UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E COMPUTAÇÃO

GLAUBER BORGES LINDOLFO
IAN MARCOS DA CRUZ CHAVES

COMPARAÇÃO DE ARQUITETURAS DE REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO DE DOENÇAS FOLIARES DO TOMATEIRO

Goiânia
2024

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as):

Glauber Borges Lindolfo 201802681 e Ian Marcos da Cruz Chaves 201802684

Título do trabalho:

COMPARAÇÃO DE ARQUITETURAS DE REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO DE DOENÇAS FOLIARES DO TOMATEIRO

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Cassio Dener Noronha Vinhal**, **Professor do Magistério Superior**, em 22/12/2024, às 13:07, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Ian Marcos Da Cruz Chaves**, **Discente**, em 22/12/2024, às 13:10, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Glauber Borges Lindolfo**, **Discente**, em 22/12/2024, às 13:12, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5060541** e o código CRC **63A1B1C4**.

Referência: Processo nº 23070.045015/2024-54 SEI nº 5060541

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E COMPUTAÇÃO

GLAUBER BORGES LINDOLFO
IAN MARCOS DA CRUZ CHAVES

COMPARAÇÃO DE ARQUITETURAS DE REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO DE DOENÇAS FOLIARES DO TOMATEIRO

Trabalho de Conclusão apresentado à Coordenação do Curso de Engenharia de Computação da Escola de Engenharia Elétrica, Mecânica e Computação da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. CASSIO DENER NORONHA VINHAL

Goiânia
2024

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Lindolfo, Glauber Borges
Comparação de Arquiteturas de Redes Neurais Convolucionais para a Detecção de Doenças Foliares do Tomateiro [manuscrito] / Glauber Borges Lindolfo, Ian Marcos da Cruz Chaves. - 2024.
xxiv, 24 f.: il.

Orientador: Prof. Cassio Dener Noronha Vinhal.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Engenharia da Computação, Goiânia, 2024.
Bibliografia.

1. Visão Computacional. 2. Inteligência Artificial. 3. Redes Neurais Convolucionais. 4. Agricultura 4.0. 5. Tomateiro. I. Chaves, Ian Marcos da Cruz. II. Vinhal, Cassio Dener Noronha, orient. III. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Aos **vinte dias do mês de dezembro do ano de 2024, 16h00** , iniciou-se a sessão pública de defesa do Projeto Final de Curso II - Trabalho de Conclusão de Curso - TCC intitulado **"COMPARAÇÃO DE ARQUITETURAS DE REDES NEURAIS CONVOLUCIONAIS PARA A DETECÇÃO DE DOENÇAS FOLIARES DO TOMATEIRO"**, de autoria de **Glauber Borges Lindolfo (**201802681**) e Ian Marcos da Cruz Chaves (**201802684**)**, do curso de Engenharia de Computação, da Escola de Engenharia Elétrica, Mecânica e Computação (EMC) da UFG. Os trabalhos foram instalados pelo(a) **Prof. Dr. Cassio Dener Noronha Vinhal (EMC/UFG)**, presidente, com a participação dos demais membros da Banca Examinadora: **Prof. Flávio Geraldo Coelho Rocha (EMC/UFG), Prof. Dr. Gelson da Cruz Junior (EMC/UFG) e Prof. Dr. Marco Antonio Assfalk de Oliveira (EMC/UFG)**. Após a apresentação, a banca examinadora realizou a arguição dos estudantes. Posteriormente, de forma reservada, a Banca Examinadora atribuiu a nota final de **9,0 (nove)**, tendo sido o TCC considerado **aprovado**.

Proclamados os resultados, os trabalhos foram encerrados e, para constar, lavrou-se a presente Ata que segue assinada pelos Membros da Banca Examinadora.

Documento assinado eletronicamente por **Cassio Dener Noronha Vinhal**, **Professor do Magistério Superior**, em 15/01/2025, às 20:33, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Gelson Da Cruz Junior**, **Professor do Magistério Superior**, em 15/01/2025, às 20:34, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Marco Antonio Assfalk De Oliveira**, **Professor do Magistério Superior**, em 15/01/2025, às 20:40, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Flavio Geraldo Coelho Rocha**, **Professor do Magistério Superior**, em 16/01/2025, às 07:49, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5097729** e o código CRC **DF8BEFD7**.

Referência: Processo nº 23070.045015/2024-54 SEI nº 5097729

Comparação de Arquiteturas de Redes Neurais Convolucionais para a Detecção de Doenças Foliares do Tomateiro

Glauber Borges Lindolfo[1], Ian Marcos da Cruz Chaves[2], Cássio Dener Noronha Vinhal[3]
Universidade Federal de Goiás (UFG) - Escola de Engenharia Elétrica, Mecânica e de Computação (EMC) - Goiânia, Goiás, Brasil 74605-010. E-mails: glauber_borges@discente.ufg.br[1], ianchaves@discente.ufg.br[2], vinhal@ufg.br[3]

***Resumo*—O reconhecimento automatizado de doenças foliares é um dos principais desafios da Agricultura 4.0, demandando metodologias que integrem conhecimentos agronômicos, coleta de imagens para monitoramento e técnicas avançadas de aprendizado de máquina. Este trabalho tem como objetivo realizar uma análise comparativa entre diferentes arquiteturas de redes neurais convolucionais (CNNs) aplicadas à detecção de doenças foliares em tomateiros, incluindo a pinta-preta. Foram utilizadas três arquiteturas amplamente reconhecidas, ResNet-50, Inception-v3 e VGG-16, explorando combinações de hiperparâmetros como taxa de aprendizado, otimizadores e o uso de decaimento. Além disso, mapas de ativação foram empregados para identificar padrões visuais relevantes que influenciam as decisões dos modelos. Os resultados demonstram que o ResNet-50 apresentou o melhor desempenho e estabilidade geral, seguido pelo Inception-v3, enquanto o VGG-16 mostrou maior sensibilidade às configurações de treinamento. Por meio desta análise, busca-se compreender o impacto dessas variações no desempenho das redes, oferecendo informações valiosas para o aprimoramento de modelos aplicados ao manejo fitossanitário e à agricultura de precisão.**

***Palavras-chave*—Visão Computacional, Inteligência Artificial, Redes Neurais Convolucionais, Agricultura 4.0, Tomateiro**

***Abstract*—The automated recognition of leaf diseases is one of the main challenges of Agriculture 4.0, requiring methodologies that integrate agronomic knowledge, image collection for monitoring, and advanced machine learning techniques. This study aims to perform a comparative analysis of different convolutional neural network (CNN) architectures applied to the detection of leaf diseases in tomato plants, including target spot. Three widely recognized architectures were used: ResNet-50, Inception-v3, and VGG-16, exploring combinations of hyperparameters such as learning rate, optimizers, and the use of weight decay. Additionally, activation maps were employed to identify relevant visual patterns that influence the models' decisions. The results show that ResNet-50 achieved the best overall performance and stability, followed by Inception-v3, while VGG-16 exhibited greater sensitivity to training configurations. Through this analysis, we aim to understand the impact of these variations on network performance, providing valuable insights for improving models applied to crop protection management and precision agriculture.**

***Index Terms*—Computer Vision, Artificial Intelligence, Convolutional Neural Networks, Agriculture 4.0, Tomato Plant**

I. Introdução

O Tema da inteligência artificial (IA) tem dominado as manchetes de notícias de tecnologia em todo o mundo, principalmente impulsionado pelos avanços recentes na capacidade da IA de realizar funções tipicamente humanas, como fala e escrita, artes visuais e até criação de música. Essa aproximação da IA aos humanos despertou um interesse generalizado no alcance das capacidades da IA e em quais áreas essas novas tecnologias podem ser aplicadas. Com a ampla disponibilidade de dados acessíveis pela internet, o uso da IA é praticamente ilimitado, capaz de se integrar de várias formas na vida das pessoas.

Segundo um relatório publicado em dezembro de 2022 pela empresa de consultoria Gartner, Inc. [1], a IA pode criar até 2 milhões de novos empregos até 2025. Do ponto de vista acadêmico, o conhecimento sobre o desenvolvimento de redes neurais tem despertado maior interesse, dada a complexidade de desenvolver softwares que utilizam aprendizado de máquina para criar modelos inteligentes capazes de classificações automáticas.

A agricultura tem muito a se beneficiar dessas tecnologias emergentes, incluindo o potencial de aumentar a produtividade para atender à crescente demanda global por alimentos enquanto reduz simultaneamente o impacto ambiental. Neste contexto, a Agricultura 4.0 emergiu como uma abordagem inovadora, integrando tecnologias como Internet das Coisas (IoT), big data, IA e visão computacional para transformar práticas agrícolas tradicionais em sistemas mais eficientes, precisos e sustentáveis. [2].

Dentre as diversas aplicações da visão computacional no setor agrícola, a detecção precoce de doenças em plantas se destaca como uma ferramenta crucial para garantir a saúde das culturas e mitigar perdas econômicas. A utilização de CNNs tornou-se um dos principais métodos para analisar e classificar imagens agrícolas, devido à sua capacidade de identificar padrões intricados e sutis em folhas, caules e frutos de plantas.

Baseado em um modelo de CNN para classificação de imagens, podemos obter uma compreensão mais profunda do desenvolvimento e funcionalidade de uma rede neural, elucidando seus conceitos fundamentais. Ao analisar cada segmento que constitui uma CNN, podemos explorar os processos envolvidos no pré-processamento de imagens, no desenvolvimento de filtros convolucionais, no desenvolvimento de camadas densamente conectadas, no treinamento do modelo, na validação e na utilização de erros para propagar mudanças dentro das camadas.

As CNNs são amplamente empregadas em aplicações de visão computacional, proporcionando uma abordagem escalável para tarefas de classificação de imagens e reconhecimento de objetos. Através de filtros baseados em multiplicação de matrizes, as CNNs extraem informações dos píxeis da imagem e correlacionam essas informações com as características dos objetos correspondentes. Ao estabelecer conexões entre essas características, as CNNs podem identificar características gerais que possibilitam a classificação de objetos. [3].

Essa correlação entre características é um pilar da funcionalidade da CNN, com cada neurônio na camada convolucional indicando a presença de uma característica na imagem. A ativação desse neurônio facilita a transmissão de informações para as camadas subsequentes, que interpretam a combinação de neurônios ativados simultaneamente para gerar novas informações. O poder computacional de uma CNN está mais concentrado na camada densamente conectada, onde os neurônios recebem os resultados das camadas convolucionais e geram probabilidades de classificação para cada filtro. Este processo culmina na geração do resultado final de classificação.

Todo o processo de classificação de doenças em plantas de tomate, particularmente a pinta-preta, é semelhante à visão humana, embora com uma complexidade distinta. Embora o avanço da programação tenha possibilitado o desenvolvimento de tecnologias que automatizam procedimentos complexos e repetitivos, a criação de uma CNN para classificação de doenças permanece um empreendimento significativo. O objetivo não é substituir o trabalho humano ou técnicas de cultivo tradicionais, mas sim introduzir novas tecnologias que aprimorem a eficiência agrícola. [4]

Este trabalho tem como foco o estudo comparativo de diferentes arquiteturas de CNNs aplicadas à classificação de doenças foliares do tomateiro, com ênfase na identificação das configurações mais eficazes para essa tarefa. Três modelos amplamente utilizados, ResNet-50, Inception-v3 e VGG-16, foram selecionados para análise, permitindo uma avaliação detalhada de seus desempenhos em cenários variados. A pinta-preta do tomateiro, por exemplo, uma doença prevalente e prejudicial ao cultivo de tomate, serve como estudo de caso. Dada a capacidade de identificar a doença através de características distintas presentes nas folhas afetadas, a Visão Computacional é empregada para discernir essas características, enquanto o Aprendizado de Máquina é utilizado para definir sua significância para fins de identificação. Para isso, foram exploradas diferentes combinações de hiperparâmetros, incluindo taxa de aprendizado, otimizadores (SGD e Adam) e a utilização ou não de decaimento, com o objetivo de identificar como essas variações influenciam a eficiência, robustez e capacidade de generalização dos modelos para reconhecimento das doenças.

Além disso, o estudo buscou investigar como a distribuição das classes e a complexidade visual das doenças foliares impactam o desempenho de cada arquitetura, contribuindo para uma melhor compreensão de suas limitações e pontos fortes. Por meio da análise de mapas de ativação, foram avaliadas as regiões das imagens consideradas mais relevantes pelos modelos, elucidando os padrões visuais que mais influenciam a tomada de decisão. Essa abordagem não apenas permite uma avaliação quantitativa, mas também oferece informações qualitativas sobre o funcionamento interno de cada modelo.

Ao realizar essa análise comparativa, este trabalho visa contribuir para o desenvolvimento de sistemas mais eficientes e escaláveis na área de visão computacional aplicada à agricultura, com potencial para auxiliar no monitoramento de doenças e na tomada de decisões em tempo real. Os resultados esperados podem servir como base para futuros aprimoramentos de modelos e para a implementação prática de soluções tecnológicas no manejo fitossanitário do tomateiro, promovendo maior produtividade e sustentabilidade no setor agrícola.

II. Revisão Bibliográfica

Uma análise comparativa do desempenho de APIs baseadas em visão computacional na detecção e reconhecimento de quatro doenças prevalentes em plantas (Peronospora, Diplocarpon rosae, oídio e Cancro Cítrico) é apresentada neste estudo [5]. Cada doença foi treinada usando um conjunto de 50 imagens, divididas em duas etapas, e subsequentemente avaliada quanto à sua precisão. Os resultados demonstram que as APIs alcançaram alta precisão de reconhecimento, com a API 3 exibindo mais de 80% de precisão e recall na identificação de doenças em plantas.

Uma visão abrangente das doenças e distúrbios que afetam as plantas de tomate é fornecida na literatura [6]. Esta abrange questões fúngicas, bacterianas, virais e fisiológicas. Uma das principais doenças discutidas é a mancha preta, comumente relatada em plantações de tomate em Iowa. Os autores apresentam medidas de controle cultural e químico para o manejo da mancha preta, assim como práticas recomendadas para controlar outras doenças significativas do tomate.

Outro trabalho [7] propõe princípios de design e modificações arquitetônicas para escalar eficientemente CNNs, resultando no modelo Inception-v2. Este modelo alcança desempenho de ponta no *benchmark* de classificação de imagens ILSVRC 2012 com requisitos computacionais relativamente modestos. Os autores exploram técnicas como evitar gargalos representacionais, empregar representações de dimensões mais altas, fatorar convoluções, usar classificadores auxiliares e aplicar regularização por suavização de rótulos para melhorar o desempenho da rede.

Pereira et al. [8] fornecem uma análise detalhada das estratégias de manejo para a mancha preta. O artigo sublinha a importância de uma abordagem integrada, enfatizando práticas culturais preventivas e a aplicação criteriosa de fungicidas como medidas de controle essenciais. Além disso, os autores destacam a disponibilidade de sistemas de previsão de doenças, que podem otimizar as aplicações de fungicidas e minimizar impactos ambientais.

Uma abordagem abrangente baseada em visão computacional para a detecção de maturidade e doenças em tomates, com foco específico na Mancha de Septoria, é apresentada em [9]. Os autores avaliam o desempenho dos algoritmos de segmentação por limiarização e agrupamento k-means na determinação da maturidade do tomate com base em características de cor. Além disso, propõem um sistema para

analisar folhas de tomate e quantificar a porcentagem de penetração fúngica.

Um método para reconhecimento de doenças capaz de classificar o tipo de doença (saudável, mancha preta inicial, requeima) e o grau de infecção (geral ou grave) em imagens de folhas de batata é apresentado em [10]. Aspectos-chave da abordagem proposta incluem a segmentação automática das regiões das folhas utilizando um algoritmo baseado em corte por grafos, extração de características de cor e textura das folhas segmentadas, e avaliação do desempenho da classificação usando modelos de aprendizado de máquina como k-NN, SVM, RNA e Random Forest.

Outro método também baseado em visão computacional para estimar a gravidade da queima das folhas de chá em imagens de cenas naturais é proposto em [11]. A abordagem incorpora restauração por B-spline para lidar com folhas doentes que estão ocultas, deformadas ou danificadas, e a aplicação de um modelo de máquina de aumento de gradiente (GBM) para fornecer estimativas precisas da gravidade utilizando apenas um pequeno número de amostras rotuladas manualmente.

Um sistema de visão computacional para identificar doenças e pragas em lavouras de soja utilizando imagens capturadas por veículos aéreos não tripulados (VANTs) é proposto em [12]. O método integra segmentação SLIC para isolar folhas e emprega técnicas de aprendizado de máquina tanto superficiais (SVM, k-NN) quanto profundas (Inception-v3, ResNet-50) para classificar doenças e pragas. O sistema alcança alta precisão na identificação, gerando também mapas codificados por cores para informar os agricultores sobre os níveis de infestação nas lavouras. A pesquisa destaca o potencial dessas tecnologias para apoiar especialistas e agricultores na gestão eficaz de cultivos de soja.

III. Fundamentação Teórica

Nessa secção apresentaremos alguns conceitos abstraídos das revisões bibliográficas, onde pretendemos aprofundar os conhecimentos sobre conceitos bases para a compreensão do projeto desenvolvido. Essa conceituação tem por objetivo melhorar o entendimento sobre as doenças estudadas e o funcionamento de uma CNN.

A. Convolutional Neural Network (CNN)

Uma Rede Neural Convolucional é uma categoria de rede neural do tipo *feedfoward*, alimentada por dados que seguem uma direção: do nó de entrada até o nó de saída, e é especificada por realizar processos de filtragem convolucional, *pooling*, entre outros. Quando utilizada em imagens, o modelo atribui pesos e vieses para características das imagens, apresentadas por padrões nos píxeis da imagem. [13].

Esse processo é amplamente inspirado no funcionamento biológico do córtex visual humano. A entrada da imagem real no globo ocular causa um estímulo nos receptores responsáveis por uma região específica do campo visual. Esse estímulo é transportado para um neurônio específico, e a sobreposição de múltiplos neurônios corresponde à interpretação de uma característica, permitindo, ao final, a interpretação dos objetos maiores presentes no campo visual.

De maneira semelhante, a CNN é alimentada por uma matriz que é formada pelos píxeis de uma imagem. Na primeira camada, a camada convolucional, cada neurônio corresponde a um filtro especificado por uma matriz. Realiza-se, então, a operação de convolução entre essa matriz e a matriz que corresponde a uma região da imagem original, retornando um valor que corresponde à probabilidade da presença dessa característica naquela região da imagem. Esse procedimento é realizado até que se cubra toda a região da imagem.

Após esse procedimento, os resultados são passados para a camada de *pooling*. Nessa camada, cada mapa de característica obtido de cada neurônio passa por procedimentos individuais. Primeiro, os valores de baixa probabilidade de presença da característica são transformados em zero, normalmente por um filtro ReLU (Unidade Linear Retificada). Em seguida, o tamanho da matriz é reduzido através de um *pooling*, sendo o *max-pooling* o mais utilizado. Ou seja, há a redução do tamanho da matriz pelo agrupamento de uma região da matriz, preservando o maior valor presente nessa região, o que preserva a posição daquela característica em relação às outras [14].

Analisando o funcionamento das camadas Convolucionais e de *pooling*, é possível entender o motivo da utilização de CNNs na classificação de imagens. Devido à importância da localização de cada píxel para a interpretação de uma característica, não é possível realizar um flattening em uma imagem e alimentá-la em uma rede feedforward comum. Por isso, utiliza-se a forma matricial da imagem para preservar a relação de posição dos píxeis.

Os resultados da camada de *pooling* são encaminhados para uma nova camada convolucional, onde novamente ocorre a operação de convolução com um filtro definido por uma matriz. Essa nova camada tem por objetivo sobrepor as características obtidas anteriormente, para identificar características maiores e mais complexas presentes na imagem. O processo de convolução e *pooling* é contínuo até que se obtenham as características que serão, então, encaminhadas para a camada densamente conectada.

É notável que as primeiras camadas têm por objetivo reduzir o tempo de processamento da camada densamente conectada, visto que a transformação de píxeis em características reduz os dados a serem alimentados nos neurônios das camadas densamente conectadas.

As características da última camada de *pooling* são, então, passadas para os neurônios da camada densamente conectada, que irão atuar da mesma maneira que os neurônios presentes em Redes Neurais comuns. Os neurônios dessa camada produzirão valores probabilísticos para classes, baseados nas características recebidas, utilizando pesos e tendências. Esses neurônios serão, então, responsáveis pela classificação, em termos de probabilidade, da imagem avaliada.

B. Aprendizado de Máquina

Como dito por [9], aprendizado de máquina refere-se à capacidade dos sistemas de aprender a partir de dados específicos do problema, a fim de automatizar o processo de

construção de modelos analíticos e resolver tarefas associadas. Em vez de codificar explicitamente o conhecimento em computadores, o aprendizado de máquina busca aprender automaticamente relações e padrões significativos a partir de exemplos e observações. Esse campo visa melhorar o desempenho de programas de computador com base na experiência, aplicando algoritmos que iterativamente aprendem com dados de treinamento específicos do problema, permitindo que os computadores encontrem informações ocultas e padrões complexos sem serem explicitamente programados.

IV. Métodos Propostos

O desenvolvimento do projeto, como mencionado anteriormente, foi realizado em Python, devido à disponibilidade da biblioteca *PyTorch*, *Torchvision*, entre outras bibliotecas, como *utils*, *numpy*, e ferramentas para álgebra linear e manipulação de dados. Dessa forma, a linguagem oferece a possibilidade de desenvolver um modelo personalizado com o uso de ferramentas que facilitam os processos.

Este trabalho utiliza arquiteturas como ResNet, VGG e Inception, explorando suas capacidades em tarefas de detecção de doenças foliares no tomateiro. A utilização de modelos pré-existentes de CNN envolve uma série de etapas, descritas a seguir na Figura 1. Primeiramente, realiza-se o pré-processamento das imagens, que inclui o armazenamento e a aplicação de uma série de transformações nos dados para adequá-los ao modelo escolhido. Essa etapa é fundamental para garantir que as imagens estejam normalizadas e padronizadas de acordo com os requisitos das arquiteturas utilizadas.

Na segunda etapa, ocorre a configuração e ajuste do modelo pré-treinado. Esse processo envolve a definição de parâmetros como taxa de aprendizado, otimizador e função de perda, além da possível adaptação das camadas densamente conectadas, quando necessário, para alinhar a saída do modelo às classes específicas do problema em estudo.

A terceira etapa corresponde ao treinamento do modelo, no qual o modelo é alimentado com o banco de imagens em múltiplos ciclos (épocas). Durante esse processo, o modelo ajusta seus pesos por meio de retropropagação, a fim de minimizar o erro e otimizar seu desempenho na classificação.

Por fim, realiza-se a etapa de teste do modelo, utilizando um banco de imagens separado para validar o modelo treinado. Nessa etapa, a validação das imagens é considerada fundamental, pois constitui os principais resultados do trabalho.

A. Preparação do Ambiente de Experimentação

Para a realização dos procedimentos descritos neste artigo, foi utilizada a plataforma Google Colab, que fornece um ambiente virtual com kernel Python 3. Além desse ambiente padrão, foi necessário instalar as bibliotecas relacionadas ao uso da biblioteca *rembg*. As funções utilizadas foram extraídas de diferentes bibliotecas, como aquelas referentes a modelos, transformações e otimizadores do torch, além de bibliotecas matemáticas e de exibição, como pandas, *numpy*, *Image* e *cv2*.

Os dados utilizados neste projeto foram obtidos na plataforma Kaggle e têm origem em um repositório público no GitHub. Esses dados foram reunidos e disponibilizados

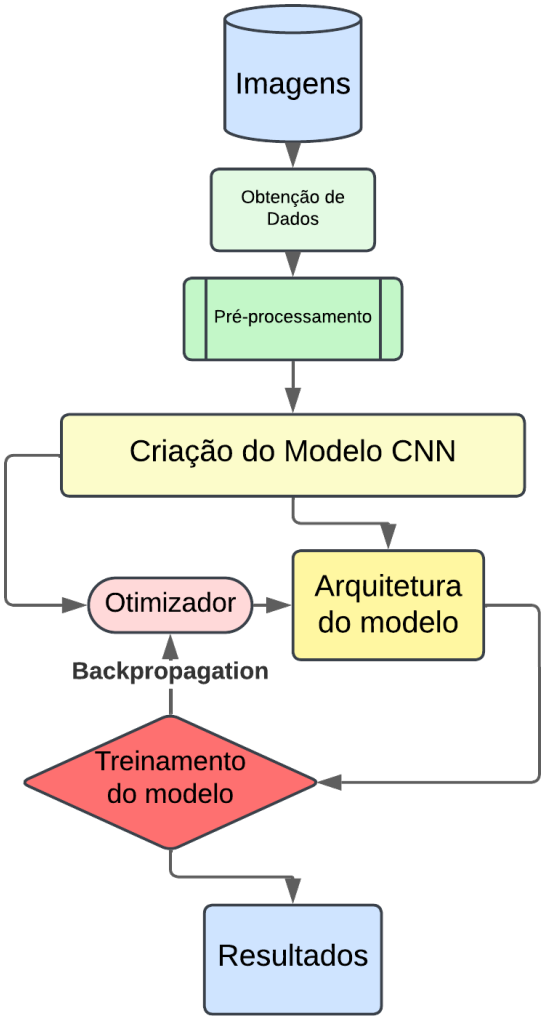


Figura 1: Diagrama dos métodos propostos. Fonte: Próprios autores.

publicamente por Sharada Mohanty[1]. A fonte dos dados está relacionada a um projeto de detecção de diversas doenças em plantas, contendo imagens de várias espécies e condições. Para este estudo, foram considerados apenas os dados relacionados a folhas de tomate, em conformidade com o cultivo específico monitorado.

O conjunto de dados utilizado é composto por imagens de folhas de tomate, todas no formato PNG e com resolução de 256x256 píxeis. A padronização presente nesses dados consiste em uma pequena quantidade de folhas, normalmente apenas uma, com essa folha centralizada e próxima da base da imagem. As folhas estão bem iluminadas, com um *background* cinza padrão, apresentando variações na predominância da doença, quando existente, e variações na idade da folha, podendo ser mais madura ou mais jovem.

Além da classe de folhas saudáveis, os dados estão organi-

[1] https://github.com/spMohanty/PlantVillage-Dataset

zados e indexados de acordo com as doenças que as afetam, podendo estas ser de origem bacteriana, fúngica, viral ou resultantes de pragas, como ácaros. Cada uma dessas doenças apresenta características visuais distintas, o que possibilita sua diferenciação. Tradicionalmente, essa identificação é realizada por especialistas.

Neste projeto, assume-se que as imagens contêm informações suficientes para identificar as doenças. Assim, aplica-se a visão computacional para desenvolver modelos capazes de reconhecer as características-chave presentes nas imagens e, com isso, prever as doenças. A Figura 2 apresenta o histograma de frequências de cada classe analisada.

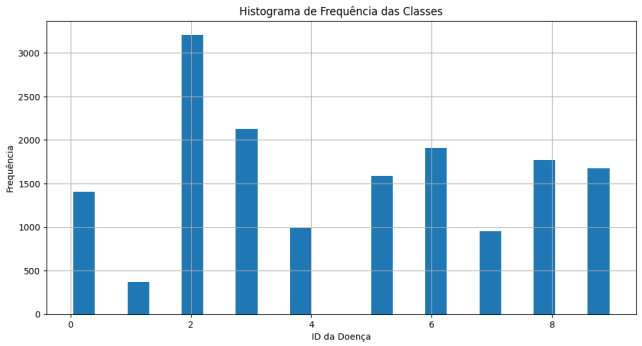


Figura 2: Histograma de frequência das classes de doenças do tomateiro. Fonte: Próprios autores.

A classe *Tomato Healthy* (Tomate Saudável), ilustrada na Figura 3, representa folhas saudáveis do tomateiro, sem a presença de patógenos. As folhas saudáveis possuem coloração verde, com pequenas variações devido à iluminação, mas apresentam constância em cor, textura e forma.

Figura 3: Folhagem do tomateiro saudável. Fonte: Próprios autores.

A classe *Tomato Early Blight* (Pinta-preta do tomateiro), ilustrada na Figura 4, é causada pelo patógeno *Alternaria linariae* (anteriormente *A. solani*) e caracteriza-se pela presença de manchas marrons circulares em folhas mais velhas, com halos amarelados ao redor das manchas. Além disso, observa-se a presença de anéis concêntricos, semelhantes a alvos, nas manchas marrons. As folhas afetadas podem murchar e morrer, expondo os frutos ao sol. Geralmente, a doença inicia-se na base da planta, região mais suscetível devido à depleção de nutrientes durante o surgimento dos frutos [15].

A classe *Tomato Spider Mites* (Ácaros de duas manchas), ilustrada na Figura 5, é causada pelo patógeno *Tetranychus urticae* (ácaro) e caracteriza-se por pontos amarelados ou manchas claras nas folhas, levando à clorose. Observa-se também a presença de teias finas na parte inferior das folhas. Em infestações severas, as folhas podem secar e cair [16].

Figura 4: Folhagem com Pinta-preta. Fonte: Próprios autores.

Figura 5: Folhagem com Ácaros de duas manchas. Fonte: Próprios autores.

A classe *Tomato Septoria Leaf Spot* (Mancha de Septoria), ilustrada na Figura 6, é causada pelo patógeno *Septoria lycopersici* e caracteriza-se por pequenas manchas marrons circulares, com centros acinzentados ou brancos. As margens das manchas são bem definidas e mais escuras em relação ao centro. As folhas infectadas geralmente caem prematuramente, comprometendo o desenvolvimento da planta [17].

Figura 6: Folhagem com Mancha de Septória. Fonte: Próprios autores.

A classe *Tomato Leaf Mold* (Mofo das folhas do tomateiro), ilustrada na Figura 7, é causada pelo patógeno *Cladosporium fulvum* e caracteriza-se por manchas amarelas na parte superior das folhas. Observa-se o crescimento de fungos com textura aveludada de cor verde-oliva na parte inferior das folhas, podendo levar à queda das folhas infectadas e à exposição dos frutos [18].

Figura 7: Folhagem com Mofo do tomateiro. Fonte: Próprios autores.

A classe *Tomato Late Blight* (Requeima tardia), ilustrada na Figura 8, é causada pelo patógeno *Phytophthora infestans*. Caracteriza-se pela presença de manchas irregulares de cor verde-pálido, que se tornam marrons ou pretas conforme a doença avança. Observa-se o crescimento de mofo branco na parte inferior das folhas em condições de alta umidade. As folhas infectadas secam rapidamente, e a doença pode afetar também caules e frutos [19].

Figura 8: Folhagem com Requeima tardia. Fonte: Próprios autores.

A classe *Tomato Bacterial Spot* (Mancha bacteriana do tomateiro), ilustrada na Figura 9, é causada pelo patógeno *Xanthomonas spp*. Caracteriza-se pela presença de pequenas manchas marrons ou pretas, com halos amarelados. As folhas afetadas podem apresentar uma aparência de folha queimada, o que permite diferenciá-la de outras doenças com características visuais semelhantes, como o *Early Blight*. Em casos severos, a infecção provoca queda das folhas e redução da produtividade [20].

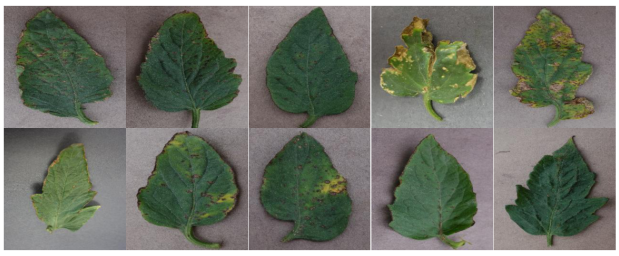


Figura 9: Folhagem com mancha bacteriana. Fonte: Próprios autores.

A classe *Tomato Yellow Leaf Curl Virus* (Vírus do enrolamento amarelo das folhas do tomateiro), ilustrada na Figura 10, é causada pelo vírus transmitido pela mosca-branca (*Bemisia tabaci*). Caracteriza-se visualmente pelo enrolamento e amarelamento das bordas das folhas. Em estágios iniciais, observa-se crescimento reduzido das folhas, e a planta apresenta aparência atrofiada. Essa doença também provoca queda de flores, reduzindo a quantidade de frutos produzidos [21].

Figura 10: Folhagem com vírus do enrolamento amarelo. Fonte: Próprios autores.

A classe *Tomato Mosaic Virus* (Vírus do mosaico do tomateiro) é causada pelo patógeno *Tobamovirus* sp. Caracteriza-se visualmente por manchas claras e escuras, que formam padrões irregulares nas folhas. As folhas afetadas podem apresentar aparência enrugada ou deformada. Em infecções severas, ocorre também redução significativa do crescimento da planta [22].

Figura 11: Folhagem com vírus do mosaico. Fonte: Próprios autores.

A classe *Tomato Target Spot* (Mancha-alvo do tomateiro) é causada pelo patógeno *Corynespora cassiicola*. Visualmente, é caracterizada pela presença de manchas marrons com padrões de anéis concêntricos nas folhas. As folhas afetadas frequentemente caem prematuramente, resultando em impacto negativo na produtividade [23].

Ao analisar as características visuais das doenças podemos perceber algumas semelhanças visuais entre algumas doenças, essas semelhanças serão um desafio para os modelos neurais, que devem buscar características chaves que diferenciem as folhagens afetadas pelas doenças. Como por exemplo o padrão de ”alvo”pode ser diferenciado pois o *Early Blight* possui anéis finos e distintos, já o *Target Spot* possui anéis mais amplos e menos precisos. Com o uso de redes neurais mais profundas, com mais camadas convolucionais, será possível diferenciar as características chaves e achar características que diferenciem as folhagens.

Figura 12: Folhagem com Mancha-alvo. Fonte: Próprios autores.

Os dados foram organizados em pastas, divididas de acordo com as doenças representadas, e estão sendo acessados por meio da função *ImageFolder*, presente na seção de *datasets* da biblioteca *torchvision*. Em seguida, os dados são transformados em tensores utilizando a função *transforms*, também da biblioteca *torchvision*, permitindo que sejam utilizados como entrada nos modelos de aprendizado de máquina. Após a transformação, são adicionados os rótulos de classificação correspondentes às doenças presentes em cada imagem, os quais serão empregados no treinamento supervisionado dos modelos.

B. Pré-processamento do Banco de Imagens

Após a leitura do conjunto de dados, inicia-se o processo de transformação das imagens. Considerando que o modelo será responsável por extrair características visuais, é fundamental preservar informações que possam identificar a doença durante as transformações. Como o modelo interpreta a imagem de maneira distinta da visualização real da folha, é necessário manter a disposição dos píxeis e as relações entre eles. Entretanto, transformações como normalização e redimensionamento podem ser aplicadas.

A transformação de redimensionamento alterará o tamanho da imagem, normalmente uma redução do tamanho original, para um padrão de 299x299 ou 244x244 píxeis a depender do modelo de CNN. Esse redimensionamento tem por objetivo diminuir a complexidade do modelo sacrificando nitidez, pois ao diminuir a entrada será necessário menos processamento para extrair as características de todos os píxeis da imagem. Porém, essa perda de informações não irá impactar na precisão do modelo, pois a distinção entre as características da imagem ainda estará presente, mesmo que para o olho humano se torne mais difícil de visualizá-las, o modelo CNN será capaz de identificá-las.

Também será realizado um conjunto de transformações com o mesmo objetivo de aumentar a robustez do modelo. Primeiramente será realizado uma rotação randômica da imagem, essa transformação parte do princípio que é possível identificar a doença mesmo sem que a folha esteja posicionada de forma padrão. Outra transformação será um espelhamento randômico da imagem, que parte do princípio que não há conexão entre nenhuma doença ao sentido da imagem ser direito ou esquerdo. E será realizado também uma alteração randômica do brilho, contraste e saturação da imagem, essas

Figura 13: Redução de resolução da imagem. Fonte: Próprios autores.

alterações tem por objetivo não criar nenhuma associação da classificação da imagem com as condições de luz e de ambiente do momento da foto da folhagem, ou em relação ao equipamento de coleta das imagens.

Essas alterações irão dar maior robustez ao modelo, pois esse processo irá forçar a Rede Neural a buscar melhores características gerais que possam identificar a doença, evitando possíveis vícios em relação a padrões que não sejam das doenças, mas sim do contexto em que as fotos foram tiradas. Esse procedimento é importante para a aplicabilidade desse projeto, já que aumentará a precisão para casos de uso do modelo em imagens externas ao conjunto de dados que estamos utilizando.

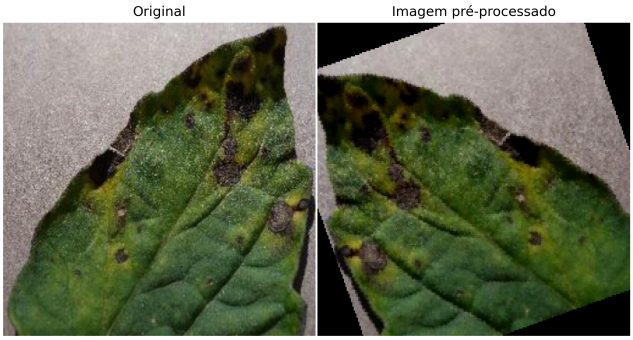


Figura 14: Pré-processamentos aplicados. Fonte: Próprios autores.

Neste projeto, o uso de *Padding* nas imagens não será adotado. Esse processo consiste em adicionar uma borda de píxeis zeros ao redor de toda a imagem, com o objetivo de estendê-la sem alterar suas características, permitindo que o filtro utilizado seja aplicado também nas bordas. No entanto, essa etapa é considerada desnecessária, uma vez que as folhas estão centralizadas nas imagens e as bordas, na maioria das vezes, correspondem ao plano de fundo ou não contêm informações relevantes. O centro da imagem já possui os dados necessários para a classificação das doenças.

Outro processo realizado trata-se da normalização, que é o processo de ajustar os valores dos píxeis da imagem para uma escala específica, geralmente visando melhorar o desempenho e a estabilidade dos modelos de aprendizado

profundo. A normalização utilizada muda a escala dos píxeis de [0, 255] para a escala [-1, 1], e subtrai a média e divide pelo desvio padrão da distribuição dos píxeis. Esse processo evita que grandes variações nos valores dos píxeis causem gradientes explosivos ou desaparecidos durante o treinamento, e padroniza os dados de entrada para facilitar o aprendizado do modelo.

E por fim a imagem será transformada em um tensor em Python, um tensor é uma generalização de matrizes que pode ter dimensões arbitrárias. Imagens coloridas são convertidas em tensores tridimensionais: Altura, Largura e Canais (RGB). Transformar uma imagem em um tensor significa converter a representação da imagem (geralmente armazenada como *arrays* de píxeis) em uma estrutura numérica que pode ser processada eficientemente por *frameworks* de aprendizado profundo, como TensorFlow ou *PyTorch* que estamos utilizando.

Tensors são fundamentais para o treinamento dos modelos de rede neural pois otimizam o uso de hardware acelerado, como GPUs e TPUs, aumentando a eficiência e velocidade do treinamento. Além de facilitar o processo de *Backpropagation*, já que a estrutura de tensor permite a computação automática de gradientes, essencial para o cálculo do erro e ajuste dos pesos durante o treinamento.

Após isso, será realizado uma segmentação da imagem, separando a imagem da folha do *background*, e utilizando uma máscara de píxeis, o *background* será removido. Esse processo tem por objetivo remover informações desnecessárias para o treinamento, focando o aprendizado de máquina na folha presente na imagem. Os métodos de segmentação propostos serão apresentados para justificar a escolha de qual processo será utilizado em todo o conjunto de dados.

1) Segmentação por K-means: O processo de segmentação de imagem usando o algoritmo K-means é amplamente utilizado para separar diferentes regiões dentro de uma imagem, com base em características como cores ou intensidades de píxel, especialmente em casos de imagens com diferenças claras de intensidade ou cor. A geração e aplicação de máscaras permitem isolar a região de interesse de forma eficaz. No caso de segmentar uma imagem em *background* e *foreground*, o número de *clusters* (k) é definido como 2, e o modelo particiona os píxeis da imagem entre esses dois *clusters* [24].

Cada píxel é tratado como um ponto de dados com características, como intensidade ou valores RGB. E assim o algoritmo calcula as distâncias entre os píxeis e os centroides de cada *cluster*. O objetivo do K-means é minimizar a soma dos quadrados das distâncias entre os píxeis e seus centroides atribuídos, através do processo de inicialização aleatória dos *clusters*, atribui-se cada píxel ao *cluster* cujo centroide está mais próximo. Recalcula-se os centroides com base nos píxeis atribuídos a cada *cluster* e repete-se os dois passos anteriores até que centroides alcancem uma convergência.

Tendo em vista a existência de dois *clusters* para separar a imagem, os *clusters* serão definidos como uma classificação em 0 ou 1. Através desse processo é gerado uma máscara binária da imagem, sendo 0 o *background* que deve ser removido, e 1 o *foreground* que deve ser preservado. Dado a configuração dessa máscara, basta multiplicar a imagem píxel a píxel pela máscara para remover o *background*.

Figura 15: Segmentação por K-means. Fonte: Próprios autores.

2) Segmentação com Rembg: A função remove da biblioteca Rembg é uma ferramenta avançada de segmentação de imagens projetada para separar o primeiro plano (*foreground*) do fundo (*background*). Ela utiliza modelos pré-treinados baseados na arquitetura U-2-Net, entre outras, otimizados para diversas aplicações, incluindo segmentação de objetos, pessoas e elementos em imagens. A função é amplamente utilizada em tarefas de visão computacional, como a preparação de imagens para treinamento em CNNs, simplificando dados visuais para que o modelo possa se concentrar nos elementos mais relevantes [25].

A função recebe uma imagem como entrada e gera uma nova imagem com o fundo removido, mantendo apenas o objeto de interesse. Isso é alcançado por meio de redes neurais profundas especializadas em detecção de bordas e separação de regiões de contraste visual. Essa abordagem elimina a necessidade de pré-processamento manual, o que aumenta a eficiência do pipeline de machine learning.

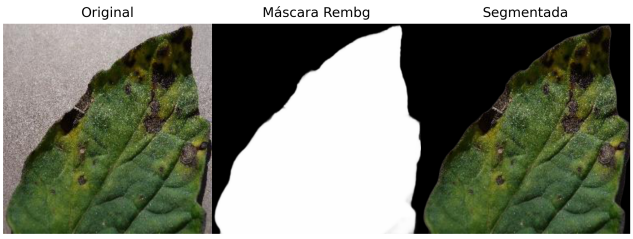


Figura 16: Segmentação por Rembg. Fonte: Próprios autores.

Remover o fundo de imagens pode melhorar significativamente a precisão de CNNs em tarefas como classificação e segmentação. Ao eliminar ruídos visuais e informações irrelevantes, a segmentação permite que o modelo concentre seus recursos computacionais na análise das características mais importantes do objeto de interesse. Isso pode resultar em um melhor desempenho, tempos de convergência reduzidos e menor probabilidade de *overfitting* [26].

Além disso, a padronização dos dados por meio da remoção de fundos complexos cria um conjunto de treinamento mais consistente, o que é essencial para modelos que precisam generalizar bem para novos dados.

C. Divisão do Dataset

Após as transformações, os dados serão divididos em conjuntos de treino, validação e teste de forma não estratificada,

ou seja, de maneira aleatória, sem garantir a manutenção proporcional das classes em cada conjunto. A divisão será realizada com a seguinte proporção: 80% dos dados serão destinados ao treino, 10% à validação e 10% ao teste.

D. Modelos de CNN

O modelo de Rede Neural Convolucional se baseia na especialização de uma Rede Neural *feedfoward*, adicionando camadas convolucionais e de *pooling* no inicio da rede neural. Essas camadas tem por objetivo, respectivamente, extrair características dessa imagem, e realçar essas caraterísticas para facilitar o processamento das próximas camadas.

Neste projeto será utilizado três modelos de CNN baseados em três arquiteturas populares de CNN, para que elas sejam comparadas em relação aos resultados obtidos e o custo de processamento da fase de treinamento. As arquiteturas utilizadas são a Residual Neural Network (ResNet), Visual Geometry Group (VGG) e Inception, que serão apresentadas em seguida. Esses modelos estão sendo obtidos através da biblioteca *Torchvision* do Python, é feito a implementação da arquitetura padrão, e moldamos ela para se adequar as necessidades e especificidades da demanda.

1) Residual Neural Network (ResNet): Desenvolvida pelos pesquisadores da Microsoft Kaiming He, Xiangyu Zhang, Shaoqing Ren, e Jian Sun em 2015, para reconhecimento de imagem, a arquitetura de rede neural profunda ResNet trata-se do modelo que introduziu o conceito de conexões residuais em redes neurais. A rede venceu o ImageNet Large Scale Visual Recognition Challenge (ILSVRC) daquele ano, com uma taxa de erro de 3,57%, a proposta dessa rede veio da necessidade de solucionar dois problemas que afetavam o desempenho das redes neurais quando elas se tornavam muito profundas [27].

Primeiramente, à medida que as redes ficavam mais profundas, os gradientes usados para treinar os pesos das camadas distantes da saída diminuíam exponencialmente, ou podiam crescer descontroladamente dependendo da configuração da rede. Esse desbalanceamento entre as camadas próximas e distantes da saída tornavam o treinamento ineficiente. O outro problema que a ResNet objetiva corrigir se trata da perda de desempenho na validação que a rede sofria quando se aumentava muito a profundidade das redes neurais comuns, devido a maior facilidade que essas redes comuns mais profundas tinham de gerar *overfit* e aumentar o erro na validação [28].

Para a utilização do modelo, será importada a biblioteca de modelos do *Torchvision*, que disponibiliza uma CNN com a arquitetura ResNet de 50 camadas (ResNet-50). O modelo será instanciado inicialmente com pesos pré-treinados. Além disso, a última camada do modelo, correspondente à camada de saída, será ajustada para se adequar à quantidade de classes presentes no conjunto de dados.

2) Visual Geometry Group (VGG): Criado pelo grupo Visual Geometry Group do departamento de Engenharia da Computação da Universidade de Oxford, esse modelo de rede neural se inspira em arquiteturas convencionais de CNN, mas com um acréscimo notável na profundidade da rede para melhorar a performance. A arquitetura do modelo VGG possui cinco blocos convolucionais intercalados por Max *Pooling* e ReLu, e um bloco com três camadas densamente conectadas. A arquitetura representou um grande avanço ao utilizar pequenos filtros convolucionais de tamanho 3x3 e aumentar a profundidade máxima funcional de uma rede neural, pois esse design permitiu aumentar a quantidade convoluções aplicadas as imagens [29].

O aumento na quantidade de convoluções permite uma melhora na receptividade de detalhes das imagens de entrada, gerando características mais poderosas para decisões na camada densamente conectada. Além disso, utilizar filtros pequenos para gerar mapas de características se reduz o número de parâmetros por camadas, principalmente nas camadas iniciais, o que resulta em uma menor proeminência da rede para gerar *overfitting* e maior viabilidade de treino.

Por padrão, a rede VGG usa imagens RGB com tamanho fixo de 224x224 como *input*, e realiza um pré-processamento subtraindo os píxeis pelo valor médio de RBG extraído das imagens de treinamento. A imagem então alimenta uma sequência de camadas convolucionais, e para preservar a resolução da imagem pós convolução utiliza-se um deslocamento fixo de um píxel e um preenchimento zero *padding* de um píxel. As camadas de max *pooling*, utilizadas após as convolucionais, são de tamanho 2x2, com um deslocamento de dois píxeis. Esse uso reduz os mapas de características das convoluções, tornando as representações resultantes invariantes a distorções e pequenas translações [27].

O modelo também será importado da biblioteca de modelos do *Torchvision*, que disponibiliza a arquitetura VGG-16, composta por 16 camadas. Ele será instanciado inicialmente com pesos pré-treinados. Será adicionada uma última camada densamente conectada, configurada com o número de classes do conjunto de dados, para a realização das classificações.

3) Inception: O modelo de Rede Neural Convolucional Inception, também conhecido como GoogLeNet, foi introduzido por pesquisadores do Google em 2014 no artigo “Going Deeper with Convolutions”. Ele trouxe avanços significativos em tarefas de visão computacional, como classificação de imagens, ao explorar o conceito de extração de recursos em múltiplas escalas. O nome ”Inception”é uma referência à ideia de ”redes dentro de redes”, inspirada no filme ”A Origem”.

O núcleo da arquitetura é o Inception Module, que realiza convoluções de diferentes tamanhos (1x1, 3x3, 5x5) simultaneamente, combinando os resultados por concatenação de canais. Isso permite que o modelo capture padrões em diferentes escalas espaciais em uma única camada, imitando como o olho humano percebe os detalhes em diferentes níveis. Além disso, camadas de *pooling* (normalmente *max-pooling*) são adicionadas para extrair características espaciais relevantes [7].

O Inception-v3 é uma evolução significativa na família de arquiteturas Inception, desenvolvida com base em melhorias observadas no Inception-v1 e Inception-v2. Apresentado em 2015, ele introduz técnicas avançadas para aumentar a eficiência computacional e a precisão em tarefas de visão computacional, especialmente na classificação de imagens.

Será então instanciado um modelo Inception-v3 obtido através da mesma biblioteca do *Torchvision*. Esse modelo será então alterado para que realize a classificação relativa ao número de classes utilizadas. Durante o uso do modelo, será

necessário redimensionar as imagens para a configuração de 299x299, pois essa é uma demanda da arquitetura do modelo, pois a primeira camada possui esse tamanho.

E. Otimizadores

Com a estruturação dos modelos definida, torna-se necessário implementar um otimizador para garantir a eficácia do processo de Machine Learning durante o treinamento. Utilizando a biblioteca Torch, estão disponíveis dois otimizadores que serão detalhados posteriormente: o Stochastic Gradient Descent (SGD) e o Adaptive Moment Estimation (ADAM). Esses otimizadores utilizarão os erros do treinamento para aprimorar a precisão dos modelos.

Otimizadores são algoritmos fundamentais no treinamento de redes neurais, pois ajustam os pesos do modelo para minimizar a função de perda. O progresso dos otimizadores acompanhou a evolução do aprendizado profundo, começando com métodos simples, como Gradiente Descendente (GD), até abordagens mais sofisticadas como Adam e RMSprop. Esses algoritmos são especialmente relevantes para CNNs, que possuem aplicações em visão computacional, como classificação de imagens e detecção de objetos [30].

Os otimizadores utilizam gradientes calculados durante o processo de *backpropagation* para ajustar os pesos do modelo. *Backpropagation*, por sua vez, aplica a regra da cadeia para propagar o erro da saída para as camadas anteriores, permitindo a atualização incremental dos pesos. A atualização segue a fórmula:

$$w_{t+1} = w_t - \eta \cdot \nabla L(w_t) \tag{1}$$

onde w_t representa os pesos atuais, η é a taxa de aprendizado, e $\nabla L(w_t)$ é o gradiente da função de perda em relação aos pesos. Diferentes otimizadores incorporam técnicas adicionais, como *momentum*, regularização ou adaptação da taxa de aprendizado, para melhorar a eficiência e estabilidade da convergência. Os otimizadores influenciam diretamente o desempenho de CNNs no treinamento e na generalização dos dados. Otimizadores como SGD com *momentum* são amplamente utilizados para evitar oscilações nos gradientes e alcançar convergência mais rápida. Métodos adaptativos como Adam ajustam a taxa de aprendizado com base no histórico dos gradientes, sendo eficazes em cenários com dados ruidosos ou funções de perda complexas. A escolha do otimizador varia de necessidade para necessidade, pois essa decisão leva em conta vários fatores, como a velocidade convergência e epócas de treinamento, generalização e complexidade computacional.

1) Stochastic Gradient Descent (SGD): O otimizador SGD (Stochastic Gradient Descent) é uma ferramenta fundamental na otimização de redes neurais, especialmente no treinamento de modelos convolucionais (CNNs). Ele é implementado na biblioteca *PyTorch* por meio do módulo torch.optim, permitindo ajustar os pesos dos modelos para minimizar a função de perda.

O SGD é uma variante do gradiente descendente que processa pequenas amostras de dados (*minibatches*) em vez de todo o conjunto de treinamento. Isso reduz o custo computacional por iteração, permitindo atualizações mais frequentes e potencialmente mais rápidas. Em *PyTorch*, sua implementação pode incluir ajustes como *momentum*, que acelera o treinamento ao manter uma direção persistente no espaço de parâmetros, e regularização L2, que combate *overfitting* ao penalizar valores grandes de pesos [31].

Um exemplo de configuração básica do SGD é demonstrada no código a seguir:

```
optimizer = torch.optim.SGD(model.parameters()
    , lr=0.01, momentum=0.9, weight_decay=0.1)
```

Aqui, lr é a taxa de aprendizado, *momentum* adiciona inércia às atualizações, e weight_decay implementa regularização L2.

O SGD utiliza o método de *backpropagation* para calcular os gradientes dos pesos em relação à perda. Esses gradientes orientam as atualizações dos parâmetros. Em uma CNN, o fluxo de cálculo começa com a propagação dos dados pela arquitetura de camadas convolucionais e totalmente conectadas. A perda é então calculada, e o *backpropagation* ajusta os filtros das camadas convolucionais para melhorar a detecção de características relevantes.

Durante o treinamento do modelo, o SGD atua no processamento dos *minibatches* do conjunto de treinamento, em seguida calcula a perda e os gradientes do treinamento. Com essas informações, o SGD atualizará os pesos utilizando a fórmula:

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L}{\partial w_t} \tag{2}$$

Onde η é a taxa de aprendizado, w são os pesos, e L é a perda.

O SGD é amplamente utilizado em CNNs devido à sua simplicidade e eficácia. Ele permite um ajuste gradual dos parâmetros, o que é ideal para capturar padrões complexos em dados visuais. Contudo, por operar em *minibatches*, a variação dos gradientes introduz ruído, o que pode ajudar o modelo a escapar de mínimos locais ruins, mas também pode causar flutuações desnecessárias [32].

Em comparação com otimizadores como Adam, que ajustam a taxa de aprendizado dinamicamente, o SGD pode ser mais lento para convergir. No entanto, ele tende a generalizar melhor, especialmente em modelos grandes, como ResNets. Comparado com o Adam, o SGD requer mais ajustes manuais para taxas de aprendizado e hiperparâmetros, mas pode alcançar soluções mais generalizáveis em problemas específicos.

2) Adaptive Moment Estimation (ADAM): O otimizador ADAM (Adaptive Moment Estimation) foi introduzido em 2014 por Kingma e Ba para abordar limitações de métodos anteriores como SGD, AdaGrad e RMSProp. Ele combina as vantagens do decaimento adaptativo da taxa de aprendizado do AdaGrad com a aceleração do momento do RMSProp, tornando-se uma escolha popular em tarefas de aprendizado profundo, como visão computacional e processamento de linguagem natural [33].

O otimizador ADAM utiliza dois conceitos principais, em princípio ele realiza uma média móvel dos gradientes anteriores, o momento da média, e em um segundo momento, de variância, ele realiza a média móvel dos quadrados dos gradientes. Esses momentos são atualizados em cada iteração

com parâmetros de suavização β_1 e β_2, geralmente 0,9 e 0,999, respectivamente. Para corrigir viés inicial, são aplicadas correções aos momentos antes de ajustar os pesos. O ADAM então combina esses valores para ajustar os parâmetros do modelo com taxas de aprendizado adaptativas, garantindo estabilidade e eficiência [34].

Em CNNs, o ADAM permite atualizações eficientes mesmo em *loss surfaces* complexas, acelerando o treinamento. Durante o *backpropagation*, o otimizador ajusta os pesos com base nas derivadas do erro, adaptando a taxa de aprendizado para diferentes parâmetros. Isso é particularmente útil em CNNs profundas e em tarefas com dados variados, como classificação de imagens.

F. Treinamento do Modelo

O treinamento do modelo se baseia na execução das etapas preparadas anteriormente, durante o pré-processamento e construção da arquitetura da CNN. Primeiramente se divide o conjunto de dados em três conjuntos, treino, validação e teste. Para dar início ao treino do modelo, se configura o modelo para treino, e se alimenta o modelo com as imagens do conjunto de treino.

Para gerar uma previsão teste se divide o conjunto de treino em *batchs*, e esses *batchs* são passados um a um através do *Foward Pass* pelo modelo. Cada *batch* gera uma classificação prevista para a quantidade de imagens referente ao tamanho do *batch*, essas classificações do treino são então comparadas a classificação verdadeira da imagem. O erro entre a classificação verdadeira e a obtida pelo modelo é calculado através da Cross-Entropy Loss, e é utilizado para gerar um gradiente que será propagado de maneira reversa através dos neurônios do modelo, levando em consideração os pesos dos neurônios.

Esses gradientes são então utilizados pelo otimizador para atualizar os pesos do neurônio, de modo a corrigir o funcionamento do modelo, levando a classificações mais precisas. A correção dos pesos é feita com base na fórmula (2).

O processo de *forward pass*, cálculo do erro, *backpropagation* e atualização dos pesos é repetido por várias iterações (ou *epochs*), até que a função de perda convirja ou atinja um valor aceitável. Durante o treinamento, o desempenho no conjunto de validação é monitorado para evitar *overfitting*.

G. Métodos de Teste

Após o treinamento do modelo, é necessário desenvolver métodos para avaliar a eficácia das classificações realizadas. Esses métodos permitem não apenas analisar a porcentagem de acertos do sistema, mas também identificar onde ocorrem os erros e de que forma eles se manifestam.

A análise mais aprofundada dos erros possibilita uma compreensão detalhada dos resultados e orienta possíveis modificações no modelo para aprimorar sua performance. Além disso, fornece métricas mais robustas para a comparação entre os modelos utilizados. Os métodos de análise de erros aplicados incluem a Matriz de Confusão, a Sensibilidade, a Especificidade e o F1-Score.

1) Matriz de Confusão: A matriz de confusão é uma ferramenta para a avaliação de modelos de classificação, fornecendo uma representação visual e quantitativa do desempenho. Para modelos de CNNs, que são amplamente aplicados em tarefas de visão computacional, a matriz de confusão oferece uma visualização detalhada dos erros do modelo [35].

A matriz de confusão organiza as previsões do modelo em relação às classes reais. Em problemas de classificação binária, a matriz possui quatro entradas principais: o TP (*True Positives*), que representa o número de exemplos corretamente classificados como positivos; o FN (*False Negatives*), que indica o número de exemplos positivos incorretamente classificados como negativos; o TN (*True Negatives*), correspondente ao número de exemplos corretamente classificados como negativos; e o FP (*False Positives*), que refere-se ao número de exemplos negativos incorretamente classificados como positivos.

Essa matriz também pode ser expandida para problemas de classificação com múltiplas classes, onde cada classe possui uma linha e uma coluna correspondentes. Em tarefas de classificação de imagens, a matriz de confusão é útil para identificar padrões de erro, como classes frequentemente confundidas entre si, o que pode indicar similaridade visual entre as imagens analisadas. Na detecção de objetos, a matriz auxilia na compreensão do desempenho do modelo em diferentes categorias. Além disso, é amplamente utilizada como base para calcular métricas como acurácia, precisão, sensibilidade (*recall*) e especificidade.

A matriz de confusão representa os resultados da classificação com base na comparação entre os valores reais (*ground truth*) e os valores previstos pelo modelo. Formalmente, considera-se um conjunto de dados D com n amostras e C classes.

Na estrutura da matriz, cada entrada M_{ij} indica o número de exemplos da classe verdadeira i classificados como j. E a soma das diagonais principais (M_{ii}) corresponde ao número total de predições corretas.

2) Sensibilidade (Recall): A sensibilidade é uma métrica fundamental para avaliar o desempenho de modelos de classificação, especialmente em aplicações de visão computacional onde a identificação de falsos negativos é crítica. A sensibilidade mede a proporção de verdadeiros positivos (TP) em relação ao total de elementos que realmente pertencem à classe positiva (TP + FN) [36]. Em termos matemáticos, é definida como:

$$\text{Sensibilidade} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

Uma alta sensibilidade indica que o modelo é eficaz em detectar a classe positiva, o que é essencial em contextos onde a falha em identificar exemplos positivos pode ter consequências severas, como na detecção de doenças.

Na Visão Computacional, a sensibilidade avalia a capacidade do modelo de localizar corretamente objetos de interesse em uma imagem, minimizando os falsos negativos. Embora a sensibilidade seja crítica, é geralmente usada em combinação com outras métricas, como a especificidade e a precisão, para obter uma visão equilibrada do desempenho do modelo.

A sensibilidade deriva da análise da matriz de confusão, primeiramente é construída a matriz de confusão para a classificação do modelo. Então é identificado os TP e FN, e aplicada a fórmula de sensibilidade. Quando os dados têm classes altamente desbalanceadas, a sensibilidade é mais útil que a precisão para avaliar o desempenho em classes minoritárias.

3) Especificidade: A especificidade é uma métrica amplamente usada para avaliar o desempenho de modelos de classificação, particularmente em cenários onde a distinção entre classes negativas e positivas é crítica. Na visão computacional, seu papel é especialmente relevante em problemas com classes desbalanceadas ou em aplicações onde minimizar falsos positivos é essencial [36].

Ela mede a proporção de verdadeiros negativos (TN) corretamente identificados em relação ao total de elementos que pertencem à classe negativa (TN + FP). Sua fórmula matemática é dada pela equação 4.

$$\text{Especificidade} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{4}$$

A Especificidade é essencial para avaliar a habilidade do modelo em rejeitar adequadamente instâncias da classe negativa. Enquanto a sensibilidade avalia a habilidade do modelo em identificar corretamente a classe positiva, a especificidade mede sua competência em excluir a classe negativa. Juntas, oferecem uma visão equilibrada do desempenho do modelo.

Trata-se de uma métrica indispensável na avaliação de modelos de visão computacional, oferecendo uma análise sobre como o modelo lida com a classe negativa. Juntamente com a sensibilidade, permite uma avaliação equilibrada, sendo especialmente relevante em contextos onde a minimização de falsos positivos é crítica.

4) F1-Score: O F1-Score é uma métrica amplamente utilizada na avaliação de modelos de classificação, especialmente em problemas com classes desbalanceadas. Ele fornece uma medida equilibrada entre a precisão e o *recall*, sendo particularmente útil para avaliar o desempenho em tarefas críticas onde ambas as métricas devem ser consideradas [36].

O **F1-Score** é a média harmônica entre a precisão e a sensibilidade, e é calculado pela equação 5.

$$F1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{5}$$

Precisão: Proporção de verdadeiros positivos (TP) em relação a todas as predições positivas.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{6}$$

Recall: Proporção de verdadeiros positivos (TP) em relação a todas as instâncias reais da classe positiva. É cálculada com a fórmula 3.

O F1-Score varia de 0 a 1, sendo 1 o desempenho ideal. Ele é particularmente útil em cenários onde há *trade-offs* entre precisão e *recall*. A média harmônica penaliza casos em que uma métrica (precisão ou *recall*) é muito baixa, garantindo que o F1-Score seja significativo apenas quando ambas as métricas têm valores altos.

5) Acurácia: A acurácia é uma das medidas obtidas através da matriz de confusão. Ela aponta uma performance geral sobre o modelo mas acaba não sendo muito eficaz para para datasets que apresentem distribuição desbalanceada das classes, mesmo assim ainda apresenta resultados que podem ser análisadas e discutidas [35]. Sua fórmula segue a equação 7.

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

V. Resultados

Com base no treinamento, os resultados são apresentados de acordo com as configurações específicas de cada modelo. Os modelos foram organizados conforme o otimizador utilizado, considerando também a configuração dos hiperparâmetros.

Os otimizadores avaliados foram ADAM e SGD, ambos com pré-processamento padrão realizado pelo Rembg. A taxa de aprendizado (learning rate) foi ajustada para os valores de 0.01, 0.001 e 0.0005. As métricas de avaliação incluem Matriz de Confusão, Sensibilidade, Especificidade e F1-Score, permitindo uma análise detalhada do desempenho dos modelos.

Adicionalmente, foi gerado um mapa de ativação com base em uma mesma imagem para todos os modelos. Esse mapa representa as características mais relevantes detectadas pela última camada do modelo, que são utilizadas como elementos-chave na classificação realizada pelas camadas densamente conectadas.

A. ResNet-50 com ADAM

As Figuras 17 até 20 apresentam os resultados do modelo ResNet, treinado com o otimizador Adam, taxa de aprendizado de 0.0005 e sem decaimento explícito. Os mapas de ativação da Figura 20 mostram que o modelo foca em regiões específicas das folhas para a classificação, com as áreas mais relevantes destacadas em vermelho. A análise visual dos mapas sugere que o modelo é capaz de identificar padrões críticos, mesmo na ausência de decaimento, com boa precisão em áreas chave das imagens.

A matriz de confusão na Figura 18 revela uma acurácia geral de 98.94%, destacando um bom desempenho do modelo. As classes *Tomato Target Spot*, *Tomato Leaf Mold* e *Tomato Spider mites Two spotted spider mite* apresentaram alta precisão, com erros mínimos entre as classes. As divergências ocorreram em casos isolados, como na classe *Tomato YellowLeaf Curl Virus*, onde algumas amostras foram classificadas incorretamente. Os gráficos de perda e acurácia, mostrados na Figura 17, indicam que o modelo converge rapidamente, com a perda de treinamento e validação estabilizando-se em valores baixos. A acurácia de treinamento e validação aproxima-se de 99% ao longo das épocas, com flutuações mínimas e consistência entre as métricas.

Ainda nos gráficos da Figura 17 é notado uma leve tendência de *overfitting* durante a décima época, evidenciada pelo aumento da acurácia no treinamento enquanto a acurácia de validação apresentou uma queda. Esse comportamento, entretanto, foi temporário, sendo corrigido nas épocas subsequentes.

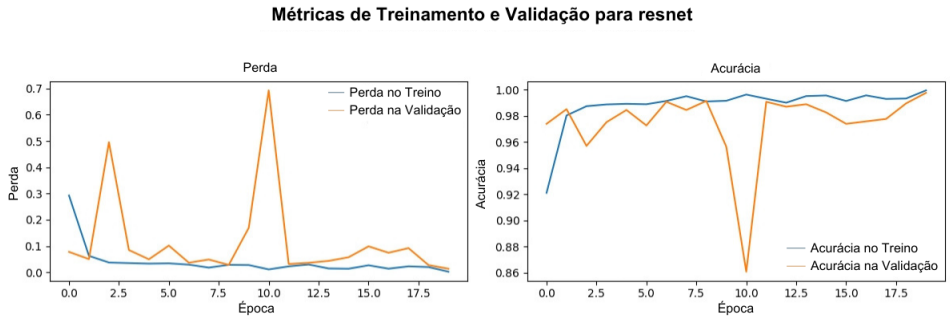
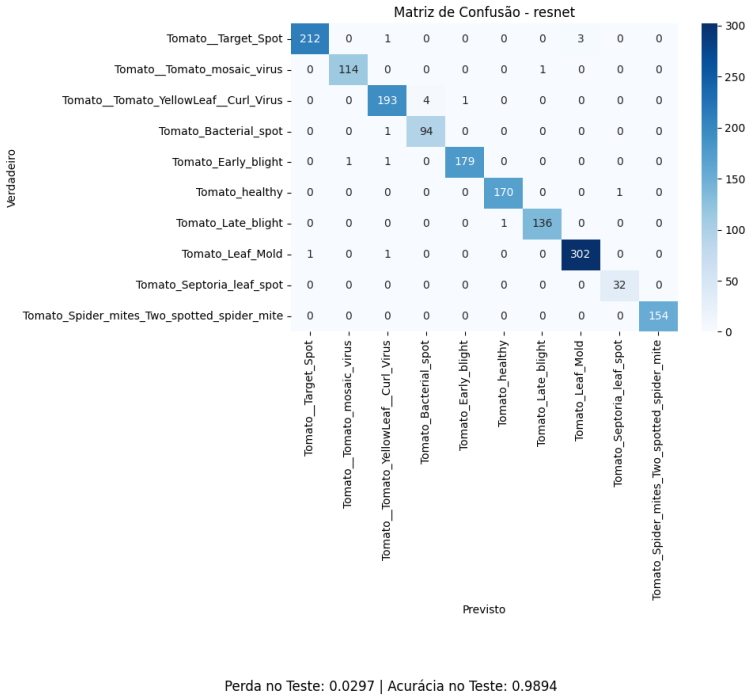


Figura 17: Métrica do treinamento ResNet-50 com Adam e taxa de aprendizado 0.0005. Fonte: Próprios autores.



Perda no Teste: 0.0297 | Acurácia no Teste: 0.9894

Figura 18: Matriz de Confusão do ResNet-50 com Adam, sem taxa de decaimento explícito. Fonte: Próprios autores.

O modelo retomou um padrão de melhoria contínua até o final do treinamento.

A Tabela I apresenta as métricas de avaliação para as diferentes classes de doenças e condições em tomates para este modelo. Observa-se que a classe *Tomato Spider Mites Two Spotted Spider Mite* possui métricas perfeitas (1.0000) para precisão, sensibilidade, especificidade e F1-Score.

As classes *Tomato Target Spot*, *Tomato Mosaic Virus*, *Tomato Early Blight*, *Tomato Healthy*, *Tomato Late Blight* e *Tomato Leaf Mold* também apresentam métricas muito altas, com precisão, sensibilidade, especificidade e F1-Score acima de 0.9815. Já a classe *Tomato Bacterial Spot* e *Tomato Septoria Leaf Spot* apresenta métricas ligeiramente inferiores, com precisão de 0.9592 e 0.9697 respectivamente, e F1-Score de 0.9741 e 0.9846, indicando uma maior dificuldade do modelo em identificar corretamente todos os casos positivos.

Por fim, a classe *Tomato Yellow Leaf Curl Virus* apresenta resultados sólidos, com a precisão em 0.9797, sensibilidade de 0.9797, especificidade 0.9972 e F1-Score de 0.9772.

Classe	Precisão	Sensibilidade	Especificidade	F1-Score
Tomato_Target_Spot	0.9953	0.9815	0.9993	0.9883
Tomato_Tomato_mosaic_virus	0.9913	0.9913	0.9993	0.9913
Tomato_Tomato_YellowLeaf_Curl_Virus	0.9797	0.9747	0.9972	0.9772
Tomato_Bacterial_spot	0.9592	0.9895	0.9973	0.9741
Tomato_Early_blight	0.9944	0.9890	0.9993	0.9917
Tomato_healthy	0.9942	0.9942	0.9993	0.9942
Tomato_Late_blight	0.9927	0.9927	0.9993	0.9927
Tomato_Leaf_Mold	0.9902	0.9934	0.9977	0.9918
Tomato_Septoria_leaf_spot	0.9697	1.0000	0.9994	0.9846
Tomato_Spider_mites_Two_spotted_spider_mite	1.0000	1.0000	1.0000	1.0000

Tabela I: Métricas de avaliação para as classes do modelo ResNet-50 utilizando Adam. Fonte: Próprios autores

Foi obtido o mapa de ativação do melhor modelo para identificar as porções da imagem consideradas as melhores características-chave para a classificação da folhagem, esses resultados estão presentes na Figura 19. Vale destacar que, devido ao processo de redimensionamento do mapa de características para adequá-lo às dimensões da imagem original da folhagem, podem ocorrer distorções no posicionamento aproximado dos píxeis do mapa de ativação.

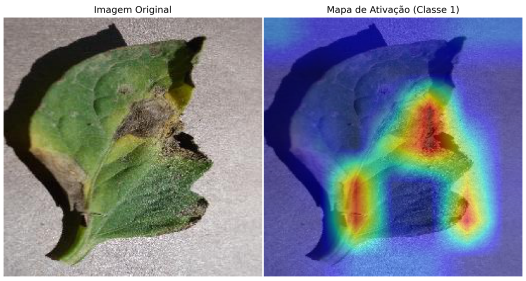


Figura 19: Mapa de Ativação do ResNet-50, lr = 0.0005. Fonte: Próprios autores.

Pode-se confirmar então, uma centralização das características chaves na folha. Nota-se a semelhança do mapa com a capacidade humana de visualizar e extrair informações da folha, onde é possível observar a doença se manifestando. É possível notar ainda a evolução do mapa de ativação de acordo com a alteração da taxa de aprendizado.

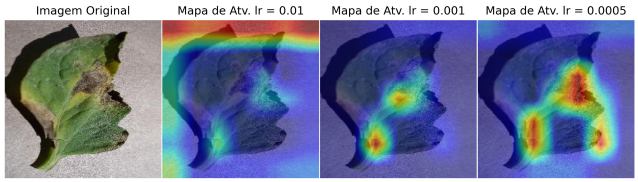


Figura 20: Mapas de Ativação dos ResNet-50. Fonte: Próprios autores.

B. *ResNet-50 com SGD*

As Figuras 21 até a 24 apresentam os resultados do modelo ResNet-50, treinado com o otimizador SGD, taxa de aprendizado de 0.01 e decaimento explícito de 0.01.

A matriz de confusão apresentada na Figura 22 revela uma acurácia geral de 99.31%, evidenciando a alta performance do modelo. Classes como *Tomato Target Spot*, *Tomato Leaf Mold* e *Tomato Spider mites Two spotted spider mite* se destacaram com excelentes resultados, com a grande maioria das amostras corretamente classificadas. No entanto, alguns pequenos desvios de predição podem ser notados, especialmente na classe *Tomato Late blight*. Além disso, a Figura 21 ilustra a rápida convergência da perda de treinamento e validação para valores baixos, enquanto a acurácia permanece alta tanto para o treinamento quanto para a validação, indicando um processo de aprendizado eficiente e uma boa capacidade de generalização do modelo.

A Tabela II apresenta as métricas de avaliação para as diferentes classes no modelo em questão. Destacam-se as classes *Tomato Septoria leaf spot* e *Tomato Spider mites Two spotted spider mite*, que obtiveram valores máximos em todas as métricas, incluindo um F1-Score de 1.000. Esses resultados indicam a alta eficácia do modelo na identificação precisa dessas condições, com total acerto na detecção tanto dos verdadeiros positivos quanto dos verdadeiros negativos.

Além disso, outras classes como *Tomato Target Spot*, *Tomato Tomato mosaic virus* e *Tomato Late blight* também apresentam ótimos resultados, com F1-Scores superiores a 0.9828, refletindo um desempenho muito equilibrado entre precisão e sensibilidade. Embora as classes *Tomato Bacterial spot* e *Tomato Leaf Mold* apresentem um desempenho ligeiramente inferior em termos de sensibilidade, elas ainda têm valores muito próximos de 1.0000 em precisão e especificidade. No geral, o modelo ResNet-50 com SGD demonstrou ser altamente eficiente e confiável para a classificação das diferentes doenças e condições em tomates, com a maioria das classes apresentando resultados excepcionais.

Classe	Precisão	Sensibilidade	Especificidade	F1-Score
Tomato_Target_Spot	0.9955	0.9910	0.9993	0.9932
Tomato_Tomato_mosaic_virus	0.9744	0.9913	0.9980	0.9828
Tomato_Tomato_YellowLeaf_Curl_Virus	0.9939	0.9939	0.9993	0.9939
Tomato_Bacterial_spot	1.0000	0.9808	1.0000	0.9903
Tomato_Early_blight	0.9838	0.9945	0.9979	0.9891
Tomato_healthy	0.9942	0.9942	0.9993	0.9942
Tomato_Late_blight	0.9858	0.9858	0.9986	0.9858
Tomato_Leaf_Mold	1.0000	0.9967	1.0000	0.9984
Tomato_Septoria_leaf_spot	1.0000	1.0000	1.0000	1.0000
Tomato_Spider_mites_Two_spotted_spider_mite	1.0000	1.0000	1.0000	1.0000

Tabela II: Métricas de avaliação para as classes do modelo ResNet-50 utilizando SGD. Fonte: Próprios autores

C. *VGG-16 com ADAM*

As Figuras 25 até a 28 apresentam os resultados do modelo VGG-16, treinado com o otimizador Adam, uma taxa de aprendizado de 0.0005 e sem decaimento explícito. A Figura 28 ilustra o mapa de ativação gerado para uma amostra específica, destacando as áreas mais importantes que o modelo utilizou na tomada de decisão. Assim como nos modelos anteriores é possível notar as regiões em vermelho que indicam maior relevância, com foco nas áreas mais afetadas da folha.

A Figura 25 exibe as métricas de treinamento e validação ao longo das épocas. A perda diminui rapidamente nas primeiras iterações e se estabiliza em valores baixos, tanto para o treinamento quanto para a validação, sugerindo uma boa convergência do modelo. Já a acurácia aumenta de forma consistente, atingindo aproximadamente 96% na validação ao final do treinamento. Pequenas flutuações são observadas, mas o comportamento geral das métricas indica que o modelo possui um aprendizado estável e eficiente, com uma boa capacidade de generalização.

Por fim, a matriz de confusão mostrada na Figura 26 confirma o desempenho do modelo, com uma acurácia geral de 96.63%. As classes *Tomato Leaf Mold* e *Tomato Target Spot* apresentaram os maiores acertos, com uma quantidade mínima de erros. Apesar disso, há confusão em classes como *Tomato Tomato mosaic virus* e *Tomato healthy*, onde foram identificados falsos positivos e falsos negativos em pequena proporção. No geral, o modelo VGG-16 mostrou-se eficiente na classificação das imagens, mesmo sem o uso de uma taxa de decaimento, alcançando um desempenho robusto e consistente.

A Tabela III apresenta as métricas de desempenho para cada classe no modelo VGG-16, com taxa de aprendizado de 0.0005. Observa-se que a maioria das classes alcançou altos valores de sensibilidade, especificidade, precisão e F1-

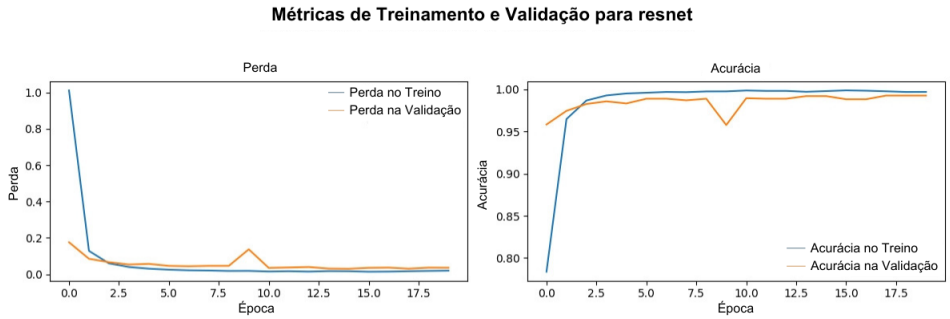
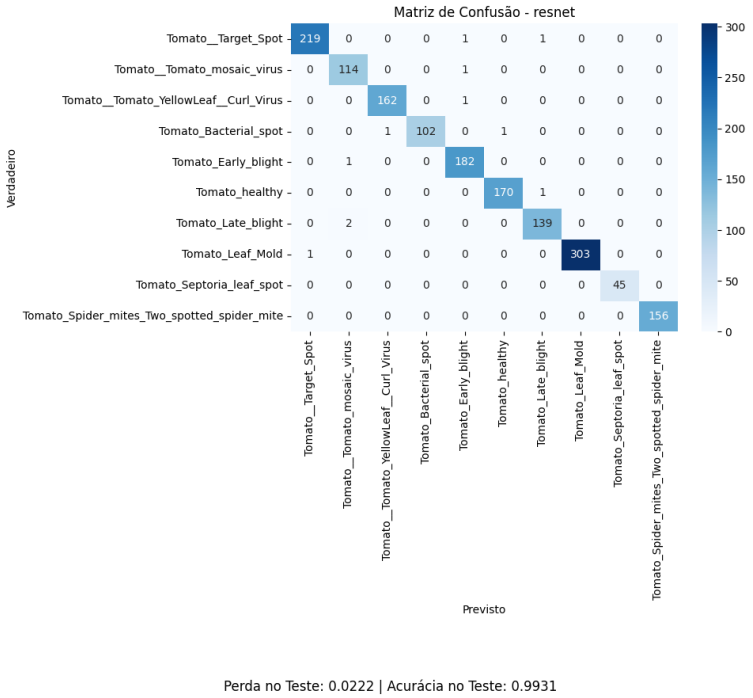


Figura 21: Métrica do treinamento ResNet-50 com SGD e taxa de aprendizado 0.01. Fonte: Próprios autores.



Perda no Teste: 0.0222 | Acurácia no Teste: 0.9931

Figura 22: Matriz de Confusão do ResNet-50 com SGD, com taxa de decaimento explícito. Fonte: Próprios autores.

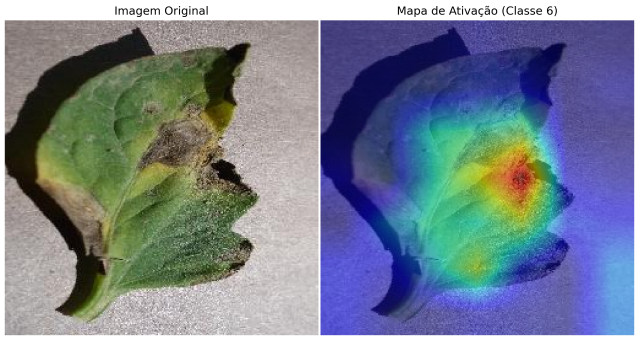


Figura 23: Mapa de Ativação do ResNet-50, lr = 0.01, weight_decay = 0.01. Fonte: Próprios autores.

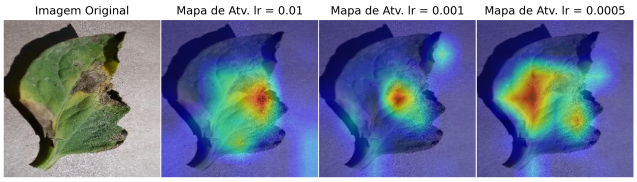


Figura 24: Mapas de Ativação dos ResNet-50. Fonte: Próprios autores.

Score, indicando um desempenho consistente do modelo em identificar corretamente as imagens das diferentes categorias.

A classe *Tomato Leaf Mold* destaca-se com os maiores valores, obtendo 0.9938 de sensibilidade e 0.9954 de F1-Score, sugerindo uma excelente capacidade do modelo em identificar os casos positivos dessa classe com alta precisão. Da mesma forma, a classe *Tomato Target Spot* apresentou uma alta sensibilidade de 0.9755, embora a precisão tenha sido ligeiramente inferior (0.9900), o que resultou em um F1-Score de 0.9827. Por outro lado, a classe *Tomato Late Blight* apresentou a menor sensibilidade (0.9084) e um F1-Score de

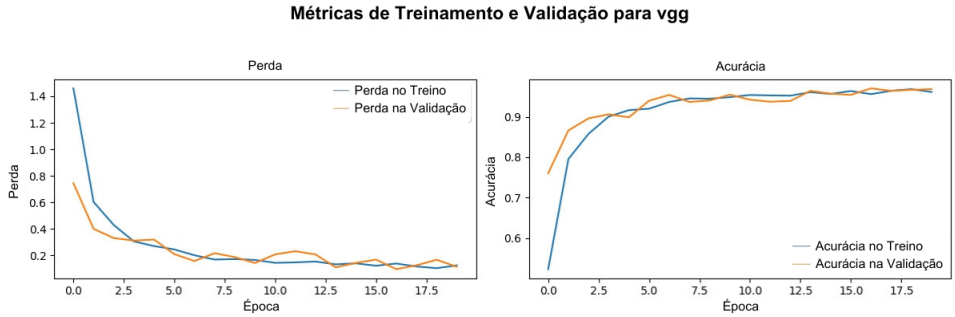
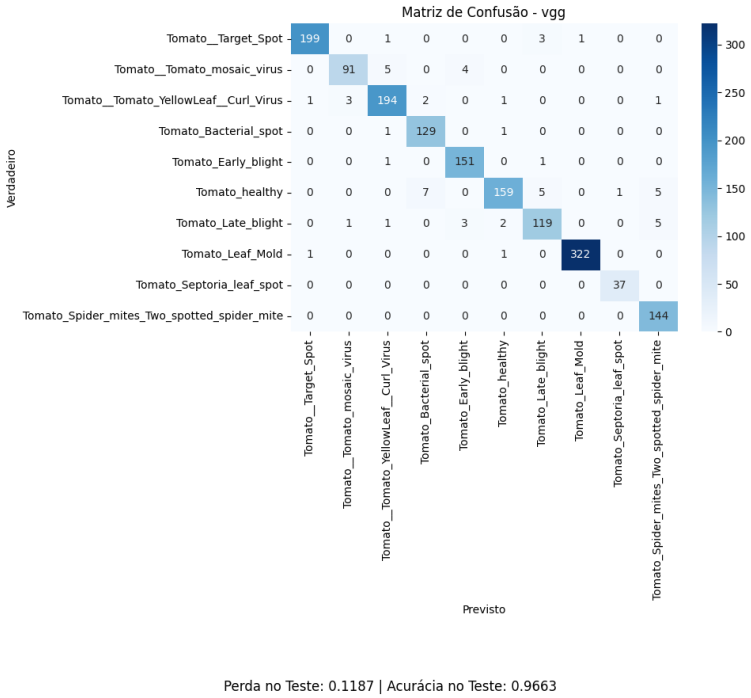


Figura 25: Métrica do treinamento VGG-16 com Adam e taxa de aprendizado 0.0005. Fonte: Próprios autores.



Perda no Teste: 0.1187 | Acurácia no Teste: 0.9663

Figura 26: Matriz de Confusão do VGG-16 com Adam, sem taxa de decaimento explícito. Fonte: Próprios autores.

0.9189, indicando que essa classe foi a mais desafiadora para o modelo, com uma maior proporção de falsos negativos.

Classes como *Tomato Early blight* e *Tomato Bacterial spot* mantiveram um desempenho equilibrado, com F1-Scores de 0.9711 e 0.9591, respectivamente, refletindo a capacidade do modelo de balancear sensibilidade e precisão para esses casos. Já a classe *Tomato Septoria leaf spot* apresentou uma precisão limitada (0.9737), mas com uma especificidade elevada (0.9994) e sensibilidade de 1.0000, demonstrando uma tendência do modelo em minimizar falsos positivos nessa classe.

No geral, os resultados mostram que o modelo VGG-16 com a configuração utilizada apresenta um desempenho robusto, com F1-Score superior a 0.9189 para todas as classes. Pequenas variações nos valores de sensibilidade e precisão em classes específicas sugerem que o modelo ainda encontra desafios em classes com características visuais mais sutis ou semelhantes.

Ao analisar todos os três mapas de ativação obtidos na

Classe	Precisão	Sensibilidade	Especificidade	F1-Score
Tomato_Target_Spot	0.9900	0.9755	0.9986	0.9827
Tomato_Tomato_mosaic_virus	0.9579	0.9100	0.9973	0.9333
Tomato_Tomato_YellowLeaf_Curl_Virus	0.9557	0.9604	0.9936	0.9580
Tomato_Bacterial_spot	0.9348	0.9847	0.9939	0.9591
Tomato_Early_blight	0.9557	0.9869	0.9952	0.9711
Tomato_healthy	0.9695	0.8983	0.9965	0.9326
Tomato_Late_blight	0.9297	0.9084	0.9939	0.9189
Tomato_Leaf_Mold	0.9969	0.9938	0.9992	0.9954
Tomato_Septoria_leaf_spot	0.9737	1.0000	0.9994	0.9867
Tomato_Spider_mites_Two_spotted_spider_mite	0.9290	1.0000	0.9925	0.9632

Tabela III: Métricas de avaliação para as classes do modelo VGG-16 utilizando Adam. Fonte: Próprios autores

Figura 28, pode-se perceber a tendência a procurar características maiores quanto maior é a taxa de aprendizado. Essa configuração indica um bom treinamento do modelo para uma taxa de aprendizado de 0.0005, porém ainda inferior ao modelo de Resnet-50.

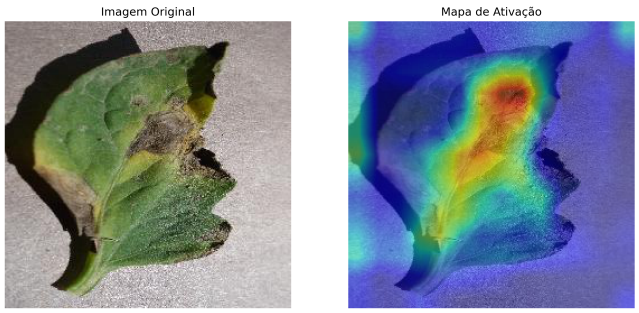


Figura 27: Mapa de Ativação do VGG-16, lr = 0.0005. Fonte: Próprios autores. Fonte: Próprios autores

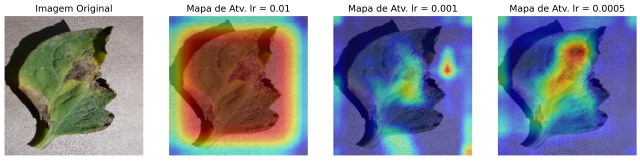


Figura 28: Mapas de Ativação dos VGG-16. Fonte: Próprios autores.

D. VGG-16 com SGD

As Figuras 29 até 32 apresentam os resultados do modelo VGG-16, treinado com o otimizador SGD, taxa de aprendizado de 0.01 e decaimento de 0.01. Os mapas de ativação da Figura 32 destacam as regiões mais relevantes da imagem para a classificação. Observa-se que, com a redução do decaimento, as áreas de ativação tornam-se mais suaves e focalizadas, indicando que o modelo consegue identificar características críticas das folhas de maneira progressivamente eficiente.

Na Figura 29, os gráficos de perda e acurácia indicam boa convergência do modelo. A perda de treinamento e validação estabiliza em valores baixos, com pequenas flutuações na validação após a época 10. A acurácia de treinamento atinge valores próximos a 95%, enquanto a acurácia de validação se mantém próxima de 90%, demonstrando que o modelo generaliza bem sem evidências de *overfitting* ou *underfitting*.

A matriz de confusão da Figura 30 mostra uma acurácia geral de 90.95%, com as classes *Tomato Leaf Mold* e *Tomato Spider mites Two spotted spider mite* apresentando as melhores taxas de acerto. O maior desvio ocorreu entre as classes *Tomato Target Spot* e *Tomato Tomato mosaic virus*.

A tabela IV apresenta as métricas de avaliação para as diferentes classes de doenças e condições em tomates, utilizando o modelo VGG. Observa-se que algumas classes, como *Tomato Target Spot*, *Tomato Bacterial Spot* e *Tomato Septoria Leaf Spot*, possuem precisão e especificidade perfeitas (1.0000), o que indica que o modelo é altamente eficaz em identificar corretamente os negativos e positivos para essas classes. No entanto, a sensibilidade dessas classes é relativamente mais baixa, especialmente para *Tomato Target Spot*, o que sugere que o modelo pode ter dificuldades em identificar todos os casos positivos dessas doenças.

Por outro lado, classes como *Tomato Tomato mosaic virus* e *Tomato Tomato YellowLeaf Curl Virus* apresentam alta sensibilidade, indicando boa capacidade de identificar corretamente os casos positivos. No entanto, sua precisão é mais baixa, refletindo uma maior taxa de falsos positivos. O F1-Score, por sua vez, é alto para a maioria das classes, indicando que o modelo tem um desempenho equilibrado entre precisão e sensibilidade.

Classe	Precisão	Sensibilidade	Especificidade	F1-Score
Tomato_Target_Spot	1.0000	0.6402	1.0000	0.7806
Tomato_Tomato_mosaic_virus	0.6503	0.9688	0.9668	0.7782
Tomato_Tomato_YellowLeaf_Curl_Virus	0.9116	0.9751	0.9864	0.9423
Tomato_Bacterial_spot	1.0000	0.9479	1.0000	0.9733
Tomato_Early_blight	0.8812	0.9889	0.9831	0.9319
Tomato_healthy	0.9368	0.9588	0.9923	0.9477
Tomato_Late_blight	0.9268	0.8636	0.9939	0.8941
Tomato_Leaf_Mold	0.9652	0.9591	0.9914	0.9621
Tomato_Septoria_leaf_spot	1.0000	0.8571	1.0000	0.9231
Tomato_Spider_mites_Two_spotted_spider_mite	0.9326	0.9881	0.9916	0.9595

Tabela IV: Métricas de avaliação para as classes do modelo VGG-16 utilizando SGD. Fonte: Próprios autores

E. Inception-v3 com ADAM

As Figuras 33 até a 36 apresentam os resultados obtidos pelo modelo Inception-v3, treinado com o otimizador Adam, taxa de aprendizado de 0.0005 e sem o uso de decaimento explícito.

Na Figura 33, os gráficos de perda e acurácia demonstram a convergência do modelo porém com algumas oscilações. A perda de treinamento diminui drasticamente nas primeiras épocas e permanece próxima de zero até o final, enquanto a perda de validação mantém valores baixos, com pequenas flutuações ao longo do treinamento. A acurácia, por sua vez, atinge valores superiores a 98% tanto para treinamento quanto para validação já nas primeiras épocas.

A matriz de confusão exibida na Figura 34 confirma o desempenho robusto do modelo, com uma acurácia do teste sendo de 98.88%. Classes como *Tomato Target Spot, Tomato Late blight* e *Tomato Bacterial spot* apresentaram uma alta quantidade de acertos, com pouquíssimos erros.

A Tabela V, assim como nos modelos anteriores,apresenta as métricas de desempenho do Inception-v3 para cada classe. Os resultados evidenciam a capacidade do modelo em classificar corretamente as imagens, com métricas elevadas em quase todas as classes, refletindo a robustez e a consistência do modelo.

A classe *Tomato Septoria leaf spot* se destaca com métricas ideais, alcançando 1.0000 em todas as métricas, o que demonstra que o modelo identificou todos os casos sem gerar falsos negativos ou falsos positivos. Da mesma forma, as classe *Tomato Target Spot*, *Tomato Yellow Leaf Curl Virus*, *Tomato Bacterial spot* e *Tomato Late Blight* apresentou valores máximos de precisão e especificidade, demonstrando que o modelo identificou os casos sem gerar falsos positivos e sem gerar falsos negativos.

Por outro lado, a classe *Tomato Mosaic Virus* apresentou um desempenho ligeiramente inferior em comparação às demais, com precisão de 0.9533 e F1-Score de 0.9761, indicando que o modelo teve dificuldades em identificar alguns casos negativos, resultando em falsos positivos. Contudo, a sensibilidade para essa classe permaneceu elevada (1.0000), mostrando que o

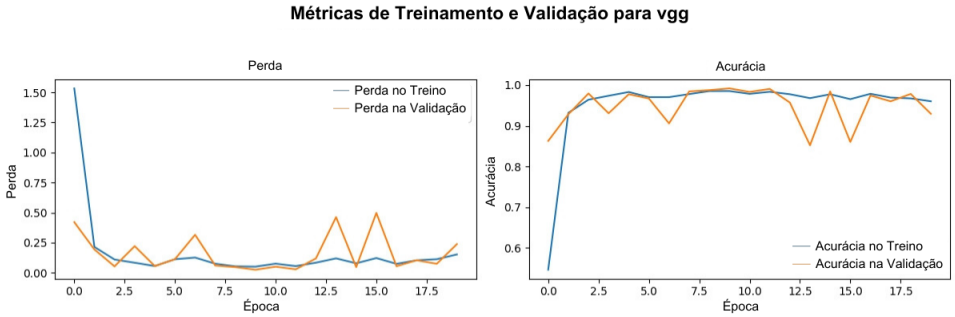
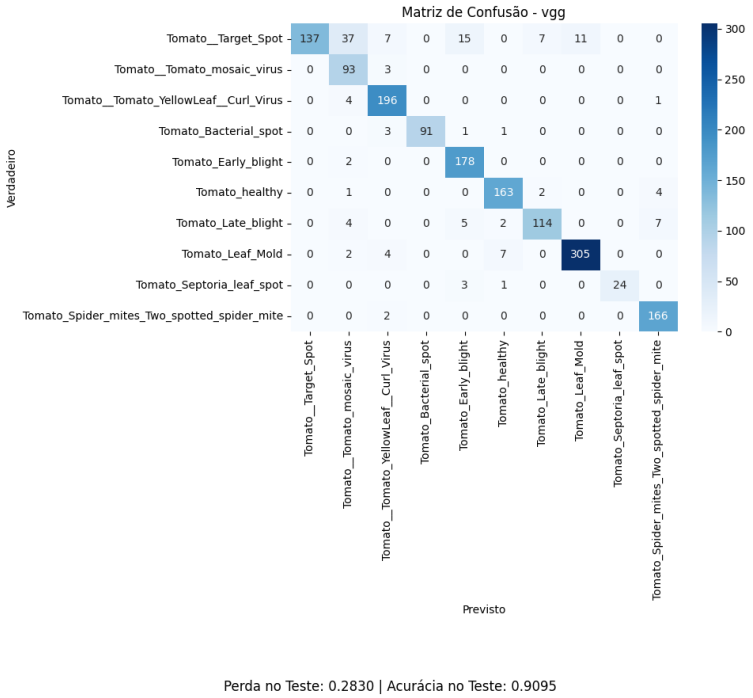


Figura 29: Métrica do treinamento VGG-16 com SGD e taxa de aprendizado 0.01. Fonte: Próprios autores.



Perda no Teste: 0.2830 | Acurácia no Teste: 0.9095

Figura 30: Matriz de Confusão do VGG-16 com SGD, com taxa de decaimento explícito. Fonte: Próprios autores.

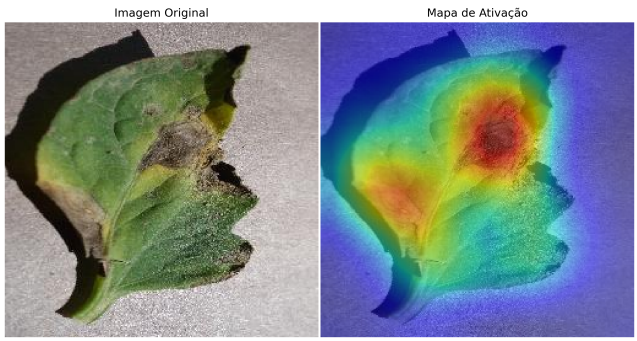


Figura 31: Mapa de Ativação do VGG-16, lr = 0.01. Fonte: Próprios autores.

modelo tem resistência em falsos negativos. Esses resultados foram similares aos da classe *Tomato Healthy*.

As demais classes, como *Tomato Early Blight*, *Tomato*

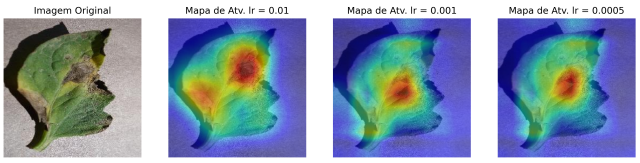


Figura 32: Mapas de Ativação dos VGG-16. Fonte: Próprios autores.

Leaf Mold e *Tomato Spider mites Two spotted spider mite*, mantiveram resultados equilibrados, com F1-Scores superiores a 0.9866, demonstrando a capacidade do modelo de balancear precisão e sensibilidade.

Apesar da distorção ocasionada pelo redimensionamento, observa-se pela Figura 35 que o modelo é capaz de identificar múltiplas características associadas à doença na folha.

Com o conjunto de mapas que representam a evolução do mapa final de características do modelo, podemos perceber que

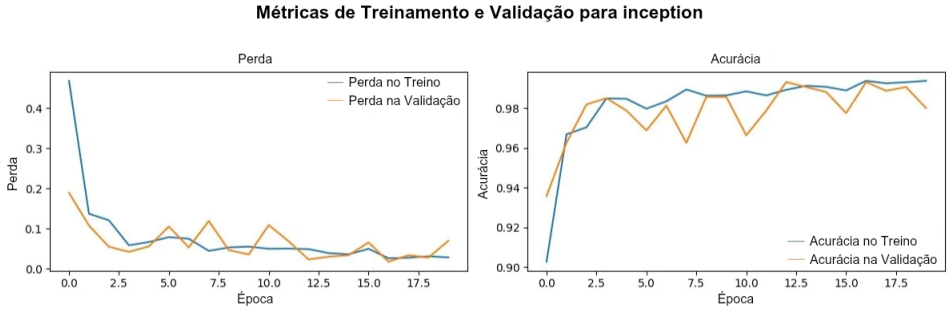
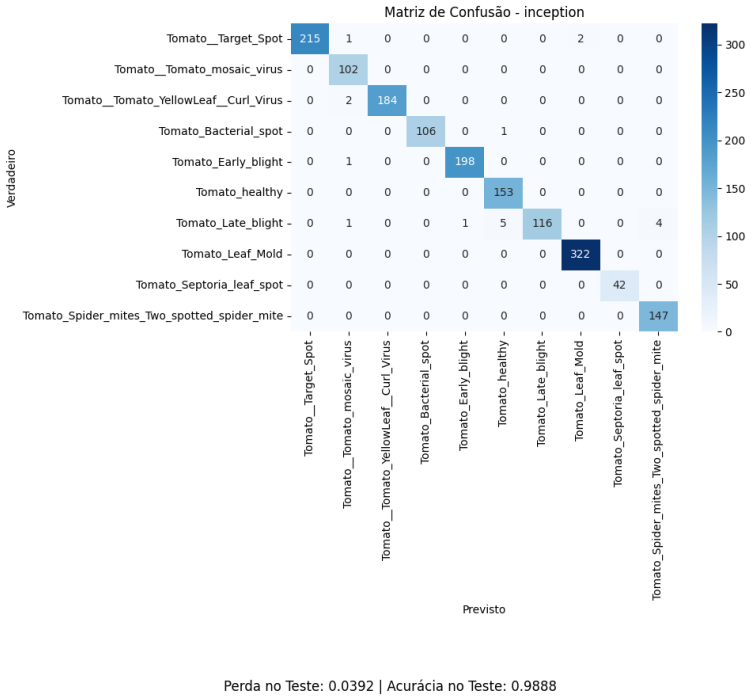


Figura 33: Métrica do treinamento Inception-v3 com Adam e taxa de aprendizado 0.0005. Fonte: Próprios autores.



Perda no Teste: 0.0392 | Acurácia no Teste: 0.9888

Figura 34: Matriz de Confusão do Inception-v3 com Adam, sem taxa de decaimento explícito. Fonte: Próprios autores.

Classe	Precisão	Sensibilidade	Especificidade	F1-Score
Tomato_Target_Spot	1.0000	0.9862	1.0000	0.9931
Tomato_Tomato_mosaic_virus	0.9533	1.0000	0.9967	0.9761
Tomato_Tomato_YellowLeaf_Curl_Virus	1.0000	0.9892	1.0000	0.9946
Tomato_Bacterial_spot	1.0000	0.9907	1.0000	0.9953
Tomato_Early_blight	0.9950	0.9950	0.9993	0.9950
Tomato_healthy	0.9623	1.0000	0.9959	0.9808
Tomato_Late_blight	1.0000	0.9134	1.0000	0.9547
Tomato_Leaf_Mold	0.9938	1.0000	0.9984	0.9969
Tomato_Septoria_leaf_spot	1.0000	1.0000	1.0000	1.0000
Tomato_Spider_mites_Two_spotted_spider_mite	0.9735	1.0000	0.9973	0.9866

Tabela V: Métricas de avaliação para as classes do modelo Inception-V3 utilizando Adam. Fonte: Próprios autores

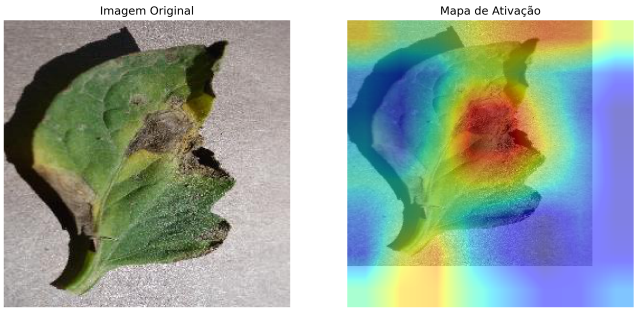


Figura 35: Mapa de Ativação do Inception-v3, lr = 0.0005. Fonte: Próprios autores.

o modelo evolui ao conseguir focar melhor nas regiões dominadas pela doença. Essa melhora na coleta de características indica uma vantagem do uso da taxa de aprendizagem de 0.0005, em relação as outras.

F. Inception-v3 com SGD

As Figuras 37 até a 40 apresentam os resultados do modelo Inception-v3, treinado com o otimizador SGD, taxa de aprendizado de 0.01 e decaimento explícito de 0.01.

A Figura 39 mostra o mapa de ativação extraído da última camada convolucional, destacando as regiões mais relevantes da imagem utilizadas pelo modelo para realizar a classificação.

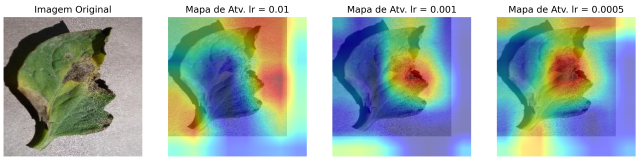


Figura 36: Mapas de Ativação dos Inception-v3. Fonte: Próprios autores.

Observa-se que o modelo concentra sua atenção nas áreas mais afetadas da folha, com maior intensidade nas regiões em vermelho.

Na Figura 37, os gráficos de perda e acurácia ao longo das épocas mostram um comportamento estável durante o treinamento. A perda diminui rapidamente e se mantém próxima de zero após poucas iterações, enquanto a perda de validação permanece em níveis baixos, sugerindo uma boa capacidade de generalização. A acurácia, por sua vez, aumenta de maneira consistente, atingindo valores próximos de 99% tanto no treinamento quanto na validação, com mínima diferença entre as curvas, o que demonstra um treinamento eficiente sem sinais de *overfitting*.

Por fim, a matriz de confusão na Figura 38 revela uma acurácia geral de 98.94%, confirmando um desempenho do modelo. Classes como *Tomato Late blight* e *Tomato Target Spot* apresentam alta taxa de acertos, enquanto pequenos desvios ocorrem em classes como *Tomato Tomato mosaic virus*, com um total de 3 previsões incorretas.

Classe	Precisão	Sensibilidade	Especificidade	F1-Score
Tomato_Target_Spot	1.0000	1.0000	1.0000	1.0000
Tomato_Tomato_mosaic_virus	0.9725	0.9381	0.9980	0.9550
Tomato_Tomato_YellowLeaf_Curl_Virus	0.9840	0.9840	0.9979	0.9840
Tomato_Bacterial_spot	0.9903	1.0000	0.9993	0.9951
Tomato_Early_blight	0.9824	1.0000	0.9979	0.9911
Tomato_healthy	0.9938	0.9755	0.9993	0.9845
Tomato_Late_blight	0.9714	0.9784	0.9973	0.9749
Tomato_Leaf_Mold	0.9939	1.0000	0.9984	0.9969
Tomato_Septoria_leaf_spot	1.0000	1.0000	1.0000	1.0000
Tomato_Spider_mites_Two_spotted_spider_mite	1.0000	1.0000	1.0000	1.0000

Tabela VI: Métricas de avaliação para as classes do modelo Inception-V3 utilizando SGD. Fonte: Próprios autores

A Tabela VI apresenta as métricas de desempenho do modelo Inception-v3 para cada classe, avaliadas por meio de precisão, sensibilidade, especificidade e F1-Score.

As classes *Tomato Target Spot*, *Tomato Septoria leaf spot* e *Tomato Spider mites Two spotted spider mite* obtiveram precisão, sensibilidade, especificidade e F1-Score iguais a 1.0000, o que confirma a capacidade do modelo em classificar todos os exemplos corretamente, sem a ocorrência de falsos positivos ou negativos. A classe *Tomato Bacterial spot* também apresentou excelente desempenho, com uma sensibilidade de 1.0000 e um F1-Score de 0.9951, destacando a precisão do modelo na identificação dessa condição.

Por outro lado, a classe *Tomato Tomato mosaic virus* apresentou os menores valores entre as métricas avaliadas, com sensibilidade de 0.9381 e F1-Score de 0.9550. Embora esses valores ainda sejam elevados, indicam que o modelo apresentou dificuldades na detecção de alguns exemplos positivos dessa classe, resultando em uma pequena quantidade de falsos negativos. A classe *Tomato healthy*, com um F1-Score de 0.9845 e sensibilidade de 0.9755, também apresentou um leve desafio, embora mantenha uma alta precisão.

De forma geral, o modelo InceptionV3 apresentou métricas superiores, com F1-Scores próximos ou iguais a 1.0000 para a maioria das classes.

VI. Outros Resultados

As Tabelas VII, VIII e IX apresentam os valores de acurácia [36] obtidos através da equação 7 para os 36 modelos treinados variando os hiperparâmetros de taxa de aprendizado (lr), otimizador (SGD e Adam) e a utilização ou não de decaimento com valor 0.1. Os resultados permitem observar o impacto direto dessas variações no desempenho dos modelos ResNet-50, VGG-16 e Inception-v3 em diferentes cenários.

No ResNet-50 (Tabela VII), os valores de acurácia variam significativamente em função do otimizador e da taxa de aprendizado. Os melhores desempenhos foram registrados com o otimizador SGD, especialmente na configuração com lr=0.01 e decaimento, onde a acurácia chegou a 99.31%. Já o Adam apresentou desempenho instável, com quedas expressivas na presença de decaimento, especialmente para lr=0.01, enquanto alcançou valores superiores com lr=0.0005 sem decaimento.

A Tabela VIII apresenta os resultados para o VGG-16, onde os valores de acurácia foram mais sensíveis às variações dos hiperparâmetros. O otimizador SGD obteve os melhores resultados para lr=0.01 sem decaimento, atingindo 99.06%, mas sofreu uma queda significativa com o uso de decaimento, registrando 90.95%. O Adam, por outro lado, apresentou desempenho instável, especialmente com decaimento, onde a acurácia chegou a apenas 9.23% com lr=0.01. Contudo, com lr=0.0005 e sem decaimento, o desempenho foi mais consistente, alcançando 96.63%.

Os valores de acurácia para o Inception-v3 estão apresentados na Tabela IX. O SGD obteve os melhores resultados, com destaque para lr=0.01 sem decaimento, onde a acurácia atingiu 99.38%. Mesmo com a introdução de decaimento, o desempenho do SGD permaneceu estável, com acurácia acima de 98.81% em todas as configurações. O Adam apresentou comportamento mais sensível, com uma queda acentuada para 49.59% com lr=0.01 e decaimento. No entanto, para lr=0.0005 sem decaimento, o Adam obteve 98.88%, demonstrando melhor adaptação a taxas de aprendizado menores.

Otimizador	Decaimento (0.1)	LR=0.01	LR=0.001	LR=0.0005
ADAM	Não	0.9638	0.9894	0.9975
ADAM	Sim	0.1890	0.7679	0.9476
SGD	Não	0.9869	0.9738	0.9507
SGD	Sim	0.9931	0.9726	0.9613

Tabela VII: Resultados para o modelo RESNET com diferentes configurações de otimizador, taxa de decaimento e taxa de aprendizado. Fonte: Próprios autores.

VII. Conclusão

Este estudo comparou os modelos ResNet-50, Inception-v3 e VGG-16 na tarefa de classificação de doenças e condições

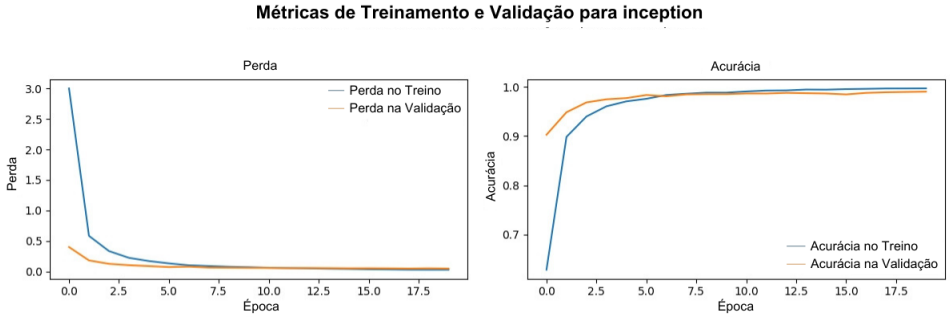
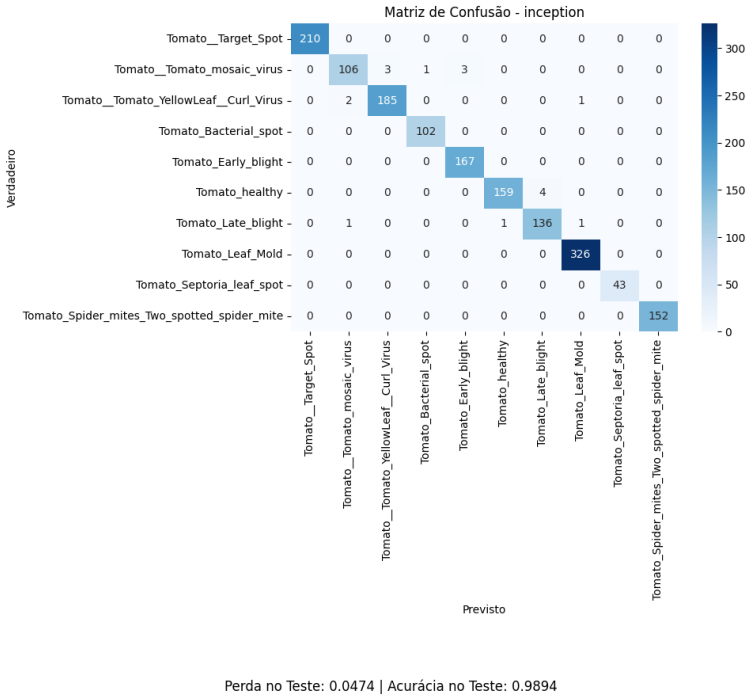


Figura 37: Métrica do treinamento Inception-v3 com SGD e com taxa de aprendizado 0.001. Fonte: Próprios autores.



Perda no Teste: 0.0474 | Acurácia no Teste: 0.9894

Figura 38: Matriz de Confusão do Inception-v3 com SGD, com taxa de decaimento explícito. Fonte: Próprios autores.

Otimizador	Decaimento (0.1)	LR=0.01	LR=0.001	LR=0.0005
ADAM	Não	0.1853	0.8677	0.9663
ADAM	Sim	0.0923	0.8400	0.8846
SGD	Não	0.9906	0.9894	0.9869
SGD	Sim	0.9095	0.9838	0.9850

Tabela VIII: Resultados para o modelo VGG com diferentes configurações de otimizador, taxa de decaimento e taxa de aprendizado. Fonte: Próprios autores.

Otimizador	Decaimento (0.1)	LR=0.01	LR=0.001	LR=0.0005
ADAM	Não	0.9488	0.9838	0.9888
ADAM	Sim	0.4959	0.9052	0.9707
SGD	Não	0.9938	0.9863	0.9788
SGD	Sim	0.9881	0.9894	0.9832

Tabela IX: Resultados para o modelo INCEPTION com diferentes configurações de otimizador, taxa de decaimento e taxa de aprendizado. Fonte: Próprios autores.

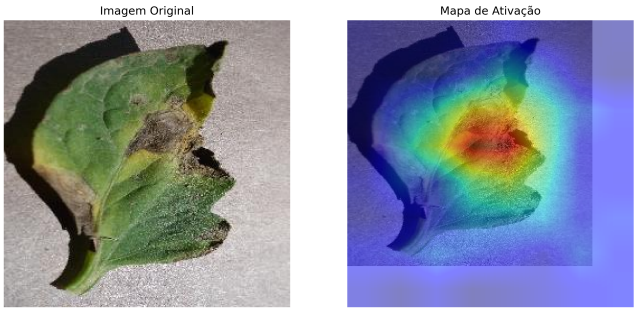


Figura 39: Mapa de Ativação do Inception-v3, lr = 0.001. Fonte: Próprios autores.

em folhas de tomate, utilizando diferentes configurações de taxa de aprendizado, decaimento e otimizadores. Os resultados obtidos demonstraram um desempenho satisfatório para todos os modelos, com acurácia geral superior a 90%, destacando

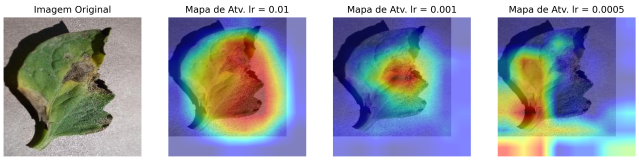


Figura 40: Mapas de Ativação dos Inceptions-v3. Fonte: Próprios autores.

a capacidade de generalização das três arquiteturas. Entretanto, diferenças importantes foram identificadas em relação ao desempenho individual dos modelos, à influência dos hiperparâmetros e à distribuição desigual das classes, o que impactou diretamente os resultados.

O ResNet-50 apresentou o melhor desempenho geral, alcançando acurácias de até 99.75% quando treinado com o otimizador ADAM e sem taxa de decaimento explícita de 0.0005. A profundidade da arquitetura permitiu a extração de características mais relevantes, resultando em F1-Scores próximos ou iguais a 1,0000 em diversas classes, como *Tomato Target Spot* e *Tomato Spider mites Two spotted spider mite*. Além disso, o comportamento do ResNet otimizado com SGD mostrou-se estável mesmo com a introdução de decaimento, especialmente em taxas de aprendizado mais altas. No entanto, o desempenho do Adam no ResNet foi altamente sensível ao decaimento, resultando em uma queda de acurácia para 18.90% com lr=0.01. Isso indica que o ResNet responde melhor ao ADAM com a configuração adequada de hiperparâmetros e sem uso do decaimento, mas o uso do SGD tem uma maior garantia de ótimos resultados em diferentes situações.

No caso do Inception-v3, os resultados também foram bastante positivos, com uma acurácia máxima de 99.38% obtida com SGD e lr=0.01 sem decaimento. O modelo demonstrou consistência em quase todas as configurações do SGD, com desempenho acima de 98.8%, mesmo com o uso de decaimento. Por outro lado, o Adam apresentou maior variabilidade, com uma queda expressiva na acurácia (49.59%) ao utilizar lr=0.01 com decaimento explícito, o que sugere uma maior instabilidade desse otimizador em condições mais agressivas de treinamento. Taxas de aprendizado menores, como 0.0005, permitiram ao Adam recuperar o desempenho, atingindo 98.88% sem decaimento. Um fator diferencial do Inception, no entanto, é sua capacidade de convergir rapidamente durante o treinamento, necessitando um número menor de épocas para atingir ótimos resultados de acurácia.

O VGG-16 foi o modelo com desempenho inferior entre os três, alcançando uma acurácia máxima de 99.06% com SGD e lr=0.01 sem decaimento. No entanto, a introdução de decaimento reduziu o desempenho do SGD para 90.95%, assim como em todas as outras configurações, destacando uma maior sensibilidade da arquitetura a essa configuração. O Adam, por sua vez, mostrou-se ainda mais instável no VGG-16, com uma queda drástica na acurácia (9.23%) ao utilizar lr=0.01 com decaimento. Contudo, em taxas de aprendizado menores, como 0.0005, o modelo atingiu 96.63% sem decaimento, sugerindo que o VGG-16 se beneficia de um ajuste mais cuidadoso dos hiperparâmetros e tende a apresentar dificuldades em cenários mais complexos.

Os mapas de ativação gerados ao longo do estudo também evidenciaram diferenças importantes entre os modelos. O ResNet-50 mostrou-se mais sensível e preciso na identificação das regiões afetadas nas folhas, enquanto o Inception-v3 apresentou resultados intermediários, com bom foco nas características relevantes, mas menos detalhado que o ResNet. O VGG-16, por sua vez, apresentou mapas de ativação mais difusos, o que pode explicar seu desempenho inferior, especialmente em classes com características visuais mais sutis.

Outro ponto relevante foi o impacto da distribuição das classes nos resultados. Classes com maior quantidade de amostras, como *Tomato Tomato YellowLeaf Curl Virus* (3209 amostras) e *Tomato Target Spot* (1404 amostras), apresentaram métricas superiores em todos os modelos, com F1-Scores próximos de 1.0000. Em contraste, classes menos representadas, como *Tomato Tomato mosaic virus* (373 amostras) e *Tomato Leaf Mold* (952 amostras), apresentaram desempenho inferior em determinadas configurações, indicando que a baixa representatividade dessas classes pode ter limitado a capacidade dos modelos de generalizar corretamente.

Portanto, os resultados mostram que o ResNet-50 foi o modelo mais eficaz, apresentando alto desempenho e estabilidade nas diferentes configurações avaliadas. O Inception-v3 obteve resultados competitivos, mas apresentou maior sensibilidade ao otimizador e ao ajuste de hiperparâmetros. Já o VGG-16, apesar de alcançar boas acurácias em cenários específicos, demonstrou maior dependência de taxas de aprendizado menores e configurações sem decaimento o que pode ter impactado seu desempenho no experimento, um possível fator é a divisão não estratificada das classes para realização do treino validação e teste, haja visto que o modelo do VGG-16 é menos robusto quanto os demais. A análise evidencia a importância de um ajuste cuidadoso dos hiperparâmetros e do balanceamento dos dados para otimizar o desempenho dos modelos, sugerindo que abordagens futuras, como técnicas de aumento de dados e reequilíbrio das classes, podem contribuir para resultados ainda melhores.

Referências

[1] A. Jaffri, “The AI Hype Cycle 2023: New Technologies on the Innovation Trigger”, Gartner, Inc., dez. de 2023, Acessado em 1 de dezembro de 2024. endereço: https://www.gartner.com/en/documents/hype-cycle-2023-ai.

[2] E. TOTVS. “Agricultura 4.0: Conceito, tecnologias, vantagens e desafios”. Acessado em 12 de novembro de 2024. (2024), endereço: https://www.totvs.com/blog/gestao-agricola/agricultura-4-0/.

[3] “Deep Learning Book”. Acessado em 2 de dezembro de 2024. (2024), endereço: https://www.deeplearningbook.com.br/introducao-as-redes-neurais-convolucionais/.

[4] P. Giovannini. “Introdução às Redes Neurais Convolucionais (CNNs) para Classificação de Imagens”. Acessado em 8 de novembro de 2024. (2023), endereço: https://pt.w3d.community/paulogio/introducao-as-redes-neurais-convolucionais-cnns-para-classificacao-de-imagens-3jah.

[5] D. Kaufman e L. L. V. Boas, "Visão computacional na agricultura: APIs de detecção e reconhecimento de doenças das plantas", *Revista Digital de Tecnologias Cognitivas*, p. 112, 2019, Acessado em 17 de setembro de 2024. endereço: https://revistas.pucsp.br/index.php/teccogs/article/view/48548.

[6] M. L. Gleason e B. A. Edmunds, "Tomato Diseases and Disorders", *Department of Plant Pathology*, 2006, Acessado em 18 de agosto de 2024. endereço: https://ncmg.ucanr.org/files/180088.pdf.

[7] C. Szegedy, V. Vanhoucke, S. Ioffe e J. Shlens, "Rethinking the Inception Architecture for Computer Vision", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016, Acessado em 8 de setembro de 2024. endereço: https://ieeexplore.ieee.org/document/7780677.

[8] R. B. Pereira, A. D. F. de Carvalho e J. B. Pinheiro, "Manejo da pinta-preta: uma ameaça às lavouras de tomateiro a céu aberto", *Embrapa Hortaliças. Comunicado técnico, 95*, v. 48, n. 2, pp. 123–134, 2013, Acessado em 6 de agosto de 2024. endereço: http://www.infoteca.cnptia.embrapa.br/infoteca/handle/doc/960748.

[9] T. Mehra, V. Kumar e P. Gupta, "Maturity and Disease Detection in Tomato Using Computer Vision", *Fourth International Conference on Parallel*, 2016, Acessado em 23 de setembro de 2024. endereço: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7913228.

[10] C. Hou, J. Zhuang, Y. Tang, Y. He e A. Miao, "Recognition of Early Blight and Late Blight Diseases of Potato Leaves Based on Graph Cut Segmentation", *Journal of Agriculture and Food Research*, v. 5, 2021, Acessado em 18 de setembro de 2024. endereço: https://www.sciencedirect.com/science/article/pii/S2666154321000569.

[11] G. Hu, M. Wan, K. Wei e R. Ye, "Computer Vision Based Method for Severity Estimation of Tea Leaf Blight in Natural Scene Images", *European Journal of Agronomy*, v. 144, 2023, Acessado em 20 de agosto de 2024. endereço: https://www.sciencedirect.com/science/article/pii/S1161030123000242.

[12] E. C. Tetila, "Detecção e classificação de doenças e pragas da soja usando imagens de veículos aéreos não tripulados e técnicas de visão computacional", *Programa de pós-graduação em Desenvolvimento Local*, 2019, Acessado em 4 de setembro de 2024. endereço: http://repositorio.ufgd.edu.br/jspui/handle/prefix/2385.

[13] K. Simonyan e A. Zisserman, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION", *International Conference on Learning Representations 2015*, 2015, Acessado em 03 de novembro de 2024. endereço: http://arxiv.org/pdf/1409.1556.

[14] "Convolutional Neural Networks", 2024, Acessado em 28 de novembro de 2024. endereço: https://www.ibm.com/br-pt/topics/convolutional-neural-networks.

[15] G. Holmes, S. Center, C. P. S. L. Obispo e Bugwood.org. "Early blight symptoms". Acessado em 5 de novembro de 2024. (2023), endereço: https://extension.umd.edu/resource/early-blight-tomatoes/.

[16] A. BASF. "Saiba identificar os tipos de ácaros e como tratar o tomateiro — BASF". Acessado em 7 de novembro de 2024. (2023), endereço: https://agriculture.basf.com/br/pt/conteudos/cultivos-e-sementes/tomate/tipos-acaro.

[17] R. Cultivar. "Septoriose no tomate: doença expressiva". Acessado em 7 de novembro de 2024. (2016), endereço: https://revistacultivar.com.br/artigos/efeito-letal.

[18] Plantix. "Bolor da Folha de Tomate". Acessado em 7 de novembro de 2024. (2024), endereço: https://plantix.net/pt/library/plant-diseases/100257/leaf-mold-of-tomato/.

[19] R. Cultivar. "Requeima no tomate: vilã temida". Acessado em 7 de novembro de 2024. (2016), endereço: https://revistacultivar.com.br/artigos/requeima-no-tomate-vila-temida.

[20] Plantix. "Mancha-bacteriana do Tomate". Acessado em 7 de novembro de 2024. (2024), endereço: https://plantix.net/pt/library/plant-diseases/300050/bacterial-spot-and-speck-of-tomato/.

[21] B. Queensland. "Tomato yellow leaf curl virus". Acessado em 7 de novembro de 2024. (2022), endereço: https://www.business.qld.gov.au/industries/farms-fishing-forestry/agriculture/biosecurity/plants/diseases/horticultural/tomato-yellow-leaf-curl-virus.

[22] C. Diagnostics. "What is Tomato Mosaic Virus?" Acessado em 7 de novembro de 2024. (2017), endereço: https://www.creative-diagnostics.com/blog/index.php/what-is-tomato-mosaic-virus/.

[23] L. Gasparotto, A. Reis, A. K. Inoue-Nagata, R. A. C. Netto e G. S. da Silva. "Principais doenças do tomateiro no Amazonas". Acessado em 7 de novembro de 2024. (2019), endereço: https://www.infoteca.cnptia.embrapa.br/infoteca/bitstream/doc/1118558/1/CircTec75-atual.pdf.

[24] C. do Gis. "O que é: K-Means Segmentation em Imagens". Acessado em 3 de novembro de 2024. (2024), endereço: https://clubedogis.com.br/glossario/o-que-e-k-means-segmentation-em-imagens/.

[25] R. Developers. "Rembg Documentation". Acessado em 2 de dezembro de 2024. (2024), endereço: https://pypi.org/project/rembg/.

[26] J. Liang, Y. Liu e V. Vlassov, "The Impact of Background Removal on Performance of Neural Networks for Fashion Image Classification and Segmentation", *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, 2023, Acessado em 19 de novembro de 2024. endereço: https://www.american-cse.org/csce2023-ieee/pdfs/CSCE2023-5LlpKs7cpb4k2UysbLCuOx/275900b960/275900b960.pdf.

[27] K. He, X. Zhang, S. Ren e J. Sun, "Deep Residual Learning for Image Recognition", *Microsoft Research*, 2015, Acessado em 05 de novembro de 2024. endereço: http://arxiv.org/pdf/1512.03385v1.

[28] K. Ito. “Vanishing & Exploding Gradient Problem & Dying ReLU Problem”. Acessado em 1 de dezembro de 2024. (2024), endereço: https://dev.to/hyperkai/vanishing-exploding-gradient-problem-dying-relu-problem-5fgn.

[29] L. C. da Cunha, “Redes neurais convolucionais e segmentação de imagens : uma revisão bibliográfica.”, Acessado em 3 de dezembro de 2024., 2020. endereço: http://www.monografias.ufop.br/handle/35400000/2872.

[30] M. Ponti, “Introdução à Otimização de Redes Neurais”, Universidade de São Paulo (USP), 2021, Acessado em 05 de novembro de 2024. endereço: https://edisciplinas.usp.br/pluginfile.php/7912910/mod_resource/content/1/_scc0270__RNAP_3___Introdu__o_a_Otimiza__o.pdf.

[31] M. Ponti, “Introdução à Otimização de Redes Neurais”, Acessado em 5 de dezembro de 2024., 2020. endereço: https://edisciplinas.usp.br/pluginfile.php/7912910/mod_resource/content/1/_scc0270__RNAP_3___Introdu__o_a_Otimiza__o.pdf.

[32] A. M. Ribeiro e F. de Paula S. Araujo Junior, “Um Estudo Comparativo Entre Cinco Métodos de Otimização Aplicados Em Uma RNC Voltada ao Diagnóstico do Glaucoma”, *Revista de Sistemas e Computação*, 2020, Acessado em 29 de novembro de 2024. endereço: https://revistas.unifacs.br/index.php/rsc/article/viewFile/6488/4043.

[33] A. E. D. C. D. T. D. INFORMAÇÃO. “Como o otimizador Adam otimiza o modelo de rede neural?” Acessado em 29 de novembro de 2024. (2023), endereço: https://pt.eitca.org/intelig%C3%AAncia-artificial/eitc-ai-dltf-aprendizado-profundo-com-tensorflow/fluxo-tensor/modelo-de-rede-neural/revis%C3%A3o-de-exame-modelo-de-rede-neural/como-o-otimizador-adam-otimiza-o-modelo-de-rede-neural/.

[34] D. P. Kingma e J. L. Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION”, *International Conference on Learning Representations 2015*, 2015, Acessado em 29 de novembro de 2024. endereço: https://arxiv.org/pdf/1412.6980.

[35] T. Lucas. “The Confusion Matrix in Computer Vision”. Acessado em 6 de dezembro de 2024. (2023), endereço: https://www.picsellia.com/post/the-confusion-matrix-in-computer-vision.

[36] D. M. Powers, “Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation”, *Journal of Machine Learning Technologies*, v. 2, n. 1, pp. 37–63, 2011.

Glauber Borges Lindolfo é graduando em Engenharia de Computação pela Universidade Federal de Goiás. Atualmente atua como Desenvolvedor de Software na Embrapa Arroz e Feijão. Possui experiência como analista de dados, visão computacional e séries temporais, desenvolvidos em Linguagem R e Python.

Ian Marcos da Cruz Chaves é graduando em Engenharia de Computação pela Universidade Federal de Goiás. Atualmente atua como Desenvolvedor de Software Full Stack na LG lugar de gente. Possui experiência com desenvolvimento de Sistemas Web em C# .NET além de familiaridade com tecnologias Microsoft e Azure.

Cássio Dener Noronha Vinhal possui graduação em Engenharia Elétrica - ênfase Engenharia da Computação/Eletrônica pela Universidade Federal de Uberlândia (1990), mestrado em Engenharia Elétrica, área de Automação, pela Faculdade de Engenharia Elétrica e de Computação da UNICAMP (1994) e doutorado em Engenharia Elétrica, área de Automação, pela Faculdade de Engenharia Elétrica e de Computação da UNICAMP (1998). Atualmente é Professor Titular da Escola de Engenharia Elétrica, Mecânica e de Computação da Universidade Federal de Goiás. Foi pós-doutorando junto ao Instituto de Engenharia de Sistemas e Computadores, Universidade do Porto, Portugal (2006-2007). Integra o Banco de Avaliadores BASIs - SINAES / INEP / Consultor Ad hoc do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP/MEC), para atos regulatórios de Autorização, Reconhecimento, Renovação de Reconhecimento de cursos de graduação. Tem experiência na área de Engenharia Elétrica e de Computação, atuando principalmente nos seguintes temas: Planejamento, Otimização e Simulação de Sistemas, Inteligência Artificial, Agentes Inteligentes e Autônomos, Robótica e Visão Computacional.