

USO DA YOLOV8 NA IDENTIFICAÇÃO DAS DOENÇAS MANCHA-ALVO E MANCHA OLHO-DE-RÃ NA CULTURA DA SOJA.

Carlos Pinheiro Ferreira Junior

Mario Lúcio Gomes de Queiroz Pierre Junior

Morgana Mateus Santos

RESUMO

PALAVRAS-CHAVE:

ABSTRACT

KEYWORDS:

1 INTRODUÇÃO

Os avanços tecnológicos e científicos trouxeram significativas contribuições em relação ao desenvolvimento agrícola, assim, aumentando em escala exponencial a produção de alimentos. O Brasil é um dos principais produtores e exportadores agrícolas, sendo que “as exportações brasileiras do agronegócio bateram recorde em 2023, atingindo US\$ 166,55 bilhões. A cifra foi 4,8% superior em comparação a 2022, o que representa um aumento de US\$ 7,68 bilhões.” (Brasil, 2024).

Mesmo produzindo variados grãos e frutos, o país é o principal *player* internacional quando se trata da soja, estando a frente de países como Estados Unidos e Argentina, a ascensão rápida do Brasil como um dos principais fornecedores globais de *commodities* é evidenciada pela destacada posição que o país ocupa na produção e exportação de soja, liderando esse mercado mundialmente. (Valdes; Gillespie; Dohlman, 2023).

No entanto, a produção de soja no país sofre com algumas perturbações. Doenças, como a mancha-alvo causada pelo fungo *Corynespora cassiicola*, emergem como fatores-chave na diminuição da produtividade da soja no Brasil, podendo resultar em uma queda de até 40% na produtividade. (Pivotto, 2023). A mancha olho-de-rã também, doença causada pelo fungo *Cercospora sojina* é um outro agravante problema, doença que também reduz a produtividade, podendo causar perdas de 31% a 60%. (Barro et al., 2023)

Entretanto, a melhor forma de combater esses fungos com “a aplicação de fungicidas constitui uma das principais estratégias de controle da mancha alvo” (Pivotto, 2023, p. 13), para isso o acompanhamento constante da cultura é o ideal, pois quanto mais cedo diagnosticados os problemas, mais eficaz será a aplicação do fungicida, “um dos aspectos que mais prejudicam a eficiência de fungicidas para o controle da mancha-alvo tem sido o atraso na primeira aplicação.” (Tormen; Blum; Balardin, 2017, p. 26).

Os progressos recentes no contexto computacional, principalmente no campo da inteligência artificial e visão computacional, propiciaram a viabilização de estudos que empregam essas tecnologias no reconhecimento pragas e doenças presente na agricultura, essas inovações têm o potencial de facilitar o diagnóstico, aumentar a precisão e diminuir os gastos com pesticidas. (Bever; Sikora; Hardy, 2022).

Nesse contexto, torna-se essencial o desenvolvimento de métodos que facilitem a identificação precoce dos sintomas de pragas e doenças nas culturas de soja. A eficácia dessas abordagens reside na capacidade de permitir uma aplicação adequada de fungicidas, assegurando não apenas a eficiência no controle desses agentes patogênicos, mas também a otimização dos recursos agrícolas.

O modelo da YOLOv8 oferece uma abordagem promissora para a identificação de doenças causados por fungos na folha de soja, como a mancha-alvo e a mancha olho-de-rã. Ao treinar o modelo com um conjunto diversificado de imagens de plantas infectadas, ele pode aprender a reconhecer padrões visuais específicos associados a essas doenças.

Nesse contexto, como o modelo da YOLOv8 pode ser empregado na identificação dos sintomas causados por fungos na folha de soja, incluindo a mancha-alvo e a mancha olho-de-rã, contribuindo assim para aprimorar as estratégias de diagnóstico e manejo fitossanitário nessa cultura agrícola?

2 DESENVOLVIMENTO

2.1 FUNDAMENTAÇÃO TEÓRICA

A partir daqui serão introduzidos todos os conhecimentos necessários para o entendimento deste trabalho. Assim, serão apresentadas as duas doenças, mancha-alvo e mancha olho-de-rã, juntamente com seus sintomas causadores; técnicas e ferramentas computacionais que foram aplicados para a detecção, como também trabalhos correlatos.

2.1.1 MANCHA-ALVO (*CORYNESPORA CASSIICOLA*)

A mancha-alvo, doença causada pelo fungo *Corynespora Cassiicola*, é um mal que atinge mais de 400 espécies de plantas, incluindo culturas como, mamão, feijão, tomate, etc. Também é problema que assola o cultivo da soja, sendo uma doença que se favorece com a boa distribuição de chuvas e aumento da umidade, presente em praticamente todas as regiões onde a soja é cultivada no Brasil. Parece ser endêmico e pode infectar uma ampla variedade de plantas, tanto nativas quanto cultivadas. Sua sobrevivência ocorre em restos de cultura e sementes infectadas (Embrapa, 2023).

Os sintomas predominantes da mancha-alvo incluem a formação inicial de lesões pardacentas, circundadas por um halo amarelado, que progressivamente se desenvolvem em manchas circulares. Estas variam de tonalidades castanho-claros a castanho-escuros. Dependendo da reação, as lesões podem expandir-se alcançando até 2 cm de diâmetro ou permanecer em tamanho reduzido, oscilando entre 1 mm e 3 mm, mas em maior quantidade. Tipicamente, as manchas exibem uma pontuação central, acompanhada de anéis concêntricos de coloração mais intensa. (Embrapa, 2023).

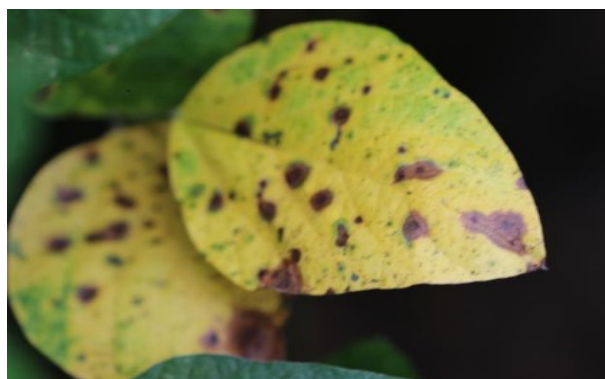


Figura 1 - mancha-alvo na folha da soja
Disponível em: <https://doi.org/10.5061/dryad.41ns1mj3>

2.1.2 MANCHA OLHO-DE-RÃ (*CERCOSPORA SOJINA*)

A segunda doença a ser identificada é a mancha olho-de-rã, que é causada pelo fungo *Cercospora Sojina*. A doença pode surgir em qualquer fase do desenvolvimento da planta, mas, é mais frequente a partir do florescimento. Afeta folhas, hastes, vagens e sementes, manifestando-se inicialmente como pequenas áreas encharcadas (anasarca), que posteriormente se transformam em manchas com centros castanho-claros na página superior das folhas e cinzentos na página inferior, com bordas castanho avermelhadas em ambas as páginas (Embrapa soja, 2014).

O fungo se dissemina através de sementes infectadas e esporos transportados pelo vento, sendo capaz de sobreviver em resíduos de plantas. As condições favoráveis para a doença incluem alta umidade e temperatura. Além disso, o patógeno tem a capacidade de gerar novas variantes (Embrapa soja, 2014).

De acordo com as recomendações, a adoção de cultivares resistentes juntamente com o tratamento de sementes utilizando fungicidas benzimidazois combinados com fungicidas de contato de maneira sistemática são medidas essenciais para o controle da doença e para prevenir a introdução do fungo ou o surgimento de novas variantes (Embrapa soja, 2014).



Figura 2 - mancha olho-de-rã na folha da soja.
Disponível em: <https://doi.org/10.5061/dryad.41ns1rnj3>

2.1.3 REDES NEURAI E APRENDIZADO PROFUNDO

As redes neurais artificiais, ou redes neurais (RNs) como são popularmente conhecidas, é um subconjunto da área de inteligência artificial e aprendizado de máquina, que modela o funcionamento cerebral para executar tarefas específicas, utilizando componentes eletrônicos ou simulação por software em computadores (Haykin, p. 2, 2009, tradução nossa).

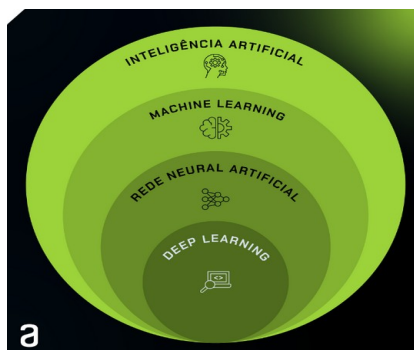


Figura 3 - rede neural artificial e deep learning como subcampo da IA.

Disponível em:
<https://www.alura.com.br/artigos/assets/machine-learning/grafico.png>

Para que as RNs possam simular o funcionamento do cérebro, elas utilizam o *perceptron*, que se trata da unidade fundamental para o funcionamento de uma rede neural (Haykin, 2009). O *perceptron* opera de forma análoga a um neurônio biológico, que recebe sinais elétricos, os processa e emite um sinal de saída apenas quando a força total dos sinais de entrada ultrapassa um limite específico. Para replicar esse comportamento, o neurônio artificial calcula a soma ponderada das entradas, representando a intensidade total dos sinais, e aplica uma função degrau (função de ativação) para determinar se a saída deve ser ativada (1) caso o sinal ultrapasse o limite ou desativada (0) caso contrário (Elgendy, 2020).

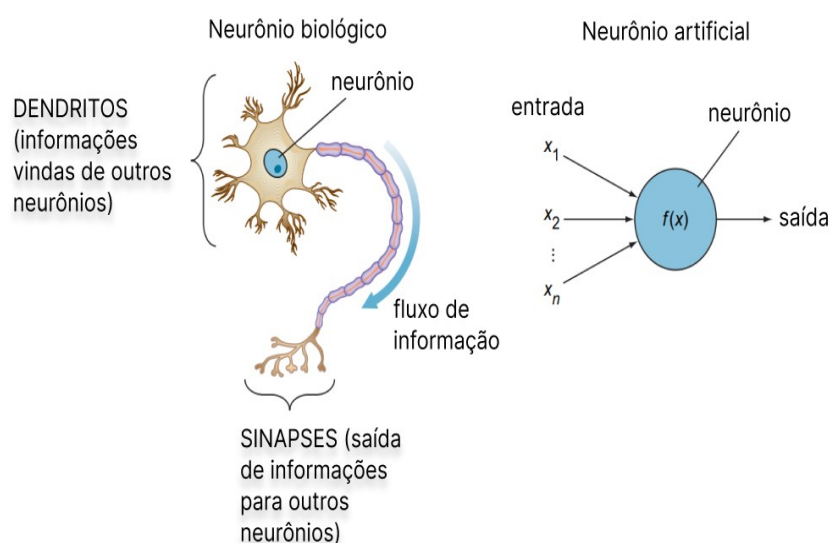


Figura 4 - neurônios artificiais foram inspirados em neurônios biológicos
Fonte: Elgendy, 2020, p. 9

Apesar dos *perceptrons* servirem como base para o funcionamento de uma rede neural, um neurônio sozinho não é capaz de realizar muita coisa, o *perceptron* pode ser entendido como uma representação linear, o que implica que, após o treinamento, o neurônio será capaz de traçar uma linha reta que divide os dados em duas categorias distintas (Elgendy, 2020, p. 43). Um único perceptron é capaz de resolver apenas problemas lineares, como o que ocorre com os casos *AND* e *OR* na imagem 5 por exemplo, no qual há uma separação entre os círculos vermelhos e a estrela azul, indicando que apenas um neurônio é o suficiente para realizar essa separação. Sendo que o mesmo já não é possível com o problema XOR presente na mesma imagem, “reserve um segundo para se convencer de que é impossível desenhar uma única linha que separa as estrelas azuis dos círculos vermelhos” (Rosebrock, 2017).

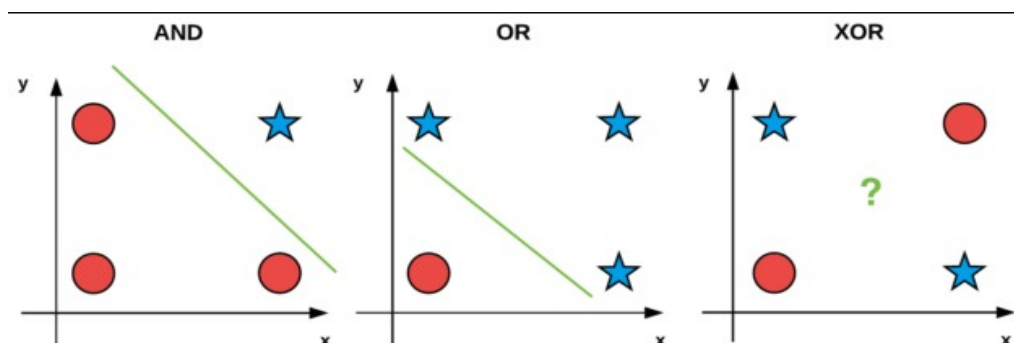


Figura 5 - separação linear que ocorre utilizando apenas um perceptron (neurônio artificial)
Disponível em: <https://pyimagesearch.com/2021/05/06/implementing-the-perceptron-neural-network-with-python/>

Para que as redes neurais possam ser utilizadas em problemas mais complexos, é necessário utilizar mais camadas de neurônios, surgindo assim o termo “aprendizado profundo”, que advém justamente da construção de múltiplas camadas neurais interconectadas, “isso significa que precisamos criar uma arquitetura para usar dezenas e centenas de neurônios em nossa rede neural.” (Elgendy, 2020, p. 46, tradução nossa).

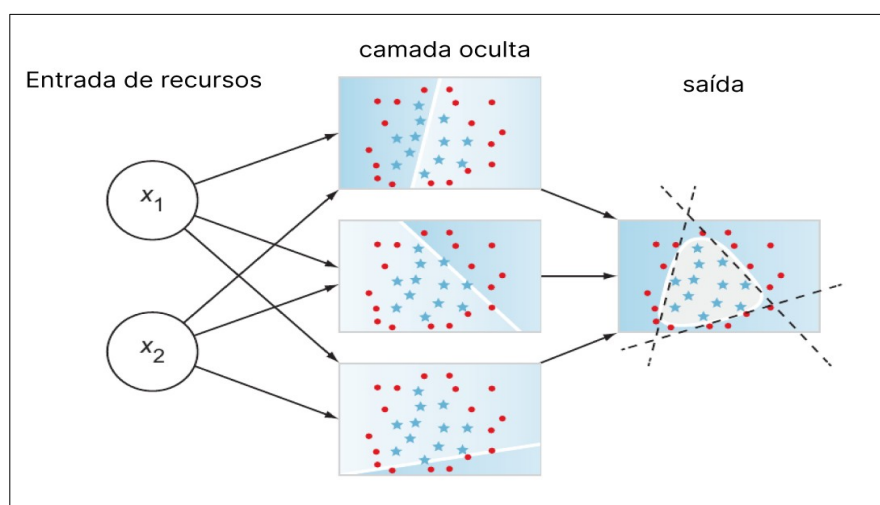


Figura 6 - utilizando 3 neurônios para separar os círculos vermelhos das estrelas azuis.
Fonte: Elgendy, 2020, p. 46

É nas camadas ocultas que a tudo acontece, é onde está o cerne do processo de aprendizado de características: as primeiras camadas detectam padrões simples, como linhas retas, enquanto as posteriores identificam padrões mais complexos dentro dessas especificações. Em redes neurais, adicionamos camadas ocultas para aprender características complexas através das camadas até ajustar nossos dados. Então, se sua rede neural não estiver se adaptando bem aos dados, talvez seja necessário adicionar mais camadas ocultas (Elgendy, 2020).

A camada oculta recebe esse nome, “porque não vemos ou controlamos a entrada que vai para essas camadas ou para a saída. Tudo o que fazemos é alimentar o vetor de recursos para a camada de entrada e ver a saída da camada final.” (Elgendy, 2020, p. 46, tradução nossa). Essa ideia causa a sensação de que as redes neurais são implacáveis, e que basta sair adicionando camadas infinitamente e você terá a solução de todos os problemas, porém, quanto mais camadas, mais recursos computacionais são necessários, além de que, o excesso de profundidade pode levar ao *overfitting*, onde a rede se ajusta demais aos dados de treinamento e não generaliza bem com novos dados.

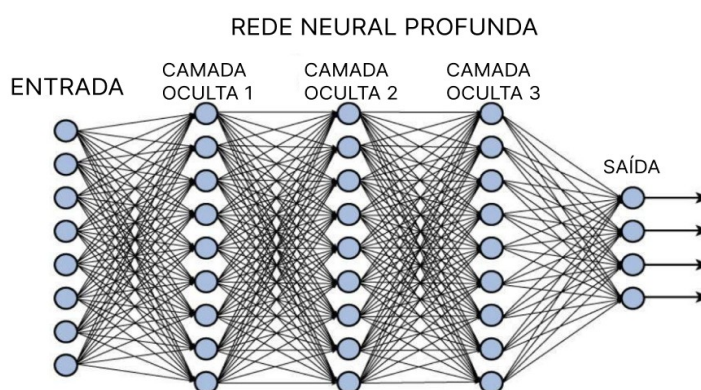


Figura 7 - rede neural profunda com 3 camadas ocultas.

Disponível em:

<https://jmvstream.com/wp-content/uploads/2023/07/Deep-Learning.jpeg>

2.1.4 YOU ONLY LOOK ONCE (YOLO)

O YOLO é um modelo de detecção em tempo real desenvolvido por Joseph Redmon e Ali Farhadi em 2015 que ficou muito popular devido sua eficiência e acurácia, obtendo uma precisão igual ou superior a outros modelos concorrentes (Alves, 2020).

O YOLO (*you only look once*) recebeu esse nome devido a sua característica de passada única (*single pass*), no qual, diferente dos outros modelo de detecção em imagem, percorre a imagem uma única vez, dessa forma ele acaba se tornando muito mais ágil que os outros modelos (Redmon, 2020).

O funcionamento do YOLO é baseado na divisão da imagem em uma grade de células, geralmente com dimensões $S \times S$. Na versão YOLOv8, por exemplo, essa grade tem 13 linhas por 13 colunas. Cada célula da grade é encarregada de prever 5 caixas delimitadoras, que são retângulos que definem a área onde um objeto pode estar presente na imagem. Essas caixas delimitadoras são acompanhadas por probabilidades de classe, indicando a chance de cada

caixa conter um objeto de uma classe específica. Essa abordagem permite uma detecção eficiente e precisa de múltiplos objetos em tempo real (Aggarwal, 2020).

Além disso, o YOLO utiliza uma única arquitetura de rede neural para realizar a detecção e classificação dos objetos, sendo uma abordagem de ponta a ponta, o que o torna mais rápido e eficiente em comparação com métodos tradicionais que exigem múltiplas etapas de processamento.

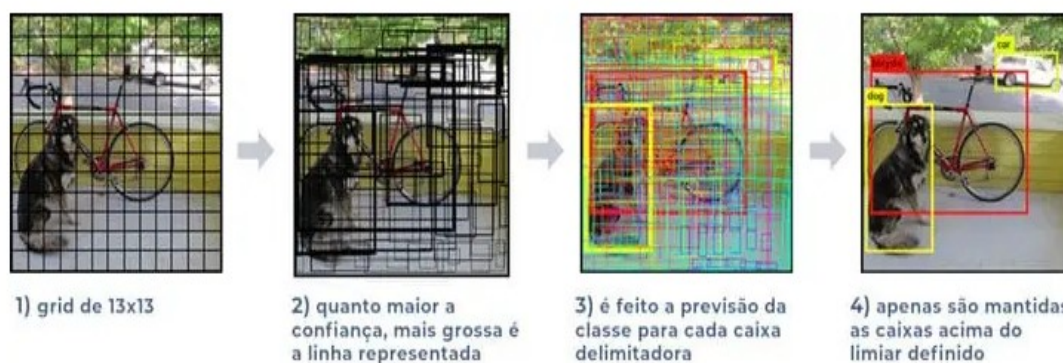


Figura 8 - demonstração do funcionamento do YOLO.

Disponível em: <https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna/>

No exemplo ilustrado na imagem 8, são evidenciadas as etapas conduzidas pelo YOLO, que resultam na inclusão de 845 caixas delimitadoras, derivadas do cálculo de 13x13, totalizando 169 células, e posteriormente multiplicadas por 5, representando o número de caixas por célula. Contudo, a maioria dessas caixas delimitadoras apresenta baixa relevância, sendo consideradas apenas aquelas cuja pontuação excede 30% do limiar de confiança, o qual é ajustável. Simultaneamente à detecção da presença de objetos dentro das caixas delimitadoras, o modelo também realiza a classificação de qual é o item contido dentro da caixa em questão, cuja a pontuação final resulta da combinação entre a predição da classe e a da caixa delimitadora. Essa abordagem ilustra a meticulosa seleção e avaliação das detecções efetuadas pelo modelo YOLO, refletindo sua capacidade de discernimento e precisão na identificação de objetos em imagens (Alves, 2020).

2.1.5 MÉTRICAS DE AVALIAÇÃO

Para avaliarmos os resultados obtidos ao final do treinamento e da validação, empregamos métricas amplamente utilizadas em tarefas de visão computacional. Essas métricas incluem, precisão, *recall*, mAP50, mAP50:95 e F1-score. Para compreender cada

métrica, é fundamental primeiro entender o conceito de matriz de confusão e a Interseção sobre União (IoU, do inglês *Intersection over Union*).

2.1.5.1 MATRIZ DE CONFUSÃO

A matriz de confusão serve para indicar os erros e acertos, comparando os resultados obtidos, com os resultados esperados (Rodrigues, 2019).

		Valor Previsto	
		Positivo	Negativo
Valor Verdadeiro	Positivo	TP Verdadeiro Positivo	FN Falso Negativo
	Negativo	FP Falso Positivo	TN Verdadeiro Negativo

Figura 9: matriz de confusão

Disponível em:

https://media.licdn.com/dms/image/C4D12AQE-hEJoe0y34w/article-inline_image-shrink_400_744/0/1538424289069?e=1721260800&v=beta&t=cP374WgNeQgMRJMpH7TXrprZtXSrv9G2zzmzYQExVT0

Como é possível ver na imagem acima, existem 4 tipos de resultados possíveis, cada um está explicado abaixo:

- **Verdadeiro positivo (TP - *true positive*)** - São os casos em que o modelo previu corretamente a classe positiva.
 - **Exemplo:** Em uma tarefa de detecção de doenças, se um exame identifica corretamente a presença de um sintoma(e a doença realmente está presente), isso é um verdadeiro positivo.
- **Verdadeiro negativo (TN - *true negative*)** - São os casos em que o modelo previu corretamente a classe negativa.
 - **Exemplo:** Na mesma tarefa de detecção de doença, se um exame identifica corretamente a ausência (e realmente não há doença), isso é um verdadeiro negativo.
- **Falso positivo (FP - *false positive*)** - São os casos em que o modelo previu incorretamente a classe positiva.

- **Exemplo:** Se o exame indica a presença de uma doença, mas na verdade não há, isso é um falso positivo. É também chamado de "alarme falso".
- **Falso negativo (FN - *false negative*)** - São os casos em que o modelo previu incorretamente a classe negativa.
 - **Exemplo:** Se o exame não detecta a doença, mas a enfermidade está presente, isso é um falso negativo.

Todos esses valores são necessários para se calcular a métrica de avaliação do desempenho do modelo treinado.

2.1.5.2 INTERSEÇÃO SOBRE UNIÃO (IoU)

Essa métrica é amplamente utilizada em modelos de identificação de objetos que empregam caixas delimitadoras (*bounding boxes*) em suas tarefas. A sua importância reside na capacidade de avaliar a precisão alcançada pelo modelo treinado, fornecendo uma medida crítica de desempenho para a eficácia do reconhecimento e delimitação de objetos (Rosebrock, 2022).

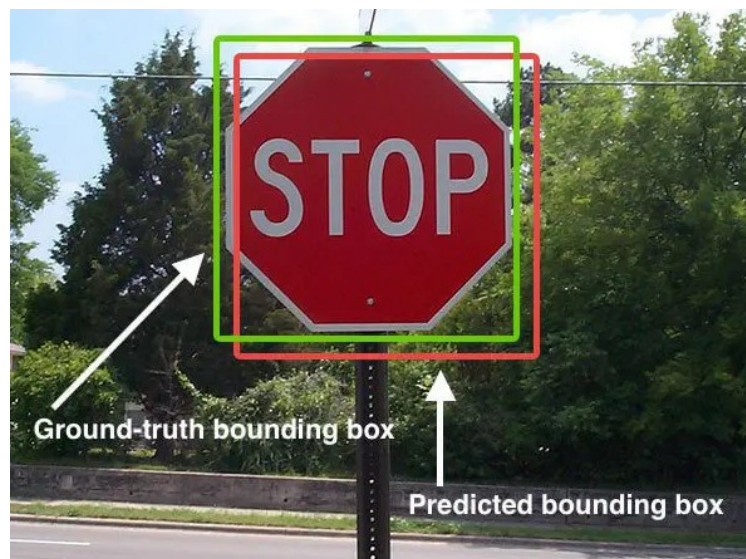
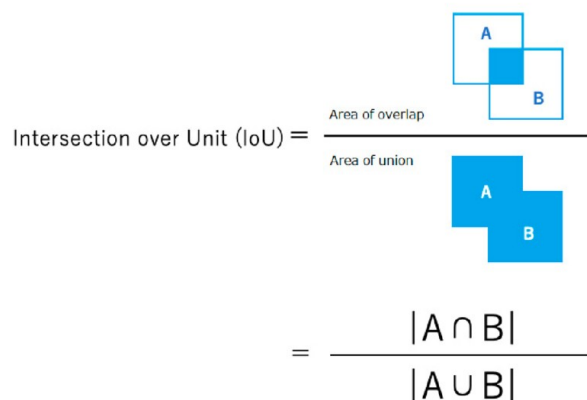


Figura 10: a caixa com linha verde representa a caixa delimitadora real e a caixa vermelha delimitada pela rede neural.

Disponível em: https://b2633864.smushcdn.com/2633864/wp-content/uploads/2016/09/iou_stop_sign.jpg?lossy=2&strip=1&webp=1

Como é possível observar na imagem acima, a IoU representa uma medida de similaridade entre duas áreas: a caixa delimitadora predita pelo modelo e a caixa delimitadora real (*ground truth*), definida como a razão entre a área de interseção das duas e a área de união das mesmas.



$$\text{Intersection over Unit (IoU)} = \frac{\text{Area of overlap}}{\text{Area of union}} = \frac{|A \cap B|}{|A \cup B|}$$

Figura 11: fórmula do IoU, sendo que o valor pode variar entre 0 e 1.

Fonte: Iwasa et al., 2022.

Acima é possível observar como é calculada essa métrica, que é um fator determinante para a classificação de uma detecção como verdadeira ou falsa. Comumente, um limiar (*threshold*) é definido, como por exemplo, 0,5 (50%). Se a IoU entre a caixa predita e a caixa real for maior ou igual ao limiar, considera-se que a detecção é verdadeira positiva (*True Positive*). Caso contrário, é considerada uma falsa positiva (*False Positive*) (Shah, 2023).

2.1.5.3 PRECISÃO

O cálculo da precisão é realizado dividindo o número de verdadeiros positivos pelo total de exemplos classificados como positivos, que é a soma dos verdadeiros positivos e dos falsos positivos. A fórmula matemática pode ser expressa como:

$$\text{precisão} = \frac{\text{verdadeiros positivos}}{\text{verdadeiros positivos} + \text{falsos positivos}}$$

Este cálculo é fundamental para avaliar a exatidão de um modelo de classificação, assim, “a precisão pode ser usada em uma situação em que os Falsos Positivos são considerados mais prejudiciais que os Falsos Negativos” (Rodrigues, 2019). refletindo a proporção de predições positivas corretas em relação ao total de predições positivas feitas pelo modelo.

2.1.5.4 RECALL

O *Recall*, também conhecido como sensibilidade, é uma métrica de desempenho utilizada para avaliar a eficácia de um modelo de classificação. O cálculo de recall é definido como a razão entre o número de verdadeiros positivos e a soma dos verdadeiros positivos e dos falsos negativos. A fórmula matemática é:

$$recall = \frac{\text{verdadeiros positivos}}{\text{verdadeiros positivos} + \text{falsos negativos}}$$

Essa métrica mede a capacidade do modelo de identificar corretamente todas as instâncias relevantes (verdadeiros positivos) no conjunto de dados. “O recall pode ser usado em uma situação em que os Falsos Negativos são considerados mais prejudiciais que os Falsos Positivos” (Rodrigues, 2019). Por exemplo, o modelo deve identificar todos os pacientes doentes, mesmo que classifique alguns saudáveis como doentes (falsos positivos). Isso significa que o modelo deve ter um alto recall, pois classificar pacientes doentes como saudáveis pode ter consequências graves (Rodrigues, 2019).

2.1.5.5 Mean Average Precision (mAP)

O *mean average precision* (mAP) é uma métrica utilizada para avaliar a performance de modelos de detecção de objetos e classificação multiclasse. O cálculo leva em consideração todas as métricas discutidas anteriormente.

Primeiro precisamos calcular a precisão média de cada classe, sendo calculada como a média ponderada das precisões em cada limiar definido, no caso do mAP50, “a precisão média calculada com um limiar de intersecção sobre união (IoU) de 0,50. É uma medida da precisão do modelo considerando apenas as detecções ‘fáceis’” (Ultralytics, 2023, tradução nossa).

No caso do mAP50:95, “a média de precisão aritmética é calculada em diferentes limiares de IoU, variando de 0,50 a 0,95. Dá uma visão abrangente do desempenho do modelo em diferentes níveis de dificuldade de detecção.” (Ultralytics, 2023, tradução nossa). O cálculo pode ser observado abaixo:

$$mAP = \frac{1}{N} \cdot \sum_{i=1}^N AP_i$$

Acima é possível observar que o mAP é uma média entre todas as médias obtidas em cada classe, ou seja, é a somatória de todos os *average precision* (AP) dividido pela quantidade de categorias.

2.1.5.6 F1-score

Essa métrica serve para caso você precise encontrar equilíbrio entre a precisão e o *recall*, são ocasiões onde as 2 medidas são importantes. O F1-score “determina o limiar de confiança ideal, onde a precisão e o recall resultam na maior pontuação F1. A pontuação F1 mede o equilíbrio entre precisão e recall.” (Shah, 2023, tradução nossa). O cálculo F1 pode ser analisado abaixo:

$$F1 - score = \frac{2 \cdot (precisão \cdot recall)}{precisão + recall}$$

Apesar de ser uma operação matemática simples, essa métrica muito importante para uma avaliação geral do desempenho de uma rede neural.

2.2 ESTADO DA ARTE

O referencial teórico para este projeto traz autores que possam contribuir com o desenvolvimento e ampliação do mesmo, de forma que apesar de desconhecidas por muitos, as aplicações de aprendizado profundo na área da agricultura estão ganhando muita força, e através de pesquisas é perceptível a importância da área no presente e principalmente no futuro.

O primeiro trabalho que vale destacar é o artigo *Efficient Tobacco Pest Detection in Complex Environments Using an Enhanced YOLOv8 Model* feito por Daozong Sun et al. (2024). O estudo realizado por Daozong Sun e sua equipe apresenta um modelo YOLOv8 aprimorado para detecção eficiente de pragas do tabaco em ambientes complexos. No seu trabalho os pesquisadores coletaram manualmente 460 imagens de 4 tipos diferentes de pragas da cultura do tabaco, sendo distribuídas da seguinte maneira: 143 para lagartas, 102 para aranhas vermelhas, 113 para afídeos e 102 para lagarta-enroladeira.

O *dataset* passou por anotações utilizando o *software* Labelme, e também foi realizado um aumento artificial utilizando um estilo mosaico, no qual uma única imagem é construída combinando várias partes de imagens diferentes, ao final o conjunto ficou com um total de 4000 imagens, sendo feita uma divisão de 80% para treinamento e 20% para validação.

O modelo incorpora o mecanismo de atenção SimAM e o módulo VoV-GSCSP para aprimorar a extração de recursos e a localização do alvo. Testes comparativos mostram que o modelo aprimorado reduz a quantidade de parâmetros e os requisitos computacionais, ao mesmo tempo que melhora as métricas de desempenho de detecção, como médias gerais, recall e precisão. O modelo YOLOv8 aprimorado demonstra precisão de detecção superior e eficiência de parâmetros em comparação com outras estruturas de detecção de objetos. Este avanço equilibra o consumo de recursos com a precisão da detecção, oferecendo benefícios significativos para a detecção de pragas na agricultura.

Ao final foi feita uma comparação entre as versões da YOLOv8L padrão e o YOLOv8L aprimorado. No fim, a versão modificada utilizando 23,4 milhões de parâmetros a menos apresentou uma melhora de média de 2% nas médias gerais e na precisão, atingindo respectivamente 88.9% e 92.7%, “o YOLOv8 aprimorado equilibra o consumo de recursos com a precisão da detecção. Alcançando uma redução significativa nos parâmetros e nos requisitos computacionais, ao mesmo tempo que aprimora as capacidades de detecção do modelo.” (Daozong et al, 2024, p. 19, tradução nossa).

O próximo trabalho *Alpha-EIOU-YOLOv8: An Improved Algorithm for Rice Leaf Disease Detection*, apresenta um estudo de pesquisa de Trinh et al. publicado na AgriEngineering em 2024. O trabalho traz um algoritmo aprimorado, Alpha-EIOU-YOLOv8, projetado para detectar doenças nas folhas do arroz com rapidez e precisão. O sistema utiliza uma configuração de hardware que consiste em um microcontrolador, câmera USB e módulo GSM para capturar imagens em tempo real de plantas de arroz no campo, identificar doenças de plantas usando uma versão modificada da YOLOv8 para detecção, e alertar os agricultores por e-mail e mensagem de texto.

Para a etapa de treinamento, os autores coletaram manualmente 1643 imagens, sendo que o dataset contava com 3 pragas comuns à cultura do arroz: lagarta desfolhadora, lagarta-enroladeira e a mancha-marrom. As imagens foram adquiridas em diferentes plantações de arroz e em diferentes condições climáticas.

Também foi realizado um aumento artificial nesse conjunto de imagens, sendo aplicadas modificações verticais, horizontais e translações. Além disso, também foram feitas imagens mosaicas. Ao final o *dataset* continha 3175 imagens, dividido em 3 partes, sendo 2608 para treinamento, 326 para validação e 241 para testes.

A seção de metodologia descreve a visão geral do sistema, design de hardware, preparação de dados, arquitetura YOLOv8, métricas de avaliação e cálculo de perdas. O

estudo emprega métricas como precisão, recall, mAP e pontuação F1 para fins de avaliação. A pesquisa demonstra melhorias no desempenho de detecção de doenças em comparação com outros métodos de última geração. No geral, a pesquisa de Trinh et al. contribui para os avanços na engenharia agrícola, introduzindo um algoritmo mais eficiente para detectar doenças nas folhas do arroz, que pode ajudar os agricultores na gestão oportuna de doenças e na proteção das culturas.

O terceiro trabalho é o *Vegetable disease detection using an improved YOLOv8 algorithm in the greenhouse plant environment* desenvolvido pelos pesquisadores Wang e Liu (2024). Este estudo apresenta o modelo YOLOv8n-vegetal, um algoritmo leve e eficiente para detecção de doenças em vegetais. O modelo incorpora vários aprimoramentos, incluindo o módulo GhostConv, módulo de atenção de percepção de oclusão, camada de detecção de objetos de pequeno porte e função de perda de HIoU.

Os autores treinaram essa rede modificada para detectar 20 doenças nas culturas do pepino, berinjela e tomate. Ao total foram coletadas cerca de 28000 imagens, divididas em 80% para treinamento, 10% validação e 10% para testes.

Os resultados demonstram o desempenho superior do modelo na detecção precisa das doenças, atingindo médias gerais de 92.91% e com uma velocidade superior quando comparadas com a versão padrão da YOLOv8n. O destaque ficou principalmente para as doenças de pequena escala e ocluídas, sendo essas uma dificuldade que a YOLO encontra devido a maneira como funciona. O modelo proposto mostra-se promissor para aplicação no mundo real na detecção e manejo de doenças vegetais.

3 METODOLOGIA

Para o desenvolvimento deste trabalho foram necessárias as seguintes etapas: Criação do Banco de Imagens e Treinamento e avaliação da Yolov8. A seguir serão descritas cada uma das etapas.

3.1 BANCO DE IMAGENS

Uma rede neural é uma máquina que replica o funcionamento do cérebro para executar tarefas específicas. Ela é implementada com componentes eletrônicos ou simulada em software. Para funcionar eficazmente, as redes neurais utilizam neurônios interconectados e

um algoritmo de aprendizagem para ajustar os pesos sinápticos e alcançar objetivos de design desejados (Haykin, 2009, p. 2, tradução nossa).

Ter um conjunto de dados com quantidade e variedade de amostras é fundamental para obter bons resultados no treinamento de redes neurais. Atualmente já existem vários sites que são focados na compartilhamento de conjuntos de dados prontos para utilização, como por exemplo o Kaggle e o Roboflow universe.

Porém, nem sempre é possível encontrar um banco de imagens com o que se procura para o treinamento de uma rede. Sendo necessário, por vezes, a criação de um dataset, passando pelas etapas de aquisição de imagens, e pré-processamento das amostras.

Para esta pesquisa, as imagens foram obtidas através da internet com os trabalhos “*Pictures of diseased soybean leaves by category captured in field and with controlled backgrounds: Auburn soybean disease image dataset (ASDID)*” (Bever, Sikora e Hardy, 2022) e “*SoyNet: Indian Soybean Image dataset with quality images captured from the agriculture field (healthy and disease Images)*” (Rajput, Shukla e Thakur, 2023).

No primeiro trabalho, os autores disponibilizaram um banco de imagens contendo oito categorias diferentes de folhas de soja, totalizando mais de 40 GB. As categorias incluem, mancha bacteriana marrom, mancha púrpura da semente, míldio, mancha olho-de-rã, mancha-alvo, ferrugem-asiática, deficiência de potássio e folha saudável. As imagens foram capturadas com uma câmera Canon EOS 7D Mark II, com resolução de 5472 x 3648 pixels e tamanho médio de arquivo de 4MB.

O segundo banco de imagens também conta com várias categorias diferentes, porém a intenção foi conseguir mais imagens da folha saudável, ignorando as outras categorias. Ao total foram adicionadas 509 capturas da folha saudável, sendo que as imagens foram capturadas utilizando uma câmera Nikon L810 e um smartphone Motorola G4, com resolução de 5472 x 3648 pixels e tamanho médio de 5,7 MB.

Com os *datasets* definidos, iniciou-se o pré-processamento das imagens. Inicialmente, as imagens foram redimensionadas utilizando a biblioteca PIL (Python Imaging Library) da *Python* para uma resolução de 640 x 480 pixels, reduzindo para uma média de 24,6 KBs por imagem, pois além de facilitar o manuseio das imagens, é fundamental remover informações desnecessárias e manter apenas os aspectos relevantes para impactar positivamente na velocidade de treinamento (Elgendy, 2020).

Em seguida, foi utilizada outra biblioteca *Python* chamada *rembg* para remover o fundo das imagens. Essa ferramenta foi especialmente projetada para essa finalidade,

ajudando a remover objetos que não são de interesse, trazendo maior eficiência na hora do treinamento da rede neural. Esse processo de tratamento foi essencial para preparar as imagens para o treinamento de modelos de aprendizado profundo.

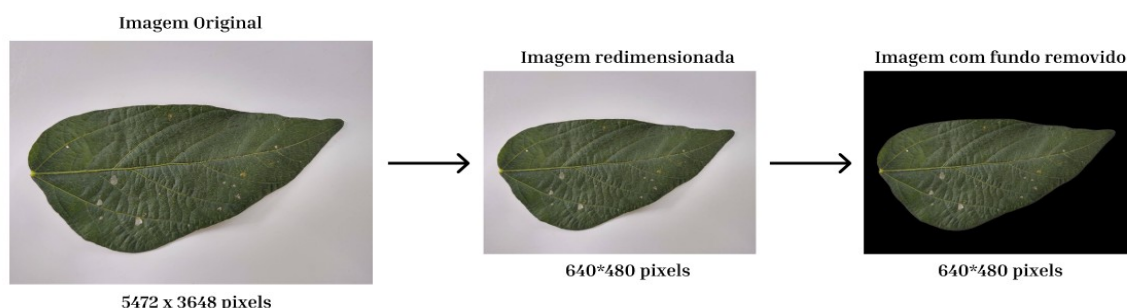


Figura 12 - tratamento realizado nas imagens.
Fonte: elaborado pelo autor

3.1.1 ROTULAGEM DAS AMOSTRAS

Após concluir as alterações nas imagens, iniciou-se a etapa de rotulagem. Este procedimento é conduzido manualmente, onde são inseridas etiquetas de coordenadas nas imagens, conhecidas como *bounding boxes* (caixas delimitadoras). Tais etiquetas fornecem à rede neural as coordenadas da localização sobre os objetos de interesse na imagem, permitindo que ela aprenda a identificar os padrões associados a esses objetos. Diversas ferramentas estão disponíveis para facilitar essa etapa, sendo o *Roboflow* escolhido para este trabalho devido a sua abrangência e facilidade de uso, possibilitando uma anotação eficiente e organização do banco de imagens.

Cada modelo de rede neural utiliza um formato de anotação específico para as *bounding boxes*. O formato de anotação utilizado para este trabalho, consiste em um arquivo de texto para cada imagem rotulada, onde cada linha representa uma etiqueta e contém as coordenadas (x, y, largura, altura) da caixa delimitadora, seguido pela classe do objeto e possíveis atributos adicionais, dependendo da aplicação específica (ULTRALYTICS, 2023).



Figura 13 - exemplo de anotação; caixa verde indica “folha da soja” e a vermelha indica “saudável”.
Fonte: elaborado pelo autor

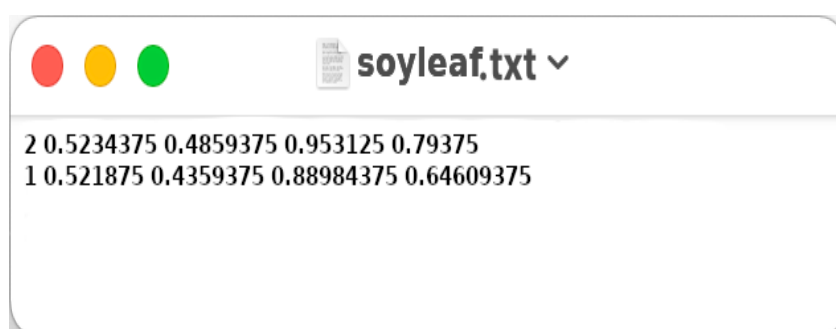


Figura 14 - arquivo de texto com as coordenadas das anotações.
Fonte: elaborado pelo autor

No total, foram anotadas 1586 imagens, distribuídas da seguinte forma: 200 para manchas olho-de-rã, 200 para manchas-alvo e 1186 para folhas saudáveis. Além dessas três categorias, foi criada uma categoria adicional para identificar se a imagem representa uma folha de soja, sendo esta aplicada a todas as imagens. Embora possa parecer incorreto o uso de um número muito maior de imagens de folhas saudáveis, é importante considerar o número de caixas delimitadoras por imagem. Nas imagens de folhas saudáveis, geralmente são necessárias apenas uma ou duas caixas delimitadoras, enquanto as imagens de manchas olho-de-rã e manchas-alvo requerem mais de 20 anotações em algumas situações.

Tabela 1 - quantidade de imagens anotadas x quantidade de bounding boxes

Categoria	Quantidade de imagens	Quant. bounding
		boxes
folha saudável	1186	1283
mancha-alvo	200	2898
mancha olho-de-rã	200	7649
folha da soja	586	1782
Total	1586	13612

3.1.2 DIVISÃO DO DATASET

Após todos esses passos, para o treinamento de um modelo de aprendizado de máquina, os dados são tipicamente divididos em conjuntos de treinamento e teste. O conjunto de treinamento é utilizado para ajustar os pesos do modelo, enquanto o conjunto de teste é reservado para avaliar a capacidade de generalização do modelo em dados não vistos anteriormente. É uma prática fundamental nunca utilizar os dados de teste durante o treinamento, a fim de evitar qualquer tipo de viés ou sobreajuste por parte do modelo. A avaliação do desempenho do modelo é realizada após cada época (iteração) de treinamento, utilizando métricas como precisão e erro. Para evitar a quebra dessa regra, é comum também utilizar um conjunto de validação para ajustar os parâmetros do modelo durante o treinamento. Após a conclusão do treinamento, o desempenho final do modelo é testado no conjunto de dados de teste para uma avaliação imparcial de sua eficácia (ELGENDY, 2020).

Normalmente os conjuntos são divididos em teste e treinamento em 66% e 33% ou 75% e 25% respectivamente (ROSEBROCK, 2017 tradução nossa), baseado nisso o dataset foi dividido em 70% treinamento, 20% validação e 10% para teste, essa uma divisão, resultando respectivamente em 1155, 305, 151 amostras imagens para cada conjunto da divisão.

3.1.3 AUGMENTATION (AUMENTO ARTIFICIAL)

Como dito anteriormente, quanto maior a quantidade de imagens disponíveis para treinamento, mais bem sucedido será a rede neural. Para isso a última etapa antes do avançar, é o *augmentation*, que é responsável por gerar um aumento artificial das imagens, a utilização de técnicas de aumento de dados pode desempenhar um papel fundamental na prevenção do *overfitting* (sobre ajuste), ao aumentar a diversidade dos dados de treinamento e permitir que o modelo generalize melhor para novos e dados não vistos. Além disso, o aumento de dados pode aprimorar a precisão e a robustez do modelo, ao expô-lo a uma variedade mais ampla de variações nos dados (FAIZAN, 2023, tradução nossa).

Para o aumento artificial dos dados, foi utilizado o Albumentations, uma biblioteca Python desenvolvida especificamente para essa finalidade. Além de realizar as modificações nas imagens, o Albumentations ajusta automaticamente a posição das caixas delimitadoras, garantindo a consistência dos dados. É importante destacar que esse processo é aplicado apenas ao conjunto de treinamento, preservando a integridade dos conjuntos de validação e

teste.

Ao todo, foram realizadas 9 iterações de aumento de dados, resultando em um total de 11.550 imagens, incluindo as originais e as modificadas. Durante esse processo, foram aplicadas diversas transformações, tais como inversão horizontal, variações de brilho e contraste, adição de ruído, rotação aleatória de 90° e rotação vertical. Essas iterações contribuíram significativamente para a diversidade e robustez do conjunto de dados, beneficiando o treinamento do modelo de detecção de doenças nas folhas de soja.

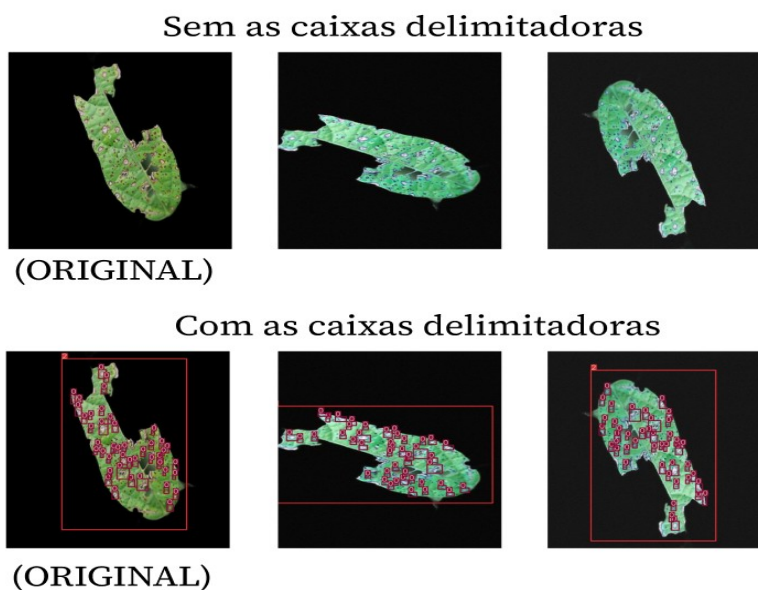


Figura 15 - aumento artificial utilizando o *Albumentations*
fonte: elabora pelo autor.

Acima é possível ver que a partir de uma mesma imagem original é possível gerar várias outras utilizando modificações. O lado bom de utilizar o *Albumentation* é que a ferramenta além de gerar as novas imagens, também gera as anotações das caixas delimitadoras.

3.2 TREINAMENTO

Inicialmente, a plataforma Google *Colab* foi empregada devido a sua facilidade por trazer um ambiente pronto para utilização, porém na versão gratuita há limitações com o tempo de uso e quantidade de memória disponível, fazendo com que outra estratégia tivesse que ser adotada. Por conseguinte, optou-se por utilizar um computador pessoal equipado com um processador Intel © Core™ i7-12700 CPU @ 2.10GHz x 12; memória de 8GB, DDR4, 3200MHz; SSD M.2. de 1TB; Placa gráfica NVIDIA R © GeForce © RTX 3070 TI de 8GB, GDDR5; Sistema operacional Windows 11 Pro.

Para viabilizar o treinamento, procedeu-se à configuração do ambiente, empregando-se a plataforma Anaconda e o *Jupyter Notebook* como ambiente de desenvolvimento e editor de código. Adicionalmente, foi necessário utilizar o *CUDA Toolkit*, uma suíte de ferramentas essencial para aproveitar ao máximo o poder de processamento das placas gráficas NVIDIA durante o treinamento de modelos de aprendizado profundo.

O treinamento foi feito utilizando a YOLOv8s, uma versão mais leve, que conta com 225 camadas e 11.137.148 parâmetros. Ao total foram utilizadas 200 *epochs*, com uma “paciência” de 100 *epochs* (*patience* serve para suspender a execução caso o algoritmo não encontre melhoras durante um determinado período de épocas no treino), e foi utilizado 16 *batch size* correspondente ao grupo de imagens selecionadas para o treinamento da rede, os resultados evidenciam que lotes pequenos asseguram estabilidade e generalização superiores durante o treinamento, com os tamanhos de lote mais eficazes geralmente variando entre 2 e 32 (Masters e Luschi, 2018).

O otimizador foi selecionado automaticamente pela rede neural, optando pelo método do Gradiente Descendente Estocástico (SGD) com uma taxa de aprendizado de 0.1. O SGD, um dos algoritmos mais amplamente empregados na atualidade, destaca-se por sua eficácia na atualização iterativa dos pesos da rede neural com base em gradientes calculados em amostras de dados de treinamento (Carvalho, 2021).

3.2.1 VALIDAÇÃO CRUZADA

A validação cruzada é uma técnica amplamente utilizada para avaliar o desempenho de modelos em conjuntos de dados limitados. Ela consiste na divisão dos dados em subconjuntos mutuamente exclusivos, onde um subconjunto é utilizado para treinamento do modelo e os demais são reservados para validação. Esse processo é repetido várias vezes, com diferentes combinações de dados de treinamento e validação, permitindo uma avaliação mais abrangente do desempenho do modelo. A validação cruzada é particularmente útil para evitar o sobreajuste (*overfitting*) e fornecer estimativas mais confiáveis da capacidade de generalização do modelo para novos dados.

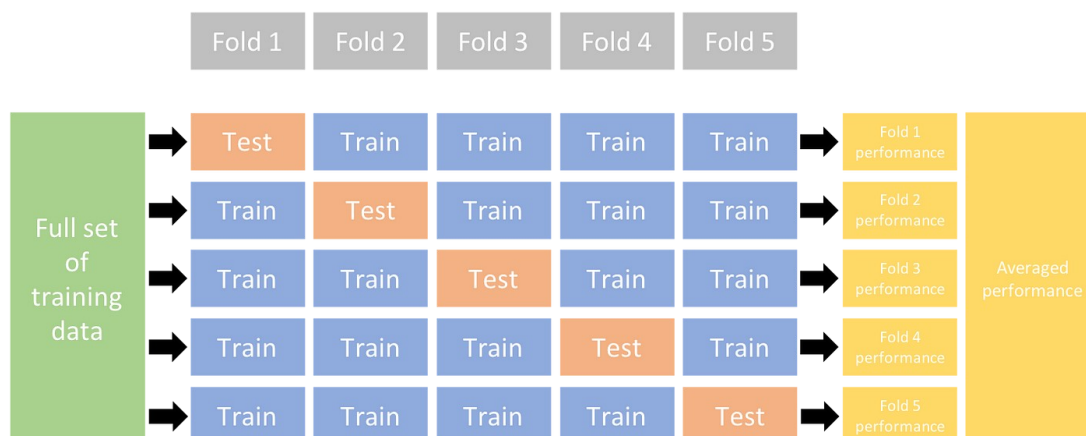


Figura 16: Validação cruzada com 5-fold

Disponível em: <https://docs.ultralytics.com/pt/guides/kfold-cross-validation/#introduction>

Como ilustra a figura acima, os dados para o trabalho também foram divididos em 5-*fold*, o que indica que foram feitos 5 subconjuntos ao total, alternando aleatoriamente entre o conjunto que seria usado para validação e os conjuntos que seriam utilizado para treinamento, visando garantir a confiabilidade e consistência do desempenho do modelo em diversos conjuntos de dados. Isso resulta em um modelo mais generalizável e confiável, com menor propensão a se ajustar excessivamente a padrões específicos dos dados (Ultralytics, 2023).

4 RESULTADOS

Nesta seção serão mostrados os resultados obtidos, trazendo medidas de desempenho, juntamente com o custo computacional mensurado pela medida de tempo de cada treinamento. Ressaltando que apenas a melhor *epoch* de cada treinamento é considerada.

Com base nas etapas delineadas na metodologia, foi identificado que a vigésima quarta época apresentou o melhor desempenho, seguido de uma diminuição gradual até que o modelo atingiu sua estabilidade e alcançou o limite de tolerância, encerrando-se na época 179, levando um total de 5 horas e 5 minutos, os resultados podem ser avaliados nos gráficos e na tabela abaixo:

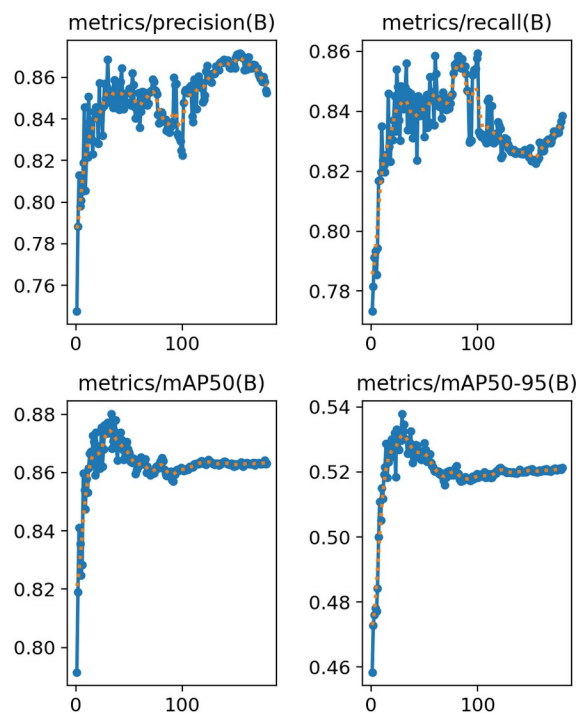


Figura 17: gráfico com mAP50, mAP50:95, precisão e recall obtidos durante o treinamento.

Fonte: elaborado pelo próprio autor.

O gráfico mostra uma rápida ascensão e após isso uma normalização, não ocorrendo grande variação entre as épocas. As maiores variações foram encontradas no recall e na precisão, sendo que a primeira entrou em uma leve queda após a época 100, porém mostra recuperação ao final. A precisão também apresentou uma variação e atingiu um pico de aproximadamente 89% por volta da época 150. Porém essas métricas não podem ser analisadas isoladamente, uma vez que elas fazem parte do cálculo do F1-score.

As métricas mAP50 e mAP50:95 mostraram pouquíssimas variações, após atingirem seu ápice na época 24, houve uma leve queda e então um estabilidade. Os dados obtidos da melhor *epoch* também podem ser analisados na tabela abaixo:

Categoria	Precisão	Recall	mAP50	MAP50:95	F1-score
Folha saudável	0.924	0.972	0.981	0.753	0.947
Mancha-alvo	0.717	0.682	0.714	0.283	0.699
Mancha olho-de-rã	0.821	0.739	0.819	0.369	0.778
Folha da soja	0.96	0.983	0.99	0.747	0.971
Total	0.856	0.844	0.876	0.538	0.85

Tabela 2: métricas obtidas a partir da validação da melhor epoch.

Por hora será levado em consideração principalmente as métricas F1-score e mAP, sendo que com ambas dá para se ter uma visão geral do desempenho encontrado. Como dito anteriormente, F1-score é uma métrica harmônica entre precisão e recall, indicando que o modelo atingiu uma média de 85% no total.

É possível observar que as categorias mancha-alvo e mancha olho-de-rã atingiram métricas inferiores quando comparadas com as outras categorias, em especial a mAP50:95, isso se deve a principal característica de passada única do YOLO, “embora possa identificar rapidamente objetos em imagens, ele tem dificuldade em localizar com precisão alguns objetos, especialmente os pequenos.” (Redmon et al, 2016, tradução nossa), e uma vez que as caixas delimitadoras dessas doenças são pequenas, esse fator acaba ficando evidente. Apesar dessa limitação, o modelo treinado ainda identifica as pragas investigadas corretamente, mesmo apresentando um baixo limiar de confiança, abaixo pode ser analisada uma demonstração do modelo treinado identificando uma folha com a mancha olho-de-rã.

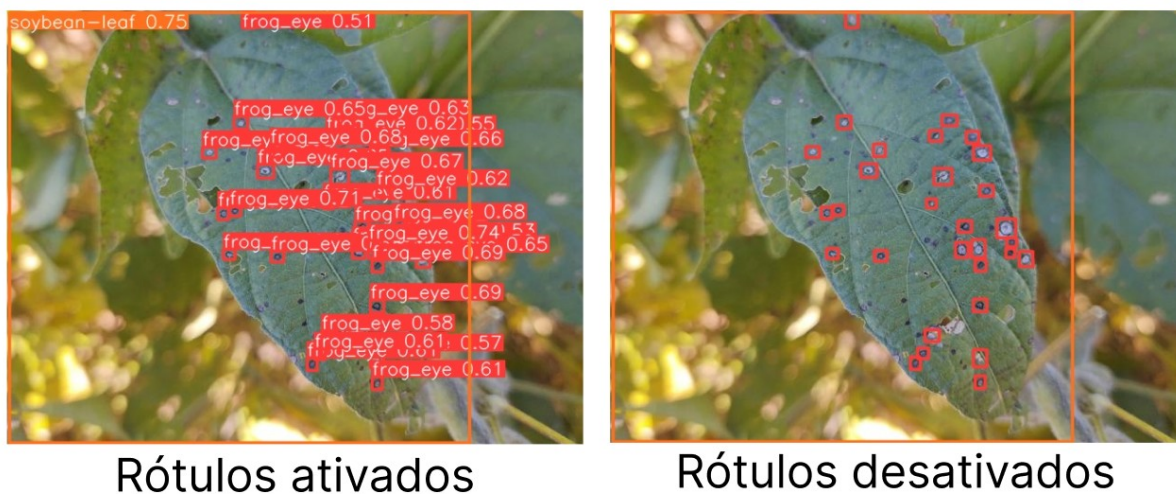


Figura 18: Mostra do modelo detectando a mancha olho-de-rã (categoria que obteve as piores métricas).

Fonte: elabora pelo autor

TEXTO EXPLICANDO A IMAGEM ACIMA.

5 APÊNDICE

Entende-se como “Apêndice” tudo aquilo que foi elaborado pelo pesquisador. Por exemplo, questionários criados pelo pesquisador, roteiro de entrevistas, etc.

Elemento opcional, devendo ser apresentado em conformidade com as orientações da ABNT - NBR 14724, ou substituta, desde que vigente no período em que o trabalho estiver sendo realizado.

6 ANEXO

Entende-se como “Anexo” tudo o que não foi elaborado pelo autor. Você pode anexar qualquer tipo de material ilustrativo, tais como tabelas, lista de abreviações, documentos ou parte de documentos, resultados de pesquisas, etc.

Elemento opcional, devendo ser apresentado em conformidade com as orientações da ABNT - NBR 14724, ou substituta, desde que vigente no período em que o trabalho estiver sendo realizado.

