UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Aplicação de redes neurais convolucionais para classificação de doenças em plantações de soja.

Lucas Henrique Mateo

Trabalho de Conclusão de Curso

MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

Lucas Henrique Mateo

Aplicação de redes neurais convolucionais na classificação de doenças em plantações de soja.

Monografia apresentada ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Fernando Pereira dos Santos

Versão Original

USP - São Carlos 2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi e Seção Técnica de Informática, ICMC/USP, com os dados inseridos pelo(a) autor(a).

M425a

Mateo, Lucas Henrique Aplicação de redes neurais convolucionais para classificação de doenças em plantações de soja / Lucas Henrique Mateo; orientador Fernando Pereira dos Santos. -- São Carlos, 2024.
83 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. Computer Vision. 2. Convolutional Neural Network. 3. Deep Learning. 4. Transfer Learning. 5. Fine Tunning. I. Pereira dos Santos, Fernando , orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Lucas Henrique Mateo

Application of convolutional neural networks to classify diseases in soybean plantations.

Monograph presented to the Department of Computer Science of the Institute of Mathematical and Computer Sciences, University of São Paulo - ICMC/USP, as part of the requirements for obtaining the title of Specialist in Artificial Intelligence and Big Data.

Concentration area: AI and Big Data

Advisor: Fernando Pereira dos Santos

Original Version

USP - São Carlos 2024

DEDICATÓRIA

Este trabalho é dedicado a todos aqueles que reconhecem o potencial revolucionário da Inteligência Artificial (IA) e da Visão Computacional no contexto do agronegócio. Aqueles que enxergam além das práticas agrícolas tradicionais, vislumbrando um futuro em que a tecnologia desempenha um papel crucial na detecção precoce e na prevenção de doenças em plantações.

À comunidade de pesquisadores, engenheiros e entusiastas da IA e da visão computacional, este trabalho é uma homenagem ao seu compromisso com a inovação e à sua busca incansável por soluções que aprimorem a eficiência e a sustentabilidade do agronegócio.

Ao Instituto ICMC, que proporcionou a oportunidade de adquirir conhecimentos valiosos em Inteligência Artificial e Big Data, e a todos os professores envolvidos, expresso minha sincera gratidão. Suas orientações e ensinamentos foram fundamentais para o desenvolvimento deste trabalho.

Aos meus irmãos, que sempre me apoiaram durante esta trajetória, meu sincero agradecimento por estarem ao meu lado em todos os momentos.

Que este estudo sirva como um tributo ao potencial ilimitado da IA e da visão computacional para transformar o agronegócio, demonstrando o poder da colaboração e da inovação na resolução dos desafios que enfrentamos em nosso mundo.

Aos meus pais, a quem devo tudo, por proporcionarem-me a oportunidade de ter uma excelente educação.

AGRADECIMENTOS

Aos meus pais, que trabalharam duro para que eu tivesse uma boa educação e sempre me apoiaram incondicionalmente. À minha irmã, que me motivou a migrar para o mundo da tecnologia, me encorajando e estando sempre ao meu lado.

Aos professores que me acompanharam durante toda a ementa do curso, contribuindo para o meu desenvolvimento acadêmico e profissional.

Ao professor e orientador Fernando Pereira dos Santos, pelo apoio e orientação inestimáveis e contribuições no desenvolvimento do projeto.

À professora e coordenadora do curso, Solange, por me proporcionar a oportunidade de realizar um curso de ponta em uma grande universidade com bolsas escolares.

E por fim, ao centro ICMC da USP, que disseminou conhecimento para que eu pudesse realizar este trabalho.

“Excelência nunca é um acidente. É sempre o resultado de elevada intenção, esforço sincero e execução inteligente; representa a escolha sábia de muitas alternativas – a escolha, e não o acaso, determina o seu destino”

Aristotle

RESUMO

MATEO, H. L. **Aplicação de redes neurais convolucionais na classificação de doenças em plantações de soja**, 2024. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

A produção agrícola desempenha um papel fundamental na alimentação global e na economia, tornando a monitoração da saúde das plantas uma prioridade. Este estudo tem como objetivo utilizar técnicas de visão computacional para desenvolver um algoritmo capaz de classificar imagens de plantações de soja entre saudáveis e duas doenças específicas, contribuindo para a prevenção de perdas de alimentos e para a sustentabilidade do agronegócio. Este estudo aborda a seguinte questão de pesquisa: "A utilização de técnicas de modelos de redes neurais convolucionais possibilita ao computador interpretar e compreender imagens de plantas, de forma a realizar a classificação de doenças em fotografias inéditas de plantações de soja?" Para isso, foram realizados experimentos com arquiteturas de redes neurais convolucionais (CNNs), incluindo VGG, ResNet, Inception e EfficientNET, comparando sua eficácia na classificação das plantações de soja entre as classes “diabrotica speciosa”, “caterpillar” e “saudável”. A organização do estudo inclui uma revisão da literatura sobre a importância da saúde das plantas na agricultura, seguida pela descrição detalhada do problema de pesquisa e dos objetivos do estudo. Em seguida, são exploradas as arquiteturas de CNNs selecionadas e suas respectivas aplicações na detecção de doenças em plantas, destacando suas características e contribuições para o campo da visão computacional agrícola. A metodologia do estudo compreende a coleta e preparação dos dados, treinamento e avaliação dos modelos de CNNs, e análise dos resultados obtidos. Por fim, são discutidas as conclusões do estudo, incluindo insights sobre a eficácia das diferentes arquiteturas de CNNs na detecção de doenças em plantações de soja e possíveis direções futuras para pesquisa na área de visão computacional agrícola.

Palavras-chave: CNN; VGG; ResNet; Inception; EfficientNET.

ABSTRACT

MATEO, H. L. **Application of convolutional neural networks to classify diseases in soybean plantations**, 2024. Final paper (MBA in Artificial Intelligence and Big Data) - Institute of Mathematical and Computer Sciences (ICMC), University of São Paulo, São Carlos, 2024.

Agricultural production is crucial for global food security and the economy, making plant health monitoring a top priority. This study aims to leverage computer vision techniques to develop an algorithm capable of classifying soybean crop images into healthy and two specific disease categories, thus contributing to food loss prevention and the sustainability of agribusiness. The research addresses the question: "Can Convolutional Neural Network (CNN) models enable computers to interpret and understand plant images, thereby classifying diseases in new photographs of soybean crops?" To explore this, experiments were conducted using CNN architectures, including VGG, ResNet, Inception, and EfficientNet, to compare their effectiveness in classifying soybean crops into the classes "Diabrotica speciosa," "Caterpillar," and "Healthy." The study is structured to include a literature review on the importance of plant health in agriculture, followed by a detailed description of the research problem and objectives. It then explores the selected CNN architectures and their applications in plant disease detection, highlighting their characteristics and contributions to the field of agricultural computer vision. The methodology encompasses data collection and preparation, training and evaluation of CNN models, and analysis of the results. The study concludes with discussions on the effectiveness of different CNN architectures in detecting diseases in soybean crops and potential future research directions in agricultural computer vision.

Keywords: CNN; VGG; ResNet; Inception; EfficientNET.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1. INTRODUÇÃO

A organização das nações unidas para alimentação e agricultura FAO (2020) estima que as plantas são responsáveis por 80% dos alimentos que ingerimos e produzem 98% do oxigênio que respiramos. Destarte, a monitoração da saúde das plantas é um componente essencial da saúde global, contribuindo para a prevenção de perdas de alimentos, a segurança alimentar, o crescimento econômico e a sustentabilidade ambiental. Os agricultores enfrentam um desafio significativo associado à perda de alimentos devido a doenças nas plantações. Ainda conforme a FAO, cerca de até 40% das colheitas de alimentos são desperdiçadas devido a pragas e doenças.

A perda anual de rendimento das culturas causada por agentes patogênicos e pragas é estimada em 220 bilhões de dólares (Singh *et al*., 2023). Portanto, a identificação antecipada de doenças nas plantações é fundamental para que o agricultor possa tomar medidas de controle eficazes, minimizando perdas de produtividade e de alimentos. A visão computacional, aliada a algoritmos de aprendizado de máquina, tem o potencial de aumentar a eficiência e a produtividade da agricultura de forma significativa. Essa tecnologia permite a análise automatizada de imagens para identificar pragas, doenças e condições de estresse nas plantas, possibilitando uma intervenção mais rápida e precisa. Métodos como a detecção de padrões e a segmentação de imagens são capazes de monitorar grandes áreas de cultivo com alta precisão, o que pode reduzir a necessidade de inspeções manuais e otimizar o uso de recursos, como água e fertilizantes. Estudos demonstram que a aplicação de visão computacional na detecção precoce de doenças pode aumentar a produtividade agrícola e minimizar perdas (Li et al., 2020). Conforme Kamilaris e Prenafeta-Boldú (2018), o uso de aprendizado profundo e visão computacional na agricultura tem mostrado ser altamente eficaz em diversas aplicações, consolidando-se como uma ferramenta essencial na agricultura moderna.

1.1 Justificativa e Motivação

A Empresa Brasileira de Pesquisa em Agropecuária (Embrapa) aponta que o produto mais utilizado como alimento no Brasil é o óleo de soja, onipresente na cozinha popular brasileira. Mais de 80% de todo óleo de soja produzido é consumido no país, na preparação de frituras, saladas e dos mais diversos pratos. Além de ser

um alimento essencial para a dieta brasileira, a soja também é uma importante fonte de proteínas, fibras e outros nutrientes essenciais para a saúde humana.

Estudos do Centro de Estudos Avançados em Economia Aplicada - CEPEA Esalq/USP afirmam que a cadeia de soja e biodiesel representou 27% do PIB do agronegócio no Brasil no ano de 2022, no valor de R$ 673,7 bi, gerando 10,8% dos empregos e 38% das exportações do agronegócio brasileiro (Cadeia de soja e do biodiesel, CEPEA).

O impacto da falta de soja para os brasileiros é significativo e justifica o desenvolvimento de tecnologias que possam auxiliar na sua produção. Nesse sentido, as aplicações de visão computacional na agricultura têm potencial para contribuir consideravelmente.

Diante do crescimento populacional e da crescente demanda por alimentos, e considerando a alta relevância da soja para a economia brasileira, este estudo buscou utilizar técnicas de visão computacional para desenvolver um algoritmo destinado à agricultura, com o objetivo de auxiliar na identificação de doenças em plantios de soja. A proposta é oferecer uma ferramenta que complemente os métodos tradicionais de inspeção, especialmente em situações onde a automação pode melhorar a escalabilidade e a consistência da análise.

1.2 Problema de pesquisa e objetivo

A visão computacional é um subcampo da Inteligência Artificial que facilita computadores e máquinas a analisar imagens e vídeos. O objetivo deste projeto foi utilizar técnicas de visão computacional para processar os dados de imagem de maneira precisa, com o intuito de processar imagens de plantações de soja e realizar a classificação entre “planta com doença” e “planta sem doença”. Para este estudo, foi elaborada a questão de pesquisa:

Q1: *“A utilização de técnicas de visão computacional possibilita ao computador interpretar e compreender imagens de plantas, de forma a realizar a classificação de doenças em fotografias inéditas de plantações de soja*”

Ao abordar a questão de pesquisa, é essencial enfrentar diversos desafios no desenvolvimento de algoritmos de aprendizado de máquina para visão computacional. Este estudo focou em:

- Explorar diversas arquiteturas de redes neurais convolucionais, como VGG, ResNet, Inception e EfficientNet, para a classificação de doenças em plantações de soja.
- Realizar ajustes finos (fine-tuning) por meio de transformações de dados e otimização de hiperparâmetros, buscando melhorar a acurácia e a generalização dos modelos.
- Implementar técnicas avançadas como o ajuste dinâmico do learning rate e a utilização de callbacks para potencializar o treinamento do modelo.
- Avaliar a performance dos modelos através de métricas como precisão, recall, F1-score, e análise da matriz de confusão para garantir uma análise robusta e confiável.

Esses objetivos garantiram uma abordagem abrangente e detalhada para o desenvolvimento do algoritmo, aumentando sua eficácia.

2. FUNDAMENTAÇÃO TEÓRICA

Este trabalho se fundamenta em uma sólida base teórica, centrada na área de visão computacional. Por meio de uma revisão criteriosa da literatura pertinente, foram explorados os principais conceitos e teorias que permeiam essa disciplina, fornecendo um arcabouço conceitual para a análise e discussão dos temas abordados. Destacando os fundamentos teóricos essenciais da visão computacional, este estudo busca aprofundar a compreensão deste campo e seu papel no desenvolvimento de soluções e aplicações práticas. A seguir, serão apresentadas as teorias fundamentais que embasam este trabalho.

2.1 Visão Computacional

Visão computacional é uma subárea da Inteligência Artificial (IA) cujo propósito é ensinar as máquinas a interpretarem e tomar decisões com base em dados visuais. Em outras palavras, através do processamento de imagens e técnicas de aprendizado de máquina, trata-se de proporcionar às máquinas a capacidade de “enxergar”, simulando a visão humana.

A visão computacional é frequentemente utilizada em combinação com técnicas de aprendizado profundo, especialmente Redes Neurais Convolucionais (CNN), com o intuito de melhorar a precisão e eficácia em tarefas complexas.

2.2 Processamento de imagem

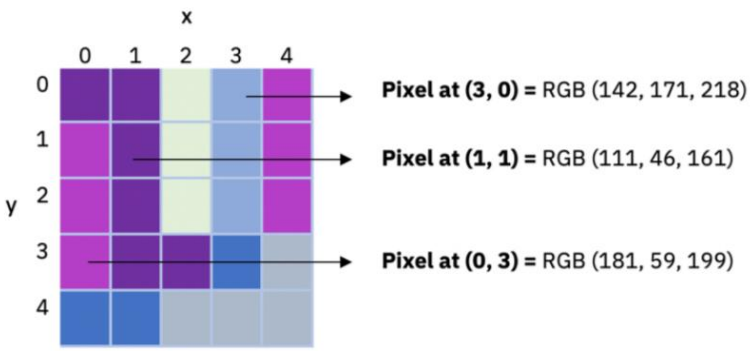
Antes de adentrarmos nas especificidades dessa aplicação na detecção de doenças em plantações de soja por meio de visão computacional, é imprescindível compreendermos a definição básica de uma imagem digital e seus atributos fundamentais.

> “Uma imagem pode ser definida como uma função bidimensional, $f(x, y)$, em que x e y são coordenadas espaciais (planas), e a amplitude de f em qualquer par de coordenadas (x, y) é chamada de intensidade ou nível de cinza da imagem naquele ponto. Quando x, y e os valores de intensidade de f são todas quantidades finitas e discretas, chamamos a imagem de imagem digital.” (Gonzales; Woods, 2018, p.18).

Diante disso, é possível inferir que a representação visual de uma imagem digital é composta por três amplitudes de coordenadas (x, y), correspondentes aos componentes de cor no modelo RGB (Red, Green, Blue). Essa estrutura tridimensional de coordenadas permite a codificação precisa das informações de cor

em cada pixel, proporcionando uma representação detalhada e fiel da imagem observada, conforme ilustrado na Figura 1.

Figura 1: Modelo de cores RGB



Fonte: Altunay, 2019.

É importante destacar as alternativas existentes para processamento de imagem. Os modelos de cores em escala de cinza e preto e branco oferecem opções simplificadas em comparação ao modelo RGB. Ao processar imagens nessas escalas, apenas uma amplitude de matriz (x, y) é necessária para operações de convolução, em contraste com as três amplitudes de matrizes (x, y) requeridas em uma imagem colorida.

Portanto, transformar a imagem para modelos de cores em escala de cinza ou preto e branco pode ser uma estratégia eficaz para otimizar a velocidade do processamento em projetos de visão computacional, especialmente em aplicações agrícolas, onde a eficiência do processamento de imagem é crucial para análises precisas e rápidas. A redução da carga computacional resultante dessa escolha pode possibilitar um processamento mais rápido mesmo em ambientes com recursos computacionais limitados, como é frequentemente o caso em sistemas utilizados em campo.

Adicionalmente, variações na resolução, iluminação e ângulo de captura das imagens podem exercer impacto significativo na precisão do reconhecimento de padrões. Neste contexto, a padronização dos tamanhos das imagens desempenha um papel crucial, favorecendo a análise mais precisa e robusta ao permitir uma identificação eficiente dos padrões dentro dos pixels.

2.3 Data Augmentation: Aumento de dados

Conforme a Amazon, O aumento de dados é um processo fundamental no campo do Machine Learning (ML) que envolve a criação artificial de novos dados a partir de dados existentes. Este método é crucial para o treinamento de modelos de ML, que requerem grandes volumes de dados diversificados. Contudo, reunir conjuntos de dados reais suficientemente variados pode ser desafiador devido a restrições como silos de dados, regulamentações e outras limitações.

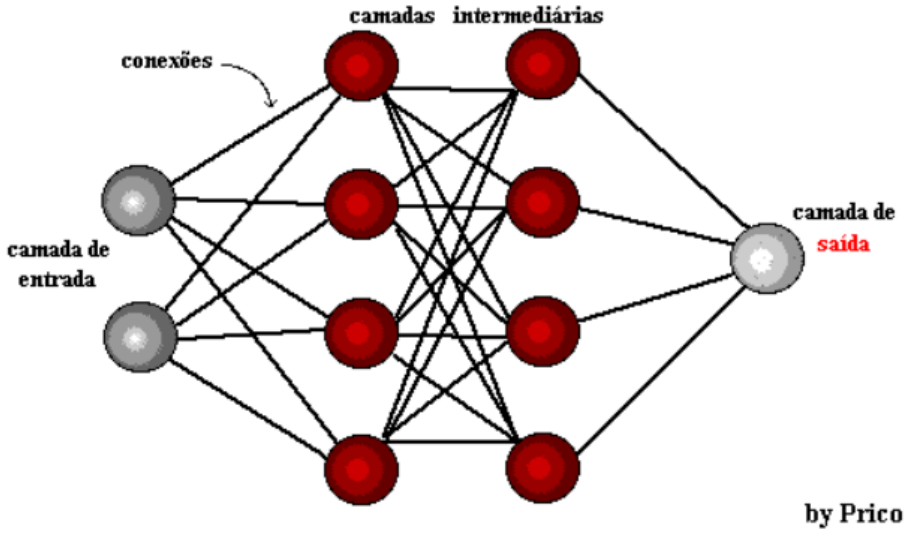
A prática do aumento de dados resolve essa questão ao fazer pequenas alterações nos dados originais, criando assim um conjunto de dados ampliado e diversificado. Com a evolução das tecnologias, as soluções de inteligência artificial generativa (IA) têm se destacado, proporcionando um aumento rápido e de alta qualidade de dados em diversos setores.

Diversas técnicas podem ser exploradas para realizar o aumento de dados, e a aplicação dessas técnicas varia conforme o problema de negócio.

2.4 Redes neurais artificiais

Uma rede neural é um modelo computacional inspirado na estrutura e funcionamento do cérebro humano. Ela é composta por unidades básicas chamadas neurônios artificiais, organizados em camadas e interconectados por meio de pesos sinápticos (Figura 2).

Figura 2: Organização de RNA em camadas



Fonte: ICMC/USP

As camadas são classificadas em três grupos:

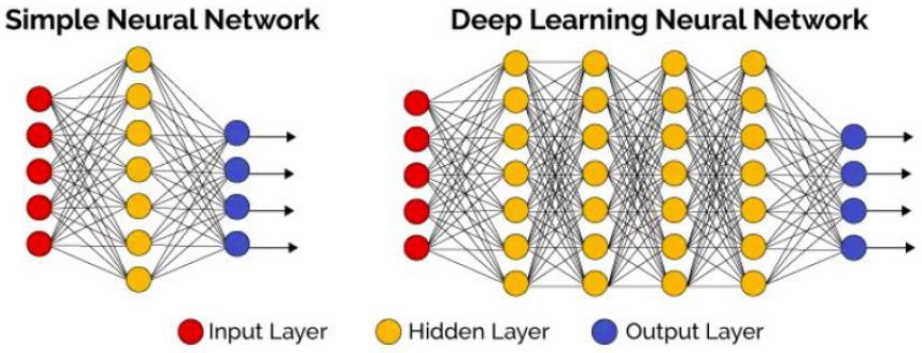
- **Camada de Entrada**: É a primeira camada de uma rede neural, onde os dados de entrada são inseridos. Esses dados podem ser imagens, texto, áudio ou qualquer outro tipo de informação que a rede neural precise processar.
- **Camadas Intermediárias (ou ocultas)**: São camadas entre a camada de entrada e a camada de saída. Cada camada intermediária contém um conjunto de neurônios, e esses neurônios estão conectados a todos os neurônios da camada anterior e da camada seguinte. O processamento dos dados acontece nessas camadas, onde padrões e características dos dados são aprendidos pela rede neural.
- **Camada de Saída**: É a última camada da rede neural, onde os resultados ou predições são gerados. Dependendo da tarefa que a rede está realizando, a camada de saída pode ter diferentes configurações, como uma única saída para problemas de classificação binária, múltiplas saídas para classificação multiclasse ou uma saída contínua para regressão.

Essas camadas trabalham em conjunto para processar os dados de entrada, aprender padrões e características nos dados intermediários e gerar resultados na camada de saída.

2.5 Redes neurais artificiais profundas

Redes neurais profundas (*Deep Learning*) é um termo geral que se refere a um conjunto de técnicas de aprendizado de máquina que utilizam redes neurais artificiais complexas com camadas ocultas para aprender com grandes conjuntos de dados.

Figura 3: Rede Neural Simples e Rede Neural Profunda



Fonte: Data Science Academy, 2022.

Conforme ilustrado na Figura 3, a diferença fundamental entre esses modelos reside na profundidade das camadas ocultas. Enquanto a rede neural simples consiste em apenas uma camada intermediária, a rede neural profunda possui múltiplas

camadas intermediárias, permitindo a aprendizagem de representações hierárquicas de características nos dados.

Esta capacidade de aprender características abstratas e complexas em diferentes níveis de abstração é crucial para tarefas de inteligência artificial, como reconhecimento de padrões em imagens e processamento de linguagem natural.

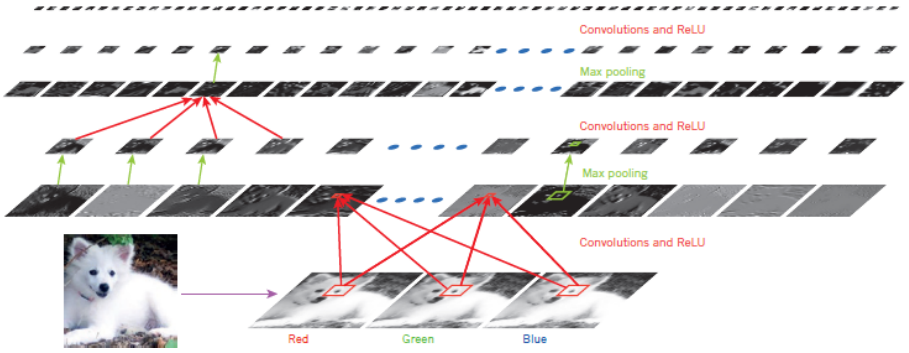
A criação de redes neurais profundas foi motivada pelo desejo de melhor representar características complexas, aprender de forma mais eficiente, generalizar para novos dados e aproveitar os avanços em hardware e software. Esses avanços têm sido fundamentais para o sucesso das redes neurais profundas em uma variedade de aplicações de inteligência artificial, inclusive classificação de imagens, o qual é o objetivo do trabalho proposto.

2.6 Redes neurais convolucionais

Rede neural convolucional (Figura 4) é um tipo específico de rede neural profunda eficaz para tarefas de visão computacional. Conforme LeCun; Bengio; Hinton (2015), existem três ideias-chave por trás dos *ConvNets* que aproveitam as propriedades de sinais naturais:

- **Conexões Locais e Pesos Compartilhados**: Nas *ConvNets*, as unidades em uma camada convolucional estão conectadas a patches locais nos mapas de características da camada anterior, utilizando um conjunto de pesos compartilhados chamado banco de filtros. Essa estrutura aproveita a correlação local dos dados, sendo especialmente eficaz para a detecção de padrões em imagens.
- ***Pooling***: As camadas de *pooling* nas *ConvNets* têm o papel de mesclar características semanticamente similares em uma, reduzindo a dimensionalidade da representação e criando invariância a pequenos deslocamentos e distorções. Isso é alcançado calculando-se o máximo (ou outra operação) em patches locais de unidades em um mapa de características.
- **Arquitetura em Estágios**: As *ConvNets* são estruturadas em uma série de estágios, cada um composto por camadas convolucionais e de *pooling*. Essa arquitetura em estágios permite a extração progressiva de características complexas, seguindo uma hierarquia semelhante à observada em sinais naturais, como imagens e texto.

Figura 4: Dentro de uma rede neural convolucional



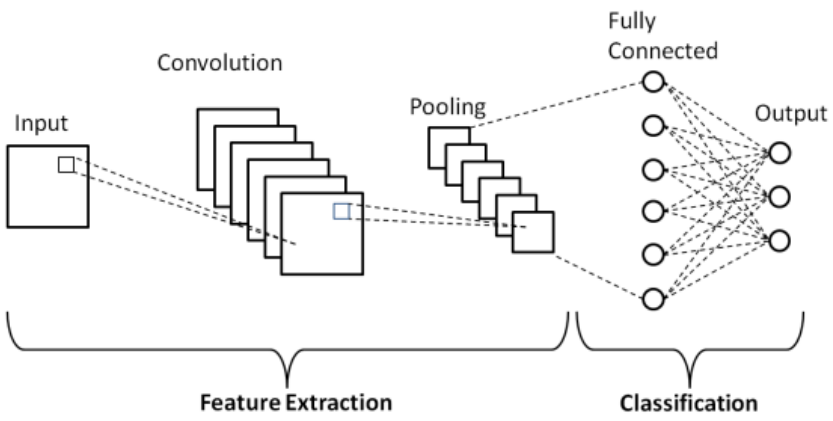
Fonte: Deep learning. Nature, v. 521, p. 438, 2015.

Em suma, a escolha de redes neurais convolucionais se baseou em sua comprovada efetividade em tarefas de visão computacional, suas características que se adaptam à natureza de imagens e sua capacidade de lidar com os desafios específicos do projeto.

2.7 Operação de Pooling

O *pooling*, também conhecido como *subsampling* ou *down-sampling*, desempenha um papel crucial nas redes neurais convolucionais, sendo essencial para a redução da dimensionalidade da imagem e aprimoramento do processo de aprendizado. O *pooling* é aplicado após cada operação de convolução (Figura 5), destacando-se como um passo fundamental no fluxo de processamento da rede. Essa abordagem permite que a rede se concentre nas características mais salientes das imagens, tornando-a mais eficiente e eficaz na extração de padrões relevantes para a tarefa em questão.

Figura 5: Rede Neural Convolucional Simples



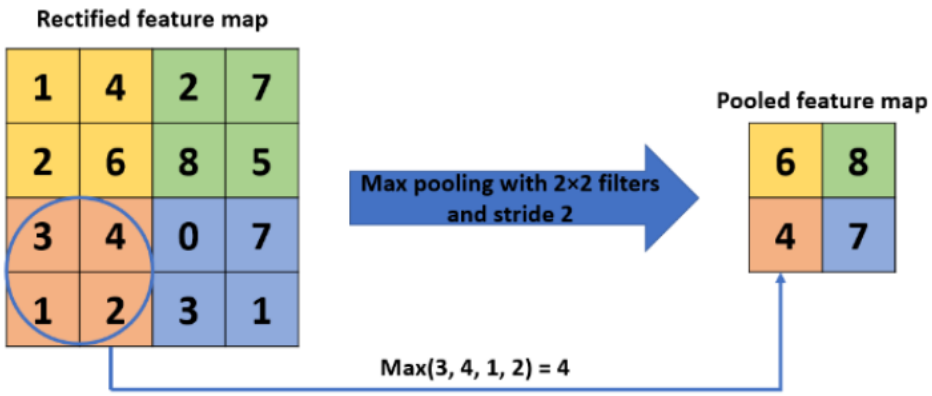
Fonte: Balaji, 2020

Existem diversas operações de *pooling*, como média, mínimo e máximo. Neste projeto, exploraremos a técnica de MAX *pooling*, reconhecida por sua capacidade de capturar as características mais proeminentes em uma região.

2.7.1 Max Pooling

O *maxpooling* seleciona o valor máximo de uma região de pixels na matriz (Figura 6).

Figura 6: Operação de Maxpooling



Fonte: Gholamalinezhad & Khosravi, 2020

Essa operação desempenha um papel fundamental nesse processo, ajudando a rede a discernir e priorizar as características mais proeminentes, ao mesmo tempo em que promove a redução da dimensionalidade e a prevenção do *overfitting*. Essa técnica oferece a preservação da invariância espacial e a promoção de uma representação mais compacta e discriminativa dos dados.

2.8 Função de ativação

De acordo com Goodfellow et al. (2016), as funções de ativação são componentes essenciais nas redes neurais artificiais, desempenhando um papel crucial no processo de aprendizagem e na capacidade da rede de resolver problemas complexos. Em sua essência, essas funções determinam se um neurônio deve ser ativado, ou seja, se a informação recebida deve ser propagada para a próxima camada da rede. Essas funções introduzem não linearidade no modelo, permitindo que ele aprenda relações complexas entre os dados de entrada e saída.

Existem diversas funções de ativação utilizadas em redes neurais, cada uma com suas características e aplicações específicas. Neste trabalho, analisaremos em

profundidade as funções ReLU e Softmax, explorando suas características e aplicações no contexto das arquiteturas de rede neural utilizadas.

2.8.1 ReLU

Conforme Data Science Academy (2022), a ReLU (Unidade Linear Retificada) é uma função amplamente utilizada em modelos de aprendizado profundo, incluindo CNNs. Essencialmente, a função ReLU Equação (1) mantém os valores positivos intactos, enquanto transforma os valores negativos em zero.

$$f(x) = \max(0, x) \tag{1}$$

Os benefícios de utilizar a função de ativação RELU (Rectified Linear Unit) incluem a facilidade de implementação e baixo custo computacional, tanto para a função em si quanto para suas derivadas (Lederer, 2021).

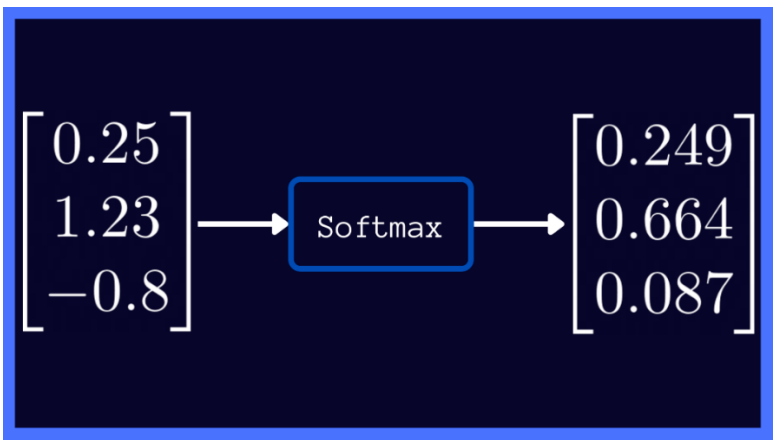
2.8.2 Softmax

A função Softmax é comumente empregada na camada final de uma rede neural em tarefas de classificação multiclasse. Seu propósito é converter os valores de entrada em um vetor de probabilidades, onde cada valor varia de 0 a 1 e a soma total é igual a 1 (Equação 2).

$$f(x) = \max(0, x) \tag{2}$$

Esse processo é ideal para interpretar a saída da rede como probabilidades associadas às diversas classes, uma vez que cada valor representa a probabilidade de a instância pertencer a uma das classes possíveis (Figura 7).

Figura 7: Saída da função Softmax



Fonte: C et al. (2023)

2.9 Loss Function: Função de perda

Conforme descrito por Bishop (2006), a função de perda mede a discrepância entre as previsões do modelo e os valores reais, orientando o ajuste dos pesos da rede para minimizar essa diferença. É um método utilizado para avaliar o modelo com base no conjunto de dados.

Para problemas de classificação multiclasse, a função de perda mais utilizada é a entropia cruzada categórica. Esta função mede a diferença entre a distribuição de probabilidade verdadeira e a distribuição prevista pela rede. Em outras palavras, o erro da entropia cruzada determina quão próximo está a distribuição prevista da distribuição verdadeira.

Segundo Murphy (2012), a Entropia Cruzada Categórica é eficaz para lidar com a natureza probabilística das saídas das redes neurais, especialmente quando a função de ativação Softmax é utilizada na camada de saída.

2.10 Aprendizado com Backpropagation

O aprendizado em CNNs é realizado através do algoritmo de *backpropagation*, que ajusta os pesos das conexões da rede para minimizar a função de perda. De acordo com Rumelhart et al. (1986), o *backpropagation* envolve duas fases principais:

- Forward Pass: As entradas são passadas através da rede para calcular as previsões.
- Backward Pass: O erro entre as previsões e os valores reais é propagado de volta pela rede, calculando os gradientes da função de perda em relação aos pesos. Esses gradientes são então usados para ajustar os pesos com o objetivo de reduzir o erro.

O algoritmo de *backpropagation* permite que a rede aprenda ajustando iterativamente os pesos para minimizar a função de perda, conforme descrito por LeCun et al. (1998). Este processo é essencial para o treinamento eficiente de redes neurais profundas.

2.11 Problema do Vanish gradient: degradação de gradiente

Conforme He *et al*. (2016), em redes neurais profundas, o "desaparecimento do gradiente" se manifesta como uma dificuldade no aprendizado de pesos nas camadas mais profundas da rede. Isso ocorre porque os gradientes, responsáveis por ajustar os pesos durante o treinamento, tendem a diminuir à medida que se propagam pelas camadas, tornando o aprendizado nessas regiões mais lento e ineficaz.

Em outras palavras, à medida que mais camadas usando certas funções de ativação são adicionadas às redes neurais, os gradientes da função de perda se aproximam de zero, dificultando o treinamento da rede (Wang, 2019).

2.12 Arquitetura das redes neurais convolucionais

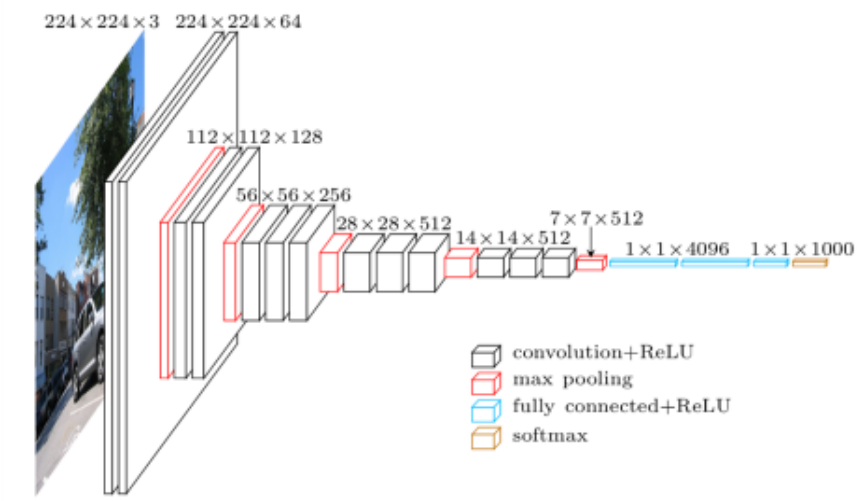
O campo do reconhecimento de imagens evoluiu drasticamente nas últimas décadas, impulsionado pelo desenvolvimento de arquiteturas neurais convolucionais (*CNNs*) cada vez mais complexas e eficientes. Entre as diversas opções disponíveis, três arquiteturas se destacam: *VGG*, *ResNet* e *Inception*.

2.12.1 VGG-16

VGG é a abreviação de Visual Geometry Group (Departamento de Ciência e Engenharia da Universidade de Oxford); é uma estrutura convolucional de redes neurais (CNN) profundamente estabelecida, contendo múltiplas camadas. A designação "profunda" refere-se à extensão dessas camadas, compreendendo 16 camadas convolucionais.

Conforme ilustrado na Figura 8, a camada de entrada da Convolução 1 processa imagens em um tamanho padrão de 224 x 224 pixels RGB. A entrada é passada por todas as 16 camadas para extrair características e alcançar resultados de alta precisão. Filtros convolucionais com um campo receptivo mínimo de tamanho 3x3 são aplicados à entrada. O passo de convolução é fixo em 1 pixel e o preenchimento espacial da entrada da camada de convolução é projetado para preservar a resolução espacial após a convolução (por exemplo, preenchimento de 1 pixel para camadas de convolução 3x3). O processo de pooling é realizado utilizando camadas de pooling max que seguem algumas das camadas convolucionais. O pooling max é realizado com uma janela de 2x2 pixels. Todas as três camadas totalmente conectadas sucedem às camadas convolucionais, possuindo diferentes profundidades em variações diferentes da arquitetura. As duas primeiras camadas possuem 4096 canais cada, e a terceira camada se conecta à classificação ILSVRC com 1000 canais, atribuindo um canal para cada categoria. A camada Softmax é a camada final e a configuração das camadas totalmente conectadas é semelhante em todas as redes VGG (Simonyan & Zisserman, 2014).

Figura 8: Arquitetura da rede neural VGG



Fonte: Boesch (2024)

O VGG-16 tem sido amplamente utilizado em várias aplicações de visão computacional devido à sua eficácia e simplicidade estrutural. Pesquisas indicam que redes profundas como o VGG-16 podem capturar representações mais ricas e discriminativas das imagens, resultando em um desempenho superior em tarefas de classificação de imagens (Szegedy et al., 2015; He et al., 2016).

2.12.2 VGG-19

A principal diferença entre o VGG-16 e o VGG-19 é a adição de três camadas convolucionais extras no VGG-19, totalizando 19 camadas convolucionais. Essas camadas adicionais permitem que a rede capture características mais complexas e sutis das imagens, potencialmente melhorando a precisão e a capacidade de generalização do modelo. Estudos indicam que a maior profundidade do VGG-19 pode resultar em um desempenho superior em benchmarks de classificação de imagens, como o ImageNet, em comparação com redes menos profundas (Liu et al., 2015).

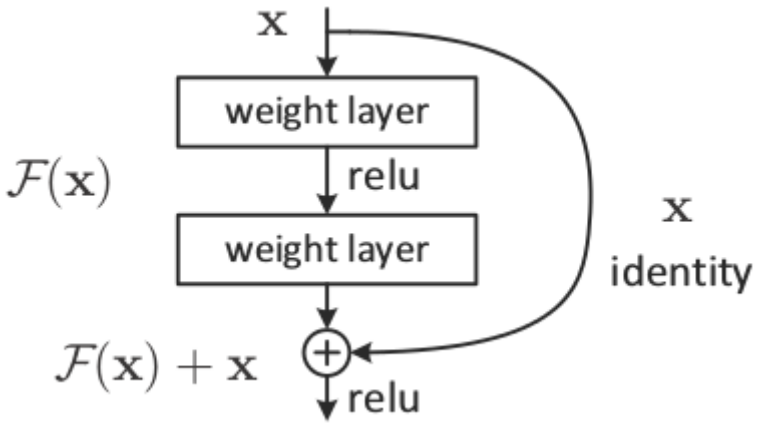
Adicionalmente, o VGG-19 tem se mostrado eficaz em tarefas de segmentação e detecção de objetos, demonstrando sua versatilidade e robustez em várias aplicações de visão computacional. Pesquisas como as de Guo et al. (2017) e Redmon & Farhadi (2018) evidenciam a aplicação bem-sucedida do VGG-19 em contextos diversos, consolidando sua importância no campo da visão computacional.

2.12.3 Resnet

A Residual Network (*ResNet*) trouxe um novo paradigma no treinamento de redes neurais profundas, permitindo a construção de redes com mais de 100 camadas, algo que era impraticável anteriormente devido do *vanish gradient*.

Para superar esse desafio, a arquitetura *ResNet* propôs uma solução inovadora: conexões residuais (Figura 9). Essas conexões introduzem um "caminho direto" para o fluxo de informações, conectando a entrada de cada bloco convolucional à sua saída. Essa abordagem, similar a "pontes" entre os blocos de construção da rede, garante que informações importantes sejam preservadas e propaguem-se facilmente pelas camadas mais profundas, mesmo em redes com grande número de camadas. Dessa forma, as conexões residuais ajudam a mitigar o problema de *vanishing gradient*, facilitando a propagação do gradiente e permitindo o treinamento de redes muito mais profundas sem perda significativa de desempenho.

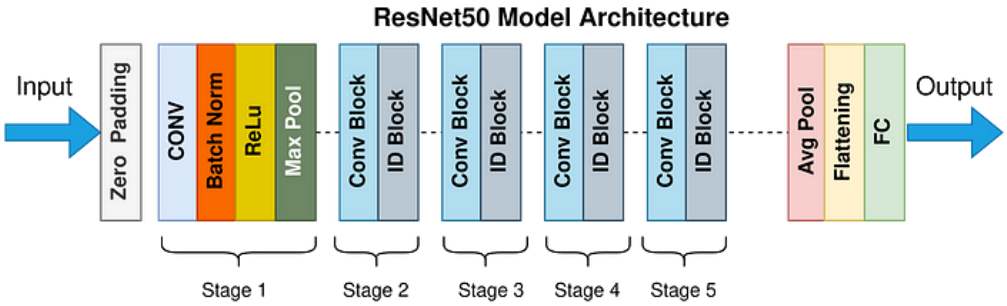
Figura 9: Bloco residual (skip connection)



Fonte: He et al. 2015

Além disso, a normalização por lotes (*batch normalization*) foi incorporada em cada bloco da arquitetura *Resnet* (Figura 10). Conforme Ioffe & Szegedy (2015), essa técnica visa normalizar a distribuição dos dados em cada camada, facilitando o aprendizado da rede e acelerando a convergência durante o treinamento de redes neurais profundas.

Figura 10: Arquitetura Resnet50



Fonte: Mukherjee, 2022.

A capacidade de treinar redes muito profundas de forma eficiente abriu novas possibilidades para a exploração de redes ainda mais complexas e poderosas, solidificando a *ResNet* como um marco na evolução das redes neurais convolucionais.

2.12.4 Inception

A arquitetura Inception (Figura 11) é uma rede neural convolucional desenvolvida para tarefas de visão computacional. Ela foi criada com o objetivo de melhorar o desempenho das redes neurais convolucionais, especialmente em termos de eficiência computacional e baixo número de parâmetros.

Conforme o Guia Avançado do Inception v3 do Google Cloud (2023), o modelo é constituído por camadas de convolução, *pooling* médio, *pooling* máximo, concatenações, *dropouts* e camadas totalmente conectadas. A normalização em lote é utilizada em todo o modelo e aplicada às entradas de ativação. A função Softmax é empregada para calcular a perda.

A camada de entrada da Inception V3 processa imagens em um tamanho padrão de 299 x 299 pixels RGB. A entrada é passada por várias camadas convolucionais para extrair características detalhadas e alcançar resultados de alta precisão. A arquitetura utiliza convoluções com diferentes tamanhos de filtro, como 1x1, 3x3 e 5x5, aplicados em paralelo, e os resultados são concatenados para formar a saída.

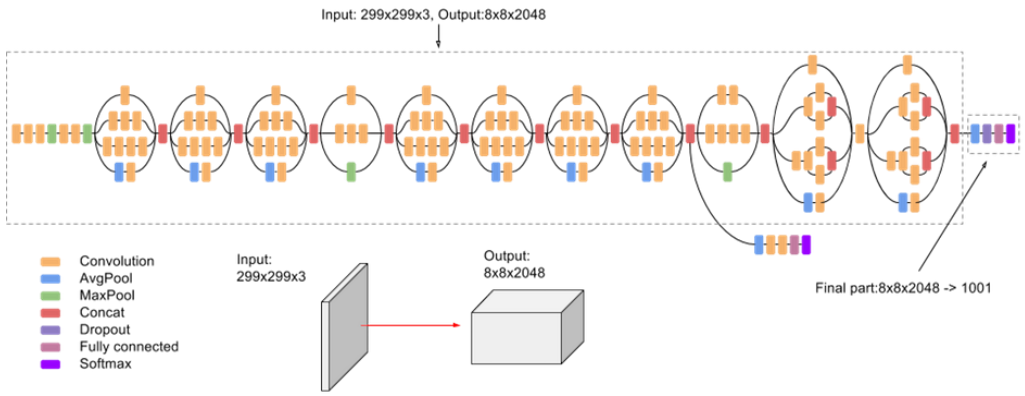
O passo de convolução é fixo em 1 pixel e o preenchimento é ajustado para preservar a resolução espacial após a convolução. A normalização em lote é extensivamente utilizada para estabilizar e acelerar o treinamento. O processo de *pooling* é realizado utilizando tanto camadas de *pooling* médio quanto *pooling* máximo, com janelas típicas de 3x3 pixels.

A arquitetura Inception V3 incorpora módulos Inception, que são blocos com várias convoluções e operações de *pooling* executadas em paralelo, cujos resultados são concatenados. Esses módulos permitem a extração de características em múltiplas escalas, melhorando a capacidade de representação do modelo.

Dropouts são aplicados antes das camadas totalmente conectadas para prevenir *overfitting*. As camadas totalmente conectadas sucedem os módulos Inception, e a última camada totalmente conectada possui 1000 unidades, correspondendo às 1000 classes da classificação ILSVRC. A função Softmax é utilizada na camada final para calcular a probabilidade de cada classe.

A configuração das camadas e módulos na Inception V3 é projetada para otimizar a eficiência computacional enquanto mantém um alto desempenho na classificação de imagens (Szegedy et al., 2016).

Figura 11: Arquitetura da rede neural Inception v3



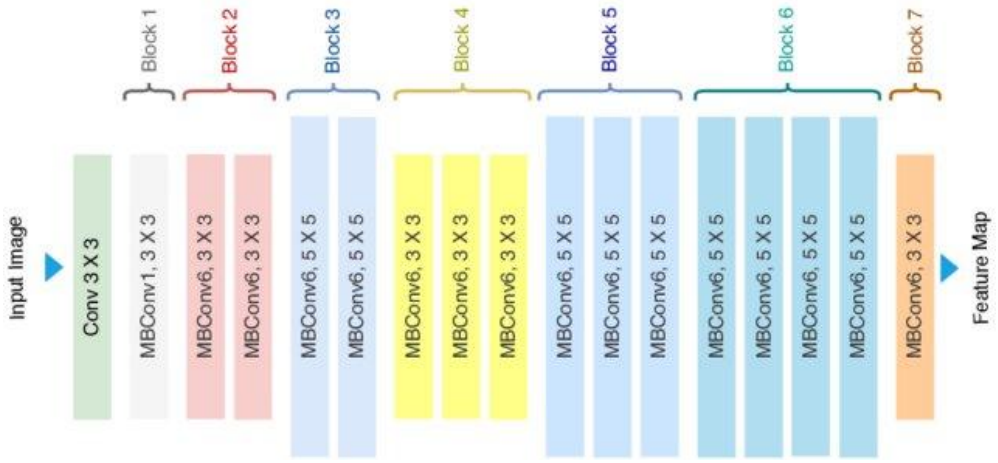
Fonte: Google Cloud

2.12.5 EfficientNET

EfficientNet é uma família de arquiteturas de redes neurais convolucionais (CNN) que otimiza simultaneamente a profundidade, a largura e a resolução da rede para melhorar a precisão e a eficiência computacional. A abordagem de dimensionamento composto, introduzida por Tan e Le (2019), permite que o modelo seja escalado de forma equilibrada, resultando em uma rede altamente eficiente.

A Figura X apresenta a arquitetura do EfficientNet. A rede base EfficientNet-B0 foi criada através de uma busca neural de arquitetura (NAS), e versões maiores, EfficientNet-B1 a B7, foram desenvolvidas escalando proporcionalmente os três parâmetros: largura, profundidade e resolução. Esta técnica de dimensionamento

composto permite que a rede atinja uma alta precisão com um custo computacional relativamente baixo.



Fonte: Ahmed *et al*. (2020)

Tan e Le (2019), afirmam que a arquitetura EfficientNet-B0, a versão base, alcança uma eficiência significativamente maior em termos de FLOPS (operações de ponto flutuante por segundo) comparado com outras arquiteturas populares, como ResNet e Inception. As versões escaladas (B1 a B7) continuam essa tendência, oferecendo melhorias incrementais em desempenho com aumentos moderados no custo computacional.

Conforme pesquisa realizada por Liu et al. (2020), a eficácia do EfficientNet foi confirmada em uma variedade de tarefas, incluindo detecção de objetos e segmentação semântica. Neste estudo, foi demonstrado que a arquitetura não só atinge uma melhor precisão em benchmarks de classificação de imagens, como o ImageNet, mas também se adapta bem a outras aplicações de visão computacional, reforçando sua versatilidade.

2.12.6 Comparando arquiteturas

Ao compararmos as arquiteturas, podemos observar uma evolução contínua e inovação no campo do aprendizado profundo, como indicado no quadro 1. A arquitetura VGG enfatizou a importância da profundidade na extração de características visuais complexas, enquanto a ResNet solucionou questões relacionadas ao treinamento de redes profundas por meio de conexões residuais. Por sua vez, a Inception introduziu uma abordagem multi-escala inovadora. Juntas, essas arquiteturas servem como base para uma variedade de pesquisas e aplicações em aprendizado de máquina e visão computacional.

Quadro 1: Arquiteturas de Redes Neurais Convolucionais (*CNN*) para Visão Computacional: Uma Comparação Detalhada

Característica	**VGG**	**ResNet**	**Inception**
Autores	Simonyan e Zisserman (Universidade de Oxford)	He et al. (Microsoft)	Szegedy et al. (Google)
Ano	2014	2016	2014
Princípios Básicos	Camadas convolucionais 3x3, max pooling, empilhamento de camadas	Conexões residuais, redes profundas (100+ camadas)	Módulos Inception, aprendizado de filtros em várias escalas
Variações	VGG-16, VGG-19	ResNet-50, ResNet-101, ResNet-152	Inception v2, v3, v4
Impacto	Demonstrou a importância da profundidade da rede	Permitiu redes muito mais profundas e eficientes (venceu ILSVRC 2015)	Resultados notáveis no ILSVRC 2014, inspirou outras arquiteturas
Pontos Fortes	Simplicidade, eficiência, robustez	Treinamento de redes profundas, melhor desempenho em tarefas complexas	Captura de informações em várias escalas, alta precisão
Pontos Fracos	Alta demanda por memória e poder computacional	Dificuldades de implementação e treinamento	Complexidade aumentada, maior consumo de memória
Aplicações	Reconhecimento de imagens, classificação de imagens, detecção de objetos	Reconhecimento de imagens, classificação de imagens, segmentação de imagens	Reconhecimento de imagens, classificação de imagens, tarefas de visão com múltiplos objetos

Fonte: Autor

2.12.7 Arquitetura utilizada

Cada uma dessas arquiteturas introduziu novos conceitos e técnicas que melhoraram significativamente o desempenho em tarefas complexas de processamento de imagens, como reconhecimento de imagens e classificação. A

escolha da melhor arquitetura para o projeto é uma decisão estratégica e depende de diversos fatores:

- Tamanho base de dados: Se a base de dados for pequena, a VGG pode ser uma boa opção. Para grandes *datasets*, a ResNet ou Inception podem oferecer melhores resultados.
- Complexidade das imagens: Se as imagens forem simples, a VGG pode ser suficiente. Para imagens complexas com múltiplos objetos ou cenas detalhadas, a Inception pode ser mais eficaz.
- Recursos computacionais disponíveis: Se os recursos computacionais forem limitados, a VGG ou ResNet podem ser mais adequadas. A Inception exige mais poder de processamento.
- Objetivos da tarefa: Se o objetivo for classificação simples, a VGG ou ResNet podem ser suficientes. Se o objetivo for extração de características complexas ou reconhecimento de objetos em imagens com múltiplos objetos, a Inception pode ser mais vantajosa.

O objetivo desta investigação é identificar não apenas a arquitetura mais adequada para a classificação de folhas de soja, mas também compreender o desempenho de cada uma em relação ao conjunto de dados em questão. Ao explorar os resultados das arquiteturas, espera-se obter insights valiosos que possam orientar pesquisas futuras e aplicações na área de visão computacional agrícola.

2.13 Hiperparâmetros

Os hiperparâmetros são parâmetros que controlam o comportamento do processo de aprendizado das redes neurais, mas que não são ajustados pelo próprio modelo durante o treinamento. Entre os principais hiperparâmetros utilizados em redes neurais estão os otimizadores, as técnicas de regularização, como Dropout, e a normalização das camadas, como a Batch Normalization. Esses elementos são essenciais para garantir a eficácia do treinamento e a capacidade de generalização dos modelos.

2.13.1 Otimizadores

No treinamento de redes neurais, os otimizadores desempenham um papel crucial ao ajustar iterativamente os pesos do modelo para minimizar a função de perda. Esses algoritmos são responsáveis por guiar o processo de aprendizado, buscando o mínimo global da função de custo, onde a rede neural apresenta o melhor desempenho. Entre os otimizadores mais utilizados na literatura estão o Stochastic Gradient Descent (SGD), Adam e AdamW, cada um com suas características particulares que os tornam adequados para diferentes aplicações e arquiteturas.

O SGD é um dos otimizadores mais antigos e amplamente utilizados. Ele funciona atualizando os pesos da rede com base no gradiente da função de perda em relação aos pesos, calculado para um pequeno lote (mini-batch) de dados de treinamento (Bottou, 2010). Embora seja eficiente em termos de memória e adequado para grandes datasets, o SGD pode ser lento para convergir, especialmente em problemas com muitos parâmetros ou em superfícies de erro complicadas.

O Adam (Adaptive Moment Estimation), proposto por Kingma e Ba (2015), combina as vantagens do SGD com momentos de primeira e segunda ordem, adaptando as taxas de aprendizado para cada parâmetro com base nas estimativas de momento acumulado. Isso faz com que o Adam seja eficiente para muitos tipos de problemas e redes profundas, oferecendo uma rápida convergência.

Já o AdamW, introduzido por Loshchilov e Hutter (2019), é uma variação do Adam que inclui a regularização por decaimento de peso de forma desacoplada do processo de atualização dos parâmetros. Essa modificação busca resolver algumas das limitações do Adam, como a acumulação excessiva de pesos, o que pode levar ao overfitting em certos cenários.

2.13.2 Dropout

O Dropout é uma técnica de regularização introduzida por Srivastava et al. (2014) para prevenir o overfitting em redes neurais profundas. A ideia central do Dropout é "desligar" aleatoriamente uma fração de neurônios durante o treinamento, forçando a rede a não depender excessivamente de características específicas aprendidas por um subconjunto de neurônios. Como resultado, a rede se torna mais robusta e generaliza melhor em dados não vistos. Srivastava et al. (2014) demonstraram que essa técnica melhora significativamente o desempenho das redes neurais em diversas tarefas de visão computacional e processamento de linguagem natural. Além disso, Hinton et al. (2012) sugerem que o Dropout pode ser visto como uma forma de ensemble de modelos, onde diferentes sub-redes são treinadas a cada iteração.

2.13.3 Batch Normalization

A Batch Normalization, proposta por Ioffe e Szegedy (2015), é uma técnica utilizada para acelerar o treinamento das redes neurais profundas e aumentar a estabilidade do processo de aprendizado. Essa técnica funciona normalizando as entradas de cada camada, garantindo que as ativações tenham uma distribuição

estável em cada mini-batch. Conforme observado por Ioffe e Szegedy (2015), a Batch Normalization reduz o problema de covariate shift interno, onde a distribuição das entradas de uma camada muda durante o treinamento, permitindo o uso de taxas de aprendizado mais altas e acelerando a convergência. Adicionalmente, Santurkar et al. (2018) mostraram que a Batch Normalization também atua como uma forma de regularização, reduzindo a necessidade de outras técnicas de regularização, como o Dropout, em algumas arquiteturas.

2.14 Transferência de aprendizado

Conforme destacado por Pan e Yang (2010), a estratégia de transferência de aprendizado implica na utilização de um modelo pré-treinado, comumente em uma tarefa genérica, e sua posterior adaptação para uma tarefa específica. Essa abordagem é amplamente empregada na área de visão computacional, notadamente na adaptação de redes neurais convolucionais (CNNs) treinadas em grandes conjuntos de dados, como o ImageNet, para tarefas específicas de classificação de imagens.

No contexto deste projeto, essa metodologia foi adotada com o propósito de agilizar o processo de treinamento e aprimorar o desempenho do modelo. Tal estratégia se justifica pela capacidade do modelo pré-treinado em extrair características relevantes das imagens, o que pode ser benéfico para a tarefa em questão.

2.14.1 Ajuste Fino (Fine-tunning)

O ajuste fino (*fine-tuning*) é um conceito fundamental na transferência de aprendizado, no qual os parâmetros internos de um modelo pré-treinado são modificados para adaptá-lo especificamente aos requisitos da nova tarefa. Esse refinamento permite uma personalização mais precisa do modelo, visando melhor adequação aos requisitos específicos dos dados e, consequentemente, um desempenho otimizado na classificação das imagens de plantas.

2.15 Avaliação do modelo

Ao desenvolver modelos de aprendizado de máquina (ML), a avaliação rigorosa é crucial para determinar sua efetividade e confiabilidade em cenários reais. As métricas de avaliação fornecem insights valiosos sobre o desempenho do modelo, permitindo identificar pontos fortes, fracos e oportunidades de aprimoramento.

Os métodos de avaliação utilizados (Quadro 2) visaram mensurar vários aspectos do modelo, como acurácia, precisão, recall e F1-score. Esses métodos também consideraram os Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN), que são fundamentais para avaliar a performance do modelo.

Esses indicadores nos permitem avaliar a previsibilidade, precisão e confiabilidade do modelo. A métrica de Acurácia – conforme demonstrado na Equação (3) – abrange todos os elementos contributivos e foi utilizada como métrica de avaliação porque fornece uma visão geral simples e intuitiva da performance do modelo. Além disso, a acurácia é especialmente útil em cenários onde a distribuição das classes é balanceada, permitindo uma avaliação direta da proporção de previsões corretas.

Quadro 2: Métricas utilizadas para avaliação do modelo

Métricas	**Fórmula**		**Interpretação**
Acurácia	$\frac{VP + VN}{VP + FN + VN + FP}$	(3)	Proporção de previsões corretas.
F1-Score	$\frac{2 * \text{Precisão} * \text{Recall}}{\text{Precisão} + \text{Recall}}$	(4)	Combinação de precisão e recall em uma única medida.
Precisão	$\frac{VP}{VP + FP}$	(5)	Proporção de amostras positivas classificadas corretamente.
Recall	$\frac{VP}{VP + FN}$	(6)	Proporção de amostras positivas identificadas corretamente.

Fonte: Autor

3. METODOLOGIA

3.1 Problema de negócio

A produção de soja é uma das principais atividades agrícolas em várias regiões do mundo, sendo uma cultura de extrema importância econômica. Entretanto, a presença de pragas e doenças, como a *Diabrotica speciosa* e a *Caterpillar*, pode resultar em perdas significativas de produtividade e aumento dos custos de produção. A detecção precoce e precisa dessas doenças é crucial para mitigar danos e assegurar a qualidade e o rendimento da colheita. Assim, o desenvolvimento de um sistema automatizado, baseado em visão computacional, visa fornecer uma solução tecnológica avançada para os agricultores, permitindo uma resposta rápida e eficaz na gestão de suas plantações.

3.2 Base de dados

3.2.1 Origem

Para o desenvolvimento deste projeto, foi utilizada uma base de dados proveniente do Kaggle, uma plataforma amplamente reconhecida por hospedar *datasets* de alta qualidade e pela sua comunidade ativa de cientistas de dados e pesquisadores. A escolha do Kaggle como fonte de dados deve-se à sua confiabilidade, diversidade de base de dados disponíveis e facilidade de acesso, permitindo uma rápida integração com ferramentas de análise e algoritmos de aprendizado de máquina.

3.2.2 Composição

O *dataset* é composto por três classes: *Caterpillar*, *Diabrotica Speciosa* e *Healthy*. As imagens representam folhas de soja, sendo algumas danificadas pela lagarta (*Caterpillar*), outras pela *Diabrotica Speciosa*, e algumas saudáveis (*Healthy*). Todas as imagens foram padronizadas com dimensões de 500 x 500 pixels, facilitando o processamento e a análise posterior.

Este conjunto de dados foi projetado para ser utilizado em diversas áreas de inteligência artificial, aprendizado de máquina e aprendizado profundo, entre outras aplicações. A padronização e a alta qualidade das imagens tornam este *dataset* particularmente adequado para tarefas de classificação.

3.2.3 Desafios adicionais

Embora a base de dados proveniente do Kaggle seja de grande valia para este estudo, é importante destacar alguns desafios adicionais que surgem na aplicação prática desta solução. A disponibilidade de conjuntos de dados de imagens de plantas com as doenças específicas abordadas neste trabalho, como *Caterpillar* e *Diabrotica Speciosa*, é limitada na internet. Isso pode dificultar a validação e o teste do algoritmo em outras bases de dados.

Este trabalho possui um caráter acadêmico e de estudo. Para que uma empresa possa aplicar esta solução em um ambiente real, recomenda-se a coleta de suas próprias imagens. A qualidade das fotografias pode variar significativamente devido a fatores como iluminação, resolução da câmera e condições climáticas, o que pode influenciar a eficácia do algoritmo. Portanto, é essencial que as empresas façam ajustes e modificações no algoritmo, adaptando-o às suas necessidades específicas. Dessa forma, garante-se que ele funcione de maneira eficiente nas condições particulares de cada situação.

3.3 Exploração de dados

3.3.1 Divisão dos dados

Um aspecto fundamental a ser considerado é o desbalanceamento da base de dados, que pode influenciar negativamente a performance do algoritmo. Conforme Silva et al. (2020), o desbalanceamento pode prejudicar o desempenho do modelo, uma vez que a predominância de uma classe sobre as outras dificulta a capacidade do algoritmo em aprender e generalizar adequadamente. No *dataset* utilizado, há uma disparidade no número de imagens entre as classes *Caterpillar*, *Diabrotica Speciosa* e *Healthy*, com as classes *Caterpillar* e *Diabrotica Speciosa* apresentando um número significativamente maior de amostras em comparação à classe *Healthy*, conforme demonstrado na tabela 1.

Tabela 1: Distribuição de imagens por classe

Classificação	**Quantidade de imagens**
Caterpillar	3.309
Diabrotica Speciosa	2.205
Saudável	896

Fonte: Autor

Esse desbalanceamento pode levar o modelo a ser enviesado para as classes majoritárias, resultando em uma baixa acurácia na detecção de plantas saudáveis. Quando os dados são desbalanceados, o algoritmo tende a aprender melhor as características das classes com mais amostras, enquanto pode negligenciar as classes minoritárias. Esse fenômeno pode comprometer a generalização do modelo e sua eficácia em aplicações práticas.

Para mitigar o efeito do desbalanceamento, foram empregadas técnicas de *Data Augmentation*. Essas técnicas visam aumentar a diversidade do conjunto de treinamento através da criação de novas amostras sintéticas a partir das existentes. Isso é realizado aplicando diversas transformações nas imagens originais, como rotação, translação, escala, espelhamento, entre outras. O objetivo é enriquecer o conjunto de dados com variações que o algoritmo possa encontrar no mundo real, melhorando assim sua robustez e capacidade de generalização.

3.3.2 Características das classes

a) Diabrotica speciosa

Diabrotica speciosa (Figura 12), conhecida popularmente como vaquinha, é uma praga polífaga que afeta várias culturas no Brasil e em outros países da América do Sul (ROSA; TRECHA; MEDINA, 2013).

Figura 12: Diabrótica Speciosa

Fonte: Agrolink

Na fase adulta, esse inseto se alimenta de folhas, brotos, vagens e frutos, resultando em uma significativa redução na produtividade das plantas. Os adultos dessa praga preferem folhas mais jovens e tenras, onde realizam pequenos orifícios, o que compromete a qualidade e o rendimento das culturas (SOSA GOMEZ, 2021).

Conforme ilustrado na Figura 13, é possível notar características como:

- Furos irregulares: Bordas irregulares e tamanhos variados, indicando a ação de lagartas.
- Falta de tecido foliar: Áreas significativas de tecido foliar faltando, evidenciando o consumo pelas lagartas.
- Variações de cor: Descoloração ou perda de clorofila nas áreas danificadas, afetando a pigmentação das folhas.

Figura 13: Plantas afetadas por *Diabrotica Speciosa*

Fonte: Autor

b) Caterpillar

Caterpillar, *Anticarsia gemmtalis* (Figura 14) é também conhecida como lagarta-da-soja, é um inseto lepidóptero, pertencente à família *Noctuidae*. Esse inseto é encontrado em todos os locais de cultivo, sendo o desfolhador mais comum da soja no Brasil.” (Hoffmann-Campo, C.B. *et al*., 2000, p.11).

Figura 14: Lagarta da soja (*Caterpillar*)

Fonte: Agrolink

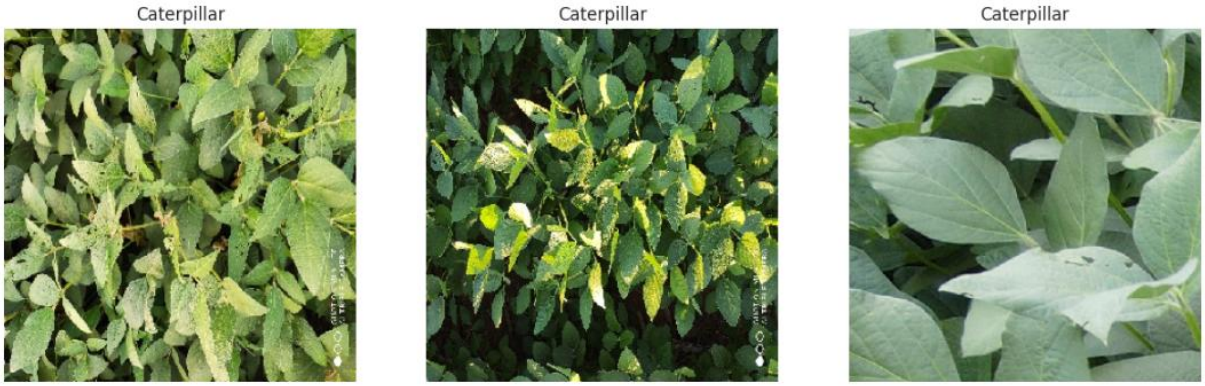
De acordo com Praça et al. (2006), este inseto causa grandes danos à lavoura de soja, desde o desfolhamento até a destruição completa da planta. Se não

controlado no momento oportuno, este inseto pode provocar desfolhas elevadas, causando perdas de produtividade da cultura. (Koppert, 2024).

A figura 15 ilustra características específicas que podem ser detalhadas como:

- Furos irregulares: Bordas irregulares e tamanhos variados, indicando a ação de lagartas.
- Falta de tecido foliar: Áreas significativas de tecido foliar faltando, evidenciando o consumo pelas lagartas.
- Variações de cor: Descoloração ou perda de clorofila nas áreas danificadas, afetando a pigmentação das folhas.

Figura 15: Plantas afetadas por *Caterpillar*



Fonte: Autor

c) Saudáveis

A observação cuidadosa dos aspectos físicos das plantas permite avaliar seu estado de saúde e identificar eventuais problemas. Plantas saudáveis apresentam características distintas que as diferenciam de plantas doentes ou estressadas (Figura 16):

- Ausência de furos: Folhas sem furos ou perfurações, indicando a integridade do tecido foliar.
- Cor verde uniforme: Pigmentação verde consistente em toda a superfície das folhas, sinal de saúde vegetal.
- Textura lisa: Superfície foliar sem rugosidades ou ondulações, indicando crescimento normal.

Figura 16: Plantas Saudáveis

Fonte: Autor

d) Percepção de padrões e aprendizado

Para o desenvolvimento de um algoritmo de aprendizado de máquina eficiente na classificação de imagens de plantações de soja, é fundamental que ele interprete e extraia informações relevantes das imagens. Durante o processo de treinamento, é esperado que o algoritmo aprenda a identificar essas diferenças de padrões de características que são indicativas das diferentes classes (Quadro 3).

Quadro 3: Diferenças de características entre as classes

Característica	**Saudável**	**Caterpillar**	**Diabrotica Speciosa**
Furos	Ausente	Presente, irregular	Presente, arredondado
Danos nas bordas	Ausente	Frequente	Ausente
Falta de tecido foliar	Ausente	Significativa	Mínima / ausente
Variações de cor	Ausente	Presente, Descoloração	Ausente
Textura da folha	Lisa	Pode ser irregular	Lisa

Fonte: Autor

O algoritmo de aprendizado de máquina não se limita a memorizar pixels individuais, mas sim a identificar padrões e relações complexas entre eles. Essa capacidade de abstração permite que o modelo generalize o conhecimento adquirido para novas imagens, mesmo que apresentem variações de luz, posição ou outros fatores.

3.3.3 Balanceamento de classes

Para tratar o desbalanceamento de classes na base de dados, onde as classes de *Caterpillar* e *Diabrotica Speciosa* possuem um número significativamente maior de imagens em comparação com a classe Saudável, foi utilizado o pacote ImageDataGenerator do Keras. Este pacote oferece uma variedade de transformações que podem ser aplicadas às imagens para aumentar a diversidade do conjunto de dados de treinamento. As transformações escolhidas são comuns em tarefas de visão computacional e ajudam a generalizar o modelo sem alterar a semântica das imagens (Quadro 4).

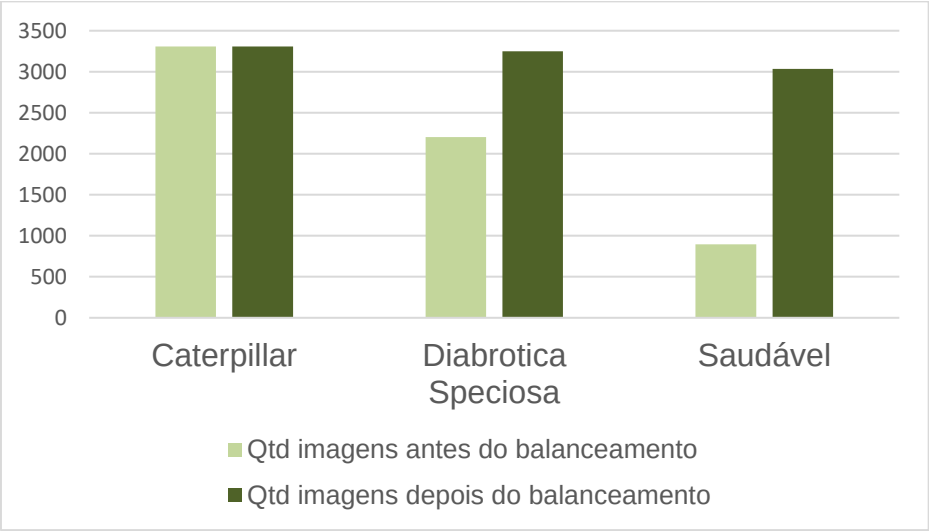
Quadro 4: Configurações do pacote ImageDataGenerator para aumento de dados

Configuração	**Valor**	**Ação**
rotation_range	20	Rotaciona as imagens em até 20 graus. Isso ajuda a simular diferentes orientações das folhas, que podem ocorrer naturalmente.
width_shift_range	0.2	Desloca a imagem horizontalmente em até 20% da largura da imagem. Simula pequenas variações no posicionamento da folha.
height_shift_range	0.2	Desloca a imagem verticalmente em até 20% da altura da imagem. Simula pequenas variações no posicionamento da folha.
shear_range	0.2	Aplica cisalhamento às imagens, distorcendo-as em até 20%. Simula deformações nas folhas.
zoom_range	0.2	Aumenta ou diminui o zoom das imagens em até 20%, simulando diferentes distâncias da câmera à folha.
horizontal_flip	TRUE	Inverte horizontalmente as imagens, útil porque as folhas podem aparecer em orientações espelhadas.
fill_mode	nearest	Preenche os pixels gerados após a transformação usando o valor do pixel mais próximo, mantendo a integridade visual da imagem.

Fonte: TensorFlow

As técnicas de *data augmentation* escolhidas visam aumentar a diversidade do conjunto de dados de treinamento de maneira que as características visuais críticas para a classificação de doenças sejam preservadas. O gráfico 1 apresenta as classes devidamente balanceadas após rodar o pacote ImageDataGenerator.

Gráfico 1: Quantidade de imagens antes e após balanceamento



Fonte: Autor

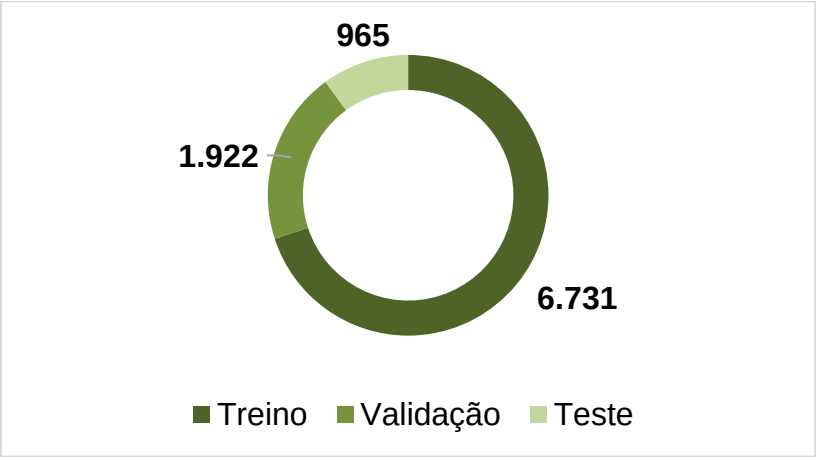
3.3.4 Transformações de dados

Devido à arquitetura distinta das redes neurais convolucionais utilizadas neste estudo, as transformações aplicadas aos dados de entrada variam conforme o modelo adotado. Os experimentos de diferentes transformações nas imagens foram essenciais para garantir a compatibilidade e eficácia do modelo, adaptando os dados às exigências e peculiaridades de cada arquitetura de rede neural convolucional.

3.4 Divisão entre treino e teste

Para realizar a divisão dos dados em conjuntos de treino, validação e teste, foi utilizado o pacote splitfolders do Python. A divisão foi realizada com uma proporção de 70% dos dados para treino, 20% para validação e 10% para teste e com a semente 100. Essa abordagem garante que o modelo tenha dados suficientes para aprender durante o treinamento, avaliar seu desempenho durante a validação e ser testado de forma robusta com dados não vistos anteriormente. A divisão total dos dados é mostrada no gráfico 2.

Gráfico 2: Distribuição de dados entre treino e teste



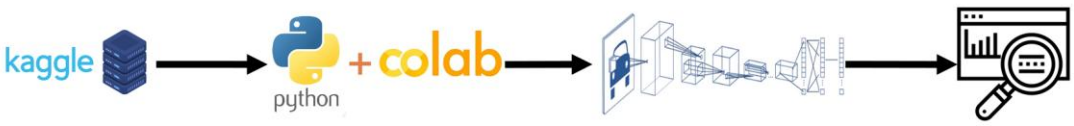
Fonte: Autor

3.5 Pipeline de dados

A arquitetura desenvolvida (Figura 17) e implementada no presente trabalho reflete um ajuste necessário às limitações de recursos e infraestrutura disponíveis, mantendo-se funcional e eficaz para a tarefa proposta. A seguir, são descritos os passos realizados:

- Extração de Dados do Kaggle: Os dados utilizados foram extraídos da plataforma Kaggle.
- Transformações e Desenvolvimento do Modelo no Google Colab: Utilizando Python e o ambiente Jupyter Notebook, foram realizadas as transformações necessárias nos dados e desenvolvimento do modelo de redes neurais convolucionais.
- Treinamento e Testes: Foram utilizadas diferentes arquiteturas de redes neurais convolucionais para treinar e testar os modelos, permitindo uma comparação detalhada de seu desempenho.
- Avaliação da modelagem: Nesta etapa, foi destacada a interpretação dos resultados utilizando métricas como acurácia, F1 score, recall e precisão, além da análise da matriz de confusão e gráficos de curvas de aprendizado.

Figura 17: Arquitetura desenvolvida



Fonte: Autor

3.6 Metodologia do desenvolvimento

3.6.1 Extração de características

Durante a etapa de extração de características, foram utilizadas diferentes arquiteturas pré-treinadas para extrair representações relevantes das imagens de plantas. Essas características foram então utilizadas como entrada para o modelo de classificação, permitindo capturar informações importantes e distintivas de cada classe, representando o baseline preliminar.

3.6.2 Teste e avaliação do modelo

Após o treinamento inicial do modelo, foram realizados testes e avaliações para verificar o desempenho das arquiteturas na classificação das imagens de plantas. Utilizaram-se métricas como precisão, recall e F1-score para avaliar a capacidade do modelo em distinguir entre as diferentes classes de plantas com precisão e robustez.

3.6.3 Fine-Tuning

Com base nos resultados obtidos durante os testes e avaliações, foram realizados ajustes no modelo por meio do fine-tuning. Nesta fase, diferentes parâmetros foram otimizados para adaptar o modelo mais especificamente aos dados do projeto, visando melhorar a capacidade de classificação ao lidar com as nuances das imagens de plantas.

4. DESENVOLVIMENTO

Nesta seção, são expostos os procedimentos empregados para a construção dos modelos de redes neurais convolucionais (CNNs), nos quais foram experimentadas diversas técnicas de transformações de dados e diferentes otimizadores, com o objetivo de aprimorar o desempenho dos modelos. O intuito foi avaliar de forma progressiva o impacto de cada técnica aplicada, observando o desempenho em diferentes etapas do desenvolvimento.

Para cada arquitetura de rede neural convolucional, foram testados cinco modelos distintos, com incrementos progressivos de técnicas de aprimoramento. Inicialmente, as imagens foram submetidas a transformações básicas para adequação ao formato exigido pelas arquiteturas. Em seguida, técnicas adicionais de pré-processamento foram aplicadas para aumentar a diversidade e robustez do conjunto de dados de treinamento.

4.1 Transformações de dados

As transformações aplicadas no pré-processamento dos dados são apresentadas no Quadro 5. Cada transformação teve um propósito específico, desde o ajuste do tamanho das imagens até a introdução de variações aleatórias para melhorar a capacidade de generalização dos modelos.

Quadro 5: Transformações aplicadas no pré-processamento de dados

Index	Transformações aplicadas	Motivo	Propósito
1	Redimensionamento	Cada arquitetura de CNN tem um tamanho de entrada fixo para o qual foi originalmente projetada e treinada.	Ajustar todas as imagens para um tamanho consistente.
2	Normalização	Normalização ajuda a estabilizar o treinamento, melhorar a convergência e alcançar melhores desempenhos.	Padronizar os valores dos pixels com a média e o desvio padrão usados no treinamento da rede.
3	Conversão para Tensor	Bibliotecas de deep learning como PyTorch e TensorFlow trabalham com tensores (n-dimensional arrays) em vez de imagens brutas.	Converter imagens de PIL ou NumPy arrays para tensores PyTorch.

4	RandomHorizontalFlip	Aumenta a variedade do conjunto de dados sem precisar coletar novas imagens, ajudando a rede a aprender invariância à rotação horizontal.	Permitir que a rede generalize melhor ao aprender a reconhecer objetos independentemente da orientação horizontal.
5	RandomRotation	Aumenta a robustez do modelo ao adicionar variedade nas orientações das imagens de treino.	Ajudar a rede a ser mais invariável a diferentes ângulos de rotação
6	Color Jitter	Introduz variações aleatórias no brilho, contraste, saturação e matiz das imagens de treino, aumentando a diversidade do conjunto de dados.	Melhorar a robustez do modelo a diferentes condições de iluminação e cor, ajudando a rede a generalizar melhor em ambientes reais.

Fonte: Autor

4.2 Modelos testados

Os modelos foram construídos seguindo uma sequência lógica de incrementos em termos de transformações e técnicas de aprimoramento, conforme descrito abaixo:

.1 Modelo Base: Aplicação das transformações básicas descritas no Quadro 5 (Redimensionamento, Normalização, Conversão para Tensor).

.2 Modelo com Aumento de Dados 1: Adição de RandomHorizontalFlip às transformações básicas.

.3 Modelo com Aumento de Dados 2: Inclusão de RandomRotation e ColorJitter, além das transformações do modelo anterior.

.4 Modelo com Dropout: Incorporação de camadas de Dropout ao modelo com todas as transformações anteriores.

.5 Modelo Final: Implementação das técnicas adicionais como batch normalization e ajustes específicos nos otimizadores conforme apropriado para cada arquitetura.

4.3 Parâmetros

Os modelos foram treinados utilizando diferentes otimizadores, cada um ajustado para maximizar a performance da respectiva arquitetura. Os parâmetros específicos dos otimizadores testados são detalhados no Quadro 6.

Quadro 6: Parâmetros e otimizadores utilizados nos testes

Index	Criterion	Otimizador	Parâmetros	Épocas
1	nn.CrossEntropyLoss	SGD	learning rate = 0.001, momentum=0.9	50
2	nn.CrossEntropyLoss	Adam	learning rate = 0.001, betas = (0.9, 0.999), eps = 1e-08	50
3	nn.CrossEntropyLoss	AdamW	learning rate = 0.001, betas = (0.9, 0.999), eps = 1e-08, weight decay = 0.01	50

Fonte: Autor

4.4 Implementação

Cada modelo foi submetido a um rigoroso processo de treinamento e validação, visando garantir a precisão e a capacidade de generalização das arquiteturas testadas. A acurácia, a perda e outras métricas relevantes foram monitoradas ao longo do treinamento para avaliar o impacto das diferentes técnicas e ajustes aplicados.

Essa abordagem metodológica detalhada permitiu uma análise compreensiva do impacto de cada técnica de pré-processamento e otimização no desempenho final das CNNs, criando uma base sólida para a interpretação dos resultados que serão discutidos nas seções subsequentes.

4.5 Ambiente de desenvolvimento

O desenvolvimento inicial deste projeto foi realizado no Google Colab, onde utilizou-se uma GPU NVIDIA T4 para o treinamento das redes neurais. No entanto, devido às limitações de recursos computacionais, o ambiente foi posteriormente migrado para o Kaggle, o que proporcionou uma maior capacidade de processamento, essencial para o ajuste fino das arquiteturas complexas.

Foram utilizados TensorFlow e Keras para o treinamento inicial das arquiteturas devido à sua robustez e facilidade de uso, especialmente em ambientes com suporte otimizado para GPUs. Esses frameworks são amplamente adotados na comunidade de deep learning, o que facilita a implementação de técnicas avançadas e o acesso a uma vasta documentação e suporte.

PyTorch foi explorado durante as fases de fine-tuning por sua flexibilidade e capacidade de permitir uma depuração mais intuitiva, sendo ideal para personalizações específicas dos modelos. Para a avaliação dos modelos, utilizou-se o Scikit-learn, que ofereceu ferramentas robustas para a análise das métricas de

desempenho, como precisão, recall, F1-score e a construção de matrizes de confusão, garantindo uma avaliação precisa e detalhada das arquiteturas testadas.

5. RESULTADOS OBTIDOS

5.1 Extração de características

No desenvolvimento de modelos de visão computacional, a extração de características é uma etapa fundamental, onde as transformações aplicadas aos dados desempenham um papel central. Diferente de humanos, que enxergam imagens como um todo, os computadores interpretam imagens como matrizes de pixels, extraindo padrões e características que ajudam no reconhecimento de classes. As transformações de dados, como redimensionamento, normalização e aumento de dados (data augmentation), são aplicadas para preparar essas imagens, permitindo que as redes neurais convolucionais (CNNs) identifiquem padrões relevantes. Por exemplo, a normalização padroniza os valores dos pixels para melhorar a convergência durante o treinamento, enquanto a conversão para tensor adapta as imagens para o formato requerido pelas bibliotecas de aprendizado profundo, como PyTorch. Transformações como RandomHorizontalFlip e RandomRotation aumentam a variedade do conjunto de dados, ajudando o modelo a aprender a reconhecer objetos independentemente da orientação ou ângulo, enquanto o Color Jitter introduz variações de iluminação, tornando o modelo mais robusto a diferentes condições de iluminação e cor.

O objetivo principal dessas transformações é garantir que o modelo possa aprender com uma representação diversa e enriquecida dos dados, o que melhora a sua capacidade de generalização e precisão na identificação de padrões complexos em novas imagens.

5.2 Acurácia em testes iniciais

Inicialmente, os modelos foram avaliados com base na métrica de acurácia. As Tabelas 2,3,4,5 e 6 apresentam as acurácias obtidas para cada combinação de modelo, arquitetura, número de épocas, transformações de dados aplicadas e parâmetros dos otimizadores. Estas tabelas permitem identificar qual configuração alcançou a melhor acurácia nos dados de teste. Para todos os testes, foram utilizadas 50 épocas no treinamento, os index das transformações de dados são relacionados ao quadro 5, bem como o index dos parâmetros são relacionados ao quadro 6.

Tabela 2: Acurácia em teste: VGG-16

Arquitetura	Transformações de dados	Parâmetros	Acurácia em dados de teste
VGG-16	1,2,3	1	94,30%
VGG-16	1,2,3,4	1	95,63%
VGG-16	1,2,3,4,5,6	1	96,68%
VGG-16	1,2,3,4,5,6	1	95,64%
VGG-16	1,2,3,4,5,6	1	95,63%

Fonte: Autor

Tabela 3: Acurácia em teste: VGG-19

Arquitetura	Transformações de dados	Parâmetros	Acurácia em dados de teste
VGG-19	1,2,3	1	94,39%
VGG-19	1,2,3,4	1	96,26%
VGG-19	1,2,3,4,5,6	1	94,69%
VGG-19	1,2,3,4,5,6	1	96,58%
VGG-19	1,2,3,4,5,6	1	96,27%

Fonte: Autor

Tabela 4: Acurácia em teste: Resnet50

Arquitetura	Transformações de dados	Parâmetros	Acurácia em dados de teste
Resnet50	1,2,3	1	89,94%
Resnet50	1,2,3	2	95,74%
Resnet50	1,2,3	3	93,87%
Resnet50	1,2,3,4	1	90,98%
Resnet50	1,2,3,4	2	94,81%
Resnet50	1,2,3,4	3	94,18%
Resnet50	1,2,3,4,5,6	1	92,01%
Resnet50	1,2,3,4,5,6	2	94,81%

Resnet50	1,2,3,4,5,6	3	95,75%
Resnet50	1,2,3,4,5,6	1	91,07%
Resnet50	1,2,3,4,5,6	2	93,77%
Resnet50	1,2,3,4,5,6	3	95,53%
Resnet50	1,2,3,4,5,6	2	95,93%
Resnet50	1,2,3,4,5,6	3	95,53%

Fonte: Autor

Tabela 5: Acurácia em teste: Inception_v4

Arquitetura	**Transformações de dados**	**Parâmetros**	**Acurácia em dados de teste**
Inception_v4	1,2,3	1	93,65%
Inception_v4	1,2,3	2	93,24%
Inception_v4	1,2,3	3	91,67%
Inception_v4	1,2,3,4	1	93,55%
Inception_v4	1,2,3,4	2	91,25%
Inception_v4	1,2,3,4	3	93,11%
Inception_v4	1,2,3,4,5,6	1	96,77%
Inception_v4	1,2,3,4,5,6	2	92,57%
Inception_v4	1,2,3,4,5,6	3	95,23%
Inception_v4	1,2,3,4,5,6	1	95,86%
Inception_v4	1,2,3,4,5,6	3	95,33%

Fonte: Autor

Tabela 6: Acurácia em teste: EfficientNET

Arquitetura	Transformações de dados	Parâmetros	Acurácia em dados de teste
EfficientNET	1,2,3	1	95,22%
EfficientNET	1,2,3	2	92,00%
EfficientNET	1,2,3	3	92,52%
EfficientNET	1,2,3,4	1	94,70%
EfficientNET	1,2,3,4	2	95,74%
EfficientNET	1,2,3,4	3	96,26%
EfficientNET	**1,2,3,4,5,6**	**1**	**97,40%**
EfficientNET	1,2,3,4,5,6	2	95,43%
EfficientNET	1,2,3,4,5,6	3	96,05%
EfficientNET	1,2,3,4,5,6	1	96,78%
EfficientNET	1,2,3,4,5,6	2	96,05%
EfficientNET	1,2,3,4,5,6	3	95,96%
EfficientNET	1,2,3,4,5,6	1	97,29%

Fonte: Autor

A análise do desempenho dos modelos VGG-16, VGG-19, Inception_v4, ResNet50, e EfficientNet (Tabelas 2-6), com base nas configurações testadas, revela insights importantes sobre o comportamento das redes neurais em termos de acurácia. Os modelos EfficientNet consistentemente alcançaram as melhores acurácias, com destaque para a configuração que utilizou transformações de dados completas (1, 2, 3, 4, 5, 6), combinado com o otimizador SGD, atingindo 97,40%. Essa arquitetura demonstrou ser altamente eficaz em extrair características relevantes dos dados, beneficiando-se do conjunto completo de transformações e de um otimizador que equilibrou bem a exploração e a exploração.

Os modelos ResNet50 também mostraram resultados robustos, especialmente quando utilizaram otimizadores Adam e AdamW, alcançando acurácias de até 95,93% na configuração que incluía técnicas adicionais de regularização. O uso do Dropout foi eficaz em evitar o overfitting em alguns casos, como observado na configuração com SGD que, mesmo apresentando uma ligeira redução de acurácia, ajudou na generalização do modelo.

Os modelos VGG-16 e VGG-19 apresentaram resultados consistentes, mas inferiores comparados ao EfficientNet e ResNet50. A melhor acurácia do VGG-16 foi 96,68%, enquanto o VGG-19 alcançou 96,58%, ambos na configuração mais completa. O Dropout teve um efeito misto nesses modelos, sugerindo que sua aplicação deve ser cuidadosamente ajustada para evitar perda de acurácia.

Por fim, o Inception_v4 mostrou-se mais sensível à escolha do otimizador, com o SGD levando a melhores resultados em comparação ao Adam e AdamW. A configuração mais bem-sucedida para Inception_v4 foi com o SGD e transformações completas, alcançando 96,77% de acurácia.

Esses resultados indicam que a combinação de transformações de dados, escolha do otimizador e técnicas de regularização, como Dropout, são cruciais para otimizar o desempenho dos modelos, sendo necessário um ajuste fino de cada componente para maximizar a acurácia final.

5.3 Avaliação detalhada com novas métricas

Após identificar o modelo com a melhor acurácia (Tabela 7), foram realizados testes adicionais para avaliar outras métricas de desempenho e garantir uma análise abrangente do modelo. Essas métricas incluíram precisão, recall, F1-score, curvas de aprendizado e matriz de confusão, proporcionando uma visão mais detalhada sobre a capacidade do modelo de lidar com diferentes classes.

Tabela 7: Modelo com melhor acurácia

Arquitetura	Épocas	Transformações de dados	Parâmetros	Acurácia em dados de teste
EfficientNET	50	1,2,3,4,5,6	1	97,40%

Fonte: Autor

O relatório de classificação gerado após os testes demonstra um desempenho excelente do modelo selecionado, com uma acurácia global de 97,4%, conforme indicado na Tabela 8. As métricas de precisão, recall e F1-score refletem um bom equilíbrio entre as classes, o que é crucial para evitar que o modelo favoreça uma classe em detrimento das outras.

Tabela 8: Relatório de Classificação

	precision	recall	f1-score	support
Caterpillar	0,98	0,94	0,96	332
Diabrotica speciosa	0,96	0,98	0,97	326
Healthy	0,97	1	0,99	302
accuracy			0,97	960
macro avg	0,97	0,97	0,97	960
weighted avg	0,97	0,97	0,97	960

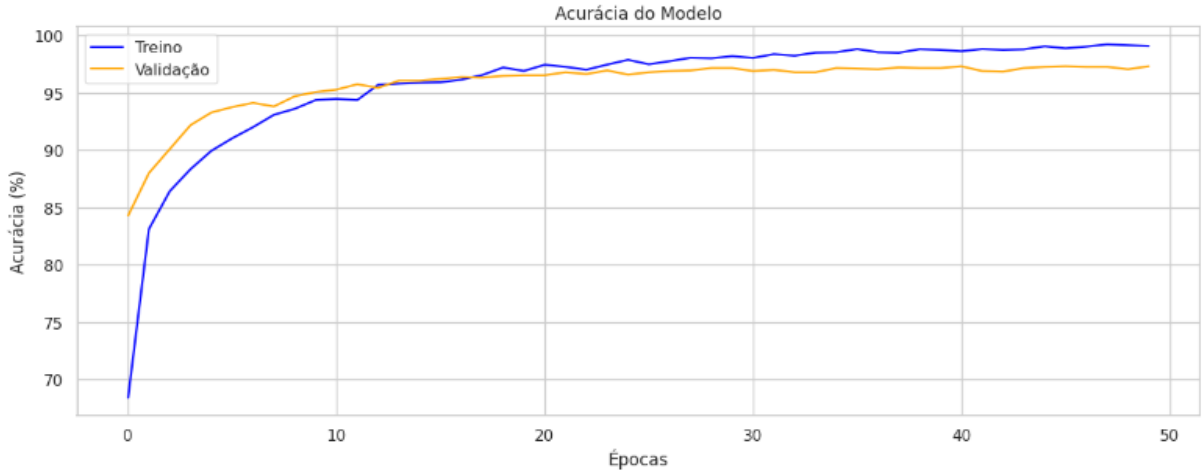
Fonte: Autor

Notavelmente, a classe "Healthy" apresentou um F1-score de 0,99, indicando que o modelo identificou corretamente quase todos os exemplos dessa classe. A classe "Diabrotica speciosa" também obteve um F1-score elevado de 0,97, enquanto a classe "Caterpillar" obteve um F1-score de 0,96. Esses resultados destacam a eficácia do modelo em distinguir entre as diferentes classes, minimizando erros de classificação e garantindo uma performance confiável em diferentes cenários.

Essas análises demonstram que, além da alta acurácia, o modelo tem uma boa capacidade de generalização, sendo capaz de manter um desempenho consistente e equilibrado entre as classes, o que é essencial para a robustez e confiabilidade em aplicações práticas.

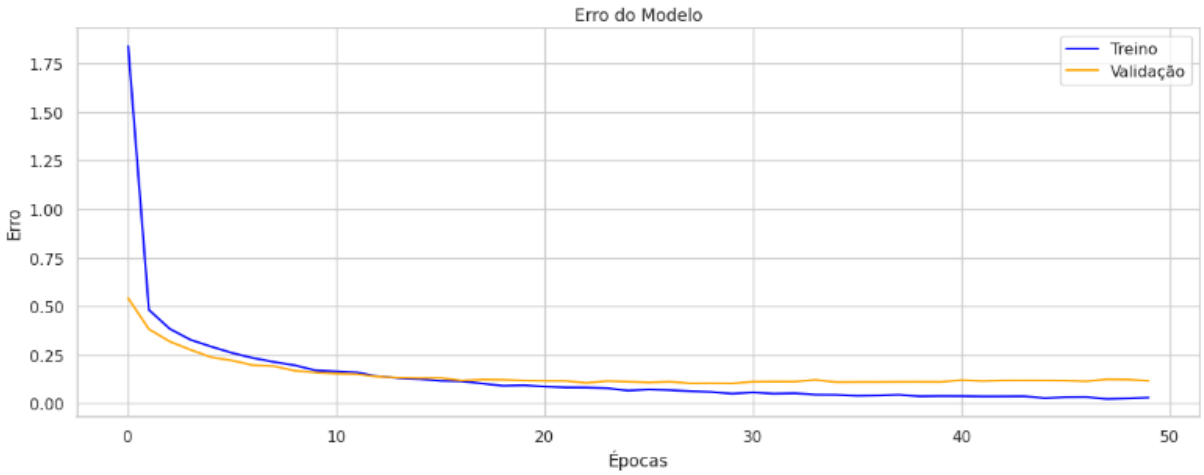
Os gráficos 3 e 4 ilustram a evolução do modelo ao longo das épocas, apresentando a curva de aprendizado para acurácia e erro do modelo, respectivamente. Com base no gráfico 3, observa-se que o modelo treinado atingiu um alto nível de acurácia, tanto nos dados de treino quanto nos de validação, logo nas primeiras épocas, estabilizando-se a partir de aproximadamente 10 a 15 épocas. Isso sugere que o modelo foi capaz de aprender as características relevantes dos dados de forma eficiente.

Gráfico 3: Curva de Aprendizado: Acurácia do modelo



Fonte: Autor

Gráfico 4: Curva de aprendizado: Erro do modelo



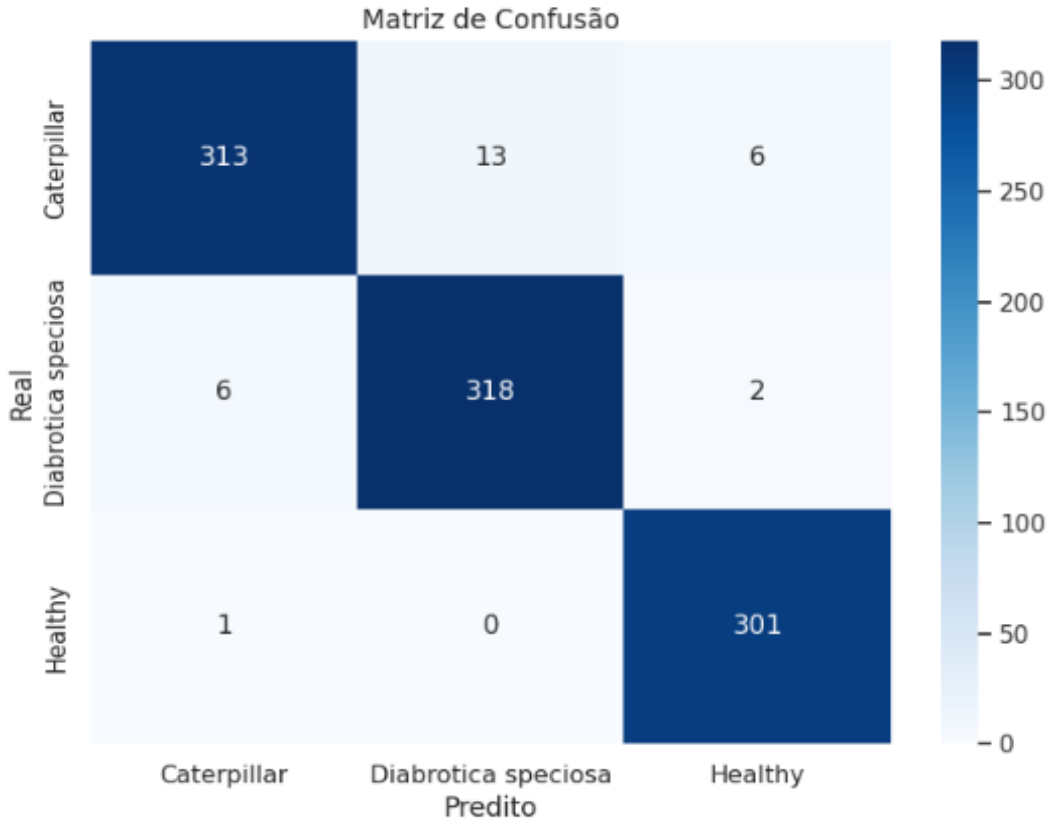
Fonte: Autor

Além disso, no gráfico 4, nota-se uma rápida redução nos erros tanto para o conjunto de treino quanto para o de validação, com uma estabilização semelhante após cerca de 15 épocas. A proximidade entre as curvas de erro de treino e validação indica que o modelo não sofreu de overfitting significativo.

A análise da matriz de confusão (Gráfico 5) demonstra que o modelo apresenta um desempenho sólido em distinguir entre as três classes. A maioria das previsões feitas pelo modelo foi correta, com apenas um pequeno número de erros de classificação. A classe "Caterpillar" teve 19 instâncias incorretamente classificadas, sendo a maioria confundida com "Diabrotica speciosa", o que indica uma pequena

sobreposição nas características dessas duas classes. Para "Diabrotica speciosa" e "Healthy", o modelo foi ainda mais preciso, com apenas 8 erros combinados.

Gráfico 5: Matriz de confusão



Fonte: Autor

5.4 Teste adicional: aumento de número de épocas e ajuste de learning rate

Como último experimento, foi testado um modelo adicional, aumentando o número de épocas de treinamento para 100 e utilizando o ajuste dinâmico da taxa de aprendizado por meio do call-back “ReduceLROnPlateau”. Essa técnica ajusta automaticamente a taxa de aprendizado conforme o desempenho do modelo, permitindo um refinamento mais preciso dos parâmetros à medida que o treinamento avança. O objetivo é melhorar a acurácia do modelo, minimizando o risco de overfitting, enquanto o modelo se aproxima de uma solução ideal. A Tabela 9 apresenta as métricas de desempenho resultantes deste novo modelo.

Tabela 9: Relatório de Classificação - Modelo Adicional

	precision	recall	f1-score	support
Caterpillar	0,98	0,95	0,96	332
Diabrotica speciosa	0,97	0,97	0,97	326
Healthy	0,97	1	0,98	306
accuracy			0,97	964
macro avg	0,97	0,97	0,97	964
weighted avg	0,97	0,97	0,97	964

Fonte: Autor

Além disso, durante o treinamento, o callback "ReduceLROnPlateau" ajustou a taxa de aprendizado em várias ocasiões, como detalhado na Tabela 10. Esses ajustes foram realizados automaticamente pelo algoritmo na tentativa de melhorar a convergência do modelo, reduzindo a taxa de aprendizado à medida que o erro de validação estabilizava.

Tabela 10: Ajuste da taxa de aprendizado durante o treinamento

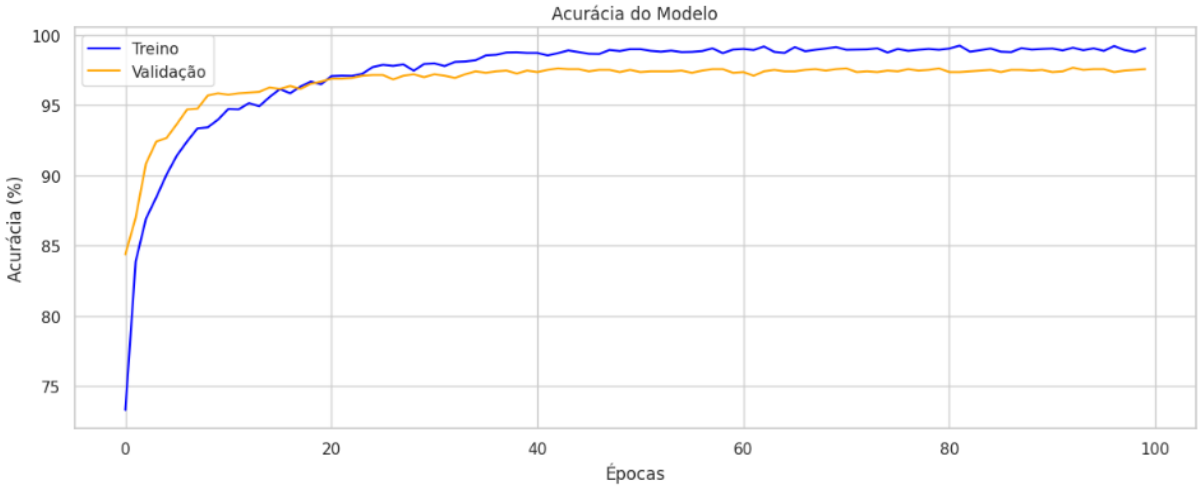
Epoch	Learning Rate
35	5,00E+00
41	2,50E+00
50	1,25E+00
56	6,25E-01
62	3,13E-01
68	1,56E-01
74	7,81E-02
80	3,91E-02
86	1,95E-02
92	1,00E-02

Fonte: Autor

Os gráficos de curva de aprendizado para acurácia e erro do modelo, apresentados respectivamente nos Gráficos 6 e 7, mostram a evolução do

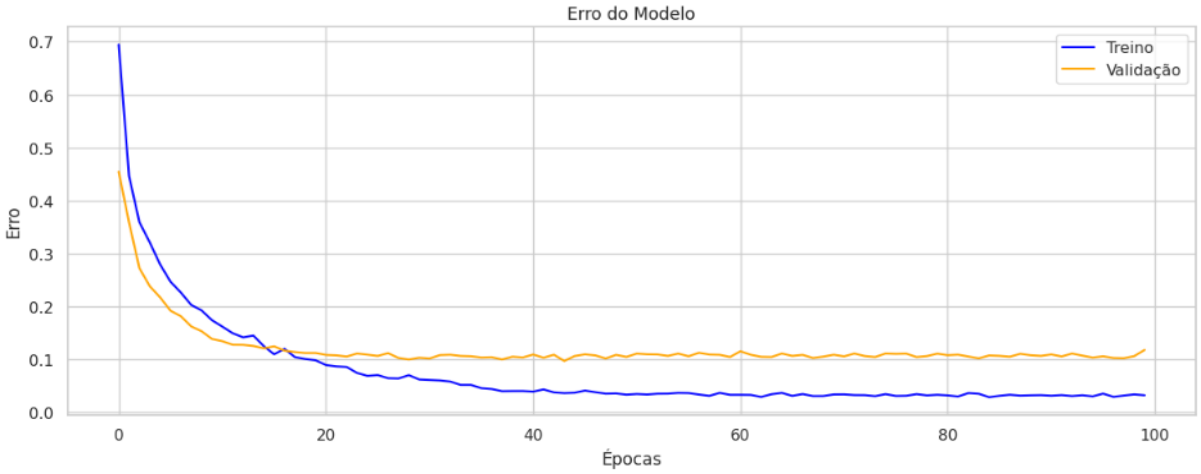
desempenho ao longo das 100 épocas. Conforme observado, a curva de acurácia estabilizou-se rapidamente, sem grandes variações após as primeiras 15 épocas. De forma similar, o erro, tanto no conjunto de treino quanto no de validação, seguiu um padrão estável, o que sugere que o aumento de recursos computacionais não trouxe benefícios significativos para este conjunto de dados em particular.

Gráfico 6: Curva de Aprendizado: Acurácia do modelo (Modelo Adicional)



Fonte: Autor

Gráfico 7: Curva de Aprendizado: Erro do modelo (Modelo Adicional)

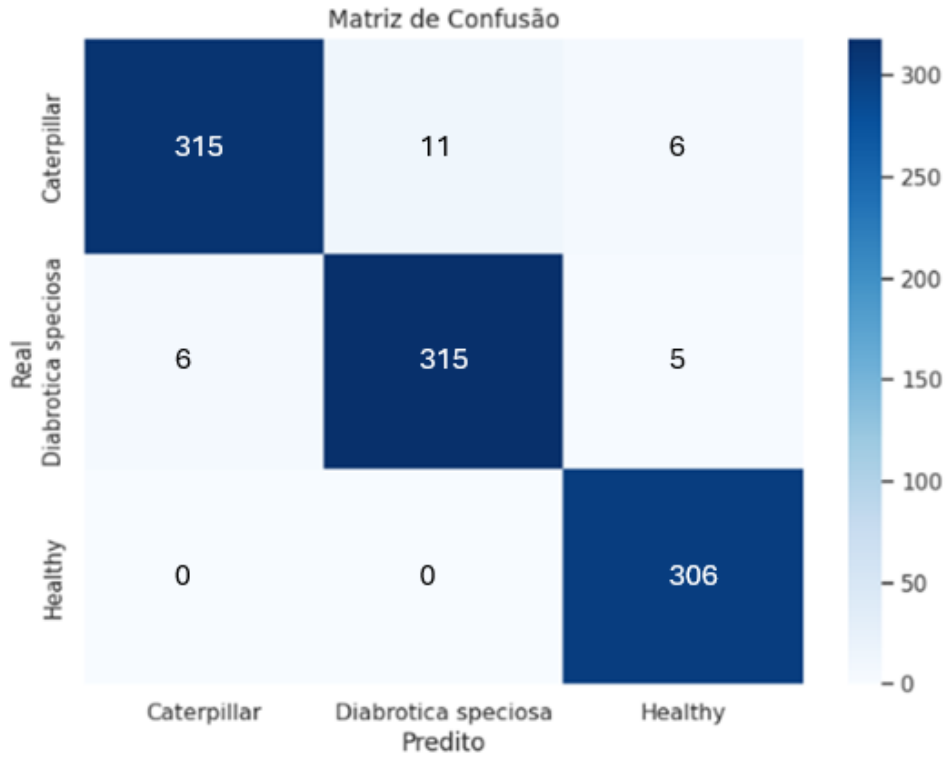


Fonte: Autor

A matriz de confusão (Gráfico 8) reflete um desempenho robusto, com a maioria das previsões feitas corretamente. No entanto, as instâncias incorretamente classificadas seguiram um padrão similar ao observado anteriormente. Notavelmente,

"Caterpillar" teve 17 classificações incorretas, predominantemente confundidas com "Diabrotica speciosa", indicando que o aumento das épocas e o ajuste da taxa de aprendizado não resolveram completamente a sobreposição nas características dessas classes.

Gráfico 8: Matriz de Confusão (Modelo Adicional)



Fonte: Autor

5.5 Considerações finais

Os resultados apresentados nesta seção evidenciam a importância das transformações de dados e dos ajustes finos nos parâmetros dos otimizadores para o desempenho das redes neurais convolucionais. Adicionalmente, o aumento do número de épocas e o uso do callback "ReduceLROnPlateau" demandaram maior esforço computacional, mas não resultaram em melhorias significativas nas métricas de desempenho.

A abordagem metodológica detalhada e as análises compreensivas realizadas nesses experimentos permitiram identificar a configuração de modelo que oferece a melhor performance, criando uma base sólida para interpretações futuras e possíveis aplicações práticas.

6. CONCLUSÃO

Neste projeto, foram exploradas diversas arquiteturas de redes neurais convolucionais (CNNs) para a tarefa de classificação de imagens em plantações de soja, com o objetivo de detectar doenças nas plantas. Os resultados indicam que as redes CNNs demonstraram uma eficácia notável, com todos os 49 modelos testados atingindo uma acurácia satisfatória, superior a 90%. O melhor modelo testado foi EfficientNET, utilizando todas as transformações de dados propostas nesse projeto, com uma acurácia de 97,40%. Esses resultados ressaltam o potencial da visão computacional como uma ferramenta eficaz para a detecção de doenças em culturas agrícolas, especialmente quando aplicada a este conjunto específico de dados.

Contudo, é importante destacar que cada projeto possui suas particularidades, e a eficácia de um modelo pode variar dependendo das condições em que as imagens são capturadas. Fatores como a resolução da câmera, o ângulo de captura das imagens, e as variâncias ambientais podem impactar significativamente o desempenho do modelo. Portanto, os modelos desenvolvidos neste estudo não são necessariamente generalizáveis para todas as situações de classificação em plantações de soja. Uma adaptação cuidadosa do modelo é necessária para atender às especificidades de diferentes contextos.

Adicionalmente, embora os experimentos realizados tenham se mostrado promissores no âmbito acadêmico, a implementação em escala comercial requer a criação e manutenção contínua de uma base de dados robusta e específica para a empresa em questão. Para empresas de grande porte, a adoção de uma abordagem automatizada para a detecção de doenças pode ser vantajosa, permitindo a aplicação direcionada de pesticidas de acordo com a praga identificada. Esta ideia pode servir de base para projetos futuros que busquem implementar e testar a tecnologia em campo, avaliando sua eficácia em condições reais de cultivo.

REFERÊNCIAS

Centro de Estudos Avançados em Economia Aplicada (Cepea) e Associação Brasileira das Indústrias de Óleos Vegetais (Abiove). Cadeia da soja e do biodiesel: PIB, empregos e comércio exterior – Primeiros Resultados e metodologia. 2023. Disponível em: <https://www.cepea.esalq.usp.br/br/pib-da-cadeia-de-soja.aspx>

AHMED, S. et al. **A survey on efficient convolutional neural networks and hardware acceleration**. IEEE Transactions on Artificial Intelligence, 2020.

AMAZON WEB SERVICES**. O que é aumento de dados?** Disponível em: https://aws.amazon.com/pt/what-is/data-augmentation/. Acesso em: 20 maio 2024.

BALAJI, Sai. **Binary Image classifier CNN using TensorFlow**. 2020. Disponível em: https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697. Acesso em: 28 maio 2024.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. Springer, 2006.

BOESCH, C. et al. **Neural Network Architectures**. International Journal of Computer Vision, 2024.

BOESCH, Gaudenz. **VGG Very Deep Convolutional Networks (VGGNet) – What Kou Need to Know.** Disponível em: https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/. Acesso em: 08 mar. 2024.

Bottou, L. **Large-Scale Machine Learning with Stochastic Gradient Descent**, 2010.

C, BALA PRIYA *et al*. **Softmax Activation Function: Everything You Need to Know**. 2023. Disponível em: https://www.pinecone.io/learn/softmax-activation/. Acesso em: 28 maio 2024.

DAMLA ALTUNAY. **The basics of image processing and OpenCV**. 2019. Disponível em: https://developer.ibm.com/articles/learn-the-basics-of-computer-vision-and-object-detection/. Acesso em: 02 fev. 2024.

DATA SCIENCE ACADEMY. **Deep Learning Book**. 2022. Disponível em: https://www.deeplearningbook.com.br/. Acesso em: 02 fev. 2024.

FAN, S.; **Understanding Deep Residual Networks**, 2018.

FOOD AND AGRICULTURE ORGANIZATION OF THE UNITED NATIONS (FAO). **International Year of Plant Health, 2020**. Rome: FAO, 2020. 28 p. Disponível em: https://www.fao.org/3/ca5188en/ca5188en.pdf. Acesso em: 03 abr. 2024.

GHOLAMALINEZHAD, H., & KHOSRAVI, H. **Pooling methods in deep neural networks, a review,** 2020.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing. 4. ed**. New York, NY, USA: Pearson, 2018.

GOODFELLOW, I., BENGIO, Y., & COURVILLE, A. **Deep Learning**. MIT Press. 2016

GOOGLE CLOUD. **Advanced Guide to Inception v3**. Disponível em: https://cloud.google.com/tpu/docs/inception-v3-advanced. Acesso em: 20 maio 2024.

GUO, Y. et al. **A review of methods and applications in image analysis for agriculture**. Computers and Electronics in Agriculture, v. 143, p. 276-295, 2017.

HE, K. et al. **Deep residual learning for image recognition**. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, p. 770-778, 2016.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian**. Deep Residual Learning for Image Recognition**. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

Hinton, G.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. **Improving neural networks by preventing co-adaptation of feature detectors**, 2012.

HOFFMANN-CAMPO, C.B. MOSCARDI, F. CORREA-FERREIRA, B.S. OLIVEIRA, L. J. SOSA-GOMEZ, D.R. PANIZZI, A. R. CORSO, I. C. GAZZONI, D. L. OLIVEIRA, E. B. de. **Pragas da soja no Brasil e seu manejo integrado**. Londrina: Embrapa Soja, 2000.

Ioffe, S.; Szegedy, C. **Batch normalization: Accelerating deep network training by reducing internal covariate shift**, 2015.

Kingma, D. P.; Ba, J. **Adam: A method for stochastic optimization,** 2015.

KOPPERT BIOLOGICAL SYSTEMS**. Anticarsia gemmatalis**, 2024. Disponível em: https://www.koppert.com.br/produtos/produtos-para-controle-de-pragas/. Acesso em: 26 mai. 2024.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. **Gradient-based learning applied to document recognition**, 1998.

LECUN, YANN, YOSHUA BENGIO, E GEOFFREY E. HINTON. **"Deep learning." Nature, vol. 521,** 2015.

LEDERER, J. **Activation functions in artificial neural networks: A systematic overview.** 2021.

LIU, W. et al. SSD: **Single Shot Multibox Detector**. European Conference on Computer Vision, Springer, Cham, 2016. p. 21-37.

LIU, X. et al**. Deep learning for visual understanding: Review and future directions**. Neurocomputing, 2020.

Loshchilov, I.; Hutter, F. **Decoupled weight decay regularization**, 2019.

MUKHERJEE, S. **The Annotated ResNet-50**. 2022.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. MIT Press, 2012.

PAN, S. J., & YANG, Q. **A survey on transfer learning**. IEEE Transactions on Knowledge and Data Engineering, 2010

PRACA, Lílian Botelho; SILVA NETO, Sebastião Pedro da; MONNERAT, Rose Gomes. **Biologia, amostragem e métodos de controle de Anticarsia gemmatalis Hübner, 1818 (Lepidoptera: Noctuidae)**. In: Embrapa. Congresso Brasileiro de Entomologia, Brasília, DF, 2024. 10 vol., p. 1-10. Disponível em: https://www.todamateria.com.br/artigo-definido-e-indefinido/. Acesso em: 26 mai. 2024.

REDMON, J.; FARHADI, A. YOLOv3: **An incremental improvement**. arXiv preprint arXiv:1804.02767, 2018.

ROSA, A. P. S. A.; TRECHA, C. O.; MEDINA, L. B. **Bioecologia de Diabrotica speciosa (Germar, 1824) (Coleoptera: Chrysomelidae) Visando Fornecer Subsídios para Estudos de Criação em Dieta Artificial**. Brasília: Embrapa Soja, 2013.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. **Learning representations by back-propagating errors**. Nature, 1986

Santurkar, S.; Tsipras, D.; Ilyas, A.; Madry, A. **How does batch normalization help optimization?**, 2018.

SIMONYAN, K.; ZISSERMAN, A. **Very deep convolutional networks for large-scale image recognition**. In: International Conference on Computer Vision (ICCV), 2014.

SIMONYAN, K.; ZISSERMAN, A**. Very Deep Convolutional Networks for Large-Scale Image Recognition**, 2014.

SINGH, Brajesh K. et al. **Climate change impacts on plant pathogens, food security and paths forward**. Nature Reviews Microbiology, v. 21, n. 10, p. 640-656, 2023.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. **Dropout: A simple way to prevent neural networks from overfitting.** The Journal of Machine Learning Research, 2014.

SZEGEDY, C. et al. **Going deeper with convolutions**. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

LI, X.; ZHANG, D.; HU, Y.; SUN, Y.; DAI, L. **Application of Deep Learning in Agricultural Informatization: Detection of Soybean Leaf Diseases Based on Deep Learning.** Journal of Sensors, 2020.

KAMILARIS, A.; PRENAFETA-BOLDÚ, F. X. **Deep learning in agriculture: A survey. Computers and Electronics in Agriculture**, 2018.

SZEGEDY, C., VANHOUCKE, V., IOFFE, S., SHLENS, J., & WOJNA, Z. **Rethinking the Inception Architecture for Computer Vision**. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

TAN, M.; LE, Q. V. **EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks**. In: Proceedings of the 36th International Conference on Machine Learning (ICML), 2019.