Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implementação de Técnicas de Atenção e Fusão em Redes Neurais Convolucionais para a Melhoria da Detecção e Segmentação de Doenças em Folhas de Café

Gabriel Henrique Souza de Melo

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Orientador
Prof. Dr. Díbio Leandro Borges

Brasília
2024

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Implementação de Técnicas de Atenção e Fusão em Redes Neurais Convolucionais para a Melhoria da Detecção e Segmentação de Doenças em Folhas de Café

Gabriel Henrique Souza de Melo

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Prof. Dr. Díbio Leandro Borges (Orientador)
CIC/UnB

Prof. Dr. Wilson Henrique Veneziano
CIC/UnB

Profa. Dra. Roberta Barbosa Oliveira
CIC/UnB

Prof. Dr. João Luiz Azevedo de Carvalho
Coordenador do Curso de Engenharia de Computação

Brasília, 10 de setembro de 2024

Dedicatória

Dedico este trabalho aos meus pais, por me apoiarem em todos os momentos importantes da minha vida. A minha namorada Natane Araujo, que é parte do Instituto de Ciências Sociais e se dedica à vida acadêmica na Universidade de Brasília e ao meu professor orientador, Díbio Borges, do Departamento de Ciências da Computação, além de todos os indivíduos que trabalham na área de computação e que foram e são importantes para a produção de conhecimento.

Agradecimentos

Gostaria de agradecer primeiramente aos meus pais, Carlos e Elida, pelo esforço e dedicação que tiveram sob as dificuldades da vida para proporcionar uma educação de qualidade aos três filhos. A minha namorada, que está ao meu lado desde o início desse percurso e me incentivou e encorajou até aqui e que, com os seus valores, orgulho e apreço pela vida acadêmica, restaurou diariamente minhas forças para continuar trilhando o meu caminho dentro da universidade.

Ainda nessa fase, bem como em várias outras, é pertinente agradecer aos amigos de décadas que puderam estar presentes e auxiliar nas dificuldades, mas também participarem dos bons momentos de descanso e alegria nesses anos de graduação. Aos colegas que fiz dentro do curso, que dividiram comigo trabalhos e desafios. Àqueles que me incentivaram, inclusive na produção deste trabalho, deixo também minha imensa gratidão.

Sou grato, ainda, a todos que contribuíram direta ou indiretamente com essa produção. Por fim, deixo meus agradecimentos ao corpo docente da universidade, que em sua excelência ajudou a mim e tantos outros discentes na conclusão dessa etapa importantíssima. Agradeço especialmente ao meu orientador, Díbio Borges, que me guiou e me instruiu durante todo o processo de pesquisa, aprimoramento e conclusão deste projeto

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Desenvolvimentos recentes em visão computacional, especialmente com técnicas de aprendizado profundo, têm promovido uma grande mudança na agricultura para o monitoramento de cultivos, a fim de identificar estresses bióticos e abióticos em plantas individuais. As plantações de café enfrentam grandes perdas de produção devido a pragas e patógenos. A identificação dessas doenças é feita principalmente com base em sintomas visuais que aparecem nas folhas do cafeeiro. Por meio de processamento computacional, as tarefas de identificação de doenças podem se tornar mais precisas utilizando dados visuais obtidos por câmeras de smartphones, permitindo que agrônomos tomem decisões mais sustentáveis e eficientes no manejo de doenças. Esta pesquisa aborda a detecção e segmentação das principais doenças nas plantações de café. São abordados os modelos de aprendizado profundo de última geração YOLACT++ e YOLOv8n, ambos para detecção e segmentação, e foi adicionado um módulo de atenção CBAM ao YOLACT++, e também um módulo de fusão C2f2 ao YOLOv8n. Todos os testes foram realizados utilizando os conjuntos de dados BRACOL e RoCoLe disponíveis publicamente, com imagens de folhas de café de laboratório e de campo, com suas respectivas anotações. Os resultados mostram que para a detecção de objetos, em ambos os conjuntos de dados, BRACOL e RoCoLe, o modelo YOLOv8n foi superior, atingindo uma precisão de 87,2% e 36,8%, respectivamente. Os resultados de segmentação de instância mostram que nosso módulo C2f2 adicionado ao YOLOv8n provou ser melhor, alcançando 72,5% para BRACOL e 45,4% de precisão para RoCoLe.

Palavras-chave: aprendizagem de máquina; aprendizagem profunda; visão computacional; segmentação de imagem; detecção de objetos; doenças do café.

Abstract

Recent developments in computer vision, especially with deep learning techniques, have promoted a big change in agriculture for monitoring crops to identify biotic and abiotic stresses in individual plants. Coffee crops face major losses in production due to pests and pathogens. The identification of these diseases is mainly made based on visual symptoms appearing in the coffee leaves. Through computational processing, the diseases identification tasks can become more precise using visual data acquired by smartphone cameras, allowing agronomists to make more sustainable and efficient decisions on how to manage the diseases. This research addresses detection and segmentation of major diseases in coffee crops. State-of-the-art deep learning models YOLACT++ and YOLOv8n were both utilized for detection and segmentation, with the addition of an attention CBAM module to YOLACT++, and also a fusion C2f2 module to YOLOv8n. All tests were done using the publicly available datasets BRACOL and RoCoLe, with coffee leaf lab and field images annotated. The results show that for object detection, in both datasets, BRACOL and RoCoLe, YOLOv8n model was superior achieving precision of 87.2%, and 36.8%, respectively. Instance segmentation results show that the added C2f2 module to YOLOv8n proved better achieving 72.5% for BRACOL, and 45.4% of precision for RoCoLe.

Keywords: machine learning; deep learning; computer vision; image segmentation; object detection; coffee disease.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A formação e a evolução das civilizações se tornou possível, ao longo da história, devido à agricultura que contribuiu para o fim do nomadismo e começo da fixação da população humana. Sob este contexto, entende-se a relevância do setor de cultivo no desenvolvimento humano, uma vez que, desde a Pré-história, este se tornou imprescindível para a sustentação e consumo da vida na cidade e no campo.

No Brasil, cerca de 7.6% do território nacional é composto de áreas agrícolas[1], isso faz do país o quinto maior em área cultivada no mundo (MIRANDA, 2018). Com sua importante extensão, o país verde-amarelo se destaca, principalmente, na produção de café e cana-de-açúcar, além de outros cultivos, como: soja, algodão, milho, tabaco e cacau.

A priori, cabe destacar a considerável dependência econômica do Brasil em relação ao agronegócio, uma vez que este pode responder por cerca de 24,4% do PIB do país (2023) podendo alcançar até R$ 2,63 trilhões, segundo estudos da Cepea (Centro de Estudos Avançados em Economia Aplicada), da Esalq/USP, junto da CNA (Confederação da Agricultura e Pecuária do Brasil)[2]. Logo, compreende-se que perdas mínimas no setor geram uma significativa perda econômica para o país, que mantém cerca de 200 milhões de cidadãos alimentados por ano. Além do próprio país, o agronegócio brasileiro alimenta, no mundo, cerca de 500 milhões de indivíduos (CONTINI; ARAGÃO, 2021), dado que corrobora, mais uma vez, a sua notável relevância para o desenvolvimento financeiro e civil no mundo.

Tendo em vista o supramencionado, os produtores querem, cada vez mais, otimizar suas produções e evitar perdas mínimas. Fatores bióticos e abióticos que geram estresse nas plantações têm ao longo do tempo ocasionado uma perda negativa para a economia e riscos à segurança alimentar global (STRANGE; SCOTT, 2005).

[1]Fonte Embrapa, disponível em: <https://www.embrapa.br/car/sintese>

[2]Fonte Cepea, disponível em: <https://www.cepea.esalq.usp.br/br/pib-do-agronegocio-brasileiro.aspx>

Para prevenir tais perdas na cultura, há a necessidade de uma hábil fenotipagem de plantas, entretanto, essa atividade, quando exercida pelo ser humano, tende a ser exaustiva e mais suscetível a erros. Hodiernamente, o avanço tecnológico possibilita a execução dessa laboração de forma automatizada, o *machine learning* (aprendizagem de máquina) e o *deep learning* (aprendizagem profunda) são métodos que vêm evoluindo e mostrando um desenvolvimento promissor. Essas tecnologias são capazes de processar e analisar uma vasta quantidade de dados, nesse caso, expressos em imagens do plantio, visando encontrar padrões e associações entre dados, podendo, então, classificar resultados que auxiliem na prevenção de perdas significativas de culturas (SINGH et al., 2021).

A classificação, detecção de objetos e segmentação de instâncias são alguns dos processos significativos em visão computacional e são altamente utilizados em sistemas de fenotipagem automatizada. A classificação é o processo de atribuição de uma classe a uma imagem ou a um objeto específico na imagem, identificando, por exemplo, se uma planta está saudável ou doente. O método de detecção de objetos leva as coisas um pouco mais longe, localizando e identificando mais de um objeto em uma imagem usando caixas delimitadoras, que são retângulos desenhados ao redor de objetos, fornecendo assim não apenas uma classificação, mas também as localizações para onde os objetos são encontrados dentro de um campo de visão. O método de segmentação de instâncias, por outro lado, é muito mais avançado, e em conjunto com a detecção e classificação de objetos, mapeia corretamente o contorno de cada instância e segmenta pixel por pixel, gerando, por fim, as máscaras de segmentação. Isso ajuda a distinguir múltiplos objetos da mesma classe em uma imagem, como folhas ou sintomas, enquanto fornece uma análise detalhada e personalizada de cada elemento, o que é importante para a fenotipagem eficiente e prevenção de perdas.

Outrossim, para que esses resultados alcancem seus objetivos futuros, bem como a implementação das soluções supramencionadas e a redução no tempo de realização dessas tarefas — ambas em condições de campo — é importante que a identificação das doenças nas folhas, após suas respectivas e corretas segmentações, sejam específicas e precisas, tanto para imagens de campo quanto para imagens de laboratório.

1.1 Hipótese

A ideia central deste estudo é avaliar dois modelos de aprendizagem profunda de detecção em estágio único —– YOLACT++ (BOLYA et al., 2020) e YOLOv8n (JOCHER; CHAURASIA; QIU, 2023) —, nas tarefas de detecção de objetos e segmentação de instâncias quando treinados em dois grandes bancos de dados públicos — Bracol (ESGARIO; KROHLING; VENTURA, 2020) e RoCole (PARRAGA-ALAVA et al., 2019) — de do-

enças em folhas de café. Ademais, foram propostas alterações de atenção e fusão para aprimorar esses modelos. A hipótese é que essas mudanças não apenas melhorarão as métricas padrão de desempenho, como mAP, precisão e revocação, mas também reduzirão o tempo de inferência, tornando os modelos mais eficientes e aplicáveis em condições de campo.

A técnica chamada atenção, que será adicionada a uma das arquiteturas (YOLACT++), é uma maneira de o modelo se concentrar em partes específicas da entrada, em vez de processar todos os elementos de forma igual. O mecanismo de atenção foca nesses trechos importantes que o modelo deve *"prestar atenção"*, permitindo maior precisão e eficiência no processamento. No caso da visão computacional, a atenção pode ser usada para destacar objetos ou detalhes específicos. Este trabalho, especificamente, utilizará um módulo já existente que combina métodos de atenção espacial e atenção de canal, isto é, indicam quais são as áreas e as características que o modelo deve focar.

Fusão refere-se ao processo de combinar diferentes tipos de informações ou características dentro de uma rede neural. Nas redes neurais convolucionais (CNNs), a fusão combina informações de diferentes camadas da rede, cada uma capturando detalhes em diferentes níveis de abstração, como bordas, texturas e formas. Essa combinação é essencial para melhorar a capacidade do modelo de identificar e classificar objetos que podem variar em tamanho, forma ou posição dentro de uma imagem. A fusão ajuda a criar uma visão mais completa e precisa dos dados, garantindo que informações de várias fontes sejam usadas para melhorar a tomada de decisão, gerando melhores informações semânticas. Esse método já é amplamente utilizado entre as arquiteturas de detecção de objetos. Neste estudo, a mudança no modelo YOLOV8n concentra-se justamente na alteração de uma de suas camadas internas, que já trabalha com esses objetivos, com o intuito de melhorá-la de acordo com as necessidades exigidas.

1.2 Objetivo Geral

Avaliar se as versões modificadas dos modelos YOLACT++ e YOLOv8n resultam em melhorias significativas nas métricas de desempenho em relação às suas versões originais. Espera-se que as modificações nos modelos aumentem a eficácia na identificação de doenças, promovendo avanços que podem otimizar a produção agrícola e minimizar perdas.

1.3 Objetivo Específico

Neste estudo, os objetivos específicos são: (1) realizar uma comparação detalhada entre os modelos YOLACT++, YOLOv8n e suas respectivas versões alteradas, analisando seus desempenhos e como isto impacta nas escalabilidades e aplicabilidades; e (2) avaliar o impacto da variação do banco de dados na performance dos modelos treinados, buscando entender como a diversidade e a forma em que estes dados são obtidos influenciam nas métricas e na generalização dos modelos.

1.4 Organização do Trabalho

Este trabalho está estruturado em diversos capítulos, cada um com uma função específica para garantir uma abordagem organizada e compreensível do tema. A introdução contextualiza a importância do setor agrícola no Brasil e a relevância do estudo na detecção e segmentação de doenças em folhas de café utilizando técnicas de aprendizado profundo. No Capítulo 2 (Fundamentação Teórica), detalha-se os modelos YOLACT++ e YOLOv8n, destacando suas principais funcionalidades e vantagens, além de explicitar as modificações implementadas em cada um desses modelos. Em seguida, no Capítulo 3 (Materiais e Métodos) , são apresentados os bancos de dados, seus principais aspectos e a forma como foram construídos. Também expõe e explica as métricas que serão utilizadas no experimento, para que sejam feitas as análises no capítulo seguinte. Além disso, explica a metodologia aplicada para que o leitor possa acompanhar todas as etapas do experimento. O Capítulo 4 (Resultados e Discussões) apresenta os testes realizados e os resultados obtidos, destacando métricas como mAP, precisão e revocação, e comparando versões modificadas e originais dos modelos. Estes resultados são analisados, considerando melhorias, limitações e a influência dos bancos de dados no desempenho dos modelos. Por fim, o Capítulo 5 (Conclusão) resume as principais descobertas, reafirma a eficácia dos modelos propostos e sugere direções futuras para pesquisas na área.

Capítulo 2

Fundamentação Teórica

Neste capítulo, é detalhada a fundamentação teórica utilizada no estudo. Este abrange a descrição dos modelos de aprendizado profundo aplicados (YOLACT++ e YOLOv8n), às modificações introduzidas (como CBAM e C2f2). O capítulo é estruturado por meio de explicações detalhadas de cada tecnologia utilizada e de como elas se inter-relacionam até que cheguem aos resultados finais.

2.1 YOLACT++

YOLACT (**Y**ou **O**nly **L**ook **A**t **C**oefficien**T**s) (BOLYA et al., 2019) é um modelo que busca uma melhora no processo de segmentação de instâncias em tempo real. Seu principal objetivo é adicionar um processo de criação de máscaras aos objetos detectados, assim como o Mask R-CNN (HE et al., 2017), porém sem uma etapa de *repooling*.

Sua proposta, exemplificada pela Figura 2.1, é dividir uma tarefa de segmentação de instâncias em dois processos mais simples:

Máscaras de protótipo: Gera um conjunto de 'máscaras de protótipo' com o tamanho da imagem representando formas gerais, sem qualquer instância especificada. Elas são feitas por uma Rede Totalmente Convolocional (FCN) (LONG; SHELHAMER; DARRELL, 2015).

Coeficientes de máscaras: Predizer um vetor de "coeficientes de máscaras" para cada caixa de âncora (região de interesse na imagem) na etapa de detecção de objetos. Esse vetor codifica como cada máscara protótipo contribui para uma máscara de uma instância específica.

Após concluir as duas etapas mencionadas, inicia-se o processo que dá origem às máscaras finais. Nesse estágio, para cada instância que resta após a *Non Maximum Suppression* (NMS), as máscaras protótipos são combinadas linearmente, utilizando os respectivos coeficientes de máscara.

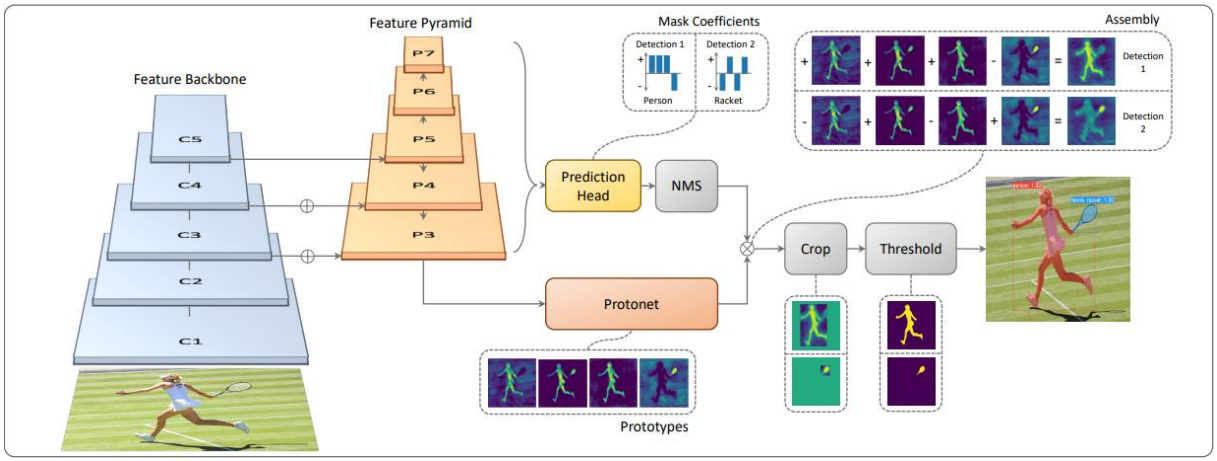


Figura 2.1: **Arquitetura YOLACT** Azul/amarelo indica valores baixos/altos nos protótipos, os nós cinzas indicam funções que não são treinadas e $k = 4$ neste exemplo, Arquitetura baseada em RetineNet usando Resnet-101 + FPN (Fonte: (BOLYA et al., 2020)).

Desta forma, o modelo aproveita os pontos fortes de ambas as camadas convolucionais e totalmente conectadas, levando em conta que protótipos e coeficientes podem ser calculados de forma independente. Sabendo que camadas convolucionais são boas em produzir máscaras espacialmente coerentes (máscaras protótipos), enquanto camadas totalmente conectadas são eficientes em gerar vetores semânticos (coeficientes de máscara).

O YOLACT++ (BOLYA et al., 2020) é uma extensão do modelo YOLACT, apresentando 3 principais melhorias:

Deformable Convolutions: Permite uma melhor adaptação do modelo à forma dos objetos, gerando uma precisão melhor da detecção.

Otimização do *Prediction Head*: O *Prediction Head* é responsável pelas previsões do modelo, como dimensões e coordenadas de cada objeto detectado.

Nova etapa de *Mask Scoring*: Melhoria significativa na velocidade da etapa de geração das máscaras dos objetos detectados.

2.1.1 Anotações de Imagens em YOLACT++

É importante salientar que o modelo YOLACT++ usa o padrão de anotação de imagens Microsoft Common Objects in Context (COCO) (LIN et al., 2014), que é estruturado da seguinte maneira:

```
1 // Padrao COCO
2 {
3     "info": info,
4     "licenses": [license],
```

```
    "images": [image],  // Lista de todas as imagens do dataset
    "annotations": [annotation],  // Lista de todas as anotacoes do
        dataset
    "categories": [category]  // lista de todas as categorias
}
```

Neste padrão, as anotações são armazenadas em um único arquivo JSON, que contém informações detalhadas sobre todas as imagens, bem como detalhes das anotações, tais como bounding boxes, segmentos, categorias de objetos, entre outros.

2.2 Convolutional Block Attention Model (CBAM)

O módulo de atenção CBAM (Convolutional Block Attention Module) (WOO et al., 2018) é uma arquitetura de atenção que pode ser usado em redes neurais convolucionais para melhorar a precisão da detecção de objetos, como mostrado na Figura 2.2.

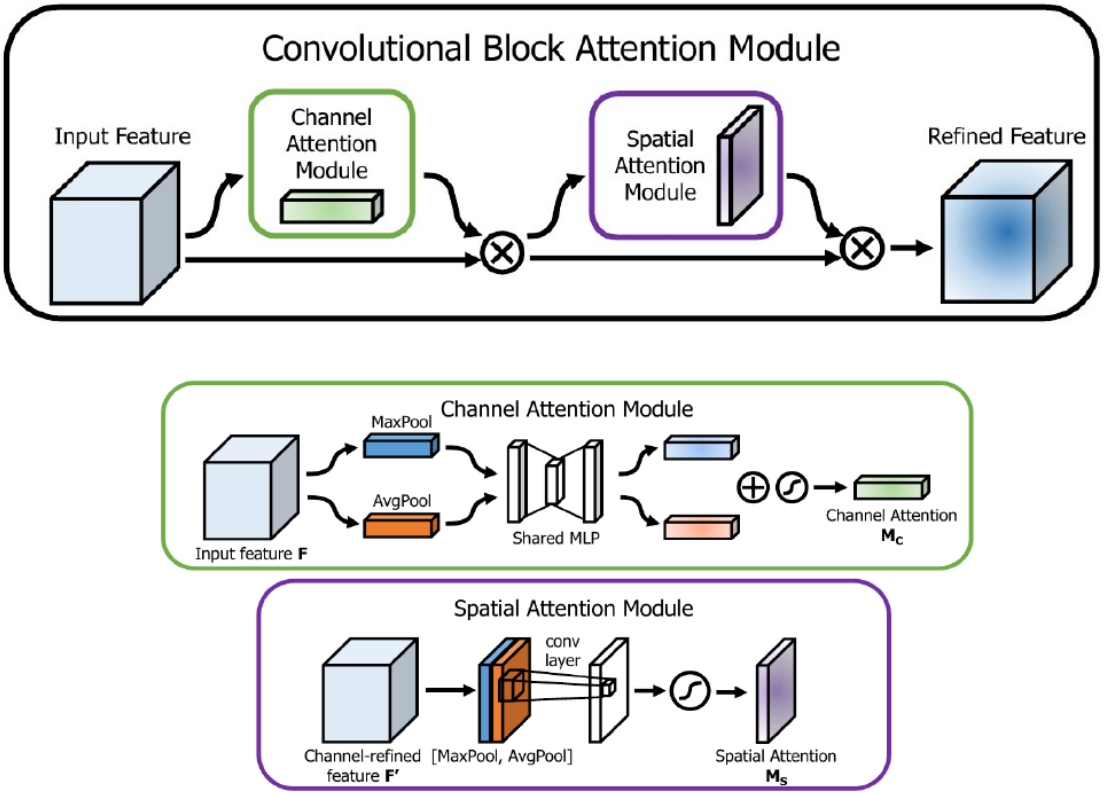


Figura 2.2: Visão geral do CBAM e seus módulos (Fonte: (WOO et al., 2018)).

Sua arquitetura é constituída de dois blocos:

Módulo de atenção de canal: é responsável por aprender a importância dos diferentes canais da imagem. Ou 'o que' é significativo dada uma imagem de entrada.

Este canal usa recursos de *average-pooling* e *max-pooling* simultaneamente, o que resulta em uma representação significativamente mais rica e informativa.

Módulo de atenção espacial: é responsável por aprender a importância das diferentes regiões da imagem. Complementar à atenção de canal, foca no 'onde' é a parte informativa. Primeiro aplicam-se operações de *average-pooling* e *max-pooling* ao longo do eixo do canal e as concatena para gerar um descritor de recurso eficiente.

2.3 Attention YOLACT++

Foi elaborada uma alteração no algoritmo original do YOLACT++ para que houvesse uma integração do modelo com o módulo de atenção CBAM. O arquivo *backbone.py* foi alterado para que no momento correto fosse chamada uma instância da classe 'CBAM' presente no arquivo *cbam.py*.

Esta abordagem foi baseada no artigo de Huang et al.(2021), que propôs a integração do módulo CBAM com a arquitetura YOLACT++ com o objetivo de segmentar doenças em plantações de milho. Para isso, foram usadas imagens obtidas em campo, as quais possuem um fundo mais complexo. Comparações foram feitas entre o modelo Attention YOLACT++, o modelo base YOLACT++ e um modelo Mask R-CNN, todos aplicados na mesma base de dados. A adição do módulo de atenção destacou-se em relação a essas outras arquiteturas nas métricas de desempenho, como a precisão. No entanto, observou-se um ligeiro aumento no tempo de inferência em comparação com o modelo original, devido à adição de uma nova etapa no processo.

Com base nos resultados obtidos no artigo citado, espera-se que a utilização da mesma abordagem traga benefícios semelhantes nas métricas de avaliação, apesar do aumento no tempo de inferência, visto que tanto este trabalho quanto o artigo em questão têm objetivos semelhantes. Acredita-se que os resultados positivos se evidenciem mais nos treinamentos realizados com o banco de dados mais semelhante ao do estudo, ou seja, aquele que contém imagens obtidas em campo.

O CBAM está equipado na saída do backbone ResNet-101 e na saída da *Feature Pyramid Network* (FPN). A Figura 2.3 representa a arquitetura com integração do módulo de atenção CBAM.

2.4 YOLOv8

YOLOv8 (JOCHER; CHAURASIA; QIU, 2023) continua com a tradição dos modelos YOLO (You Only Look Once), que são modelos usados para a detecção de objetos em imagens e vídeos. Sua maior vantagem é a velocidade de detecção de objetos sem com-

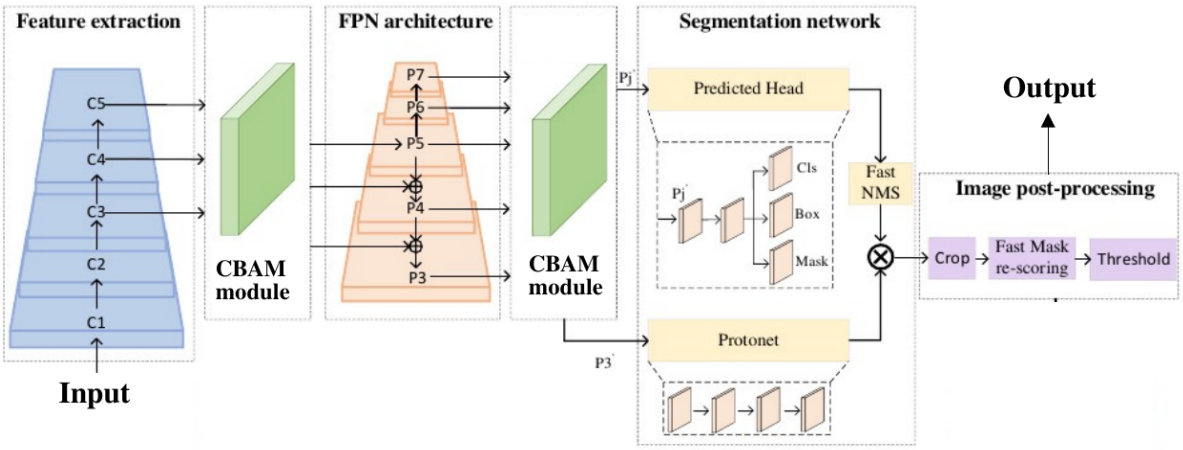


Figura 2.3: Modelo do Attention Yolact++ (Fonte: (HUANG et al., 2021)).

prometer a precisão de tal tarefa. Isso significa que ele pode analisar uma imagem inteira de uma só vez e identificar a presença, a localização e a classe dos objetos em uma única passagem pela rede neural.

Uma das inovações importantes no YOLOv8, quando comparado com seu modelo predecessor, é o método *"Anchor-Free"*. Anteriormente, os modelos de detecção de objetos usavam *anchors*, que nada mais são que regiões pré-definidas na imagem onde a rede neural esperava encontrar objetos. Esses *anchors* possuem proporções fixadas, assim como seus tamanhos. A rede neural realiza ajustes para que essas caixas se encaixem nos objetos detectados. Já o modelo "Anchor-Free" elimina toda essa etapa. Em vez de depender dessas caixas pré-definidas, o YOLOv8 detecta objetos diretamente, sem a necessidade de *anchors*. Fazendo a localização dos pontos centrais dos objetos e assim prevê o tamanho das caixas delimitadoras a partir desses pontos.

O método *"Anchor-Free"* traz algumas melhorias significativas, como por exemplo a flexibilidade do modelo em detectar objetos de diferentes tamanhos, agora que a etapa de detecção não se limita a caixas pré-definidas. Além disso, a eliminação de *anchors* reduz a complexidade e os recursos computacionais necessários na detecção dos objetos, resultando em um modelo mais rápido e mais preciso.

Outra melhoria notável no YOLOv8 é a adição da camada C2f. A principal função da camada C2f é combinar características de alto nível com informações contextuais, ou seja, estabelece conexões entre diferentes estágios da rede neural. Isso permite que informações de camadas mais profundas sejam combinadas com informações de camadas mais superficiais. Ao combinar características de diferentes níveis, a camada C2f ajuda o modelo a fazer previsões mais precisas, especialmente para objetos de diferentes tamanhos, formas e em diferentes contextos. Esta camada geralmente envolve duas convoluções,

seguidas por uma concatenação. Isso ajuda a capturar diferentes níveis de abstração na imagem, o que a torna essencial para a detecção precisa de objetos de vários tamanhos.

2.4.1 Anotações de Imagens em YOLOv8

O modelo YOLOv8 utiliza um padrão específico de anotação de imagens para treinar seus algoritmos. As anotações seguem o formato do YOLO, que é bastante compacto.

Neste padrão, cada imagem possui um arquivo de anotação correspondente em formato .txt com o mesmo nome da imagem. Cada linha do arquivo representa um objeto detectado na imagem e segue o seguinte formato:

```
// Padrao YOLO
<class_id> <x1> <y1> <x2> <y2> ... <xn> <yn>
```

Onde, <x1>, <y1>, <x2>, <y2>, ..., <xn>, <yn> são as coordenadas delimitadoras da máscara de segmentação do objeto.

2.5 YOLOv8n + C2f2

Neste estudo, foi proposto utilizar a estrutura da rede YOLOv8n com algumas modificações na camada C2f, tornando-a mais leve e melhorando o desempenho de precisão, especialmente para objetos pequenos.

Este método foi proposto em um estudo realizado pelo aluno Bruno Pinheiro de Melo Lima, da Universidade de Brasília, departamento de engenharia mecânica (LIMA et al., 2024). Seu objetivo era a detecção de percevejos em plantações de soja. Pelo fato deste inseto ser muito pequeno e difíceis de identificar em imagens de campo, esta alteração proposta visa melhorar a detecção de objetos pequenos pelo modelo e, consequentemente, diminuir o custo operacional.

Essa nova camada será chamada de C2f2. A C2f2 é, na verdade, um procedimento de fusão, ajudando a rede a processar informações em diferentes escalas de forma mais eficiente. Essa ideia, embora com designs diferentes, tem sido explorada em arquiteturas YOLO aprimoradas (LIU et al., 2022; LOU et al., 2023; LIU et al., 2023). A Figura 2.4 mostra a arquitetura da proposta YOLOv8n + c2f2, e a Figura 2.5 dá detalhes dos filtros e dá sequência da camada C2f2.

C2f2 e C2f são arquiteturas de rede semelhantes com uma diferença chave na complexidade de suas unidades de processamento internas, chamadas blocos *bottleneck*. Embora compartilhem a estrutura geral, a C2f2 possui uma ligeira alteração em seu design. Ambas as arquiteturas dependem de blocos *bottleneck* CSP, um bloco de construção comum em redes neurais convolucionais. Esses blocos geralmente envolvem uma sequência de três

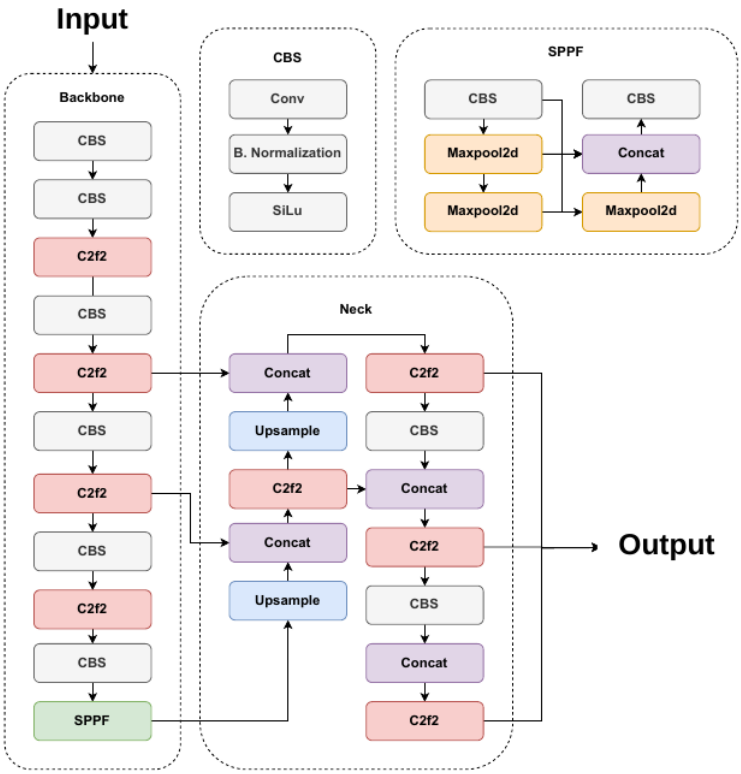


Figura 2.4: YOLOv8n proposto com uma fusão com a camada C2f2.

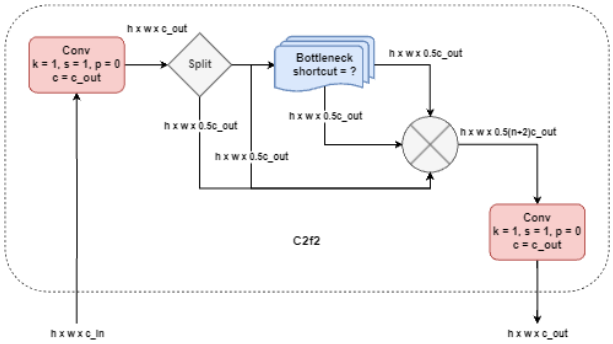


Figura 2.5: Detalhes da camada C2f2 proposta.

camadas de convolução: uma convolução 1x1 seguida por uma convolução *depthwise* (profunda) 3x3, e concluindo com outra convolução 1x1. Os dados de entrada são divididos em duas metades. A primeira metade passa pela convolução 1x1 inicial. Em seguida, é dividida novamente para processamento adicional.

A outra metade é alimentada nos módulos bottleneck. Esses módulos são, essencialmente, camadas de convolução duplas, com a vantagem de serem separáveis. Esses módulos também podem opcionalmente incluir uma conexão de atalho, permitindo que a

informação flua diretamente sem modificação.

Finalmente, as saídas de ambas as metades são mescladas usando concatenação. Esses dados combinados, então, passam por uma convolução 1x1 final, para gerar a saída do bloco. A diferença crítica reside no parâmetro n, que dita o número de módulos bottleneck dentro de um único bloco. No cenário descrito, n é definido como 1, indicando que a C2f2 emprega apenas um conjunto de convoluções separáveis.

No próximo capítulo, serão abordados os materiais e métodos utilizados nesta pesquisa, detalhando os bancos de dados que serão usados para realizar os treinamentos e testes dos modelos apresentados neste capítulo, além dos procedimentos de coleta e anotação de imagens, essenciais para a validação destes modelos. Então, serão apresentadas e explicadas as métricas que serão usadas para avaliar os treinamentos. Ao final, é explicado todo o passo a passo para que seja feito o treinamento dos quatro modelos, para que se obtenha os resultados a serem analisados.

Capítulo 3

Materiais e Métodos

Nesta seção, são apresentados os materiais utilizados, destacando-se os bancos de dados (BRACOL e RoCoLe). Também são explicados os procedimentos para a coleta e anotações das imagens e a distribuição dos dados, preparando o leitor para fatores que podem influenciar os resultados do capítulo seguinte. Em seguida, são apresentados os métodos utilizados para a avaliação de desempenho, explicando as métricas empregadas e suas importâncias. Por último, é explicada toda a metodologia adotada para os treinamentos e testes para cada arquitetura usada.

3.1 Banco de dados

A elaboração cuidadosa de um banco de imagens é uma etapa essencial quando se trabalha com algoritmos de aprendizagem de máquina. A eficácia de uma classificação depende bastante de como essa base de dados foi elaborada, levando em conta a qualidade das imagens, segmentações das áreas de interesses feitas por especialistas da área, ambientalização natural e a abundância dos dados.

Neste estudo, são utilizados dois importantes e públicos conjuntos de dados, ambos consistem em imagens de folhas de café com estresses bióticos visíveis, mas com duas principais diferenças entre os conjuntos de dados. O primeiro conjunto de dados, Bracol (ESGARIO; KROHLING; VENTURA, 2020), compreende imagens de folhas individuais em um ambiente controlado com um fundo branco, onde as anotações foram feitas nos pixels dos sintomas, com cada sintoma rotulado de acordo. O segundo conjunto de dados, RoCoLe (PARRAGA-ALAVA et al., 2019), consiste em imagens de campo em um ambiente natural, onde as anotações para este conjunto de dados são feitas a nível de folha, e cada folha é atribuída a um rótulo correspondente ao tipo de estresse predominante que ela apresenta.

3.1.1 Bracol

Bracol (ESGARIO; KROHLING; VENTURA, 2020) é um conjunto de dados constituído por imagens de folhas de café arábica saudáveis e folhas afetadas por um ou mais estresses bióticos. Foram coletadas 1747 imagens de folhas de café, no estado do Espírito Santo, Brasil, em condições parcialmente controladas e sobre um fundo branco. As fotografias foram realizadas por câmeras de smartphones: *ASUS Zenfone 2*, *Xiaomi Redmi 5A*, *Xiaomi S2*, *Galaxy S8* e *Iphone 6S*.

O procedimento de reconhecimento e rotulação dos estresses presentes nas folhas foi feito por um especialista da área. As classificações possíveis são: *healthy*, *rust*, *brown leaf spot* e *cercospora leaf spot*.

Dentro do banco de dados Bracol, têm-se, originalmente, três subconjunto de dados:

Leaf dataset: consiste em todas as 1747 imagens completas das folhas. As anotações foram feitas com base no estresse predominante na folha e sua severidade, obtendo o valor 0, caso a folha seja uma instância saudável, e valores de 1 a 4, que variam de acordo com a gravidade do sintoma presente. Todas as segmentações foram validadas visualmente, por um profissional especializado, para manter consistência e credibilidade.

Symptom dataset: este conjunto é composto por recortes dos sintomas das fotografias originais, de forma que cada sintoma é representado por uma imagem. No total foram obtidas 2157 imagens dos estresses, sendo estas organizadas de forma separada conforme a classificação adequada.

Segmentation dataset: Formado por 500 imagens extraídas do *Leaf Dataset*, escolhidas de maneira arbitrária, divididas da seguinte forma: 400 para treinamento, 50 para teste e 50 para validação. Cada imagem possui uma máscara que representa 3 objetos: folha, sintoma e plano de fundo.

Para este estudo serão usadas as 500 imagens do *Segmentation dataset*. Estas já possuem suas respectivas máscaras, porém não são o suficiente para realizar o treinamento nos modelos utilizados. Por isso é feito um *script* em *python* para que sejam feitas as conversões de máscaras para anotações em padrão COCO.

```
annotation = {
    "id": int,
    "image_id": int,  // ID da imagem que a annotation pertence
    "category_id": int,  // ID da categoria que a annotation pertence
    "segmentation": RLE or [polygon] // Pontos do contorno da
       segmentacao,
    "area": float, // Area da segmentacao
    "bbox": [x,y,width,height] // Pontos dos contornos retangular,
    "iscrowd": int,  // 0 or 1,
}
```

Tabela 3.1: Distribuição das classificações de imagem para cada sintoma do BRACOL.

Classe	**Número de imagens**
Healthy	100
Mine	118
Rust	122
Phoma	22
Cercospora	138

Esse *script* emprega a biblioteca OpenCV para extrair informações de todos os sintomas presentes em cada imagem. Para cada sintoma, são obtidas informações sobre área, contorno e posição. Com esses dados, é possível criar um arquivo JSON com todos os sintomas mapeados no padrão COCO, tornando-o pronto para uso nos modelos YOLACT++.

No entanto, para utilizar esses dados nos modelos YOLOv8, é necessária a conversão do formato COCO para o padrão YOLO. Para isso, a ferramenta Roboflow[1] é utilizada, realizando automaticamente essa conversão e fornecendo todos os arquivos necessários para iniciar os treinamentos. Isso é exemplificado na Figura 3.1, que apresenta duas imagens diferentes com seus sintomas já segmentados e classificados. Todos os arquivos necessários estão disponíveis no Github do autor[2][3].

Figura 3.1: Anotações de máscaras de segmentação da base Bracol na plataforma Roboflow.

De acordo com a distribuição mostrada na Tabela 3.1 o número de imagens varia significativamente de acordo com o sintoma apresentado. Cada sintoma pode ser visto da Figura 3.2

[1]<<https://roboflow.com/>>
[2]<<https://github.com/gabriel-smello/coco-bracol>>
[3]<<https://github.com/gabriel-smello/yolo-bracol>>

(a) Healthy. (b) Miner. (c) Rust. (d) Phoma. (e) Cercospora.

Figura 3.2: Imagens de exemplo do Bracol para todas as classes.

Tabela 3.2: Distribuição das classificações de imagem para cada sintoma do RoCoLe.

Classe	**Número de imagens**
Healthy	758
Red spider mite	166
Rust level 1	328
Rust level 2	161
Rust level 3	53
Rust level 4	27

3.1.2 RoCoLe

RoCoLe (PARRAGA-ALAVA et al., 2019) é uma base de dados pública que provê imagens de folhas de café robusta (*Coffea canephora*) para realização de treinamento e validação de algoritmos de *machine learning*, podendo esse ser de classificação binária ou de múltiplas classes. Sendo possível a realização de segmentação de objetos, especificamente, o reconhecimento de doenças em plantas.

As imagens das plantas foram captadas por uma câmera de smartphone a uma distância de 200 – 300 mm. Todas as imagens foram obtidas diariamente em condições reais com diversas iluminações, fundos e temperaturas.

Foram fotografadas a parte superior e posterior tanto de folhas saudáveis quanto de folhas infectadas. Os registros foram feitos em um campo com 390 plantas de café, resultando em um total de 1560 imagens.

Todas as imagens estão devidamente anotadas para a tarefa de segmentação de instâncias, contudo, estão classificadas em apenas duas classes (Figura 3.3): *healthy* e *unhealthy*. Entretanto, a própria base de dados fornece um arquivo denominado *RoCoLe-classes.xlsx*, no qual está registrada a doença específica presente em cada imagem.

Toda a base de dados passou por um processo de edição em suas anotações para viabilizar uma classificação mais detalhada juntamente com a segmentação. Conforme exemplificado na Figura 3.4, o arquivo disponibilizado *RoCoLe-classes.xlsx* possui a classificação das folhas de forma binária, que já vem como padrão, e a classificação multiclasse, sendo a primeira classe *healthy* referente a uma folha saudável e cinco outras classes referentes ao sintoma predominante na folha. Então, foi realizado um processo para alterar as anotações de cada imagem para que, agora, estas sejam classificadas de acordo com

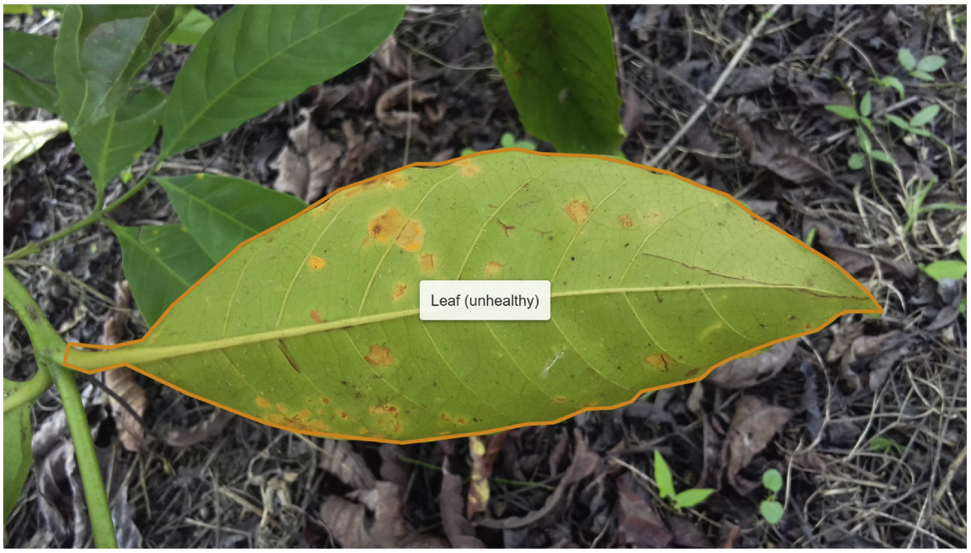


Figura 3.3: Exemplo de anotação de uma máscara de segmentação da base RoCoLe (Fonte: (PARRAGA-ALAVA et al., 2019)).

	A	B	C
1	File	Binary.Label	Multiclass.Label
2	C1P1H1.jpg	healthy	healthy
3	C1P2E2.jpg	unhealthy	rust_level_2
4	C1P2H1.jpg	healthy	healthy
5	C1P3E1.jpg	healthy	healthy
6	C1P3E2.jpg	unhealthy	rust_level_2
7	C1P3H1.jpg	healthy	healthy
8	C1P3H2.jpg	unhealthy	rust_level_3
9	...		

Figura 3.4: Primeiras linhas do arquivo *RoCoLe-classes.xlsx*.

o método multiclasse. Ao final, cada folha está rotulada de acordo com seis diferentes categorias, conforme ilustrado na Figura 3.5: *healthy*, *red spider mite*, *rust level 1*, *rust level 2*, *rust level 3*, *rust level 4*. A distribuição das imagens de acordo com cada sintoma é exibida na Tabela 3.2

A base de dados RoCoLe é originalmente disponibilizada no formato COCO. Portanto, para convertê-la para o formato YOLO, basta utilizar novamente a ferramenta Roboflow. Ao final do processo, obtém-se uma base de dados adequada para ser utilizada com os modelos YOLOv8. Ambos os formatos da base de dados estão disponíveis no Github do autor[4][5].

[4]<<https://github.com/gabriel-smello/coco-rocole>>
[5]<<https://github.com/gabriel-smello/yolo-rocole>>

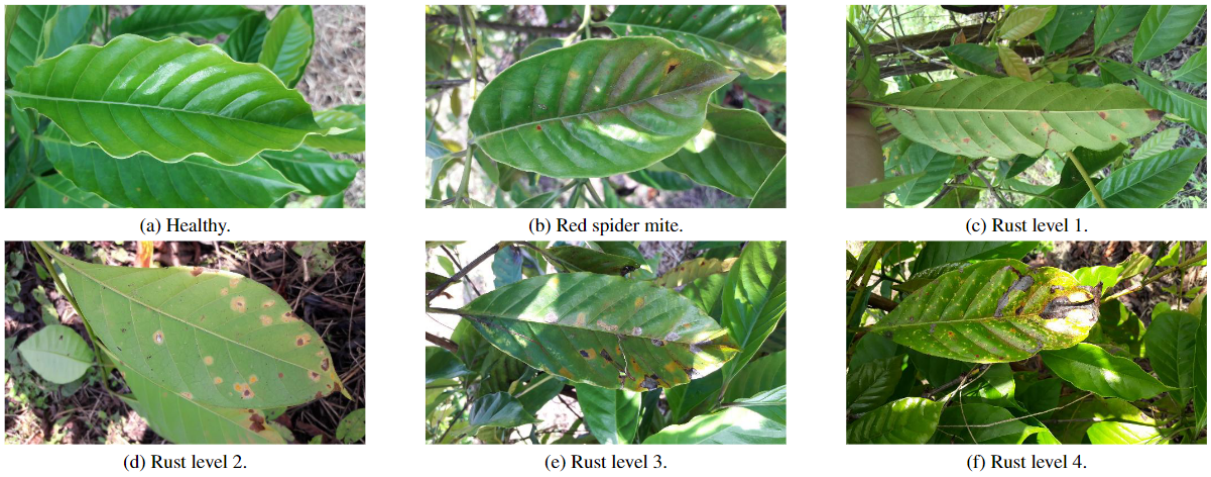


Figura 3.5: Imagens de exemplo do RoCole para todas as classes.

Figura 3.6: Anotações de máscaras de segmentação da base Rocole na plataforma Roboflow.

3.2 Avaliação de desempenho

As métricas de avaliação usadas neste trabalho são baseadas em equações amplamente utilizadas para a avaliação de modelos de aprendizagem profunda voltados para a detecção e segmentação de objetos, como Fast R-CNN (GIRSHICK, 2015), Faster R-CNN (REN, 2015) e YOLO (REDMON et al., 2016).

O *Intersection over Union* (IoU) é uma métrica usada para avaliar a precisão de algoritmos de detecção de objetos. Ela quantifica a sobreposição entre a área prevista pelo modelo e a área real do objeto (*ground truth*). O IoU é calculado dividindo a área de interseção entre a predição e a área real pela área de união dessas duas regiões, representado pela Equação 3.1. Os valores de IoU variam de 0 a 1, sendo que 1 indica uma sobreposição perfeita. Um valor alto de IoU sugere que a detecção do modelo é precisa, pois a área prevista coincide de forma significativa com a área real do objeto:

$$IoU \ = \ \frac{|A \cap B|}{|A \cup B|}. \tag{3.1}$$

As métricas usadas para avaliar os modelos incluem o *Mean Average Precision* (mAP) em diferentes limiares de *Intersection over Union* (IoU), bem como precisão e revocação. Essas métricas fornecem uma avaliação abrangente da eficiência, precisão e confiabilidade do modelo.

$$Precisão = \frac{TP}{TP + FP}, \tag{3.2}$$

a precisão mede a proporção de detecções positivas corretas, verdadeiros positivos (TP), em relação ao total de detecções positivas, a soma de verdadeiros positivos (TP) e falsos positivos (FP). Uma alta precisão indica que o modelo tem uma baixa taxa de falsos positivos, o que é particularmente útil para evitar detecções falsas.

$$Revocação = \frac{TP}{TP + FN}, \tag{3.3}$$

a revocação mede a proporção de verdadeiros positivos (TP) em relação ao total de casos reais, a soma de verdadeiros positivos (TP) e falsos negativos (FN), também conhecidos como objetos de verdadeiros positivos. Uma alta revocação indica que o modelo é eficaz na detecção do objeto alvo, minimizando a chance de perder uma condição crítica.

$$AP = \sum \left(R_n - R_{n-1}\right) P_n; \tag{3.4}$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i, \tag{3.5}$$

finalmente, o mAP (Equação 3.5) é calculado tomando a média dos escores de AP para todas as classes (Equação 3.4). O AP pode ser calculado somando todos os níveis de revocação da precisão em cada limiar, multiplicado pela mudança na revocação do limiar anterior, onde P_n e R_n são, respectivamente, a precisão e a revocação no enésimo limiar.

O mAP pode ser usado para avaliação em um único limiar de IoU ou em múltiplos (por exemplo, de 0,5 a 0,95 em passos de 0,05). Essa métrica é amplamente respeitada devido à sua capacidade de fornecer um único número de desempenho que considera tanto a precisão quanto a revocação de um modelo.

Ao final será empregada também a métrica de Tempo Médio de Inferência (ms) por imagem. Esta métrica é crucial para medir a eficiência do modelo em termos de velocidade de processamento. Especificamente, ela calcula o tempo médio que o modelo leva para realizar a inferência em uma única imagem, proporcionando uma visão clara sobre a latência e o desempenho em tempo real do modelo. Ao incluir esta métrica, é garantida

uma avaliação abrangente que considera também a praticidade e aplicabilidade do modelo em cenários de uso real. Sua fórmula é dada por

$$Tempo\,Médio\,de\,Inferência\ (ms)\ =\ \frac{Tempo\,Total\,de\,Inferência\ (ms)}{Número\,Total\,de\,Imagens}, \tag{3.6}$$

onde, Tempo Total de Inferência (ms) é a soma do tempo gasto, em milissegundos, pelo modelo para realizar a inferência em todas as imagens do conjunto de teste; e Número Total de Imagens é a quantidade de imagens no conjunto de teste.

3.3 Metodologia

Todos os processos de treinamento e avaliação dos resultados foram realizados na plataforma Google Colab[6], que disponibiliza um ambiente que permite a escrita e execução de códigos em python diretamente no navegador. Ela oferece suporte para a realização de programa voltados para análise de dados e desenvolvimentos de modelos de aprendizagem de máquina. Suas vantagens incluem a disponibilização de GPUs e TPUs, que aceleram o treinamento e análises, e a integração com o Google Drive, que permite o armazenamento dos dados utilizados e adiquiridos durante o estudo.

Todo o código, que está disponível no Github[7], e os processos de treinamentos, testes e validações foram feitos em Python 7.0[8], com as bibliotecas *torch 1.4.0* e *torchvision 0.5.0*[9]. Os compiladores *gcc* e *g++* nas versões 8.4.0 e *CUDA* na versão 10.1[10].

É importante esclarecer que os bancos de dados foram divididos de forma que uma parte da imagem seja destinada à etapa de treinamento, enquanto a outra parte será utilizada nas etapas de teste e validação. Essa separação é feita para garantir uma avaliação imparcial e evitar o *overfitting*, que ocorre quando um modelo se ajusta excessivamente aos dados, memorizando-os em vez de aprender as relações entre suas características. Diversos estudos na área utilizam essa abordagem, sendo que cada um adota uma porcentagem específica para a divisão dos dados (CHEN et al., 2021; CERÓN et al., 2021; ŽAGAR et al., 2022). Segundo seus respectivos artigos, o Bracol (ESGARIO; KROHLING; VENTURA, 2020) separa 80% para o treinamento e 20% para teste e validação, enquanto o RoCoLe (PARRAGA-ALAVA et al., 2019) adota uma divisão de 70% para treinamento e 30% para teste e validação.

[6]Disponível em: <https://colab.research.google.com/>
[7]Disnponível em: <https://github.com/gabriel-smello>
[8]Disponível em: <https://www.python.org/downloads/release/python-370/>
[9]Disponíveis em: <https://download.pytorch.org/whl/torch_stable.html>
[10]Disponível em: <https://developer.nvidia.com/cuda-10.1-download-archive-update2>

3.3.1 Treinamento em arquitetura YOLACT++

Para os treinamentos dos modelos YOLACT++, deve-se, primeiro, clonar o repositório original[11] e realizar a instalação, conforme descrito no arquivo *README.md*. Em seguida, deve ser feita todas as configurações necessárias para a utilização de um banco de dados customizado, neste caso, Bracol e RoCoLe. Então, faz-se o *download* de um modelo pré-treinado *Resnet50* para que seja possível usá-lo como parâmetro. Para o uso do YOLACT++ Cbam, faz-se necessário a alteração citada na seção 2.3, disponível no Github do autor deste estudo[12].

Conforme as alterações realizadas, os treinamentos são feitos com a execução da seguinte linha, em que o *batch size* corresponde ao valor padrão de 8:

```
1 !python train.py --config=yolact_plus_resnet50_XXXX_config
2 # 'XXXX' corresponde ao nome do banco de dados alvo do treinamento [
    bracol, rocole]
```

O modelo YOLACT++ foi análisado após cerca de 10.000 iterações no treinamento da base Bracol e após cerca de 30.000 iterações na base RoCoLe, o modelo YOLACT++ Cbam usou o mesmo número de iterações de seu modelo base.

Após finalizada a etapa de treinamento, é gerado um arquivo que corresponde ao modelo após adiquirir os conhecimentos necessários. Para obter as métricas que serão usadas para avaliação, deve executar a linha:

```
1 !python3 eval.py --trained_model=XXXX.pth
2 # 'XXXX' corresponde ao nome do arquivo gerado apos o treinamento
```

3.3.2 Treinamento em arquitetura YOLOv8

Para o treinamento do modelo YOLOv8n, deve-se, primeiro, clonar o repositório original[13]. Para o modelo YOLOv8n + C2f2, basta clonar o repositório do autor deste estudo [14], onde a camada C2f já passou pela alteração interna. Em seguida, para ambos os casos, realiza-se o *download* do banco de dados e, então é feita sua configuração dentro de um arquivo *data.yaml*:

```
1 train: # caminho para a pasta de treinamento
2 val: # caminho para a pasta de validacao
3 test: # caminho para a pasta de teste
4
5 nc: # numero de classes
6 names: # ['classe 1', 'classe 2', ..., 'classe X' ]
```

[11]Disponível em: <https://github.com/dbolya/yolact>
[12]Disponível em: <https://github.com/gabriel-smello/yolact-cbam>
[13]Disponível em: <https://github.com/ultralytics/ultralytics>
[14]Disponível em: <https://github.com/gabriel-smello/yolov8-c2f2>

Por fim, é feita a execução da seguinte linha, para que comece o treinamento:

```
!yolo train model=yolov8n-seg.yaml data=data.yaml epochs=100 imgsz=640
   batch=4 patience=100
```

Tanto o modelo YOLOv8n quanto o modelo YOLOv8n + c2f2 foram análisado após cerca de 8.200 iterações no treinamento da base Bracol e após 26.200 iterações na base RoCoLe.

Ao final do treinamento é criada uma pasta com diversos dados obtidos durante o treinamento e, o mais importante, o arquivo que corresponde aos resultados das métricas atingidas. Para obter esses valores, basta executar:

```
!yolo val data=XXXX
# 'XXXX' corresponde ao arquivo das metricas [best.pt, last.pt]
```

Agora que os dados e metodologias foram devidamente explicados, o próximo capítulo se concentrará na apresentação dos resultados obtidos com os modelos de aprendizado profundo e suas análises detalhadas. Essas análises estão separadas em relação aos dados obtidos em cada banco de dados, além de distinguir os valores entre as tarefas de detecção de objetos e de segmentação de instâncias. Na etapa final são feitas algumas discussões sobre os resultados e suas consequências.

Capítulo 4

Resultados e Discussões

Aqui, serão apresentados os experimentos realizados com os modelos de aprendizado profundo, detalhando os testes conduzidos com os conjuntos de dados apresentados na seção anterior. São discutidos os resultados obtidos para as tarefas de detecção e segmentação de objetos, incluindo comparações entre as versões originais e modificadas dos modelos. Os resultados são avaliados quantitativamente, através de métricas como mAP, precisão e revocação, e qualitativamente, com exemplos visuais das segmentações realizadas. Além de discutir as implicações derivadas dos resultados, bem como compará-los com valores obtidos em estudos semelhantes.

Neste estudo, os quatro modelos (YOLACT++, YOLACT++ Cbam, YOLOv8n e YOLOv8n + C2f2), apresentados no Capítulo 2, foram treinados e comparados com base em seus desempenhos. As comparações foram divididas em relação aos dois conjuntos de dados utilizados.

4.1 Resultados em Bracol

Considerando apenas a tarefa de detecção de objeto, a Tabela 4.1 mostra que a integração do módulo de atenção *CBAM* ao modelo *YOLACT++* diminui o mAP em limiares de IoU de 0,5. No entanto, gerou uma melhora na precisão, revocação e o mAP em limiares de IoU maiores.

Em contrapartida, adicionar C2f2 ao modelo YOLOv8n resultou em melhorias mais significativas, superando o modelo original. Isso demonstra que a adição do CBAM e do C2f2 melhora as métricas quantitativas dos modelos originais.

Mudando o foco apenas para o aspecto da segmentação, em contraste com as expectativas, a adição do módulo de atenção *CBAM* ao modelo *YOLACT++* resultou em uma redução das métricas mAP, precisão e revocação para tarefas de segmentação (Tabela 4.2).

Tabela 4.1: Desempenho de detecção de objeto para Bracol.

Modelo detec. obj.	**mAP0.5**	**mAP0.5-0.95**	**Prec.**	**Rec.**
YOLACT++	64,69	38,46	22,14	50,73
YOLACT++ CBAM	61,67	40,27	23,74	51,59
YOLOv8n	74,4	46	**87,2**	55,9
YOLOv8n + C2f2	**77,1**	**52,4**	71,5	**67,8**

Isso sugere que o módulo *CBAM*, nesse caso, só é eficaz para a detecção de objetos e não para a segmentação.

Por outro lado, a incorporação do módulo C2f2 ao modelo YOLOv8n levou a uma melhoria significativa nas métricas de mAP e precisão, com uma ligeira diminuição na revocação. Esse resultado mostra a eficácia da modificação feita no modelo YOLOv8n para detecção e segmentação de objetos, levando a resultados ainda mais confiáveis e precisos.

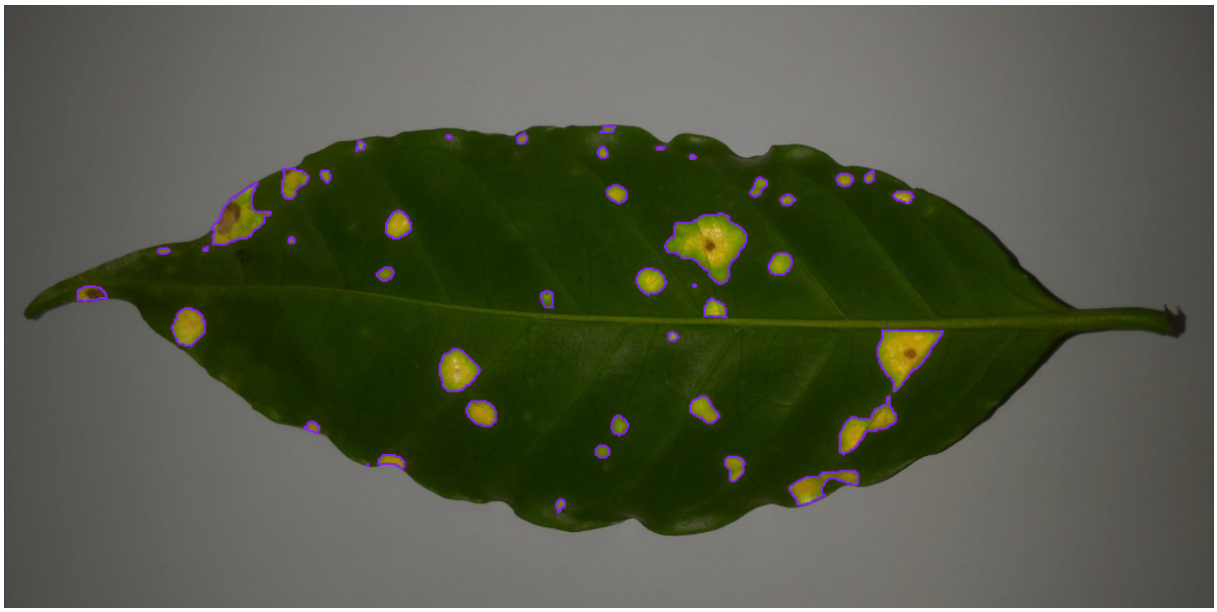


Figura 4.1: Exemplo de anotação feita na base Bracol.

É fundamental destacar que a transição da camada C2f para a camada C2f2 visa, principalmente, aprimorar a detecção e segmentação de objetos pequenos. Conforme ilustrado na Figura 4.1, as anotações neste banco de dados são realizadas diretamente sobre os sintomas presentes nas folhas. Isso frequentemente resulta em objetos consideravelmente pequenos e difíceis de detectar pelo modelo original.

A Figura 4.2 ilustra a comparação entre os modelos nas métricas de mAP, precisão e revocação.

Tabela 4.2: Desempenho de segmentação para Bracol.

Modelo segm.	mAP0.5	mAP0.5-0.95	Prec.	Rec.
YOLACT++	65,13	40,86	22,76	52,55
YOLACT++ CBAM	59,54	39,37	22,66	49,09
YOLOv8n	75,2	39,6	61,2	**68,1**
YOLOv8n + C2f2	**77,1**	**43,9**	**72,5**	66,4

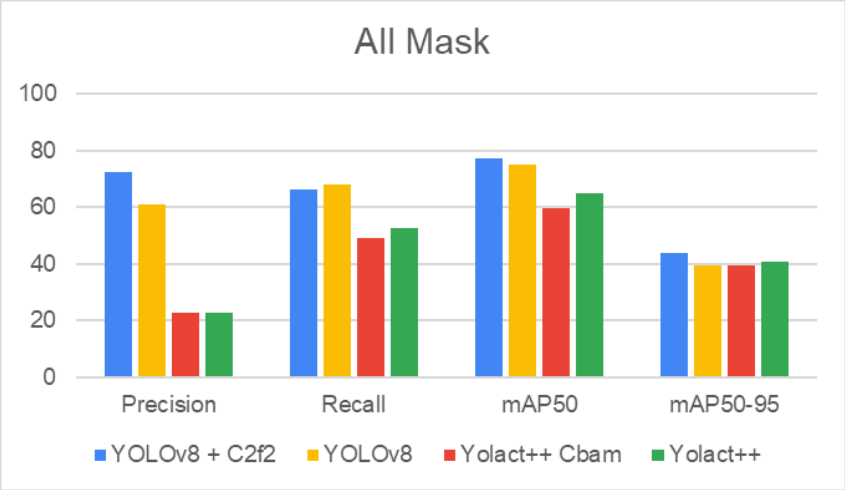


Figura 4.2: Comparação das métricas de desempenho dos modelos para o BRACOL.

A predominância do modelo YOLOv8n + C2f2 é evidente em todas as métricas, demonstrando sua superioridade sobre o YOLACT++ na tarefa de segmentação para a base de dados Bracol.

4.2 Resultados em RoCoLe

As imagens foram divididas em 70% e 30% para treinamento e teste, respectivamente.

Semelhante ao conjunto de dados anterior, a incorporação do módulo de atenção CBAM melhorou as métricas de detecção de objetos na tarefa atual (Tabela 4.3). No entanto, para a tarefa de segmentação, a adição do CBAM não melhorou as métricas das imagens Bracol, enquanto que para as imagens capturadas em um ambiente natural, como o RoCoLe, o modelo YOLACT++ Cbam superou o modelo original. Notavelmente, essas métricas de mAP superam não apenas o modelo original, mas também os modelos baseados no YOLOv8 (Tabela 4.4).

As anotações no RoCoLe foram feitas no nível da folha, ilustrado pela Figura 4.3, que são objetos significativamente maiores em comparação com os do Bracol quando se considera a imagem inteira. Isso sugere que a incorporação do C2f2 pode não ser necessária

Tabela 4.3: Desempenho de detecção de objeto para RoCoLe.

Modelo detec. obj.	**mAP0.5**	**mAP0.5-0.95**	**Prec.**	**Rec.**
YOLACT++	53,3	40,69	27,15	66,3
YOLACT++ CBAM	**61,08**	44,75	28,2	65,86
YOLOv8n	52,6	**45,4**	**36,8**	**81,9**
YOLOv8n + C2f2	44,3	35,4	45,6	64,5

Figura 4.3: Exemplo de anotação feita na base RoCoLe.

para o RoCoLe, pois essa modificação é benéfica principalmente para a detecção e a segmentação de objetos pequenos. Conforme observado nas Tabelas 4.3 a 4.4, o modelo YOLOv8n+C2f2 não só se mostrou desnecessário, mas também resultou em métricas piores em comparação com o modelo YOLOv8n sozinho, tanto para as tarefas de detecção de objetos quanto para as de segmentação.

Também é importante ressaltar a eficiência que o módulo de atenção CBAM trouxe para o modelo YOLACT++. Esse módulo é capaz de aprender 'onde' e 'o que' é significativo na imagem, sendo especialmente útil, considerando que a base RoCoLe possui um *background* com muitas informações que não devem ser levadas em conta. Observando as métricas e os resultados qualitativos na Figura 4, é evidente que o modelo YOLACT++ CBAM aprimorou seu treinamento para detectar e segmentar as folhas desejadas, descartando todas as informações indesejadas do *background*.

Outro parâmetro importante da avaliação de desempenho de todos os modelos diz respeito aos tempos de inferência. A Tabela 4.5 mostra o tempo médio de inferência

Tabela 4.4: Desempenho de segmentação para RoCoLe.

Modelo segm.	**mAP0.5**	**mAP0.5-0.95**	**Prec.**	**Rec.**
YOLACT++	50,79	43,06	28,87	71,72
YOLACT++ CBAM	**58**	**48,01**	30,01	71,77
YOLOv8n	52,2	45,8	36,5	**81,2**
YOLOv8n + C2f2	44	36	**45,4**	63,9

Tabela 4.5: Tempos médios de inferência (ms) por imagem de cada um dos modelos.

Modelo	**BRACOL**	**RoCoLe**
YOLACT++	629,89	431,83
YOLACT++ CBAM	750,03	480,02
YOLOv8n	97,0	103,8
YOLOv8n + C2f2	**69,0**	**85,4**

medido em ms por imagem para todos os modelos. Com a fusão C2f2 proposta no YOLOv8n, mostra-se que a inferência é muito mais rápida no modelo mais novo, o que é uma vantagem importante para implantar o modelo para aplicação em campos reais.

4.3 Discussões

Analisando apenas os modelos YOLACT++, notam-se algumas melhorias em troca de algumas perdas. O módulo de atenção CBAM mostra-se útil para detalhes mais grosseiros, cobrindo uma área maior, beneficiando assim a detecção de objetos. Isso pode ser observado na Figura 4.4, que ilustra claramente a melhoria do modelo com o módulo de atenção em limites intermediários de IoU. A segmentação, por outro lado, é uma tarefa mais meticulosa e precisa, trabalhando pixel por pixel em um objeto.

Os modelos YOLOv8n já alcançam ganhos significativos em comparação com o desempenho do modelo YOLACT++. O YOLOv8 é um modelo mais recente em comparação com o YOLACT++. Além de aproveitar os pontos fortes das versões anteriores, ele apresenta muitas inovações, tanto no campo da detecção de objetos quanto na segmentação e classificação. Sem nenhuma adição, já é evidente que as métricas são muito boas para a tarefa desejada, mas sempre há espaço para melhorias. A adição do C2f2 resulta em uma perda de precisão (detecção de objetos) e revocação (segmentação); no entanto, isso não impede que outras métricas, como o mAP, aumentem ainda mais sua capacidade.

Na Figura 4.2, a predominância dos modelos YOLO sobre os modelos YOLACT é observada em 3 das 4 métricas. Somente para a métrica mAP0.5-0.95 os valores se aproximam e mostram pequenos ganhos, mas, no geral, ainda refletem o que se espera ao comparar os modelos.

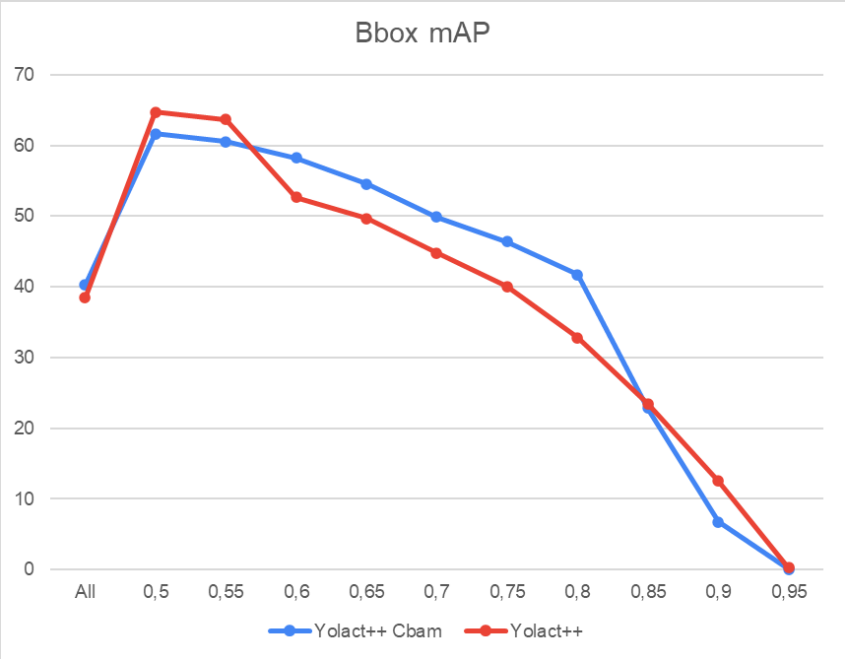


Figura 4.4: mAP da detecção de objetos dos modelos YOLACT++ e YOLACT++ Cbam para Bracol.

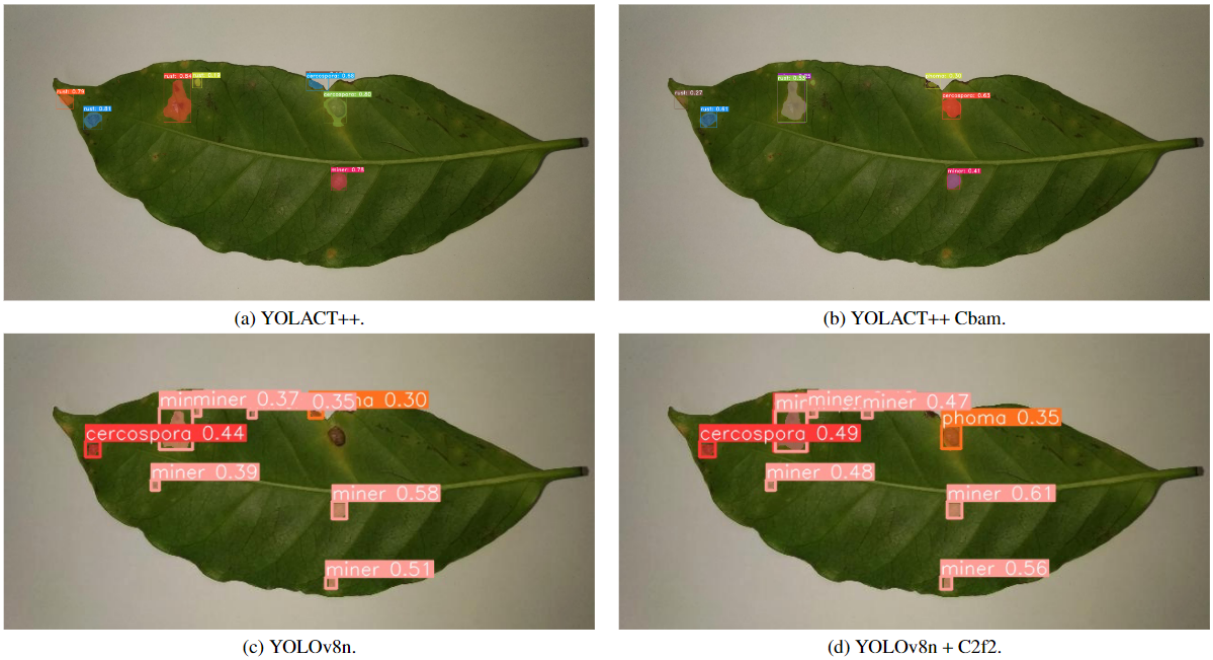
Um estudo realizado como projeto de graduação na Universidade de Brasília, pelo departamento de Ciência da Computação, com autoria do então aluno Guilherme Rodrigues Lodron Pires, (PIRES, 2022) produziu testes semelhantes aos realizados neste trabalho. O estudo também usou o modelo YOLACT++ para analisar seu desempenho nos mesmos conjuntos de dados, com algumas alterações nos bancos para aprimorar as métricas, que se limitam ao *mAP*.

Como é possível notar pela Tabela 4.6, os melhores resultados obtidos por aquele estudo superam os alcançados aqui com o modelo YOLACT++. Porém, os melhores resultados obtidos neste trabalho com os outros modelos ainda se sobressaem em comparação ao estudo mencionado. Por exemplo, ao observar os valores de *mAP0.5* na etapa de segmentação na base RoCoLe, o estudo do Guilherme Pires obteve um valor de 52,61 com o modelo YOLACT++, enquanto, aqui, o mesmo modelo YOLACT++ performou em 50,79. No entanto, ao analisar o modelo destaque YOLACT++ Cbam o resultado atingido é de 58,0, se mostrando superior ao estudo comparativo.

Observando os resultados qualitativos das Figuras 4.5 a 4.6, que apresenta as caixas delimitadoras da detecção de objetos, as máscaras de segmentação e a classificação de cada objeto, nota-se a presença de valores de confiança para cada detecção. Esses valores são calculados com base na probabilidade da classe do objeto, multiplicada pelo valor de IoU (Equação 3.1). A rede neural de detecção de objetos gera uma probabilidade para

Tabela 4.6: Valores obtidos por um estudo semelhante (Fonte: (PIRES, 2022)).

	YOLACT++			
	Bracol		**RoCoLe**	
	mAP0.5	**mAP0.5-0.95**	**mAP0.5**	**mAP0.5-0.95**
Detecção de Objeto	72,7	49,7	56,97	43,12
Segmentação	70,9	50,5	52,61	44,76



(a) YOLACT++.

(b) YOLACT++ Cbam.

(c) YOLOv8n.

(d) YOLOv8n + C2f2.

Figura 4.5: Exemplos de resultados qualitativos de cada modelo no Bracol, considerando classificação, detecção de objeto e segmentação de instâncias.

cada classe possível dentro da caixa delimitadora. Esse valor, quando multiplicado pelo IoU, que leva em consideração a sobreposição entre a caixa delimitadora prevista e a real, resulta em uma taxa de confiança, que varia entre 0 e 1:

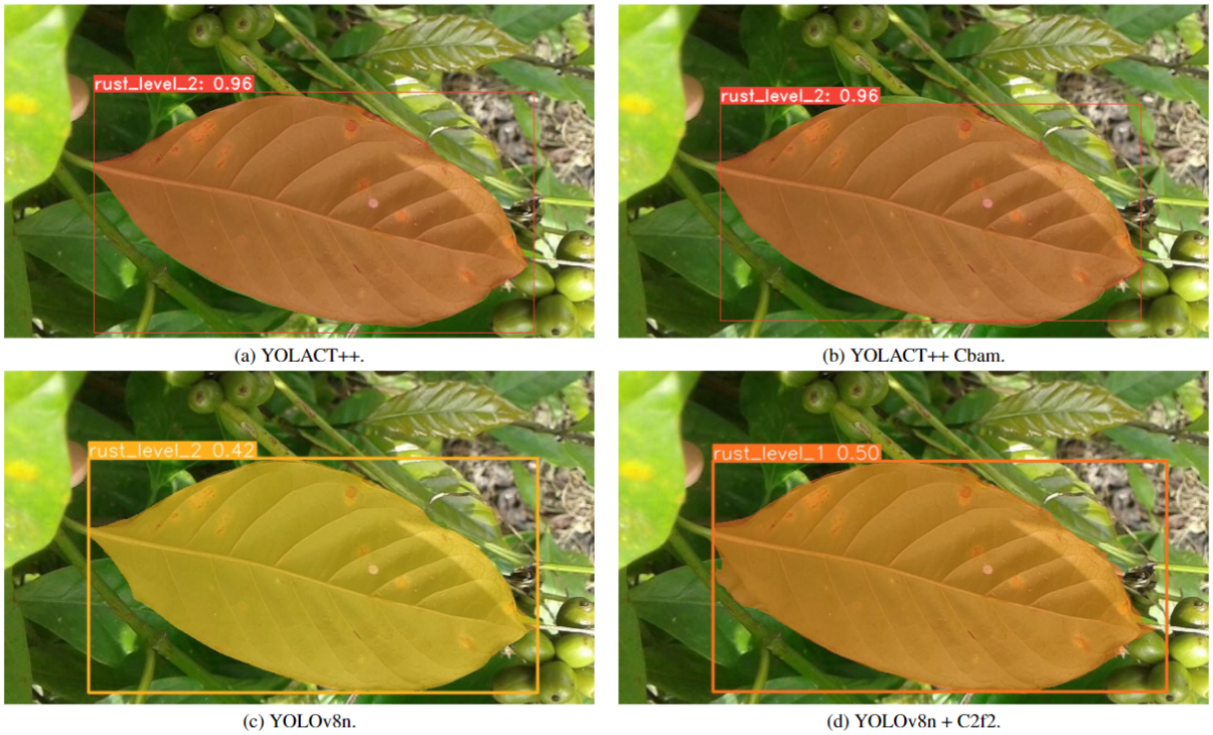
$$Confiança = Pr\left(Objeto\right) \cdot IoU\left(prev, real\right). \tag{4.1}$$

Apesar dos resultados quantitativos do YOLACT++ Cbam na Figura 4.5 não serem tão positivos, nota-se que os pixels selecionados para segmentação estão mais bem delimitados e mais próximos do sintoma. Já no modelo YOLACT++, há uma segmentação exagerada, selecionando muitos pixels que não fazem mais parte do sintoma. Isto reflete que, na prática, não houve mudanças significativas entre esses modelos. Por outro lado, os modelos baseados no YOLOv8n claramente detectam mais pontos relevantes na folha, quando comparados com modelos baseados em YOLACT++.

Na mesma figura, é possível observar o aprimoramento proporcionado pela adição de

C2f2, em que o modelo melhora suas segmentações, especialmente para objetos menores. Além disso, aumenta os valores de confiança para cada detecção presente em ambos os modelos baseados em YOLOv8n.

Na Figura 4.6, é importante notar que os valores de confiança das detecções refletem os valores obtidos na análise quantitiva, sendo os modelos com arquitetura YOLOv8n inferiores aos modelos com arquitetura YOLACT++, em especial, após a adição do módulo CBAM.



(a) YOLACT++. (b) YOLACT++ Cbam.

(c) YOLOv8n. (d) YOLOv8n + C2f2.

Figura 4.6: Exemplos de resultados qualitativos de cada modelo no RoCoLe, considerando classificação, detecção de objeto e segmentação de instâncias.

Com os resultados discutidos, o próximo capítulo encerrará este trabalho, relembrando os objetivos iniciais e debatendo se eles foram alcançados, apresentando as conclusões principais e sugerindo possíveis direções para futuras pesquisas nesta área.

Capítulo 5

Conclusões

Para concluir este trabalho, é necessário lembrar o objetivo principal. O estudo buscou verificar se as adaptações nos modelos YOLACT++ e YOLOv8n trariam grandes melhorias nos indicadores de desempenho, como mAP, precisão e revocação. Além disso, foi analisado se o tempo médio de inferência também seria favorecido. O objetivo era aumentar a eficácia na identificação e segmentação de doenças nas folhas de café, visando otimizar a produção agrícola e reduzir as perdas.

Em relação à metodologia e aos experimentos, o trabalho utilizou dois conjuntos de dados públicos: Bracol e RoCoLe, contendo imagens de folhas de café capturadas em laboratório e em campo. A metodologia envolveu o uso dos modelos de aprendizagem profunda YOLACT++ e YOLOv8n, modificados com a adição de um módulos de atenção (CBAM) e uma alteração interna em uma das camadas de convolução (C2f2) para a detecção e segmentação de doenças. Os experimentos foram comparados entre as versões originais e modificadas dos modelos, analisando o desempenho em termos de precisão, revocação, mAP e tempo de inferência.

A discussão dos resultados mostrou que a versão ajustada do YOLOv8n com o módulo C2f2 obteve o melhor desempenho no conjunto de dados Bracol, com melhorias significativas nas métricas de detecção e segmentação. No conjunto de dados RoCoLe, o YOLACT++ com o módulo CBAM foi o que mais se destacou, especialmente na segmentação de instâncias. A conclusão geral é que as modificações propostas são promissoras, mas seu impacto varia dependendo do tipo de dado utilizado, sendo mais eficazes em ambientes controlados do que em campo.

Analisando os resultados no banco de dados Bracol, que é constituído de imagens em ambiente controlado, o modelo YOLOv8n melhorado, com a camada C2f2, apresentou o melhor desempenho entre todos os modelos, indicando uma melhoria significativa tanto nas métricas de detecção de objeto (*mAP0.5* de **77,1**) quanto nas métricas de segmentação de instâncias (*mAP0.5* de **77,1**) quando comparado com sua versão original. Porém, o

modelo YOLACT++ Cbam teve um desempenho inferior à sua versão base, não atingindo o objetivo de melhoria. Em relação ao tempo médio de inferência, o modelo YOLOv8n + C2f2 teve um tempo mais rápido quando comparado com os outros modelos analisados, mostrando que as alterações reduziram o tempo de inferência (**69,0**ms por imagem), tornando-o mais escalável e aplicável em situações reais.

No banco de dados RoCoLe, que representa imagens em situação de campo, o modelo YOLACT++ Cbam melhorou a métrica *mAP0.5* em relação ao YOLACT++ na etapa de detecção de objeto (*mAP0.5* de **61,08**), mas o YOLOv8n ainda obteve os melhores resultados nas outras métricas analisadas (*mAP0.5-0.95* de **45,40**), alinhando-se parcialmente com as expectativas de melhoria. Na tarefa de segmentação de instâncias, o YOLACT++ Cbam não só melhorou em relação ao modelo YOLACT++ em ambas as métricas de *mAP*, como também superou todos os outros modelos neste aspecto, atingindo o objetivo de melhorias no modelo modificado em relação ao original. Por outro lado, o YOLOv8n + C2f2 apresentou um desempenho inferior ao modelo base, não atendendo às expectativas. Esse modelo também se destacou no tempo médio de inferência, com um tempo mais rápido (**85,40**ms por imagem), que, assim como os resultados na base de dados anterior, está alinhado com o objetivo de aumentar a escalabilidade dos modelos apresentados.

Também tinha-se como objetivo, analisar como a constituição dos bancos de dados impacta o treinamento e o desempenho dos modelos. Os resultados mostraram que as imagens em ambiente controlado (Bracol) foram mais adequadas para o modelo YOLOv8n + C2f2, que superou seu modelo original e os modelos baseados em YOLACT++ de forma notória. Em ambientes reais (RoCoLe), o YOLACT++ Cbam foi mais eficaz em algumas métricas tanto na etapa de detecção de objetos quanto em segmentação de instâncias, se sobressaindo até mesmo aos modelos baseados em YOLOv8n. Também cabe salientar a forma que as anotações foram feitas na construções desses bancos de dados. Para aquele que possui anotações a nível de folha (RoCoLe), apenas a modificação no modelo YOLACT++ obteve melhoras, já para aquele que possui imagens anotadas a nível de sintomas (Bracol), a modificação no modelo YOLOv8n se sobressaiu quando comparado com todas as outras métricas. Isso se dá ao fato de que a alteração na camada C2f para C2f2 tem como objetivo principal a de melhorar a detecção e segmentação de objetos pequenos. Esses fatores indicam que as características que constituem os bancos de dados influenciam o desempenho dos modelos, cumprindo o objetivo de analisar o impacto desse aspecto, contribuindo para uma melhor compreensão de aplicabilidade e escalabilidade.

Para o futuro, seria de extrema relevância o investimento de estudos para melhorar ainda mais a aplicabilidade dos modelos de aprendizagem de máquina com a finalidade de diminuir as perdas de culturas na agricultura, não só nacional como a mundial. Uma possi-

bilidade seria a de fenotipagem de mais doenças em mais espécies de plantas, visando uma aplicação que funcione para um maior número de espécies e tipo de culturas. Ademais, seria interessante investigar o uso de técnicas de aprendizagem de máquina não supervisionadas ou semi-supervisionadas, que podem reduzir a necessidade de dados anotados manualmente por profissionais da área, tornando o processo mais rápido e econômico. Um outro objetivo futuro seria o de achar maneiras de melhorar ainda mais as precisões e os tempos de inferências até uma aplicabilidade significativa para uma implementação em dispositivos móveis, tornando-se uma ferramenta a ser usada por profissionais que atuam diretamente com o cultivo.

Referências Bibliográficas

BOLYA, D. et al. Yolact: Real-time instance segmentation. In: *ICCV*. [S.l.: s.n.], 2019. 5

BOLYA, D. et al. Yolact++: Better real-time instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2, 6

CERÓN, J. C. A. et al. Assessing yolact++ for real time and robust instance segmentation of medical instruments in endoscopic procedures. In: IEEE. *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. [S.l.], 2021. p. 1824–1827. 20

CHEN, W. et al. Yolo-face: a real-time face detector. *The Visual Computer*, Springer, v. 37, p. 805–813, 2021. 20

CONTINI, E.; ARAGÃO, A. O agro brasileiro alimenta 800 milhões de pessoas. *Brasília: Embrapa*, 2021. 1

ESGARIO, J. G.; KROHLING, R. A.; VENTURA, J. A. Deep learning for classification and severity estimation of coffee leaf biotic stress. *Computers and Electronics in Agriculture*, Elsevier, v. 169, p. 105162, 2020. 2, 13, 14, 20

GIRSHICK, R. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. 18

HE, K. et al. Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 2961–2969. 5

HUANG, M. et al. A method for segmenting disease lesions of maize leaves in real time using attention yolact++. *Agriculture*, MDPI, v. 11, n. 12, p. 1216, 2021. 8, 9

JOCHER, G.; CHAURASIA, A.; QIU, J. *Ultralytics YOLO*. 2023. Disponível em: <https://github.com/ultralytics/ultralytics>. 2, 8

LIMA, B. P. de M. et al. A lightweight and enhanced model for detecting the neotropical brown stink bug, euschistus heros (hemiptera: Pentatomidae) based on yolov8 for soybean fields. *Ecological Informatics*, Elsevier, v. 80, p. 102543, 2024. 10

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. [S.l.], 2014. p. 740–755. 6

LIU, H. et al. Sf-yolov5: A lightweight small object detection algorithm based on improved feature fusion mode. *Sensors*, MDPI, v. 22, n. 15, p. 5817, 2022. 10

LIU, Z. et al. An improved yolov5 method for small object detection in uav capture scenes. *IEEE Access*, IEEE, v. 11, p. 14365–14374, 2023. 10

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2015. p. 3431–3440. 5

LOU, H. et al. Dc-yolov8: small-size object detection algorithm based on camera sensor. *Electronics*, MDPI, v. 12, n. 10, p. 2323, 2023. 10

MIRANDA, E. E. de. Áreas cultivadas no brasil e no mundo. *AgroANALYSIS*, v. 38, n. 2, p. 25–27, 2018. 1

PARRAGA-ALAVA, J. et al. Rocole: A robusta coffee leaf images dataset for evaluation of machine learning based methods in plant diseases recognition. *Data in brief*, Elsevier, v. 25, p. 104414, 2019. 2, 13, 16, 17, 20

PIRES, G. R. L. Segmentação de instâncias com aplicações em agricultura de café. 2022. Disponível em: <https://bdm.unb.br/handle/10483/34372>. 28, 29

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788. 18

REN, S. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015. 18

SINGH, A. et al. Challenges and opportunities in machine-augmented plant stress phenotyping. *Trends in Plant Science*, Elsevier, v. 26, n. 1, p. 53–69, 2021. 2

STRANGE, R. N.; SCOTT, P. R. Plant disease: a threat to global food security. *Annu. Rev. Phytopathol.*, Annual Reviews, v. 43, n. 1, p. 83–116, 2005. 1

WOO, S. et al. Cbam: Convolutional block attention module. In: *Proceedings of the European conference on computer vision (ECCV)*. [S.l.: s.n.], 2018. p. 3–19. 7

ŽAGAR, B. L. et al. Real-time instance segmentation of pedestrians using transfer learning. In: IEEE. *2022 27th International Conference on Automation and Computing (ICAC)*. [S.l.], 2022. p. 1–6. 20