponível em: <http://comexstat.mdic.gov.br/pt/comex-vis>. Citado na página 1.

INFORMATION, U. B. S. of. *What is Machine Learning?* 2024. Acesso em: 17 nov. 2023. Disponível em: <https://ischoolonline.berkeley.edu/blog/what-is-machine-learning>. Citado na página 5.

IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1409.4842*, 2015. Disponível em: <https://arxiv.org/pdf/1409.4842>. Citado na página 18.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1610.02357*, 2017. Disponível em: <https://arxiv.org/abs/1610.02357>. Citado na página 19.

KRIZHEVSKY, A. Convolutional deep belief networks on cifar-10. *2010 IEEE Conference on Neural Networks*, IEEE, p. 1–9, 2010. Acesso em: 05 mar. 2024. Disponível em: <https://ieeexplore.ieee.org/document/5539896>. Citado na página 26.

PAULO, F. de S. *Agro eleva PIB, renda e população, e desigualdade cai onde setor avança mais*. 2023. Acesso em: 10 nov. 2023. Disponível em: <https://www1.folha.uol.com.br/mercado/2023/07/agro-eleva-pib-renda-e-populacao-e-desigualdade-cai-onde-setor-avanca-mais.shtml>. Citado na página 1.

RURAL, C. *Soja: custo de produção cai, mas defensivos impedem retração maior*. 2023. Acesso em: 17 nov. 2023. Disponível em: <https://www.canalrural.com.br/agricultura/soja-custo-de-producao-cai-mas-defensivos-impedem-retracao-maior/>. Citado na página 2.

SYNGENTA. *Pesquisa inédita revela mapa de crescimento e danos econômicos causados*. 2023. Acesso em: 19 nov. 2023. Disponível em: <https://www.syngenta.com.br/press-release/institucional/pesquisa-inedita-revela-mapa-de-crescimento-e-danos-economicos-causados>. Citado na página 2.

SZEGEDY, C.; TOSHEV, A.; ERHAN, D. Deep neural networks for object detection. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, p. 580–587, 2013. Acesso em: 09 mai. 2024. Disponível em: <https://ieeexplore.ieee.org/document/5995665>. Citado na página 26.

ZEILER, M. D.; FERGUS, R. Visualizing and understanding convolutional networks. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, p. 818–825, 2013. Acesso em: 04 jun. 2024. Disponível em: <https://ieeexplore.ieee.org/document/6409354>. Citado na página 26.

APÊNDICE A

Em anexo estão as referências por classe e para cada imagem do banco de dados construído a partir de imagens da internet para o teste deste trabalho. A organização das referências deste banco de dados é feita da seguinte forma: as classes estão dispostas em ordem alfabética, conforme o banco de dados original, e o número antes do link corresponde à posição da imagem dentro da classe, como primeira imagem, segunda, e assim por diante.

'Apple___ Apple_scab'

1. <https://www2.gov.bc.ca/gov/content/industry/agriculture-seafood/animals-and-crops/plant-health/insects-and-plant-diseases/tree-fruits/apple-scab>
2. <https://www.apsnet.org/edcenter/disandpath/fungalasco/pdlessons/Pages/AppleScab.aspx>
3. <https://blogs.cornell.edu/applevarietydatabase/fact-sheet-apple-scab/>
4. <https://www.agric.wa.gov.au/pome-fruit/managing-apple-scab-western-australia>
5. <https://ag.umass.edu/landscape/fact-sheets/apple-scab>
6. <https://pestsofbhutan.nppc.gov.bt/crop-and-pest-identification/diseases/apple-scab/>

'Apple___ Black_rot'

1. <https://rvpadmin.cce.cornell.edu/uploads/doc_1069.pdf>
2. <https://blogs.cornell.edu/applevarietydatabase/fact-sheet-frogeye-leaf-spot-black-rot/>
3. <https://gardenerspath.com/how-to/disease-and-pests/apple-black-rot-frogeye-leaf-spot/>
4. <https://www.ipmimages.org/browse/detail.cfm?imgnum=5549164#>
5. <https://www.ontario.ca/page/black-rot>

'Apple___ Cedar_apple_rust'

1. <https://homegarden.cahnr.uconn.edu/factsheets/apple-rust-disease/>

2. <https://hort.extension.wisc.edu/articles/cedar-apple-rust/>

3. <https://extension.okstate.edu/programs/digital-diagnostics/plant-diseases/cedar-apple-rust.html>

4. <https://www.thegleaner.com/story/news/2021/06/04/rideout-cedar-apple-rust-can-cause-subs 7519390002/>

5. <https://www.pinterest.com/pin/cedar-apple-rust--267471665342684035/>

'Apple___ healthy'

1. <https://depositphotos.com/photos/apple-leaf.html>

2. <https://depositphotos.com/photos/apple-leaf.html?qview=334158958>

3. <https://depositphotos.com/photos/apple-leaf.html?qview=334985600>

4. <https://depositphotos.com/photos/apple-leaf.html?qview=333216264>

'Blueberry___ healthy'

1. <https://depositphotos.com/photo/green-leaves-blueberries-isolated-white-background-texture- html>

2. <https://yellowimages.com/stock/blueberry-leaf-yi3609185>

'Cherry___ Powdery_mildew'

1. <https://extension.usu.edu/planthealth/ipm/notes_ag/fruit-powdery-mildew>

2. <https://www.rosbreed.org/node/1032>

3. <https://attra.ncat.org/publication/cherry-diseases/>

'Cherry___ healthy'

1. <https://www.shutterstock.com/image-photo/cherry-leaf-260nw-104858582.jpg>

2. <https://www.shutterstock.com/image-photo/cherry-leafs-isolated-on-white-260nw-1660822477.
jpg>

3. <https://www.woodlandtrust.org.uk/trees-woods-and-wildlife/british-trees/a-z-of-british-trees/
bird-cherry/>

4. <https://www.tree-guide.com/images/styles/600x450-copy_/public/wild-cherry-leaf-underside.
jpg?itok=VjuAb7Ch>

5. <https://www.shutterstock.com/pt/image-photo/green-leaf-wild-cherry-tree-prunus-449371609>

'Grape___ Black_rot'

1. <https://www.mindenpictures.com/stock-photo-black-rot-guignardia-bidwellii-leaf-spot-lesions-
html>

2. <https://www.facebook.com/UMaineExtensionPenobscot/posts/black-rot-on-grape-leaf-spotted
5460994557256888/>

3. <https://www.ipmimages.org/collections/viewcollection.cfm?id=113115>

4. <https://www.ipmimages.org/collections/viewcollection.cfm?id=113115>

5. <https://ohiograpeweb.cfaes.ohio-state.edu/sites/grapeweb/files/imce/pdf_factsheets/Grape%
20Black%20Rot.pdf>

'Grape___ Esca_(Black_Measles)'

1. <https://www.goodfruit.com/management-of-grapevine-trunk-diseases-varies-by-region/
>

2. <https://universe.roboflow.com/yolo-nkpld/grape-q02uy>

3. <https://typeset.io/pdf/grape-leaf-diseases-identification-system-using-1ush38y4.pdf>

4. <https://www.researchgate.net/figure/Images-of-grape-leaf-disease-8_fig1_361865968>

5. <https://www.researchgate.net/figure/Images-of-grape-leaf-disease-8_fig1_361865968>

'Grape___ Leaf_ blight_ (Isariopsis_ Leaf_ Spot)'

1. <https://www.insectimages.org/browse/detail.cfm?imgnum=5405317>
2. <https://www.invasive.org/collections/viewcollection.cfm?ser=72222>
3. <https://www.inaturalist.org/taxa/319633-Phyllosticta-minima>
4. <https://www.forestryimages.org/browse/detail.cfm?imgnum=5508950>

'Grape___ healthy'

1. <https://medium.com/@SoftwareZone365/health-benefits-of-grape-leaves-05bbb45c55b7>
2. <https://www.pinterest.com/pin/71072500340815763/>
3. <https://pngtree.com/freepng/grape-leaves_6624166.html>

'Orange___ Haunglongbing_ (Citrus_ greening)'

1. <https://www.isaaa.org/kc/cropbiotechupdate/article/default.asp?ID=18588>
2. <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSxlB0yv83xSAVlMHypBoBTZfYC
   usqp=CAU>
3. <https://texascitrusindustry.com/pest-and-diseases/>
4. <https://ucanr.edu/blogs/blogcore/postdetail.cfm?postnum=30401>
5. <https://ucanr.edu/blogs/blogcore/postdetail.cfm?postnum=30401>
6. <https://www.apsnet.org/edcenter/apsnetfeatures/Pages/HuanglongbingImpact.aspx>

'Peach___ Bacterial_ spot'

1. <https://www.ipmimages.org/browse/detail.cfm?imgnum=0162025>
2. <https://www.ipmimages.org/browse/detail.cfm?imgnum=0162025>
3. <https://www.ipmimages.org/browse/detail.cfm?imgnum=0162025>

4. <https://ask2.extension.org/kb/faq.php?id=744179>

5. <https://www.greatplainsgrowersconference.org/uploads/2/9/1/4/29140369/shane.peachdisease
missouri.2019.pdf>

'Peach___healthy'

1. <https://www.pinterest.com/pin/peach-leaf-on-a-white-background-sponsored-leaf-peach-backg
>

2. <https://stock.adobe.com/br/search?k=%22peach+leaf%22>

3. <https://www.researchgate.net/figure/Healthy-peach-leaf-left-and-partial-center-and-complete-
fig2_273005882>

4. <https://stock.adobe.com/br/search?k=%22peach+leaf%22&asset_id=709761851>

5. <https://stock.adobe.com/br/search?k=%22peach+leaf%22&asset_id=709761851>

6. <https://stock.adobe.com/br/search?k=%22peach+leaf%22&asset_id=709761851>

7. <https://stock.adobe.com/br/search?k=%22peach+leaf%22&asset_id=709761851>

'Pepper,_bell___Bacterial_spot'

1. <https://extension.wvu.edu/lawn-gardening-pests/plant-disease/fruit-vegetable-diseases/
bacterial-leaf-spot-of-pepper>

2. <https://plantpathology.ca.uky.edu/files/ppfs-vg-17.pdf>

3. <https://www.gardeningknowhow.com/edible/vegetables/pepper/bacterial-leaf-spot-on-peppers
htm>

4. <https://extension.umd.edu/resource/bacterial-leaf-spot-peppers/>

5. <https://extension.umn.edu/disease-management/bacterial-spot-tomato-and-pepper>

'Pepper,_bell___healthy'

1. <https://www.researchgate.net/figure/Sample-bell-pepper-leaves-for-healthy-left-and-bacteria-s
fig1_358987628>

2. <https://www.facebook.com/GardenAndWindow/posts/bell-pepper-leavesbellpepperleaf-bellpe
1030221330724182/>

3. <https://www.facebook.com/photo/?fbid=1030220844057564&set=pcb.1030221330724182>

4. <https://www.facebook.com/photo/?fbid=1030220844057564&set=pcb.1030221330724182>

5. <https://www.facebook.com/photo/?fbid=1030221280724187&set=pcb.1030221330724182>

6. <https://www.facebook.com/photo/?fbid=1030221280724187&set=pcb.1030221330724182>

'Potato___Early_blight'

1. <https://www.syngenta.ca/pests/disease/early-blight/potatoes>

2. <https://www.syngenta.ca/pests/disease/early-blight/potatoes>

3. <https://www.sciencephoto.com/media/15013/view/early-blight-on-a-potato-plant>

4. <https://www.researchgate.net/figure/Early-blight-diseases-infected-potato-leaf-Introduction-E
fig1_377916096>

5. <https://www.researchgate.net/figure/Potatoes-with-Early-Blight-An-example-of-an-image-of-a
fig6_376191472>

'Potato___Late_blight'

1. <https://millerresearch.com/2018/07/late-blight-fungicide-management/>

2. <https://www.sciencephoto.com/media/15233/view/potato-leaf-blight>

3. <https://www.sciencephoto.com/media/15233/view/potato-leaf-blight>

4. <https://www.sciencephoto.com/media/15423/view/late-blight-on-a-potato-plant>

'Potato___healthy'

1. <https://www.growerexperts.com/can-i-eat-potato-leaves/>

2. <https://www.shutterstock.com/image-photo/potato-leaf-isolated-on-white-260nw-2299088565.
jpg>

3. <https://www.freepik.com/premium-photo/fresh-green-leaf-potato-plant-isolated_32515942.
htm>

4. <https://www.semanticscholar.org/paper/Potato-Leaf-Diseases-Detection-and-Classification-At
3eafa53ed355c25d0dd36aad3c04c5abe0347e32/figure/2>

'Raspberry___ healthy'

1. <https://www.euphoricherbals.com/blogs/blog/red-raspberry-leaf>

2. <https://www.euphoricherbals.com/blogs/blog/red-raspberry-leaf>

3. <https://www.euphoricherbals.com/blogs/blog/red-raspberry-leaf>

4. <https://www.shutterstock.com/image-photo/raspberry-leaf-isolated-on-white-260nw-225644190
jpg>

5. <https://melissaknorris.com/rasberry-leaf-tea-benefits/>

'Soybean___ healthy'

1. <https://www.shutterstock.com/image-photo/soybean-leaf-isolated-on-white-260nw-1712103859
jpg>

2. <https://www.shutterstock.com/image-photo/soya-bean-green-leaf-closeup-260nw-423357988.
jpg>

3. <https://www.shutterstock.com/image-photo/soya-bean-green-leaf-closeup-260nw-423357988.
jpg>

4. <https://www.shutterstock.com/image-photo/soya-bean-green-leaf-closeup-260nw-423357988.
jpg>

5. <https://www.istockphoto.com/br/foto/folha-de-soja-iluminada-%C3%A0-m%C3%A3o-gm1407

'Strawberry___ Leaf_ scorch'

1. <https://ohioline.osu.edu/factsheet/plpath-fru-35>
2. <https://ohioline.osu.edu/factsheet/plpath-fru-35>
3. <https://ohioline.osu.edu/factsheet/plpath-fru-35>
4. <https://www.dreamstime.com/strawberry-leaf-scorch-common-fungal-disease-caused-diplocarp

'Strawberry___ healthy'

1. <https://pngtree.com/freepng/green-strawberry-leaf-isolated-over-white-background-healthy_14175537.html>
2. <https://media.istockphoto.com/id/623362994/photo/green-strawberry-leaf-isolated-on-white.jpg?s=612x612&w=0&k=20&c=DKXj_Cfq5Ie6oSus4Htscmvwkn0keiYESrim2Smu__Q=>
3. <https://media.istockphoto.com/id/623362994/photo/green-strawberry-leaf-isolated-on-white.jpg?s=612x612&w=0&k=20&c=DKXj_Cfq5Ie6oSus4Htscmvwkn0keiYESrim2Smu__Q=>
4. <https://media.istockphoto.com/id/623362994/photo/green-strawberry-leaf-isolated-on-white.jpg?s=612x612&w=0&k=20&c=DKXj_Cfq5Ie6oSus4Htscmvwkn0keiYESrim2Smu__Q=>
5. <https://static1.bigstockphoto.com/2/6/2/large1500/262272706.jpg>

'Tomato___ Bacterial_ spot'

1. <https://vegetablegrowersnews.com/news/protect-greenhouse-tomato-transplants-from-bacteria>
2. <https://www.shutterstock.com/image-photo/leaf-spot-disease-tomato-600nw-1588785553.jpg>

'Tomato___ Early_ blight'

1. <https://content.ces.ncsu.edu/early-blight-of-tomato>

2. <https://content.ces.ncsu.edu/early-blight-of-tomato>

3. <https://content.ces.ncsu.edu/early-blight-of-tomato>

4. <https://plantpath.ifas.ufl.edu/u-scout/tomato/early-blight.html>

5. <https://www.walterreeves.com/food-gardening/tomato-early-blight-2/>

'Tomato___Late_blight'

1. <https://content.ces.ncsu.edu/tomato-late-blight>

2. <https://content.ces.ncsu.edu/tomato-late-blight>

3. <https://vegpath.plantpath.wisc.edu/diseases/tomato-late-blight/>

4. <https://vegpath.plantpath.wisc.edu/diseases/tomato-late-blight/>

5. <https://plantix.net/en/library/plant-diseases/100046/tomato-late-blight/>

'Tomato___Leaf_Mold'

1. <https://vegcropshotline.org/article/cercospora-leaf-mold-of-tomato/>

2. <https://vegcropshotline.org/article/cercospora-leaf-mold-of-tomato/>

3. <https://vegcropshotline.org/article/cercospora-leaf-mold-of-tomato/>

4. <https://extension.umn.edu/disease-management/tomato-leaf-mold>

5. <https://extension.umn.edu/disease-management/tomato-leaf-mold>

6. <https://extension.umn.edu/disease-management/tomato-leaf-mold>

'Tomato___Septoria_leaf_spot'

1. <https://www.missouribotanicalgarden.org/gardens-gardening/your-garden/help-for-the-home-g
advice-tips-resources/insects-pests-and-problems/diseases/fungal-spots/septoria-leaf-spot-of-tom

2. <https://www.missouribotanicalgarden.org/gardens-gardening/your-garden/help-for-the-home-g advice-tips-resources/insects-pests-and-problems/diseases/fungal-spots/septoria-leaf-spot-of-tom

3. <https://www.missouribotanicalgarden.org/gardens-gardening/your-garden/help-for-the-home-g advice-tips-resources/insects-pests-and-problems/diseases/fungal-spots/septoria-leaf-spot-of-tom

4. <https://www.missouribotanicalgarden.org/gardens-gardening/your-garden/help-for-the-home-g advice-tips-resources/insects-pests-and-problems/diseases/fungal-spots/septoria-leaf-spot-of-tom

5. <https://www.gardeningknowhow.com/edible/vegetables/tomato/septoria-leaf-spot.htm>

'Tomato___ Spider_mites Two-spotted_spider_mite'

1. <https://entomologytoday.org/tomato-leaf-with-spider-mite-damage/>

2. <https://entomology.ca.uky.edu/ef310>

3. <https://www.producegrower.com/article/pg0614-veggie-spider-mites-management/>

4. <https://www.walterreeves.com/food-gardening/spider-mites-control-on-tomatoes/>

5. <https://www.walterreeves.com/food-gardening/spider-mites-control-on-tomatoes/>

'Tomato___ Target_Spot'

1. <https://apps.lucidcentral.org/pppw_v10/text/web_full/entities/tomato_target_spot_ 163.htm>

2. <https://apps.lucidcentral.org/pppw_v10/text/web_full/entities/tomato_target_spot_ 163.htm>

3. <https://plantpath.ifas.ufl.edu/u-scout/tomato/target-spot.html>

4. <https://plantpath.ifas.ufl.edu/u-scout/tomato/target-spot.html>

'Tomato___ Tomato_Yellow_Leaf_Curl_Virus'

1. <https://www.syngenta.co.in/how-manage-virus-infections>

2. <https://content.ces.ncsu.edu/tomato-yellow-leaf-curl-virus>

3. <https://www.istockphoto.com/br/foto/tomate-amarelo-leaf-curl-virus-em-tomate-gm10753489

'Tomato___ Tomato_ mosaic_ virus'

1. <https://blogs.ifas.ufl.edu/stlucieco/2023/03/03/tomato-mosaic-virus-tomv-and-its-managemen
>

2. <https://blogs.ifas.ufl.edu/stlucieco/2023/03/03/tomato-mosaic-virus-tomv-and-its-managemen
>

3. <https://blogs.ifas.ufl.edu/stlucieco/2023/03/03/tomato-mosaic-virus-tomv-and-its-managemen
>

4. <https://extension.usu.edu/planthealth/ipm/notes_ag/veg-TMV-ToMV>

5. <https://extension.usu.edu/planthealth/ipm/notes_ag/veg-TMV-ToMV>

'Tomato___ healthy'

1. <https://www.sciencephoto.com/media/80344/view/tomato-leaf>

2. <https://www.semanticscholar.org/paper/Tomato-Leaf-Disease-Detection-Using-Convolutional-
f2d03978f41755b8407cafc229b7c10335732e6d>

3. <https://www.semanticscholar.org/paper/Tomato-Leaf-Disease-Detection-Using-Convolutional-
f2d03978f41755b8407cafc229b7c10335732e6d>

4. <https://www.researchgate.net/figure/Visual-differences-between-a-healthy-tomato-leaf-picture
fig4_339178698>