UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

MARCELO FERREIRA SALVADORI - 00278026

APLICAÇÃO DE *MACHINE LEARNING* PARA DETECÇÃO DE PRAGAS EM PLANTAS

Porto Alegre
2024

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

MARCELO FERREIRA SALVADORI - 00278026

APLICAÇÃO DE *MACHINE LEARNING* PARA DETECÇÃO DE PRAGAS EM PLANTAS

Trabalho de Conclusão de Curso (TCC-CCA) apresentado à COMGRAD-CCA da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de *Bacharel em Eng. de Controle e Automação* .

Orientador:
Prof. Dr. Heraldo José de Amorim

Co-orientador(a):
Yachel Rogério Mileski

Porto Alegre
2024

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

MARCELO FERREIRA SALVADORI - 00278026

APLICAÇÃO DE *MACHINE LEARNING* PARA DETECÇÃO DE PRAGAS EM PLANTAS

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Disciplina de TCC do curso *Eng. de Controle e Automação* e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Heraldo José de Amorim, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul - Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. Heraldo José de Amorim, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul - Porto Alegre, Brasil

Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universität Paderborn - Paderborn, Alemanha

Prof. Dr. Herbert Martin Gomes, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

_____
Alceu Heinke Frigeri
Coordenador de Curso
Eng. de Controle e Automação

Porto Alegre, Fevereiro 2024

DEDICATÓRIA

Dedico este trabalho aos meus pais, João e Carmen, pela dedicação e apoio a minha escolha de carreira, aos meus amigos e colegas por sempre me ajudarem durante o período de graduação e ao meu orientador Heraldo e coorientador Yachel, por terem auxiliado na elaboração desse trabalho.

AGRADECIMENTOS

À Universidade Federal do Rio Grande do Sul, UFRGS, pela oportunidade de realização de estudos.

RESUMO

Métodos de aprendizado de máquina são cada vez mais utilizados para soluções tecnológicas e inventivas para problemas enfrentados na indústria moderna. Combinados com visão computacional, esses métodos melhoram a capacidade dos sistemas de inspeção em diferentes campos. Os sistemas de detecção automática podem ser utilizados para identificar pragas e doenças nas culturas, permitindo a detecção precoce e a rápida implementação de técnicas de contingência, tais como medicamentos e pesticidas. Nestes casos, a correta identificação da planta doente permite o tratamento individualizado, reduzindo significativamente o uso de agrotóxicos a uma região limitada da cultura. A utilização de redes neurais permite o desenvolvimento de modelos robustos e escaláveis, que podem ser aplicados em diversos contextos. Este trabalho propõe um modelo de aprendizagem profunda para identificar duas pragas (lagarta e *Diabrotica speciosa*) em folhas de soja. Os métodos foram avaliados utilizando bancos de dados com imagens reais, e testes de modelos foram realizados para validar e melhorar o desempenho. Os modelos desenvolvidos obtiveram precisões de teste de até 98,55%.

Palavras-chave: Processamento de Imagens, **Redes Neurais**, **Detecção de Pragas**, **Indústria Moderna**.

ABSTRACT

Machine learning methods are increasingly used for technological and inventive solutions to problems experienced in modern industry. Combined with computer vision, these methods improve inspection systems capabilities in different fields. Automatic detection systems can be used for identifying crop pests and diseases, allowing early detection and fast implementation of contingency techniques such as medicine and pesticides. In those cases, the correct identification of the ill plant allows individual treatment, significantly reducing the use of pesticides to a limited region of the crop. Using neural networks enables the development of robust and scalable models, which can be applied in diverse contexts. This work proposes a deep learning model to identify two pests (caterpillar and *Diabrotica speciosa*) in soybean leaves. The methods were evaluated using databases with real images, and model tests were carried out to validate and improve the performance. The developed models obtained test accuracies of up to 98,55%.

Keywords: Image Processing, **Neural Networks**, **Pest Detection**, **Modern Industry**.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS

CNN Rede Neural Convolucional (do inglês, *Convolutional Neural Network*)

SUMÁRIO

1 INTRODUÇÃO

Durante toda a história da civilização humana, o aprimoramento das técnicas de agricultura sempre se demonstraram de extrema importância para melhorar a qualidade de vida da população. Um crescimento no setor de agricultura moderno, devido à sua importância na economia global, é capaz de aumentar a renda dos mais pobres e alimentar um número maior de pessoas. Entretanto, um dos principais fatores que ameaçam esse setor são os aparecimentos de pragas e doenças em plantas.

Doenças em plantas são causadas por organismos patogênicos, como fungos, bactérias, vírus e nematoides, que as atacam e podem prejudicar seu desenvolvimento. Para fins de diagnóstico, é possível distinguir uma planta doente de uma planta não doente a partir da cor de suas folhas, de sua taxa de crescimento, da existência de lesões e manchas na folha e a redução da produtividade de um cultivo (ISLEIB, 2012). Portanto, as plantas, assim como os humanos, também são susceptíveis a infecções e doenças, de modo que o impacto das pragas que as acometem pode ter consequências significativas na economia de uma região. Desse modo, o tratamento e prevenção dessas pragas é de fundamental importância.

A mitigação adequada de doenças em plantas e a implementação de práticas sustentáveis de controle de pragas são essenciais para minimizar o impacto negativo na economia e no meio ambiente. Uma revolução tecnológica na agricultura é necessária para resolver os problemas modernos desse setor de maneira eficiente e permanente, a fim de atingir um controle sustentável dos cultivos, evitando o uso indiscriminado de pesticidas, fungicidas e outros tipos de produtos químicos, que podem permanecer anos no solo e na água e que podem vir a ser prejudiciais para a saúde humana, principalmente para os trabalhadores que tem contato direto com estas substâncias (SAÚDE - OMS, 2020). Para isso, o monitoramento e a detecção prematura são de extrema importância pois permitem uma implementação rápida das técnicas de controle de doenças, reduzindo a velocidade de alastramento e a severidade.

As estratégias clássicas de detecção de pragas dependem exclusivamente de um observador humano, fazendo o reconhecimento da plantação e do terreno ou da colocação de armadilhas em áreas infestadas que são verificadas por operadores humanos (MICHELE PRETI, 2020). Essas técnicas tradicionais se comprovam pouco eficientes, muito custosas e estão propensas a imprecisão. Por sua vez, algoritmos de *deep learning* oferecem uma alternativa promissora para aprimorar sistemas baseados em visão computacional para monitoramento autônomo de doenças em plantas. Além disso, técnicas modernas de detecção utilizando drones são muito eficientes para rapidamente identificar doenças em plantas, pois eles conseguem percorrer grandes áreas em um período de tempo muito curto, reduzindo o trabalho manual necessário e aumentando a produtividade das plantações. Para isso, os drones extraem imagens em alta resolução e utilizam estratégias tradicionais de *machine learning* e *deep learning* para realizar o processamento das imagens. As características específicas das doenças podem ser extraídas de imagens de folhas usando

diferentes estratégias como a de detecção de manchas, diferenças de pigmentação ou formato da folha.

Neste trabalho serão aplicados métodos de visão computacional e aprendizado de máquina para desenvolver uma aplicação que consiga detectar doenças em plantas. O resultado é validado em um banco de dados de teste independente dos dados de treinamento da rede neural (dados de entrada) que foram utilizados para o treinamento. O objetivo fim dessa aplicação é a utilização para realizar a identificação precoce de doenças em plantas, reduzindo o custo de monitoramento e o impacto negativo em grandes plantações.

2 REVISÃO DA LITERATURA

Neste capítulo serão apresentados os principais conceitos necessários para a formulação deste trabalho e para a compreensão do mesmo pelo leitor. A Seção 2.1 explica a aplicação de transformações paramétricas no contexto de redes neurais. A Seção 2.2 explica os conceitos de operadores de vizinhança e filtragem linear, relacionados a operação de convolução, que é fundamental para o treinamento de redes neurais convolucionais. A Seção 2.3 explica os conceitos de aprendizado de máquina e *deep learning* (aprendizado profundo) e entra em detalhes acerca do funcionamento de uma rede neural e de um processo de treinamento iterativo. Por fim, a Seção 2.4 explica e explicita o uso de *data augmentation* e a sua importância para a melhoria do desempenho de redes neurais.

2.1 Transformações paramétricas

Transformações paramétricas são transformações matemáticas que aplicam uma deformação global a uma imagem, no qual o comportamento da transformação é controlado por um pequeno número de parâmetros ajustáveis (szeliski, 2022). As transformações paramétricas de coordenadas 2D em imagens são frequentemente utilizadas em aplicações de aprendizado de máquina e tem como objetivo modificar a posição, orientação, escala ou outros aspectos das imagens (szeliski, 2022). Além disso, são comumente usadas em aplicações de visão computacional para tarefas de aumento de dados e para correção de pequenas distorções presentes nas imagens. Os principais tipos de transformações paramétricas de coordenadas 2D incluem:

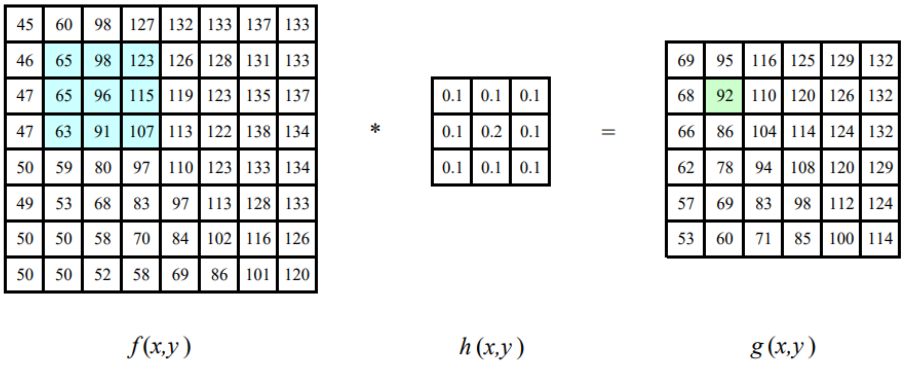
- **Translação**: Movimento de todos os pontos de uma imagem por uma quantidade fixa em direções horizontais ou verticais. A orientação original da imagem permanece inalterada;
- **Rotação**: Rotação de uma imagem em torno de um ponto de referência. As proporções originais da imagem permanecem inalteradas;
- **Mudança de escala**: Alteração da escala da imagem, aumentando ou diminuindo as suas dimensões;
- **Cisalhamento** Deslocamento paralelo dos pontos da imagem em uma direção específica, causando efeitos de inclinação ou distorção na imagem;
- **Transformações afins**: Transformações complexas que envolvem combinações de translação, rotação, mudança de escala e cisalhamento, preservando o paralelismo da imagem e os pontos colineares (pontos que pertencem a uma mesma reta).

2.2 Operadores de vizinhança e filtragem linear

Operadores de vizinhança, também conhecidos como operadores locais, são ferramentas utilizadas em processamento de imagens e processamento de sinais para manipular os valores de pixels em uma região localizada. Esses operadores têm como principal função extrair ou realçar informações importantes de um certo conjunto de dados de entrada (szeliski, 2022). No contexto de processamento de imagens, esses operadores podem ser usados para adicionar desfoque, nitidez de detalhes, acentuação de arestas ou remover ruídos de uma imagem. O principal tipo de operador de vizinhança utilizado em aplicações de visão computacional é o filtro linear, capaz de identificar características como bordas, texturas, e diversos outros padrões visuais (szeliski, 2022). Matematicamente, um filtro linear tem o seu o valor de saída de pixel (saída do filtro) como uma soma ponderada de valores de diversos pixels dentro de uma pequena vizinhança contida na imagem.

A Figura 1 representa simplificadamente como a transformação linear é aplicada a um conjunto de pixels em um determinada região de uma imagem. A região em azul da matriz à esquerda é convoluída com o filtro h para produzir a matriz à direita. Cada valor da matriz em azul é multiplicado pelo seu valor do filtro na posição correspondente e somado para se obter o valor em verde da matriz a direita. Portanto, os pixels azuis indicam a região fonte que foi aplicado o filtro e o pixel verde a região que contém o resultado da operação de filtragem. De maneira simplificada, a técnica usa uma coleção de valores de pixel nas proximidades de um determinado pixel para determinar seu valor de saída final. A saída final após a aplicação do filtro, os mapas de características, capturam padrões ou características locais, como bordas, texturas ou outras características relevantes dentro do campo receptivo do filtro. Portanto, os números de saída no mapa de características representam as ativações dos filtros aprendidos em diferentes locais espaciais das imagens. Altas ativações em determinadas regiões indicam a presença de padrões que os filtros foram projetados para detectar. A rede neural aprende a ajustar os pesos do filtro durante o treinamento para aprimorar sua capacidade de reconhecer recursos específicos. À medida que a CNN avança através de múltiplas camadas realizando essa operação, os recursos tornam-se mais abstratos e complexos, permitindo que a rede aprenda representações hierárquicas dos dados de entrada. Os operadores de vizinhança podem ser usados para filtrar imagens para adicionar desfoque suave ou aumentar a nitidez de detalhes (szeliski, 2022).

Figura 1: *Representação do processo de aplicação de um filtro linear a um conjunto de dados*



Fonte: (szeliski, 2022)

2.3 *Machine Learning* e *Deep Learning*

Machine Learning (aprendizado de máquina) é considerado um subconjunto da área de estudo de inteligência artificial que tem como objetivo desenvolver aplicações que utilizam algoritmos e modelos matemáticos que sejam capazes de automaticamente reconhecer padrões e potencialmente tomar decisões autônomas baseadas nos resultados obtidos (GOODFELLOW; BENGIO; COURVILLE, 2016). Um grande tópico de interesse desta área é o desenvolvimento de programas de computador que consigam automaticamente melhorar a performance esperada a partir de sua própria experiência adquirida em suas aplicações (MITCHELL, 1997). *Machine learning*, principalmente modelos de *Deep Learning*, que são modelos de aprendizado de máquina baseados no conceito de redes neurais, têm sido cruciais no desenvolvimento de algoritmos de visão computacional, pois permitem a criação de complexos sistemas que podem automaticamente extrair e analisar informações de imagens ou vídeos, o que revolucionou a área de reconhecimento de imagens e detecção de objetos.

2.3.1 Redes Neurais

Uma rede neural pode ser definida como um sistema composto de um grande número (geralmente milhares) de grafos computacionais interconectados entre si chamadas de *neurons* (neurônios) (MITCHELL, 1997). Um neurônio funciona recebendo informações de outros neurônios ou de dados pré selecionados e performa somas ponderadas de suas entradas. Ele geralmente possui um termo constante chamado de viés, que introduz independência com relação as entradas e auxilia no controle da faixa permitida de saída. A soma ponderada então é transformada por uma função de ativação, inserindo não linearidade no sistema, ajudando a identificar mais padrões complexos e aumentando o poder de abstração da aplicação. Matematicamente, um neurônio pode ser descrito pela seguinte equação (Szeliski; Richard, 2022):

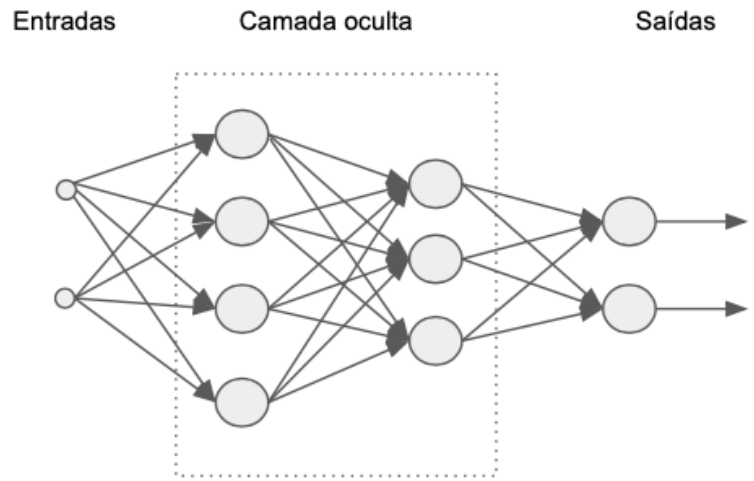
$$s_i = w_i^T x_i + b_i \tag{1}$$

seguido por uma função de ativação não linear:

$$y_i = h(s_i) \tag{2}$$

onde x_i são as entradas da enésima unidade, w_i e b_i são os parâmetros treináveis peso e viés, s_i é a saída da soma linear ponderada e y_i é a saída final após s_i ser transformada pela função de ativação h. Redes neurais são usualmente organizadas e definidas em camadas, cada uma contendo diversos neurônios que obedecem esse mesmo modelo matemático, distinguindo-se entre si através dos seus parâmetros treináveis(SZELISKI, 2022). A primeira camada é a de entrada, que recebe os dados que serão utilizados pela rede. No contexto de identificação de padrões em imagens, cada neurônio nessa camada inicial descreve um pixel ou alguma característica relevante da imagem. As camadas intermediárias, também chamadas de ocultas, recebem informação das camadas anteriores e realizam cálculos, passando adiante informação para a camada seguinte, e assim subsequentemente, até atingir a camada final. A camada de saída é a ultima camada e representa a saída de toda a rede neural. A Figura 2 mostra um diagrama que demonstra a organização de uma rede neural que possui essa configuração.

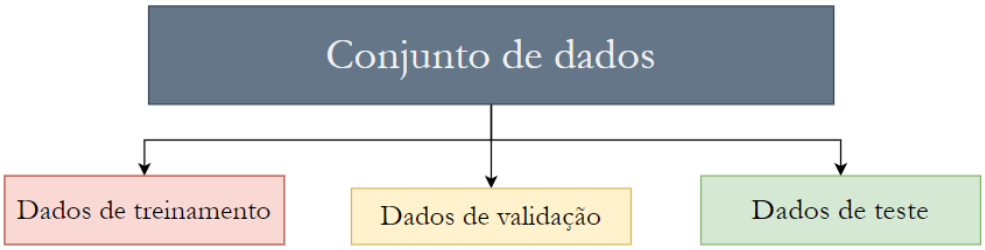
Figura 2: *Diagrama de uma rede neural básica*



Fonte: (VINICIUS, 2021)

Para evitar que a rede neural desenvolva tendências indesejadas, os dados que serão usados para o treinamento são divididos em três categorias: dados de treinamento, que compõem a maior das categorias e são utilizados para o treinamento da rede neural, auxiliando a atualizar os pesos e víeis da rede; os dados de validação, que auxiliam a harmonizar e melhorar os parâmetros e permitem a comparação de diferentes modelos; e os dados de teste, que são utilizados para obter uma avaliação final do desempenho da rede escolhida. Essa segmentação do banco de dados facilita a validação do desempenho da rede neural, pois é importante que os dados de teste não tenham sido utilizados para o treinamento da rede neural, a fim de evitar vícios no modelo obtido (SZELISKI, 2022). A Figura 3 ilustra essa divisão.

Figura 3: *Divisão básica dos dados de entrada em subconjuntos*



Fonte: Autor

2.3.1.1 Redes Neurais Convolucionais

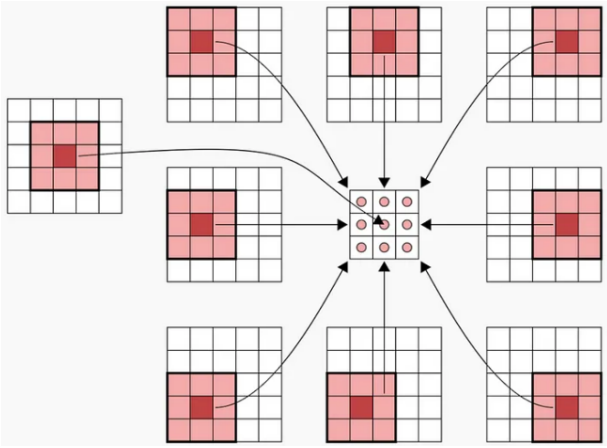
No diagrama mostrado na seção anterior (Figura 2), todas as unidades (neurônios) de uma camada são conetados a todas as unidades da camada anterior. Redes neurais convolucionais, por sua vez, organizam cada camada em unidades que são chamadas de *Feature Maps* (mapas de recursos), que podem ser representados por diversos canais paralelos dentro de uma camada da rede neural. Em uma camada de convolução, as somas ponderadas são realizadas apenas dentro de uma pequena janela local e os pesos

são idênticos para todos os pixels contidos nessa região. Nesse tipo de camada ocorre a aplicação de filtros (também chamados de *kernels*) nos dados de entrada. Cada filtro é essencialmente uma matriz de pesos que é deslocada por toda a imagem calculando ponto a ponto um valor, que é definido como o somatório dos produtos dos pixels originais da imagem pelo valor do elemento do filtro na mesma posição correspondente(SZELISKI, 2022).

A camada de convolução tipicamente possui interações esparsas, o que significa que os dados de entrada são dimensionalmente maiores que os filtros aplicados. Isso faz com que a rede neural tenha mais facilidade em extrair características específicas e identificar padrões simples como bordas, textura e cores em imagens, pois processa pequenas regiões da imagem. Os pixels próximos geralmente contêm informações visuais que estão relacionadas e que contribuem para a identificação de objetos. Portanto, esse tipo de rede se apresenta especialmente eficiente para o processamento de dados que estão dispostos em estrutura de grade, como uma imagem, que também pode ser interpretada como uma matriz de duas dimensões de pixels(SZELISKI, 2022).

A camada final da rede neural convolucional (CNN) é responsável por compor a saída final da rede utilizando os dados obtidos das diversas camadas de convolução anteriores. Para aplicações de classificação de imagens, essa camada final tem estrutura diferente da camada de convolução e retorna ao modelo básico de uma camada completamente conectada, como demonstrado na Figura 2. A Figura 4 ilustra de maneira simples e visual a operação de convolução. É possível observar que há uma redução na dimensão da imagem resultante da operação, pois a imagem original tem dimensões 5x5 e imagem final possui dimensões 3x3.

Figura 4: *Operação de convolução em uma matriz de pixels 5x5*



Fonte: (SZELISKI, 2022)

2.3.1.2 Pooling

Pooling é uma operação que tem como objetivo reduzir a dimensão espacial das saídas dos filtros da camada de convolução (GOODFELLOW; BENGIO; COURVILLE, 2016). Uma função de *pooling* transforma os dados de saída em um determinado local da camada de convolução em uma estatística resumida das saídas próximas. Essa função opera em cada mapa de recursos separadamente para criar um novo conjunto reduzido.

Uma operação matemática análoga a um filtro é selecionada para ser aplicada aos mapas de recursos. O tamanho da operação de *pooling* (ou filtro) é menor que o tamanho do mapa de recursos; geralmente, é 2 × 2 pixels aplicados com um passo de 2 pixels (que indica por quantos pixels o filtro será movido a cada iteração). No exemplo de 2x2 pixels com passo de 2, a camada será reduzida por um fator de 2 (cada dimensão da matriz de pixels é dividida pela metade), reduzindo o número de pixels em cada mapa de recursos para um quarto do tamanho.

Os dois tipos mais comuns de *pooling* são o *max pooling*, com a obtenção do valor máximo de saída do mapa de recursos após a aplicação do filtro e o *average pooling*, que extrai a média dos valores obtidos(SZELISKI, 2022). A principal funcionalidade dessas operações é permitir que a rede neural seja menos suscetível a variações dos dados de entrada. Invariância à translação local de uma imagem é especialmente útil quando é muito mais importante identificar se algum padrão está presente na imagem do que onde exatamente ele se encontra. Portanto, essa característica vai ao encontro do estudo sobre identificação de doenças em plantas através de visão computacional, visto que a localidade específica que se encontram os padrões de cor e formato típicos de uma planta doente não são de extrema relevância para a análise realizada pela rede neural, mas sim se estão presentes ou não na imagem (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.3.1.3 Treinamento de redes neurais

De acordo com (GOODFELLOW; BENGIO; COURVILLE, 2016) e (SZELISKI, 2022), a criação e treinamento de uma rede neural pode ser dividida, de maneira geral, nas seguintes etapas:

1. Preparação dos dados e pré-processamento das imagens: essa etapa envolve a coleta do banco de dados que será utilizado para o treinamento do modelo. As imagens são rotuladas de acordo com a saída esperada para auxiliar o processo de treinamento supervisionado e técnicas de pré processamento são aplicadas nas imagens, como redimensionamento, distorções, normalizações e aumento de dados. Como explicado em mais detalhes na seção sobre redes neurais, nesta etapa os dados de treinamento são divididos em três grandes categorias: dados de treinamento, dados de validação e dados de teste.

2. Design da arquitetura da rede neural: essa etapa envolve a escolha da arquitetura da rede neural que será implementada. A rede neural convolucional, escolhida para a implementação do modelo deste trabalho, é composta pelos seguintes blocos fundamentais: camadas de convolução, que são responsáveis pela aplicação dos filtros e extração de recursos; funções de ativação, que introduzem aspectos de não-linearidade ao modelo; camadas de *pooling*, que reduzem as dimensões espaciais dos dados que provém das camadas de convolução; e camadas totalmente conectadas, que realizam a classificação final do modelo.

3. Parametrização do modelo: consiste na definição da função de perda, que mede a diferença entre os valores previstos (a partir dos rótulos definidos) e os valores obtidos após o treinamento; escolha do otimizador, responsável pela atualização dos pesos do modelo com base nos gradientes calculados a partir da função de perda; escolha de métricas para a avaliação do desempenho do modelo; escolha da decadência de

peso, que adiciona um termo à função de perda que é proporcional à soma dos pesos ao quadrado, penalizando e reduzindo a ocorrência de grandes pesos no modelo; e escolha da taxa de aprendizado, que determina o tamanho do passo dado durante cada iteração do algoritmo de otimização e por conseguinte controla a taxa que a rede neural atualiza as informações que aprendeu.

4. Treinamento da rede neural: essa etapa envolve introduzir os dados de entrada no modelo. Internamente, o algoritmo irá retro propagar a perda para calcular o gradiente para cada parâmetro e utilizar o otimizador para atualizar os pesos da rede neural a fim de minimizar as perdas. Esse processo é repetido iterativamente utilizando-se o conceito de *Epochs*. Uma *Epoch* (época) é quando todos os dados de entrada de treinamento são usados para treinar o modelo. Em outras palavras, cada vez que o conjunto de dados de entrada em sua totalidade é submetido ao algoritmo, ele completa uma época. Portanto, uma época no contexto de aprendizado de máquina pode ser definida como um hiperparâmetro, pois controla o processo de treinamento e ajuda a determinar a performance do modelo. Os dados de entrada que são usados para o treinamento da rede neural geralmente são divididos em pequenos subconjuntos e introduzidos no algoritmo a cada época gradativamente, devido às limitações de armazenamento que um sistema operacional de um computador possui. Esses subconjuntos são chamadas de lotes e podem ter tamanhos entre um (todos os dados são introduzidos simultaneamente no modelo) e igual ao número de amostras que compõem os dados de entrada. Por exemplo, se o conjunto de dados de entrada é composto por 100 imagens, elas podem ser introduzidas uma a uma dentro de uma época (uma iteração) ou 1000 imagens simultaneamente. Portanto, o tamanho do lote é um hiperparâmetro que define o número de amostras que serão introduzidas em um determinado modelo antes que o mesmo atualize os seus parâmetros. Assim, facilita-se a inserção desses dados no modelo de aprendizado de máquina.

5. Avaliação do desempenho do modelo: Para minimizar o erro de um modelo ao máximo, é necessário um grande número de épocas e na maioria dos casos complexos de identificação de imagens, os algoritmos de aprendizado levam centenas ou milhares de épocas para minimizar consideravelmente o erro. Uma estratégia para acompanhar o desempenho do modelo é traçar uma curva de aprendizado com os dados de entrada, sendo um eixo do gráfico o número de épocas (tempo) e o outro eixo a precisão obtida. Esse gráfico pode fornecer informações relevantes sobre o modelo a fim de fazer correções para treinamentos futuros. Após o treinamento e obtenção do modelo, se necessário, poderá ser realizado uma última etapa para ajuste e atualização dos hiperparâmetros a fim de aumentar o desempenho do sistema.

2.3.1.4 Treinamento iterativo e atualização dos hiperparâmetros

O objetivo de treinar uma CNN ao longo do tempo é permitir que o modelo aprenda e extraia representações significativas dos dados de entrada, permitindo-lhe, em última análise, executar uma tarefa específica com precisão. O objetivo é sempre buscar o melhor desempenho possível, considerando as limitações impostas à tarefa em questão. Estas limitações podem se manifestar de diversas formas, seja de natureza técnica, como o tamanho reduzido ou a qualidade inferior do conjunto de dados de entrada ("dataset"), ou devido à complexidade e variação elevada entre as imagens utilizadas no processo de

treinamento. Além disso, é crucial antecipar obstáculos desvinculados do processo técnico de treinamento, como o tempo alocado para o projeto e a obtenção do modelo, bem como a disponibilidade de desempenho computacional. A abordagem de múltiplas sessões de treinamento permite tratar essas limitações de maneira iterativa, ajustando a estratégia conforme necessário para otimizar os resultados diante das condições específicas da tarefa em questão. Dito isso, é necessário determinar como medir o desempenho do modelo para que essas correções de percurso possam ser realizadas.

O desempenho de uma rede neural não se resume apenas à medição de parâmetros tradicionais, como precisão e erro do modelo, obtidos durante o treinamento. É importante compreender o objetivo da tarefa em execução e a saída esperada. Características como robustez do modelo, capacidade de generalização e, principalmente, a habilidade de adaptação a mudanças nos dados, tornam-se fundamentais em tarefas como a classificação binária de dados, exemplificada pela identificação de pragas em plantas. Nesse contexto, é desejável que um modelo possa enfrentar diversas situações, garantindo um desempenho consistente. Em outras palavras, é imperativo evitar o sobreajuste do modelo, uma questão recorrente no aprendizado de máquina.

O sobreajuste ocorre quando um modelo aprende demasiadamente os dados de treinamento, capturando ruídos e flutuações aleatórias presentes no conjunto de treinamento, em vez de compreender os padrões subjacentes à categoria para a qual está sendo treinado. Como resultado desse fenômeno, um modelo sobreajustado apresenta um desempenho notável nos dados de treinamento, porém falha ao lidar com dados novos e não observados, pois memoriza os exemplos de treinamento em vez de generalizar a partir deles. Para evitar este fenômeno, é essencial adotar estratégias que promovam a generalização do modelo e garantam sua capacidade de lidar eficazmente com dados inéditos.

A principal maneira de alterar o comportamento do modelo e evitar características indesejadas, atingido-se um desempenho ideal, é fazendo o ajuste fino. Hiperparâmetros, como explicado de maneira breve na seção sobre treinamento de redes neurais, são configurações que não são aprendidas durante o processo de treinamento, mas devem ser especificadas antes do início do mesmo e influenciam radicalmente o comportamento da rede neural. O ajuste fino envolve ajustar esses hiperparâmetros para encontrar a melhor configuração para uma tarefa específica. Abaixo estão listados os principais hiperparâmetros que foram ajustados durante as diversas iterações de treinamento a fim de melhorar o desempenho final da rede neural e uma breve explicação de sua importância no contexto de aprendizado de máquina e na tarefa de identificação de pragas em plantas:

- **Tamanho do Lote e Número de Épocas**: O tamanho do lote, na dinâmica do treinamento de redes neurais, é definido como o número de instâncias de treinamento utilizadas em cada iteração durante o processo de treinamento. Nesse contexto, o conjunto de dados integral é subdividido em lotes, e o modelo ajusta seus pesos após o processamento de cada lote. Optar por um tamanho de lote reduzido, significativamente inferior ao número total de amostras, proporciona diversas vantagens como economia de memória computacional. Ao empregar um tamanho de lote menor, o treinamento da rede neural é conduzido com uma quantidade reduzida de amostras em cada iteração (ou época), o que reduz a chance de ocorrer o esgotamento da memória da placa de vídeo, especialmente quando se trata do armazenamento de imagens. Além disso, o uso de tamanhos de lote menores implica uma atualização mais frequente dos parâmetros do modelo durante cada iteração de treinamento. Essa taxa de atualização alta acelera o processo de aprendizagem, permitindo que o modelo se adapte prontamente aos padrões dos dados. A introdução de uma aleato-

riedade mais pronunciada no processo de treinamento é outra consequência positiva de tamanhos de lote menores. O modelo experimenta uma variedade de minilotes pequenos em cada iteração, resultando em um conjunto mais amplo e diversificado de atualizações nos parâmetros. Essa aleatoriedade desempenha o papel crucial de regularização, agindo como uma defesa contra o sobreajuste do modelo. O resultado é uma melhoria na generalização do modelo e, possivelmente, uma aceleração do processo de convergência do mesmo (RADHAKRISHNAN, 2017))

- **Taxa de aprendizado e gradiente de perda**: O gradiente de perda é um algoritmo de otimização amplamente utilizado no treinamento de redes neurais convolucionais. A ideia básica por trás do gradiente descendente é atualizar iterativamente os parâmetros do modelo (pesos e desvios) na direção que minimiza uma função de custo ou perda. O objetivo final deste método é identificar o conjunto ideal de parâmetros do modelo que resultam no valor mais baixo possível da função de perda, ou seja, identificar o mínimo (ou um valor bem próximo do mínimo). A taxa de aprendizado é um hiperparâmetro importante que controla o quanto é ajustado os pesos da rede neural em relação ao gradiente de perda. Quanto menor o seu valor, mais lentamente o modelo desloca-se ao longo da encosta descendente do gradiente. Um valor pequeno da taxa de aprendizado aumenta as chances de se evitar a perda de um mínimo local, mas aumenta significativamente o tempo para atingi-lo. Por outro lado, uma taxa de aprendizado muito alta pode fazer com que o processo de otimização reajuste excessivamente os parâmetros, resultando na ultrapassagem do mínimo da função de perda, aumentando o tempo de conversão. Em alguns caso, é possível o modelo pular o mínimo da função de maneira a não achá-lo, fazendo com que o modelo não consiga convergir de maneira alguma ou direciona-se para uma solução considerada insuficiente. Portanto, a taxa de aprendizado deve ser cuidadosamente ajustada durante o processo de treinamento para garantir um bom progresso, evitando ultrapassagens e atualizações demasiadamente rápidas dos pesos da rede neural. Não existe um valor pré definido exato que deva ser utilizado para a taxa de aprendizado, podendo ela variar de acordo com o objetivo da tarefa que está sendo executada ou de acordo com as características específicas dos conjunto de dados e da arquitetura da rede neural. Na prática, porém, é recomendado começar o treinamento com uma taxa de aprendizagem maior (mas ainda pequena) e diminuí-la para que a otimização se estabeleça em um bom mínimo. (SZELISKI, 2022)

- **Momentum e Redução dos Pesos**: No contexto de algoritmos de otimização, momentum é um termo adicional que ajuda a acelerar o processo de otimização, introduzindo uma média móvel de gradientes anteriores. O método da descida gradiente regular tende a parar quando a solução atual atinge um plano na curva e somente computa os erros no minilote atual. Para evitar este problema, o momentum introduz um termo que é uma fração (comumente denotada como β) do vetor de atualização da iteração anterior. Isso faz com que o algoritmo possa aproveitar o impulso das atualizações de parâmetros anteriores, suavizando o processo de otimização e impedindo o processo de estagnação do método. Assim como todos os importantes parâmetros de inicialização para treinamento de redes neurais, o momentum precisa ser escolhido de maneira apropriada. Valores muitos altos de momentum (próximos de 1) podem levar a *overshooting*, onde o otimizador acumula muita velocidade e oscila ao redor do mínimo, potencialmente perdendo-o, similar ao processo que pode ocorrer ao se elevar demasiadamente a taxa de aprendizado.

Outro risco associado a altos valores deste parâmetro é a dificuldade de convergência do processo de otimização, devido ao momento acumulado movimente o otimizador rapidamente através do espaço de parâmetros. De maneira análoga, valores muito baixos de momento aumentam a probabilidade de o otimizador ficar preso nos mínimos locais devido a falta de inércia ou de tornar o processo de convergência muito lento, exigindo mais iterações para atingir o mínimo. Outro parâmetro, conhecido como redução de peso, é responsável pela diminuição dos valores dos pesos por algum pequeno fator durante cada iteração. Isto é equivalente a incluir um termo de penalidade correspondente à magnitude total dos pesos da rede. A motivação para esta abordagem é manter os valores dos pesos pequenos, reduzindo a chance de ocorrer sobreajuste e aumentando a capacidade do modelo de identificar características complexas de um conjunto de dados. Valores de pesos muito altos podem indicar que o modelo está se ajustando muito com relação aos dados de treinamento, capturando ruídos ou valores discrepantes, e perdendo a sua capacidade de generalização e adaptação a estímulos diferentes (MITCHELL, 1997)

- **Parada antecipada**: A parada antecipada é um método que permite especificar um número arbitrário e grande de épocas de treinamento e interromper o treinamento quando o desempenho do modelo parar de melhorar no conjunto de dados de validação. O treinamento é interrompido quando a medida de desempenho escolhida parar de melhorar. É de praxe, uma vez interrompido o treinamento, o retorno de chamada indicar o número exato da época em que se obteve o melhor desempenho e salvar o modelo para utilizações futuras. Porém, não é recomendado parar o processo de treinamento ao primeiro sinal de ausência de melhoria, pois não é possível prever o comportamento do modelo durante o treinamento, podendo ele piorar um pouco na próxima iteração antes de obter uma melhora considerável nas seguintes iterações. Esse problema é evitado adicionando um atraso ao gatilho em termos do número de épocas nas quais não acontece melhoria, definido-se um argumento comumente chamado de paciência. Portanto, paciência refere-se ao número de épocas consecutivas sem melhoria no conjunto de validação antes que o processo de treinamento seja interrompido. A escolha deste parâmetro é um processo empírico e depende das características específicas do conjunto de dados utilizados e da complexidade do modelo. É importante notar que definir a paciência muito baixa pode resultar em parada prematura, enquanto defini-la muito alta pode atrasar a parada quando o modelo já atingiu um platô de performance, além de potencialmente diminuir a regularização do modelo a aumentar a chance de sobreajuste(MITCHELL, 1997).

2.4 *Data Augmentation*

Data Augmentation (aumento de dados) é um conjunto de técnicas utilizadas no campo de aprendizado de máquina e visão computacional que tem como objetivo aumentar a quantidade de dados de entrada disponíveis para o treinamento de uma determinada rede neural, transformando os dados existentes (SZELISKI, 2022). Os dados são modificados através da aplicação de transformações a fim de criar imagens ligeiramente modificadas, reduzindo assim a possibilidade de *overfitting* (sobreajuste) e aperfeiçoando o desempenho do modelo quando ele é submetido a dados novos. Essa técnica é especialmente útil quando se tem um escopo pequeno, limitado e pouco diverso de dados para o treinamento e deseja-se aumentar o número de imagens utilizadas. Alguns métodos básicos e comuns

de aumento de dados incluem espelhamento horizontal ou vertical, translação, alteração de coloração e brilho, rotação e distorção. Para aplicações de classificação de imagens em que a orientação e tamanho dos padrões prezados, as transformações de rotação e espelhamento geralmente produzem um aumento significativo no desempenho do modelo.

A Figura 5 apresenta um exemplo da aplicação de uma transformação de distorção em uma imagem, e a Figura 6 mostra um exemplo da aplicação de uma transformação de translação e redimensionamento de uma imagem. A principal característica desejável do modelo que é realçada ao se utilizar corretamente técnicas de aumento de dados é a robustez, visto que o mesmo irá utilizar um número maior de imagens e com diferentes características em relação ao *dataset* original, aumentando a generalização do modelo e tornando-o capaz de identificar padrões para diferentes variações dos dados de entrada. Porém, não há garantia completa de melhora de performance do modelo ao se introduzir técnicas de aumento de dados.

Figura 5: *Aumento de dados através da transformação de distorção*

Fonte: (SZELISKI, 2022)

Figura 6: *Aumento de dados através de transformações de translação e redimensionamento*

Fonte: (SZELISKI, 2022)

O principal desafio ao usar técnicas de *data augmentation* é garantir que os novos dados sejam relevantes para o objetivo final da aplicação e não introduzam imprecisões ao modelo. O acréscimo de dados irrelevantes, de baixa qualidade ou muito distorcidos com relação aos dados originais podem introduzir ruídos, um novo viés não desejado ou inconsistências no modelo, levando a piora do desempenho da rede neural. Portanto, é evidente que as técnicas que serão utilizadas devem ser escolhidas cuidadosamente e variam consideravelmente dependendo do domínio do problema e do tipo de dados que estão sendo modificados. Por exemplo, aplicar rotações a imagens que a orientação dos dados é uma característica importante representa um erro e pode causar distorções no desempenho do modelo. Para mitigar riscos é importante selecionar cuidadosamente as técnicas de aumento de dados que serão aplicadas, evitando transformações drásticas que possam distorcer demais as imagens e manter uma aplicação consistente para todas elas, em todas

as etapas processo de treinamento. O monitoramento e a validação também é crucial, visto que o objetivo é sempre a melhora do desempenho do modelo, e não o oposto.

3 MATERIAIS E MÉTODOS

Este capítulo descreve a metodologia aplicada para a detecção de pragas em plantas. A Seção 3.1 define o modelo da rede neural escolhida, a Seção 3.2 apresenta o banco de dados escolhido para a realização do treinamento da rede neural, a Seção 3.3 aborda a biblioteca de *deep learning* e a interface de programação que foram escolhidas para o desenvolvimento do código em Python e a Seção 3.4 descreve em detalhes o processo que foi utilizado para o treinamento e a validação do desempenho da rede neural.

3.1 Rede neural escolhida

Inicialmente, foi decidido utilizar um modelo pré-treinado de uma rede neural já consolidada, a ResNet-18. A ResNet-18 é uma rede neural convolucional com 18 camadas de profundidade, sendo 17 dessas camadas convolucionais e uma camada final totalmente conectada. Essa rede foi previamente treinada em um conjunto de dados e contém os pesos e vieses que representam os recursos adquiridos a partir dos conjunto de dados de entrada que foram utilizados para o treinamento. Os modelos pré-treinados servem como ponto de partida para outras tarefas, permitindo que o desenvolvedor ajuste o modelo em um conjunto de dados diferente ou em uma tarefa diferente sem treiná-lo do zero.

Os recursos aprendidos geralmente são transferíveis e podem ser utilizados para dados de entrada diferentes. Por exemplo, um modelo treinado em um grande conjunto de dados conterá recursos que são extremamente úteis para qualquer tipo de trabalho que envolva classificação de imagens, como a identificação de bordas ou linhas horizontais. Ao usar um modelo pré-treinado, o processo de aprendizado da rede neural é acelerado, ganhando-se, assim, desempenho e popuando tempo de execução da aplicação que está sendo desenvolvida. Como resultado desse treinamento prévio, a rede aprende a identificar representações complexas com grande quantidade de recursos para uma ampla gama de imagens.

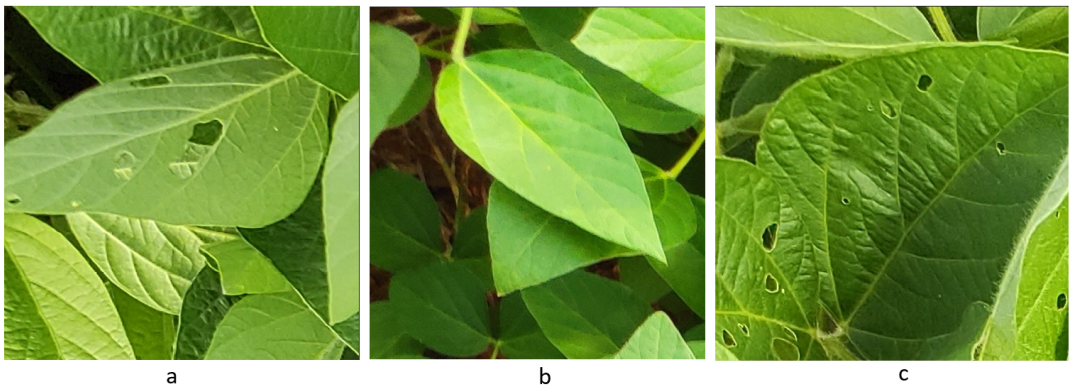
3.2 Banco de dados escolhido

Existem algumas considerações importantes ao se escolher um banco de dados para realizar o treinamento e validação da rede neural. Os modelos de *deep learning* geralmente exigem uma grande quantidade de dados, de preferência diversos entre si e com um volume equivalente de amostras de imagens por classe elevado para realizarem o aprendizado de padrões complexos com precisão. A fonte dos dados também é importante, visto que é imprescindível evitar características indesejadas nos dados de entrada, como ruídos ou imagens pré-processadas de maneira incorreta. Portanto, *datasets* que tenham sido criados

para o uso científico e especificamente para projetos de *deep learning* e que provenham de fontes confiáveis, como universidades, laboratórios ou institutos de pesquisa serão sempre preferíveis à fontes não oficiais.

Para o treinamento iterativo, foi escolhido um banco de dados de folhas de soja com imagens capturadas com smartfones e drones. As alturas de capturas foram variadas e a captura se deu em diferentes horários de diferentes dias (MIGNONI, 2021). O conjunto de dados é composto por um total de 6.410 imagens e está dividido em três categorias: 896 imagens de folhas de plantas saudáveis, 3309 imagens de folhas plantas afetadas por lagartas e 2205 imagens de folhas de plantas danificadas por *Diabrotica speciosa*, um besouro praga que afeta diversa cultura no Brasil. Na Figura 7 podemos ver os três tipos de folhas. Na imagem da esquerda, uma folha afetada por lagarta, na imagem centralizada uma folha saudável e na imagem da direita uma folha afeta por *Diabrotica speciosa*. É possível observar uma grande semelhança entre os dois tipos de folhas doentes, devido a ambas terem sido acometidas por pragas que furam as folhas para se alimentar. Essa similaridade e suas implicações será explorada com mais detalhes nos resultados deste trabalho, para os diferentes tipos de treinamento que serão realizados.

Figura 7: *Folhas de soja: a) Folha danificada por lagarta; b) Folha saudável; c) Folha danificada por Diabrotica speciosa*



Fonte: (MIGNONI, 2021)

3.3 Biblioteca de *DEEP LEARNING* e interface de programação

As bibliotecas de *deep learning* são ferramentas essenciais pois permitem a construção acessível de modelos para treinamento através de suas interfaces de prototipagem. Elas fornecem uma variedade de recursos e abstrações de funções matemáticas que facilitam o treinamento e implantação de diversas aplicações no campo de inteligência artificial, e a maioria dessas bibliotecas tem suporte para Python, linguagem de programação escolhida para a realização da aplicação. O Python é umas das linguagem de programação mais populares para a utilização de bibliotecas de aprendizado de máquina devido a sua sintaxe simples e legível aliado a sua vasta quantidade de *frameworks* específicos para aprendizado profundo. Python também possui uma comunidade de desenvolvedores e uma base de usuários muito ampla e por conseguinte, possui abundância de recursos e documentações de suporte para a resolução de problemas e implementação de diversas aplicações. Duas

das bibliotecas de código aberto mais utilizadas hoje pelo usuário comum para aplicações de aprendizado de máquina são PyTorch e TensorFlow.

Apesar das duas bibliotecas mencionadas possuírem ferramentas e suporte necessários para o desenvolvimento da aplicação neste trabalho se optou pela a biblioteca PyTorch, cuja estrutura possui prototipagem e treinamento mais veloz que TensorFlow e se destaca por integrar uma interface de programação mais acessível ao usuário. Para o desenvolvimento e edição de código para a aplicação foi escolhido o Visual Studio Code, o editor de código gratuito e de código aberto desenvolvido pela empresa Microsoft, destacando-se por sua versatilidade e acessibilidade, pois é de fácil acesso e uso pois possui amplo suporte para Python e PyTorch.

3.4 Validação experimental

Esta seção descreve em detalhes a abordagem metodológica que foi adotada para o treinamento da rede neural (atualização dos pesos) a partir do modelo pré treinado escolhido, apresentado na Seção 3.1. O conjunto de dados foi dividido de acordo com a seguinte configuração: 60% de todas as imagens destinadas ao conjunto de treinamento, 20% destinadas ao conjunto de validação e 20% destinadas ao conjunto de teste. Para alcançar a identificação eficiente de doenças em plantas, optou-se por empregar um método iterativo no treinamento da rede neural. Esse processo consistiu em conduzir diversos treinamentos, variando dois parâmetros específicos de maneira independente: a velocidade de aprendizado e a redução de peso. Esses hiperparâmetros são ajustados previamente ao treinamento e ao desenvolvimento de todas as iterações do algoritmo, portanto influenciam diretamente na velocidade de treinamento e no desempenho final da rede neural. Cada um desses parâmetros foi empregado com três valores distintos, seguindo uma abordagem de distribuição em grade, de acordo com a Figura 8 abaixo:

Figura 8: *Sessões de treinamento*

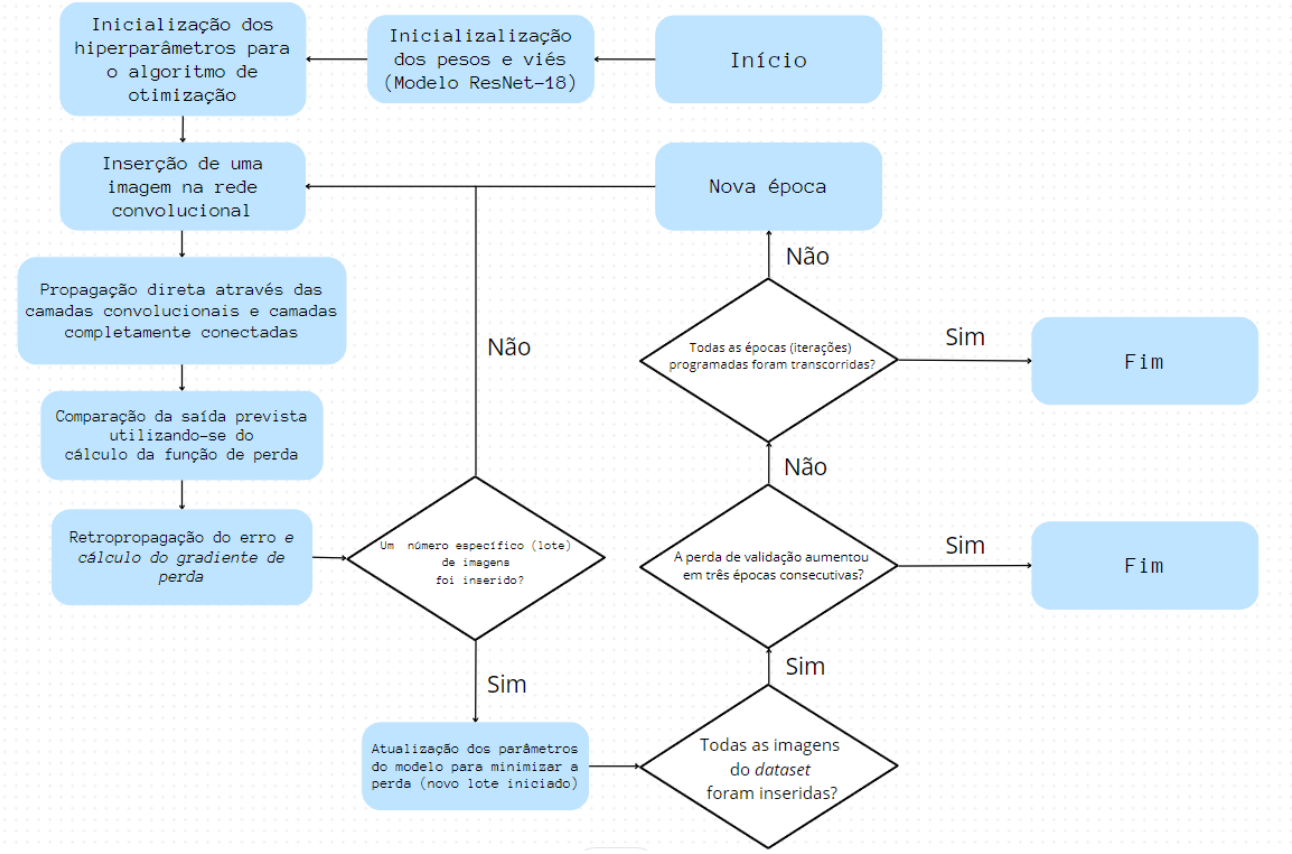
	SESSÃO DE TREINAMENTO				
	1	2	3	4	5
VELOCIDADE DE APRENDIZADO	0.02	0.00002	0.002	0.002	0.002
REDUÇÃO DE PESO	0.003	0.003	0.003	0.0003	0.00001

Fonte: Autor

Essa variação sistemática permitiu explorar diferentes configurações e otimizar o desempenho da rede. Após a realização das primeiras 5 sessões de treinamento, os melhores valores para os dois parâmetros foram escolhidos e um ajuste fino foi realizado, utilizando-se valores próximos aos encontrados, para uma tentativa de melhorar o desempenho da rede neural. Em relação aos dados de entrada, na primeira etapa de treinamento foram utilizados dois conjuntos de dados distintos, imagens de folhas saudáveis e imagens de folha acometidas por *Diabrotica speciosa*, constituindo uma identificação binária. Posteriormente, foi adicionada uma terceira classe composta por imagens de folhas danificadas por lagartas,

portanto todos os três conjuntos de dados foram combinados e empregados no treinamento subsequente da rede, constituindo uma multi classificação. Essa abordagem visa avaliar o impacto da inclusão de diferentes fontes de dados na capacidade do modelo de generalizar a identificação de plantas. Para o treinamento com as 3 classes foram utilizados diretamente os parâmetros otimizados encontrados na primeira etapa de treinamento de classificação binária. Para a análise de todas as sessões de treinamento realizadas, a métrica escolhida para avaliação dos resultados e para balizar o desempenho da rede neural foi a perda de validação do modelo, medida que reflete a capacidade do modelo de generalizar para dados não vistos durante o treinamento. Buscou-se minimizar essa métrica ao longo das iterações de treinamento, indicando um melhor ajuste do modelo aos dados de validação. O fluxograma da Figura 9 demonstra de maneira simplificada o funcionamento de uma sessão de treinamento:

Figura 9: *Fluxograma simplificado do processo de treinamento*

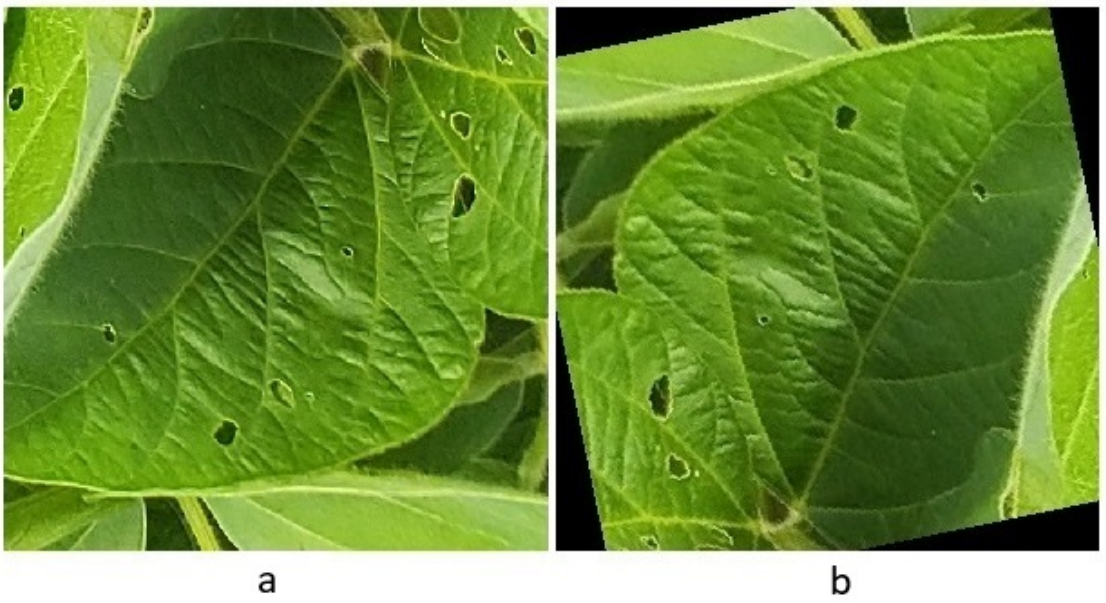


Fonte: Autor

Posteriormente, para avaliar a robustez da rede neural em relação a variações de dados de entrada e aumentar a diversidade do conjunto de treinamento, visando um aumento de performance e capacidade de generalização do modelo, foi realizado um segundo conjunto de treinamento com técnicas de aumento de dados, utilizando-se os mesmos parâmetros otimizados obtidos na etapa de treinamento sem aumento de dados. As técnicas específicas de aumento de dados utilizadas nas imagens foram: espelhamento horizontal e vertical, ambos com 50% de chance de ocorrência; rotação, de maneira aleatória, dentro de uma faixa de -45 graus a 45 graus; transformação da coloração da imagem para escala de cinza, com uma probabilidade de 10%. A Figura 10 demonstra a transformação de uma

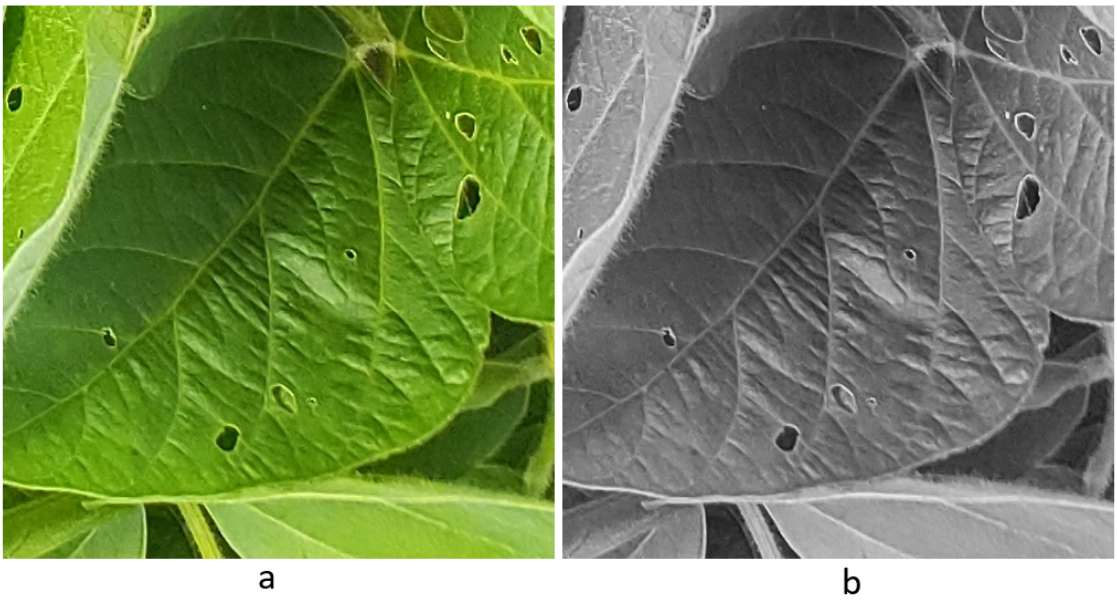
imagem utilizando rotação e espelhamento vertical e horizontal e a Figura 11 demonstra a transformação de uma imagem de folha de soja para a escala de cinza.

Figura 10: *Transformação de uma imagem de uma folha de soja: a) Imagem original; b) Imagem rotacionada e espelhada verticalmente e horizontalmente*



Fonte: Autor

Figura 11: *Transformação de uma imagem de uma folha de soja: a) Imagem original; b) Imagem transformada para escala de cinza*



Fonte: Autor

4 RESULTADOS E DISCUSSÕES

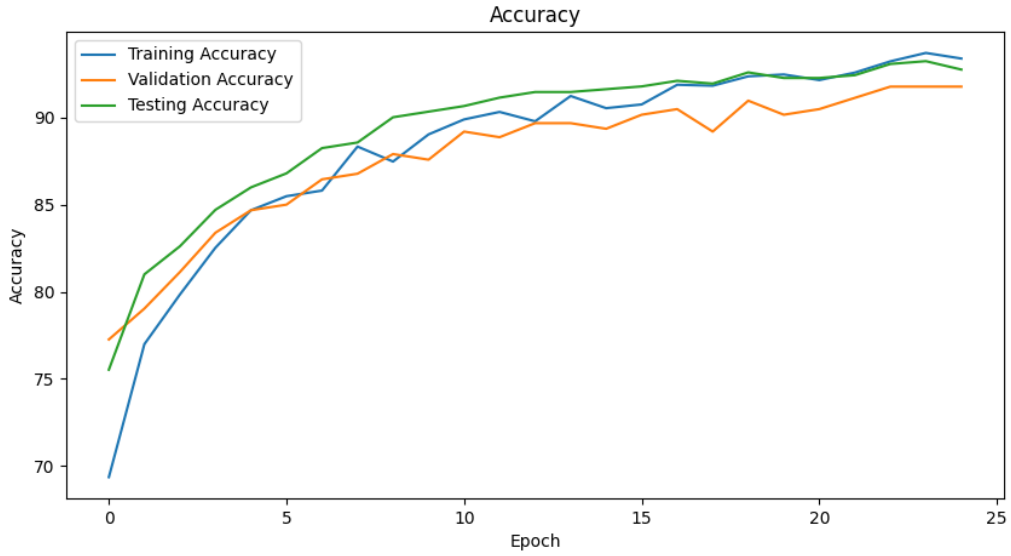
4.1 Treinamento rede neural com imagens de soja: *Diabrotica speciosa*

O treinamento da rede neural com imagens de soja foi feito inicialmente somente para as imagens de folhas saudáveis e acometidas por *Diabrotica speciosa*, consistindo portanto em uma classificação binária. Foi realizada uma pesquisa em grade em um espaço de hiperparâmetros predefinido. A pesquisa em grade é especialmente interessante devido à natureza iterativa e empírica do treinamento de redes neurais, pois explora um conjunto predefinido e ajuda na obtenção de um modelo de desempenho satisfatório. Como explicado na Sessão 3.4, foram escolhidos dois hiperparâmetros para variação: a velocidade de aprendizado e a redução de pesos.

Todas as sessões de treinamento foram realizadas com o tamanho do lote fixado em 16, que empiricamente foi definido como o valor ideal para o treinamento iterativo. Tamanhos de lote maiores ou iguais a 32 dificultam a generalização do modelo, visto que os pesos são atualizados com menor frequência, aumentando a variação da precisão do modelo e o tempo de convergência. Tamanhos de lote inferiores a 16 aumentam exacerbadamente o tempo de treinamento, visto que os pesos são atualizados com alta frequência, mas não fornecem melhora significativa na precisão do modelo. Para facilitar a influência de cada parâmetro no treinamento utilizou-se um número fixo de épocas de 25 e não foi implementado algoritmo de parada antecipada.

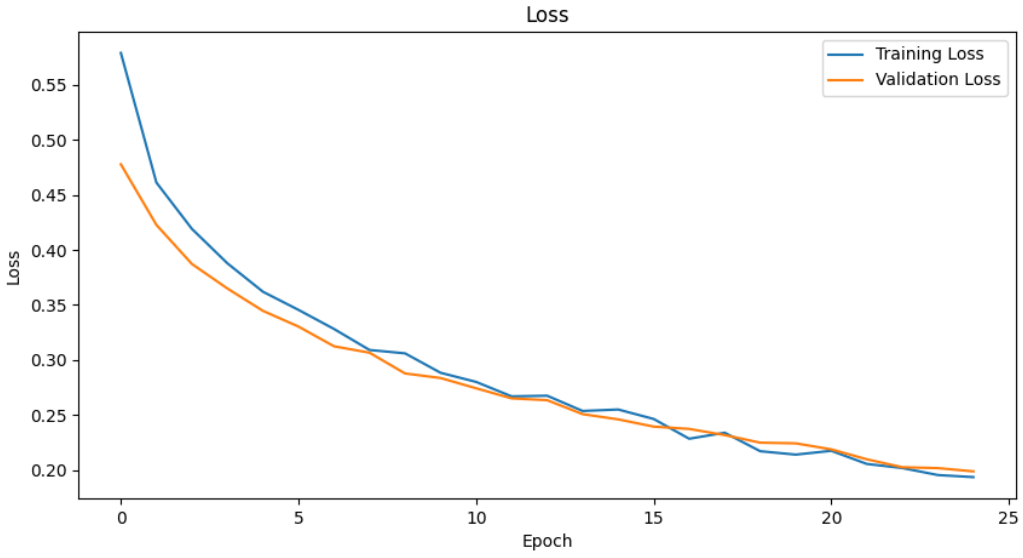
Inicialmente, a velocidade de aprendizado e a redução de pesos foram variadas de forma independente para permitir a análise da influência de cada parâmetro sobre o modelo de forma isolada. Para a velocidade de aprendizado, mantendo a redução de pesos fixa, foram utilizados 3 valores: 0,02, 0,002 e 0,00002, conforme as Figuras 12 a 15 e as Figuras 20 e 21. As linhas em azul referem-se aos dados de treinamento, as linhas em laranja aos dados de validação e as linhas em verde aos dados de teste. O eixo das abscissas (horizontal) contém o número de épocas e o eixo das ordenadas (vertical) representa a acurácia.

Figura 12: *Acurácia da rede neural durante 25 épocas com velocidade de aprendizado de 0,00002 e redução dos pesos 0,003*



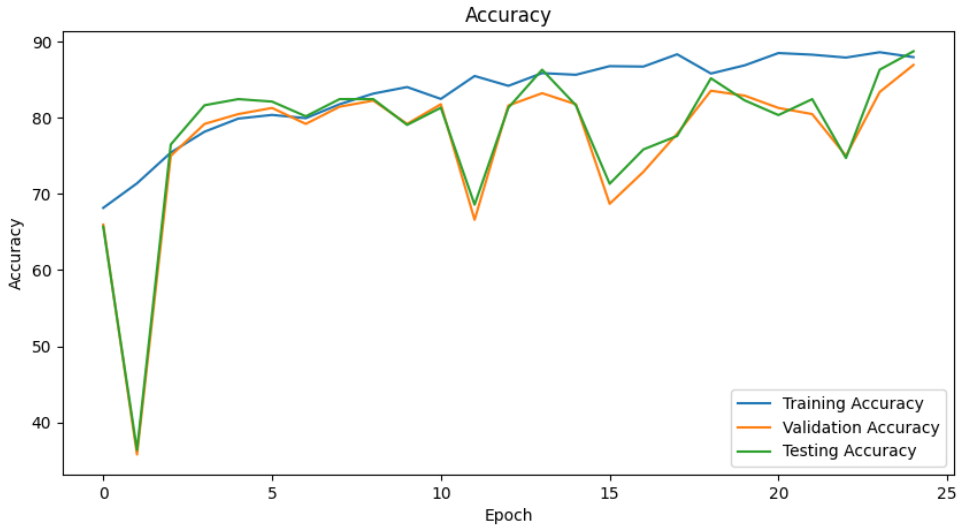
Fonte: *Autor*

Figura 13: *Perda da rede neural durante 25 épocas com velocidade de aprendizado de 0,00002 e redução dos pesos 0,003*



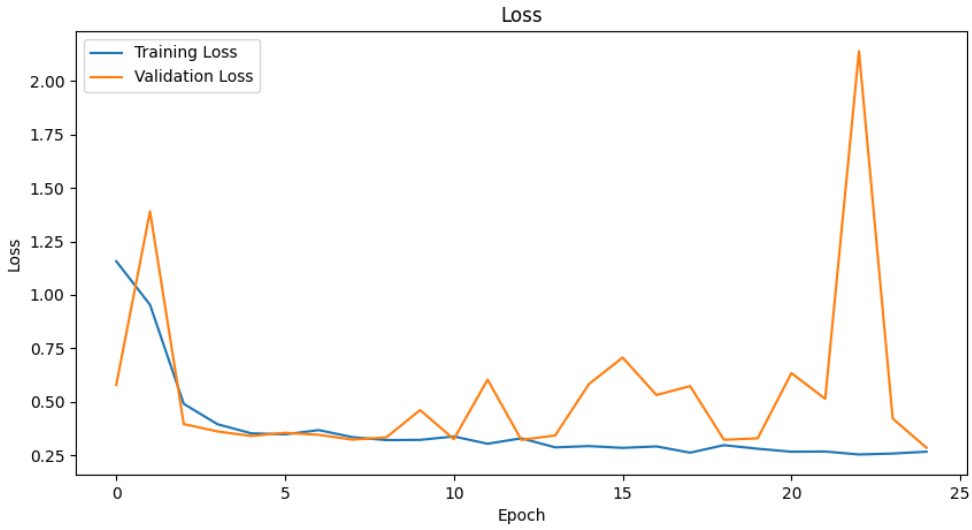
Fonte: *Autor*

Figura 14: *Acurácia da rede neural durante 25 épocas com velocidade de aprendizado de 0,02 e redução dos pesos 0,003*



Fonte: *Autor*

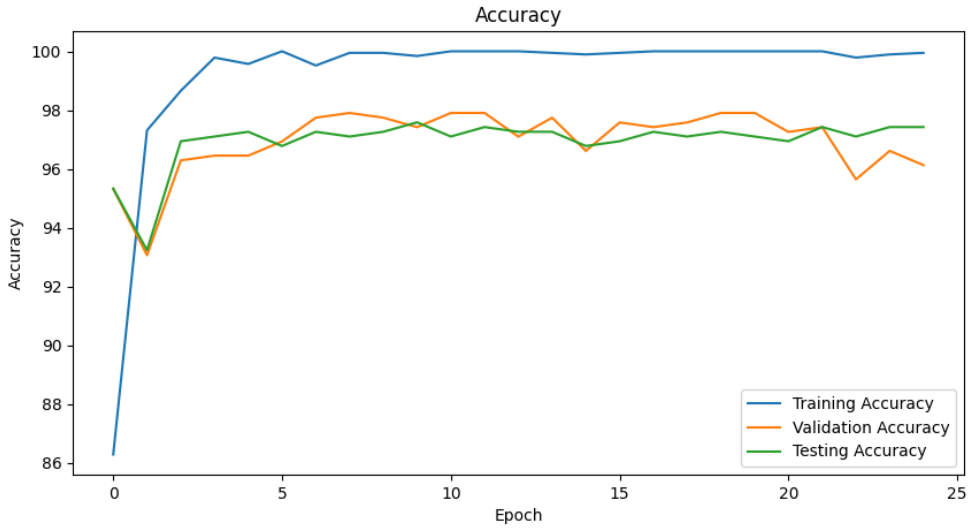
Figura 15: *Perda da rede neural durante 25 épocas com velocidade de aprendizado de 0,02 e redução dos pesos 0,003*



Fonte: *Autor*

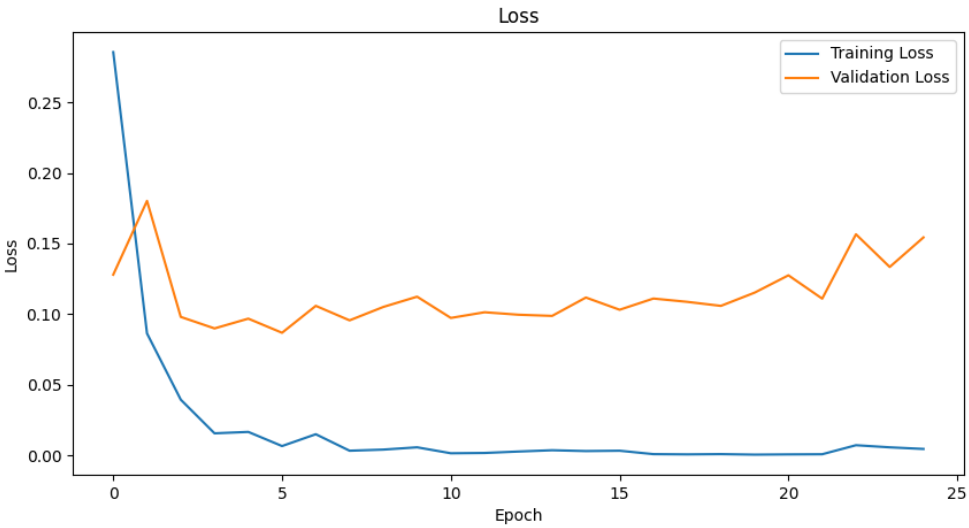
Para a redução de pesos, mantendo a velocidade de aprendizado fixa, foram utilizados os seguintes valores: 0,003, 0,0003 e 0,00001. O desempenho pode ser visualizado nas Figuras 16 a 21. Novamente, as linhas em azul referem-se aos dados de treinamento, as linhas em laranja aos dados de validação e as linhas em verde aos dados de teste.

Figura 16: *Acurácia da rede neural durante 25 épocas com velocidade de aprendizado de 0,002 e redução dos pesos 0,00001*



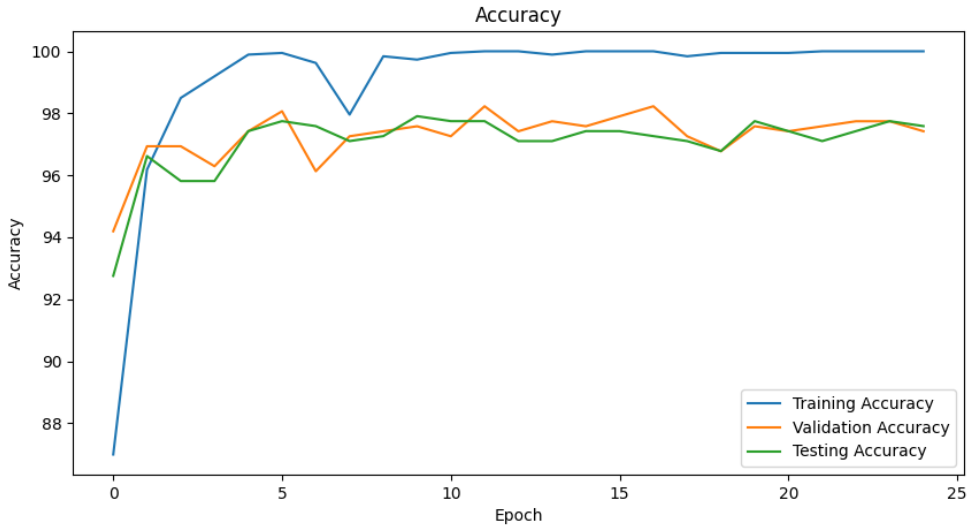
Fonte: *Autor*

Figura 17: *Perda da rede neural durante 25 épocas com velocidade de aprendizado de 0,002 e redução dos pesos 0,00001*



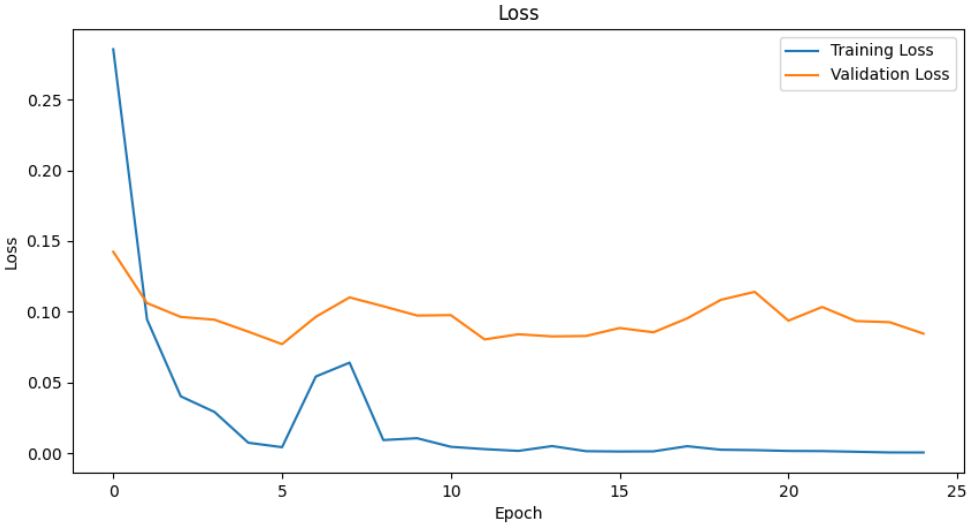
Fonte: *Autor*

Figura 18: *Acurácia da rede neural durante 25 épocas com velocidade de aprendizado de 0,002 e redução dos pesos 0,0003*



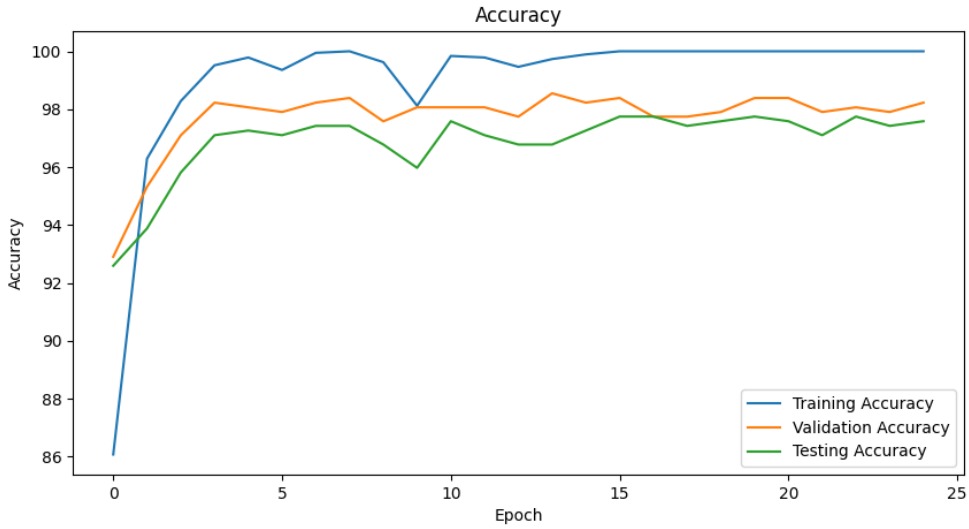
Fonte: *Autor*

Figura 19: *Perda da rede neural durante 25 épocas com velocidade de aprendizado de 0,002 e redução dos pesos 0,0003*



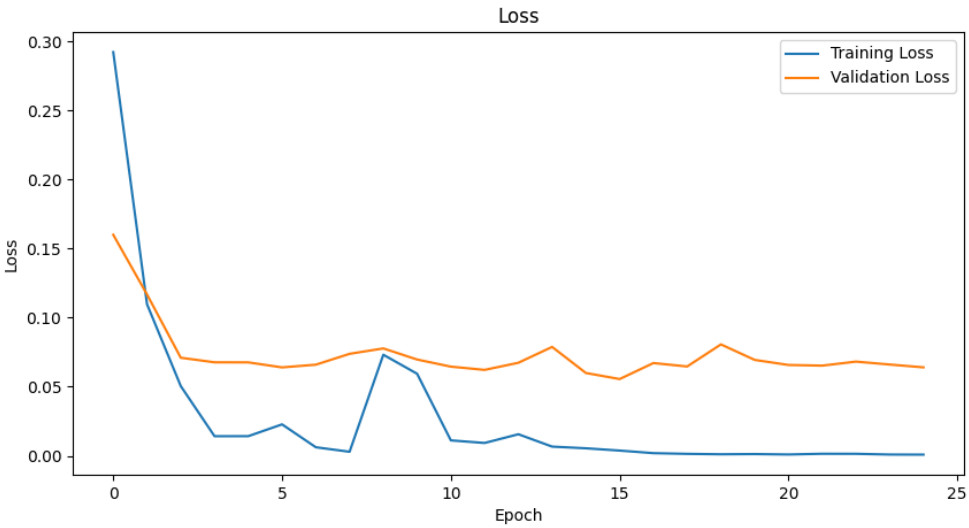
Fonte: *Autor*

Figura 20: *Acurácia da rede neural durante 25 épocas com velocidade de aprendizado de 0,002 e redução dos pesos 0,003*



Fonte: *Autor*

Figura 21: *Perda da rede neural durante 25 épocas com velocidade de aprendizado de 0,002 e redução dos pesos 0,003*



Fonte: *Autor*

É possível observar nas Figuras 12 a 21 que o modelo de melhor desempenho foi o da velocidade de aprendizado de 0,002. O modelo de 0,02 obteve uma variação grande da acurácia e perda entre algumas épocas, e a acurácia de treinamento não ultrapassou o patamar de 90%. Essa instabilidade e imprevisibilidade pode resultar em um modelo divergente, no qual os pesos oscilam muito ou crescem incontrolavelmente e a rápida atualização destes pesos pode levar a baixa capacidade de generalização. Por outro lado, uma velocidade de aprendizado de 0,00002 durante 25 épocas demonstrou uma convergência muito lenta, podendo levar muito tempo para o modelo atingir um nível de desempenho satisfatório, o que aumenta significativamente o tempo necessário para cada

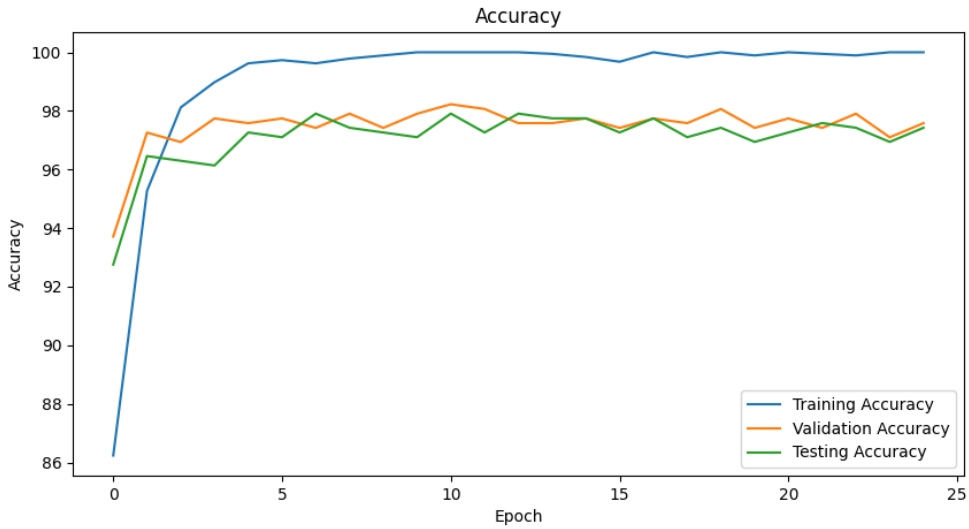
sessão de treinamento. Em relação ao parâmetro de redução de pesos, Figuras 16 a 21, o pior desempenho foi o de parâmetro 0,00001, com a perda de validação apresentando tendência de alta perto do final do treinamento. Os parâmetros de 0,003 e 0,0003 obtiveram desempenho semelhante, porém analisando o gráfico, o de 0,003 obteve menos variância entre os dados de treinamento e validação, uma convergência ligeiramente mais rápida e uma perda de validação menor, permanecendo a partir da época 3 sempre abaixo do patamar 0,1. Estabelecido que o parâmetro de melhor desempenho para a velocidade de aprendizado e redução de pesos foram, respectivamente, 0,002 e 0,003, foram realizadas mais duas sessões de treinamento, agora com os dois parâmetros variando simultaneamente. Desta vez, o espaço de pesquisa (que são os possíveis valores de atribuição para os hiperparâmetros) foi mantido pequeno, pois já foi estabelecido um bom desempenho para a rede neural e os subsequentes treinamentos serão destinados a um ajuste fino. Para facilitar a visualização, a tabela da Figura 22 apresenta o valor utilizado para cada um dos dois parâmetros para o treinamento do modelo 2 e do modelo 3, assim como os parâmetros utilizados na sessão de treinamento anterior (modelo candidato). Os gráficos de acurácia e perda são apresentados nas Figuras 23 a 26.

Figura 22: *Parâmetros utilizados para o ajuste fino*

	MODELO CANDIDATO	MODELO 2	MODELO 3
VELOCIDADE DE APRENDIZADO	0.002	0.005	0.001
REDUÇÃO DOS PESOS	0.003	0.006	0.002

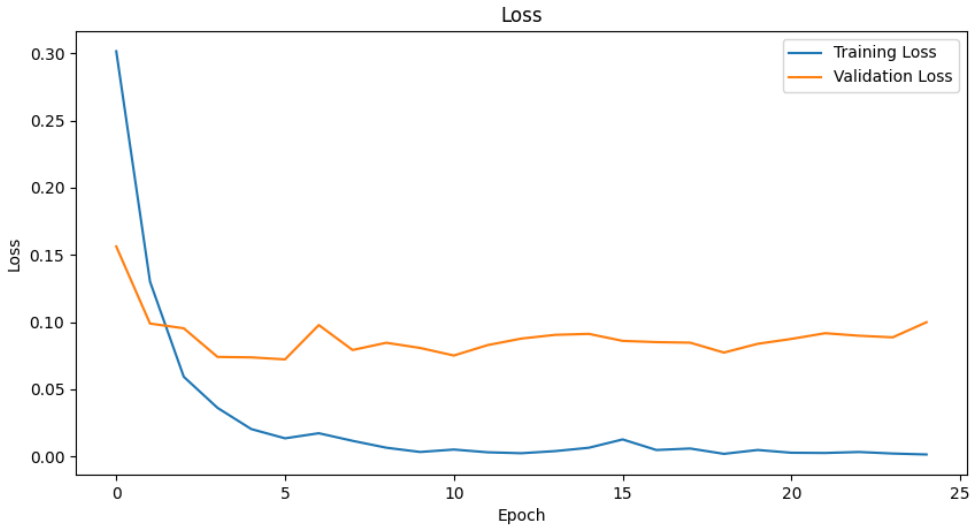
Fonte: *Autor*

Figura 23: *Acurácia da rede neural durante 25 épocas com velocidade de aprendizado de 0,001 e redução dos pesos 0,002*



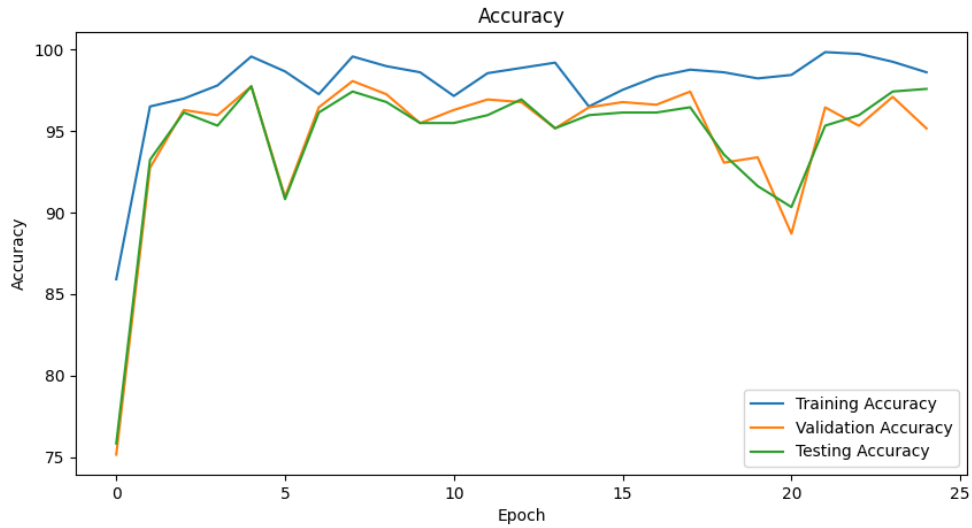
Fonte: *Autor*

Figura 24: *Perda da rede neural durante 25 épocas com velocidade de aprendizado de 0,001 e redução dos pesos 0,002*



Fonte: *Autor*

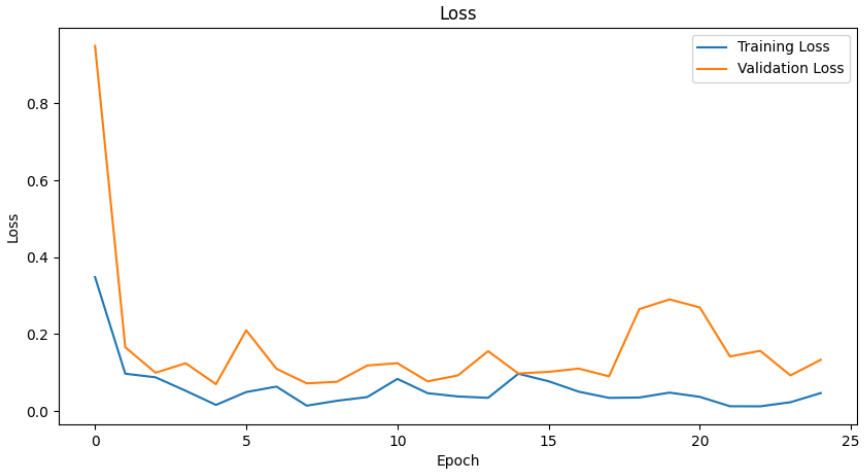
Figura 25: *Acurácia da rede neural durante 25 épocas com velocidade de aprendizado de 0,005 e redução dos pesos 0,006*



Fonte: *Autor*

O parâmetro que foi escolhido como métrica para a escolha do melhor modelo foi a acurácia nos dados de teste. Isso se deve aos seguintes motivos: o banco de dados utilizado possui uma quantidade considerável de imagens para as duas classes, portanto o desbalanceamento de classes não é tão acentuado (razão inferior a 3 entre imagens de folhas doentes e saudáveis), evitando a predição predominante da classe maioritária e desempenho fraco na classe minoritária; imagens eventualmente classificadas de maneira errada não são consideradas altamente prejudiciais para a tarefa que está sendo proposta; a acurácia de teste de todos os modelos foi alta, com todos eles obtendo uma acurácia superior a 97%. O modelo com a maior acurácia de teste foi o modelo candidato, que

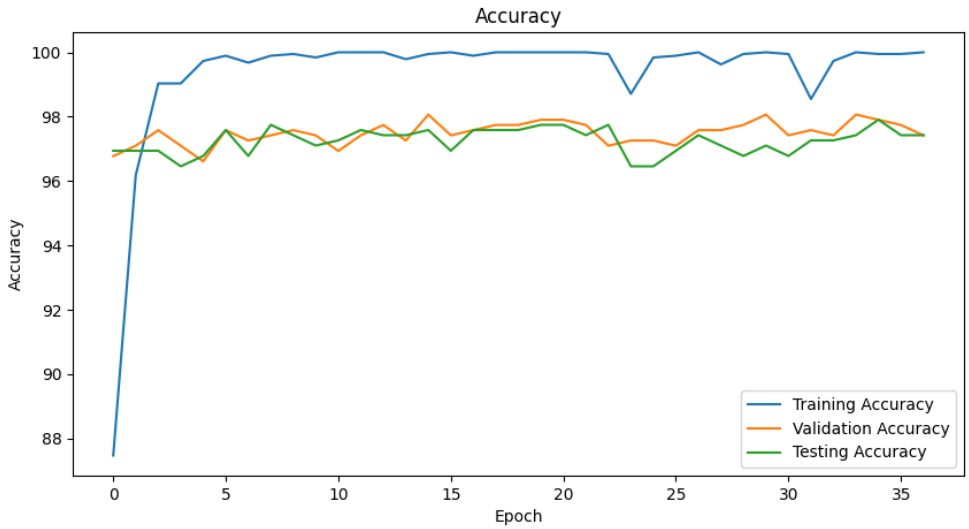
Figura 26: *Perda da rede neural durante 25 épocas com velocidade de aprendizado de 0,005 e redução dos pesos 0,006*



Fonte: *Autor*

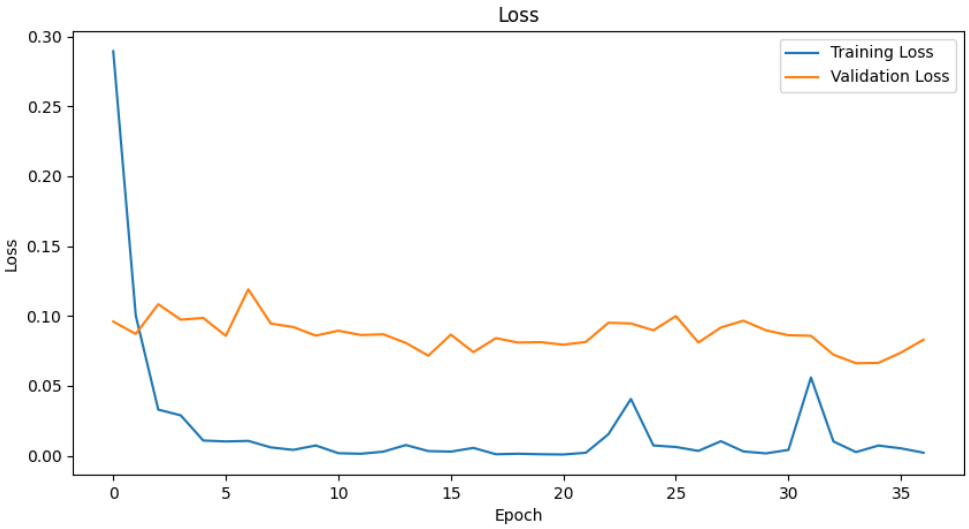
obteve 98,06% de precisão de teste, com menos de uma imagem classificada de maneira errada a cada 50. Por fim, foi feito um último treinamento com estes parâmetros escolhidos, agora com parada antecipada, o que significa que o treinamento é encerrado se em 3 épocas consecutivas a perda de validação aumente. O número de épocas foi aumentado para 50 para permitir um treinamento mais extensivo, se necessário. O resultado do treinamento é apresentado nas Figuras 27 e 28.

Figura 27: *Acurácia da rede neural com os parâmetros escolhidos e parada antecipada*



Fonte: Autor

Figura 28: *Perda da rede neural com os parâmetros escolhidos e parada antecipada*



Fonte: Autor

O treinamento decorreu em um período de 37 épocas e foi interrompido devido ao aumento consecutivo da perda de validação durante as épocas 35, 36 e 37, indicando uma redução na performance do modelo. O modelo de melhor performance foi alcançado na época número 34, na qual a acurácia de teste chegou ao patamar de 97,9%. Esse valor é inferior ao valor de 98,06% alcançado durante o treinamento sem parada antecipada ao longo de 25 épocas, realizado anteriormente na etapa de atualização de hiperparâmetros. Embora treinar uma rede neural para mais épocas possa levar a uma melhora de desempenho, isso não garante melhoria contínua. Analisando todos os treinamentos realizados durante este trabalho, é perceptível que o modelo converge rapidamente para um patamar de performance que não consegue superar, fazendo com que mais tempo de treinamento seja contraprodutivo. Se o modelo fosse treinado indefinidamente não existiria garantia de melhoria, resultando em um sobreajuste desnecessário que é evitado pela parada antecipada.

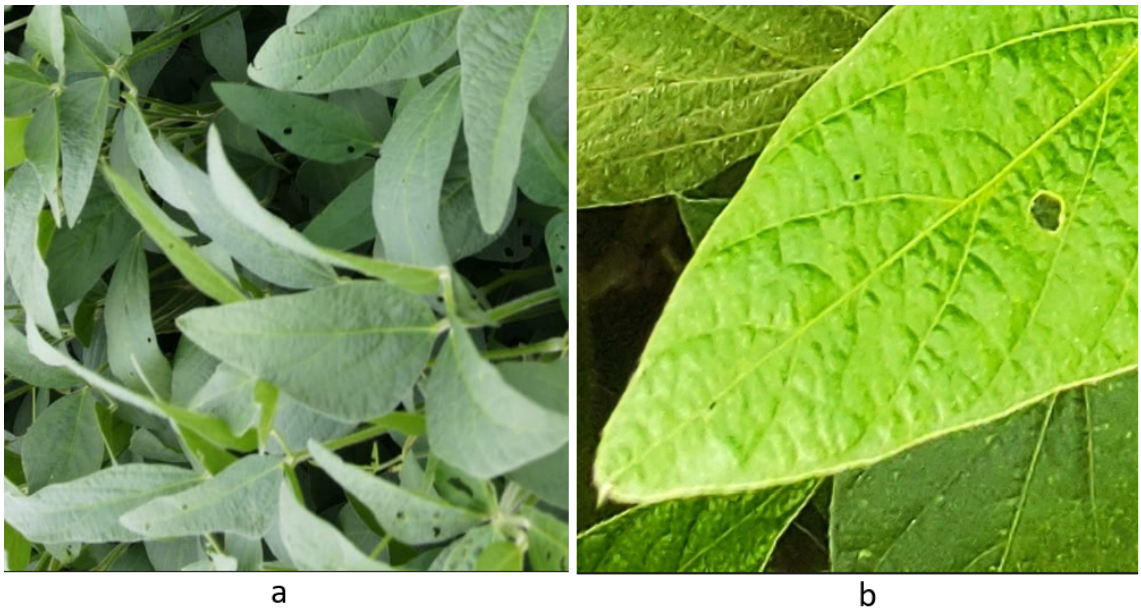
4.2 Treinamento rede neural com imagens de soja: *Diabrotica speciosa* e lagarta

Nesta nova sessão de treinamento da rede neural com imagens de soja, foram adicionadas imagens de folhas afetadas por lagartas, portanto o treinamento será feito com 3 classes distintas. O principal desafio desta classificação específica é a semelhança entre as imagens de folhas *Diabrotica speciosa* e lagarta, como pode ser visto nas Figuras 29 e 30. Porém, há algumas pequenas diferenças ao se analisar as imagens. As folhas danificadas por lagartas possuem furos, na média, maiores e com formatos distintos, enquanto os furos nas folhas com *Diabrotica speciosa* ocorrem com menos frequência, são menores e possuem quase sempre um formato que se assemelha a de um círculo.

Figura 29: *Exemplos de folhas de soja: a) Folha danificada por lagarta; b) Folha danificada por lagarta*

Fonte: (MIGNONI, 2021)

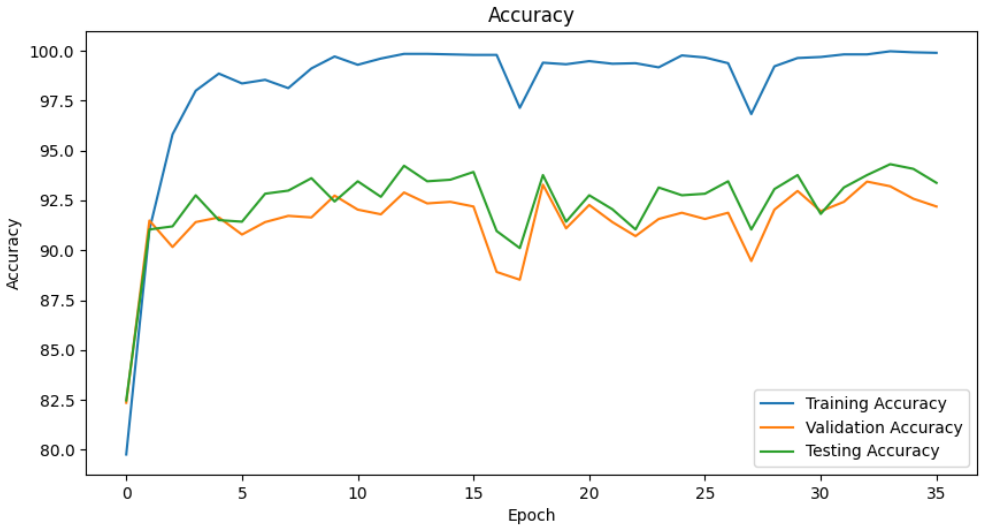
Figura 30: *Exemplos de folhas de soja: a) Folha danificada por Diabrotica speciosa; b) Folha danificada por Diabrotica speciosa*



Fonte: (MIGNONI, 2021)

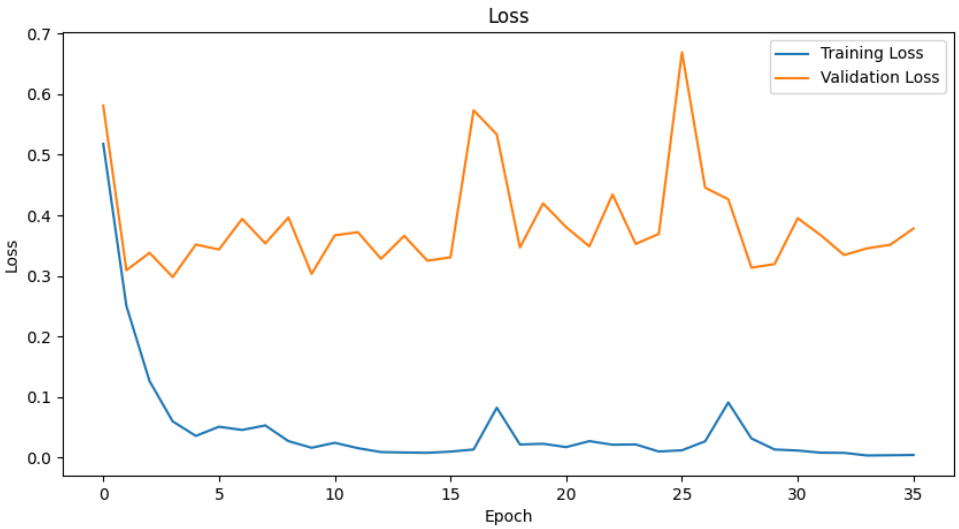
Para fins de consistência e para evitar longas sessões extras de treinamento, os hiperparâmetros utilizados foram os mesmos do que para a classificação binária somente com *Diabrotica speciosa*. O número de épocas utilizado foi 50 e com parada antecipada de 3 épocas. As Figuras 31 e 32 abaixo demonstram o desempenho da rede neural:

Figura 31: *Acurácia da rede neural para Diabrotica speciosa e lagarta com parada antecipada*



Fonte: Autor

Figura 32: *Perda da rede neural para Diabrotica speciosa e lagarta com parada antecipada*



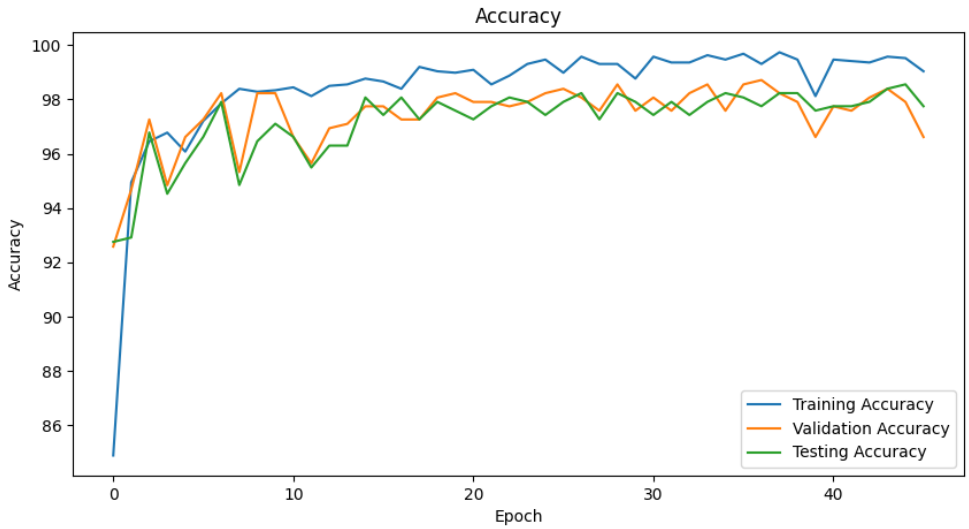
Fonte: Autor

A melhor acurácia de teste ocorreu na época 33, atingindo o valor de 94,31%, valor significativamente inferior ao obtido na classificação binária somente com folhas saudáveis e *Diabrotica speciosa*. Isso se deve não somente ao fato de ter uma classe a mais para identificação, mas principalmente pela semelhança entre as imagens das folhas acometidas por *Diabrotica speciosa* e lagarta. Um conjunto altamente semelhante de recursos entre classes pode causar ambiguidade durante o treinamento e dificultar o aprendizado de distinções significativas pelo modelo.

4.3 Treinamento da rede neural com aumento de dados: *Diabrotica speciosa*

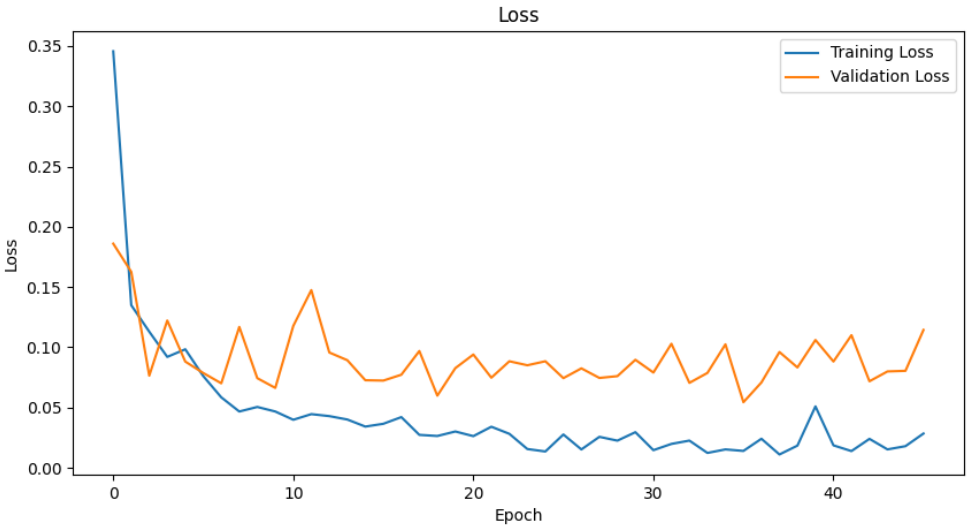
Nesta nova sessão de treinamento da rede neural, foram empregadas as técnicas de aumento de dados explicitadas na sessão de validação experimental (seção 2.4). As Figuras 33 e 34 apresentam o desempenho da rede neural com aumento de dados para a classificação binária entre folhas saudáveis e folhas com *Diabrotica speciosa*. O treinamento foi interrompido na época de número 46 e a maior acurácia de teste foi atingida na época de número 44, com valor de 98,55%, 0,49% superior ao treinamento realizado sem aumento de dados. O modelo original, obtido sem aumento de dados, apresentou um desempenho muito alto e foi treinado em um conjunto de dados grande e diversificado, capturando uma ampla gama de variações e características presentes nas imagens e potencialmente limitando o impacto das técnicas de aumento de dados empregadas. Portanto, a tarefa parece apresentar inerentemente um potencial limitado de melhoria através do aumento de dados.

Figura 33: *Acurácia da rede neural para Diabrotica speciosa com aumento de dados e parada antecipada*



Fonte: Autor

Figura 34: *Perda da rede neural para Diabrotica speciosa com aumento de dados e parada antecipada*



Fonte: Autor

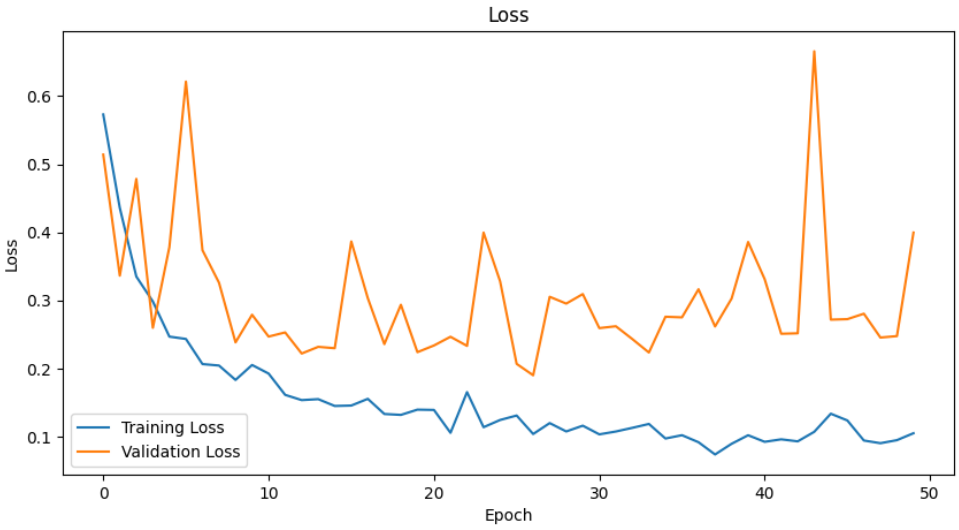
4.4 Treinamento rede neural com aumento de dados: *Diabrotica speciosa* e lagarta

Nesta nova sessão de treinamento, foi empregado o mesmo processo de treinamento com aumento de dados utilizado na classificação binária somente com *Diabrotica speciosa*, agora com a inclusão da base de dados de folhas danificadas por lagarta. As Figuras 35 e 36 demonstram o desempenho da rede neural. É possível observar que não ocorreu parada antecipada e o treinamento transcorreu todas as 50 épocas inicialmente programadas. A maior acurácia de teste foi atingida na época de número 22, com valor de 95,63%, 1,32% superior ao treinamento realizado sem aumento de dados para a classificação das 3 classes.

Figura 35: *Acurácia da rede neural para Diabrotica speciosa e lagarta com aumento de dados e parada antecipada*

Fonte: Autor

Figura 36: *Perda da rede neural para Diabrotica speciosa e lagarta com aumento de dados e parada antecipada*



Fonte: Autor

5 CONCLUSÕES

Este trabalho propôs a utilização de uma rede neural convolucional para identificação de pragas em plantas. Para compreender os principais requisitos para o desenvolvimento desta aplicação, foi feita uma pesquisa sobre os principais conceitos de aprendizado de máquina e do processo de treinamento de uma CNN. Com base nos conhecimentos adquiridos, foram definidas as estratégias de treinamento iterativo e atualização dos hiperparâmetros da rede neural, assim como a métrica que seria utilizada para avaliação dos resultados obtidos.

A aplicação da rede neural com classificação binária para imagens de folhas de soja saudáveis e imagens de folhas acometidas por *Diabrotrica speciosa* demonstrou um desempenho notável, alcançando uma acurácia de teste elevada de 98,06%. No entanto, ao expandir o escopo para uma multi classificação composta por 3 classes distintas a partir da adição de um conjunto de dados de folhas doentes atacadas por lagartas, percebeu-se uma diminuição na precisão do modelo, atribuível à semelhança entre as imagens das duas classes de plantas doentes e ao aumento da complexidade dos dados de entrada. Uma limitação que pode ter impedido um resultado superior ao obtido é o existente, apesar de pequeno, desequilíbrio no conjunto de dados de treinamento. Há uma minoria de imagens de folhas de soja saudáveis, representando aproximadamente 14% do total de imagens do conjunto de dados de treinamento.

A introdução de técnicas de aumento de dados revelou um aumento mínimo no desempenho dos modelos. Essa marginal diferença, de menos de 1,5% para os dois tipos de classificação, pode ser atribuída a já alta performance do modelo sem aumento de dados, bem como à presença, anterior à utilização das técnicas de aumento de dados, de um número significativo de imagens com elevada diversidade no conjunto de dados de treinamento, fazendo com que a tarefa de identificação tenha sido facilitada pela quantidade substancial de dados disponíveis. As imagens de validação e teste utilizadas para balizar o desempenho dos modelos obtidos não foram utilizadas para o treinamento, garantindo a independência dos conjuntos dos dados de entrado durante todo o processo. Essa separação foi crucial para avaliar de maneira correta o desempenho dos modelos.

Apesar dos bons resultados obtidos, é importante ressaltar que todos os resultados deste trabalho são decorrentes do treinamento e validação realizados em uma única base de dados, escolhida por representar variedades de soja plantadas no país. O resultado do modelo pode ser avaliado com outros conjuntos de dados para garantir a generalização dos resultados. Futuros trabalhos podem explorar a otimização desses parâmetros ou a utilização de arquiteturas de rede mais avançadas para aprimorar a precisão em cenários multiclasse, bem como o uso de bases de dados mais diversas. Em última análise, este estudo contribui para o avanço do conhecimento na aplicação de visão computacional na identificação de doenças em plantas, destacando tanto as conquistas alcançadas quanto as oportunidades de aprimoramento.

REFERÊNCIAS

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. http://www.deeplearningbook.org.

ISLEIB, J. Signs and symptoms of plant disease: Is it fungal, viral or bacterial? *Michigan State University*, p. 1, dez. 2012. Disponível em: <https://www.canr.msu.edu/news/signs_and_symptoms_of_plant_disease_is_it_fungal_viral_or_bacterial>. Acesso em: 8 out. 2023.

MICHELE PRETI, F. V. S. A. Insect pest monitoring with camera-equipped traps: strengths and limitations. *Springer Link; https://doi.org/10.1007/s10340-020-01309-4*, p. 1, dez. 2020. Disponível em: <https://link.springer.com/article/10.1007/s10340-020-01309-4>. Acesso em: 12 abr. 2020.

MIGNONI, M. E. Images of Soybean Leaves. *Mendeley Data; DOI:10.17632/bycbh73438.1*, p. 1, dez. 2021. Disponível em: <https://data.mendeley.com/datasets/bycbh73438/1>. Acesso em: 12 set. 2023.

MITCHELL, T. M. *Machine Learning*. [S.l.]: McGraw-Hill Science/Engineering/Math; (March 1, 1997), 1997.

RADHAKRISHNAN, P. *What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network?* [S.l.], 2017. P. 1. Disponível em: <https://towardsdatascience.com/what-are-hyperparameters-and-how-totune-the-hyperparameters-in-a-deep-neural-network-d0604917584a>. Acesso em: 8 out. 2023.

SAÚDE - OMS, S. Q. E. *Chemical safety: Pesticides*. [S.l.], 2020. P. 1. Disponível em: <https://www.who.int/news-room/questions-and-answers/item/chemical-safety-pesticides>.

SZELISKI, R. *Computer Vision: Algorithms and Applications: Vol 2*. [S.l.]: Springer, 2022.

VINICIUS, A. W. Arquitetura de Redes Neurais Artificiais. *Blog Ateliware*, p. 1, jun. 2021. Disponível em: <https://ateliware.com/blog/redes-neurais-artificiais>.

Apêndices

Apêndice A - CÓDIGO FONTE DESENVOLVIDO

Código desenvolvido neste trabalho disponível em: `https://github.com/MarceloFSalvadori/IdentificacaoPragasVisao.git`