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

Segmentação de instâncias com aplicações em agricultura de café

Guilherme Rodrigues Lodron Pires

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

Orientador
Prof. Dr. Díbio Leandro Borges

Brasília
2022

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

Segmentação de instâncias com aplicações em agricultura de café

Guilherme Rodrigues Lodron Pires

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

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

Prof. Dr. Camilo Chang Dórea
CIC/UnB

Profa. Dra. Roberta Barbosa Oliveira
CIC/UnB

Prof. Dr. João José Costa Gondim
Coordenador do Curso de Engenharia de Computação

Brasília, 21 de setembro de 2022

Dedicatória

Dedico esse trabalho aos meu pais, Yvete e Leônidas, por sempre estarem ao meu lado nos momentos difíceis e por me apoiarem durante toda a minha trajetória acadêmica. Também dedico a todos que se empenham em avançar o campo da tecnologia para proporcionar um mundo melhor.

Agradecimentos

Gostaria de agradecer aos meus pais, Yvete e Leônidas por sempre acreditarem no meu pontencial e me proporcionarem a vida maravilhosa que tive até aqui. Também agradeço a minha família e a minha namorada, por sempre me apoiarem e me animarem em todas as situações.

Em segundo lugar gostaria de agradecer a todos os amigos que fiz durante meus estudos e que me mostraram opiniões e conhecimentos que levarei pro resto da vida. Também à UnB, por proporcionar tanto conhecimento e vivência nesses anos.

Especialmente, gostaria de agradecer ao meu orientador Prof. Díbio, pelo comprometimento com o ensino e com sua grande paciência e conselhos durante todo o trabalho.

Por fim quero agradecer a todos que passaram por mim, direta ou indiretamente durante minha graduação. Levarei as memórias que tenho com muito carinho pela jornada que está a frente.

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

No Brasil, a agricultura tem papel muito importante na economia e no dia a dia das pessoas. A detecção e prevenção de pestes e doenças de forma automatizada pode aumentar a eficiência e a qualidade dos produtos produzidos no país e no mundo. Para tal, são necessários esforços conjuntos de especialistas e desenvolvedores de software. Atualmente, a área que se destaca no estudo desses métodos é a de aprendizado profundo, por esse motivo este trabalho busca analisar o modelo Yolact++ em diferentes bases de dados e discutir sobre seu potencial e suas limitações. O estudo foi feito sobre três bases de dados: RoCoLe, Bracol e Bracot, que são compostas por imagens de folhas de plantas de café afetadas por doenças comuns a esse tipo de plantação. Por fim, também é proposto um *pipeline* que consiste na junção do treinamento nas bases Bracol e Bracot para identificar doenças em um contexto mais próximo ao real. O trabalho foi realizado em cima da tarefa de segmentação de instâncias e a métrica utilizada para avaliar cada etapa de treinamento foi a de *mean average precision* (mAP). Diante disso os resultados provenientes do treinamento do modelo Yolact++ foram satistatórios, sendo que para a base RoCoLe foi obtido um mAP de 44.76 para segmentação e 43.12 para *bounding box*, para a base Bracot o mAP obtido para segmentação foi de 54.0 e para *bounding box* foi de 62.2 e para a base Bracol os resultados foram de 49.7 e 50.5 para segmentação e *bounding box* respectivamente. Todos os resultados obtidos foram comparados ao artigo base utilizado para referência escrito por Tassis, Tozzi e Krohling e ponderações sobre possíveis melhorias foram realizadas para cada base de dados. O pipeline proposto também gerou bons resultados qualitativos, mas devido a limitação ao acesso à engenheiros agrônomos e profissionais especializados na agricultura de café, não foi possível avaliar métricas precisas do último passo.

Palavras-chave: Segmentação de instância. Pragas de café. Aprendizagem de Máquinas. Aprendizagem Profunda. Segmentação de Imagens. Redes Convolucionais.

Abstract

In Brazil, agriculture plays a very important role in the economy and in people's daily lives. The detection and prevention of pests and diseases in an automated way can increase the efficiency and quality of products produced in the country and in the world. For this, joint efforts of specialists and software developers are needed. Currently, the area that stands out in the study of these methods is deep learning, for this reason this work seeks to analyze the Yolact++ model in different databases and discuss its potential and limitations. The study was carried out on three databases: RoCoLe, Bracol and Bracot, which are composed of images of leaves of coffee plants affected by diseases common to this type of plantation. Finally, a pipeline is also proposed, which consists of the combination of training in the Bracol and Bracot bases to identify diseases in a context closer to the real one. The work was carried out on top of the instance segmentation task and the metric used to evaluate each training step was mean average precision (mAP). Therefore, the results from the training of the Yolact++ model were satisfactory, and for the RoCoLe base a mAP of 44.76 was obtained for segmentation and 43.12 for *bounding box*, for the Bracot base the mAP obtained for segmentation was 54.0 and for bounding box it was 62.2 and for the Bracol base the results were 49.7 and 50.5 for segmentation and bounding box respectively. All the results obtained were compared to the base article used for reference written by Tassis, Tozzi and Krohling and considerations about possible improvements were carried out for each database. The proposed pipeline also generated good qualitative results, but due to limited access to agronomists and professionals specialized in coffee agriculture, it was not possible to evaluate precise metrics for the last step.

Keywords: Instance segmentation. Cooffe disease. Machine learning. Image segmentation. Segmentation analysis. Artificial neural networks. Deep learning. Loss functions

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

COCO Common Objects in Context.

CPU Central Process Unit.

FCNN Fully Convolutional Neural Network.

GB Giga bytes.

GPU Graphics Processing Unit.

JSON Javascript Object Notation.

R-CNN Region Based Convolution Neural Network.

Capítulo 1

Introdução

No Brasil, a agricultura é uma das áreas mais relevantes, tanto para economia, quanto para o desenvolvimento da população, sendo um dos setores que mais contribui com o PIB nacional, ocupa grande parte das exportações e é um dos principais geradores de emprego no país[1] . Mais especificamente, a produção de café tem papel relevante no agronegócio, ocupando para produção, em 2022, 1.82 milhões de hectares segundo a Embrapa[2]. Por esse motivo, todos os anos são colhidos milhões de toneladas do grão, como o exemplo do ano 2021 ilustrado na Figura 1.1.

Entretanto, a grande competitividade do mercado e a demanda crescente pelo café, compele os produtores a procurarem diferentes meios de aumentar eficiência e a qualidade de suas plantações. Para tal, é necessário ter um acompanhamento em tempo real de doenças e pragas que podem prejudicar a colheita (SANTOS et al., 2020b).

A tarefa de monitorar a saúde de uma planta de forma manual é extremamente expendiosa. São necessários um grande número de profissionais no campo todos os dias e um esforço de análise de inúmeras amostras colhidas. Tal esforço pode ser diminuído com o uso de tecnologias que possibilitem a localização das áreas da planta afetadas por algum sintoma e a classificação de qual doença ou praga ele pertence de forma automática em um curto período de tempo (SANTOS et al., 2020a).

Atualmente, o campo que mais se destaca no desenvolvimento e no estudo dessas tecnologias é o aprendizado de máquina (*machine learning)*, mais especificamente o aprendizado profundo (*deep learning*), pela capacidade de processar da-

[1] Fonte Embrapa, disponível em <https://www.embrapa.br/vii-plano-diretor/a-agricultura-brasileira.>

[2] Disponível em <https://www.embrapa.br/busca-de-noticias/-/noticia/64630822/producao-dos-cafes-do-brasil-ocupa-area-de-182-milhao-de-hectares-dos-quais-145-milhao-sao-de-cafe-arabica-e-37599-mil-de-conilon>

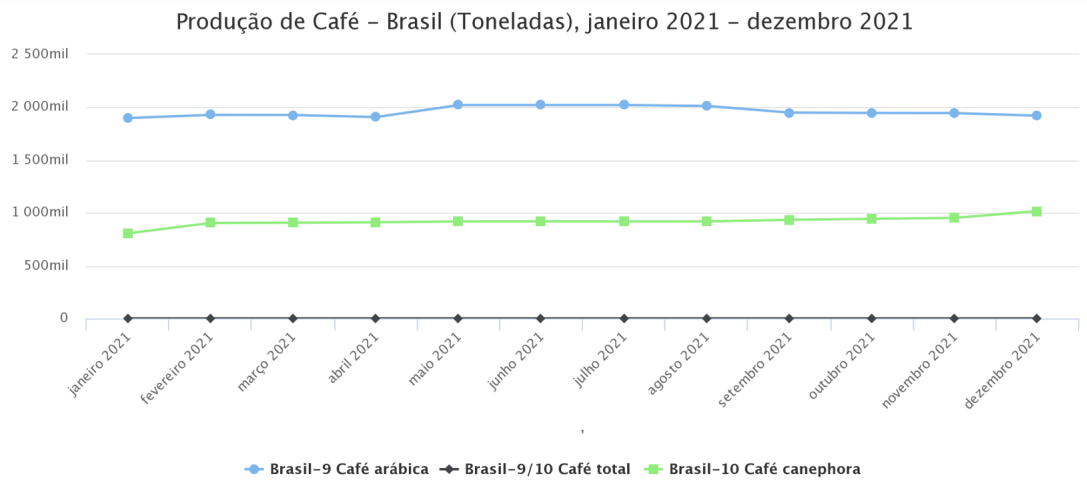

Figura 1.1: Produção de café durante o ano de 2021 para os tipos de grãos de café arábica e canephora. Fonte IBGE, Disponível em <https://www.ibge.gov.br/estatisticas/economicas/agricultura-e-pecuaria/9201-levantamento-sistematico-da-producao-agricola.html?=&t=series-historicas>.

dos complexos como imagens. Um exemplo do uso de aprendizado profundo pode ser encontrado no artigo de Santos, Souza e Avila (SANTOS et al., 2020b), que utiliza os modelos Mask R-CNN (HE et al., 2017) e YoloV3 (REDMON; FARHADI, 2018) para a detecção de cachos de uva em uma vinícola.

O desenvolvimento desses campos de estudo também pode ser aplicado em outros contextos da agricultura, como o café. A partir disso, utilizando modelos de aprendizado profundo é possível segmentar e classificar áreas afetadas em cada folha de um cafeeiro. Como pode ser observado na figura 1.2, que ilustra a segmentação de uma folha de café, representada na cor marrom, e a classificação quanto ao sintoma que a afeta, inidicada pelo texto em vermelho.

Para viabilizar o treinamento de modelos que sejam capazes de segmentar doenças em uma plantação, são desenvolvidas bases de dados compostas por imagens previamente analisadas por especialistas, cujas áreas de interesse do estudo (sintomas, folhas, objetos) são identificadas e explicitadas por meio de anotações. Essas bases de dados são disponibilizadas publicamente através de artigos científicos, repositórios *online* ou por sites voltados para comunidade de aprendizado de máquina. Como exemplo podem ser citadas as bases RoCoLe (PARRAGA-ALAVA et al., 2019b), Bracol (ESGARIO; KROHLING; VENTURA, 2019) e Bracot (TASSIS; TOZZI; KROHLING, 2021), que fornecem dados no contexto da agricultura de café e são utilizadas neste trabalho.

Figura 1.2: Exemplo de classificação e segmentação de folhas de café com diferentes níveis de sintomas produzido com o dataset RoCoLe (PARRAGA-ALAVA et al., 2019a).

A partir da utilização dessas bases de dados é possível treinar e testar modelos de aprendizado profundo desenvolvidos na literatura e atestar sua eficiência quando comparados à estudos anteriores de outros autores. Essa comparação é feita através de métricas que corroboram para a precisão dos resultados obtidos.

1.1 Objetivo Principal

Este trabalho busca avaliar a viabilidade da utilização do modelo Yolact++ (BOLYA et al., 2019a) para a segmentação de instâncias de sintomas em plantações de café para doenças comuns a esse tipo de colheita. Ademais, comparar os resultados obtidos com o modelo Mask R-CNN (HE et al., 2017).

1.2 Objetivos Específicos

Para atingir o objetivo proposto anteriormente, três bases de dados são analisadas em diferentes momentos. Para comparar o desempenho do modelo escolhido, foi utilizado um artigo base no mesmo contexto de estudo da agricultura de café escrito por Tassis, Tozzi e Krohling (TASSIS; TOZZI; KROHLING, 2021).

1.3 Organização do Trabalho

Para a fácil leitura e compreensão do trabalho, este foi dividido em 4 capítulos a seguir. O Capítulo 2 expõe os conceitos chave para o entendimento do método utilizado, bem como uma descrição detalhada das arquiteturas analisadas. Em seguida,

no Capítulo 3 são apresentadas as bases de dados escolhidas e modificações realizadas, bem como as configurações de software e hardware utilizadas e os passos realizados para a obtenção dos resultados. O Capítulo 4 conta com os resultados obtidos em cada etapa e uma análise crítica dos pontos fortes e das limitações encontradas. Por fim, no Capítulo 5, discorre-se sobre as conclusões obtidas durante o desenvolvimento do trabalho e quais perspectivas futuras surgiram do final do mesmo.

Capítulo 2

Fundamentação Teórica

Os conceitos e seus entendimentos fundamentais e necessários para a compreensão das técnicas, termos e escolhas feitas por este trabalho nos demais capítulos são detalhados neste capítulo.

2.1 Aprendizado de Máquina

O Aprendizado de Máquina, é um subcampo da inteligência artificial que estuda técnicas para que, através de algoritmos, uma máquina seja capaz de executar e se adaptar a uma tarefa a partir de um conjunto de dados. É possível definir esse aprendizado como um programa que, a partir da experiência, consegue melhorar sua performance em um objetivo definido (MITCHELL, 1997).

Dependendo do tipo de problema e de como os dados estão estruturados, o aprendizado de máquina pode ser dividido das seguintes formas (GéRON, 2019):

- **Supervisionado** [1]: nesse caso, o conjunto de treinamento possui os dados em conjunto com seus respectivos resultados desejados, denominados *labels*. A partir deles a máquina faz a otimização de seus parâmetros;
- **Não supervisionado**: ao contrário do anterior, nessa modalidade, o programa não possui nenhuma informação sobre os dados iniciais. Dessa forma o aprendizado se dá a partir dos padrões e tendências do conjunto de treinamento;
- **Semi-supervisionado**: o modelo possui alguns dados com *label* e a maioria sem informação. Dessa forma, sendo muitas vezes uma junção de técnicas supervisionadas e não supervisionadas;

[1] Esse tipo de treinamento é o que foi utilizado para o trabalho.

- **Aprendizado por reforço**: o agente executa ações baseadas no contexto que está inserido e recebe recompensas ou penalidades, dessa forma, ajustando seu aprendizado.

Para o aprendizado de máquina em conjuntos de imagens, a área que mais se destaca é a de aprendizado profundo, descrita em detalhes na seção seguinte.

2.2 Aprendizado Profundo

Uma das dificuldades encontradas por algoritmos clássicos de aprendizado de máquina se dá a partir da análise de dados complexos como imagens, áudio, ou textos. Isso deriva do desafio de selecionar quais características específicas são mais relevantes para descrever o objeto de estudo. Essas características são denominadas *features* e dados como os citados anteriormente possuem uma grande variedade de fatores que podem influenciar no resultado final. Por exemplo, para analisar uma gravação de áudio devem ser levados em conta: sexo, idade, sotaque, idioma, assunto, entre outros (GOODFELLOW; BENGIO; COURVILLE, 2016).

O aprendizado profundo é uma subárea do aprendizado de máquina que busca resolver essa dificuldade aprendendo essas representações sucessivamente em camadas de complexidade crescente. O termo "profundo"se dá a partir pelo grande número de camadas utilizadas para extrair com maior precisão as features dos objetos de estudo. Dessa forma a profundidade de um modelo é definida a partir de quantas camadas são utilizadas. (CHOLLET, 2018).

A implementação desses sistemas em camadas se apresenta geralmente a partir de redes neurais artificiais, cujo nome é inspirado pela neurobiologia, porém na prática não deve ser entendido como uma representação do cérebro humano, uma vez que não existe evidência que o cérebro implemente nenhum dos mecanismos utilizados no aprendizado profundo atualmente (CHOLLET, 2018). Essas redes podem ter uma quantidade muito grande de camadas e exigem um poder de processamento relevante, na maioria das vezes suprido através de uma *Graphics Processing Unit (GPU)*.

2.3 Detecção e segmentação de objetos em uma imagem

Ao analisar imagens, um dos principais desafios é localizar e classificar as áreas que correspondem ao objetivo do estudo, que pode ser desde um objeto bem defi-

nido como uma bicicleta ou uma área de uma folha afetada por uma doença. Dependendo do que está sendo avaliado, essas informações podem ser extraídas de maneiras diferentes. Dentre elas, podem se destacar a detecção de objetos, segmentação semântica e segmentação de instâncias.

2.3.1 Detecção de Objetos

A detecção de objetos é uma tarefa que envolve dois passos principais: localizar as coordenadas de uma ou mais ocorrências de um objeto em uma imagem e classificá-las de acordo com suas características. Visualmente isso é feito pela inserção de retângulos, denominados *bounding boxes* ao redor de cada um (ELGENDY, 2020), como pode ser observado na Figura 2.1, que exemplifica uma classificação de imagem de um gato e a detecção de objetos e sua representação visual por meio de retângulos de diferentes cores de acordo com a classe de cada um.

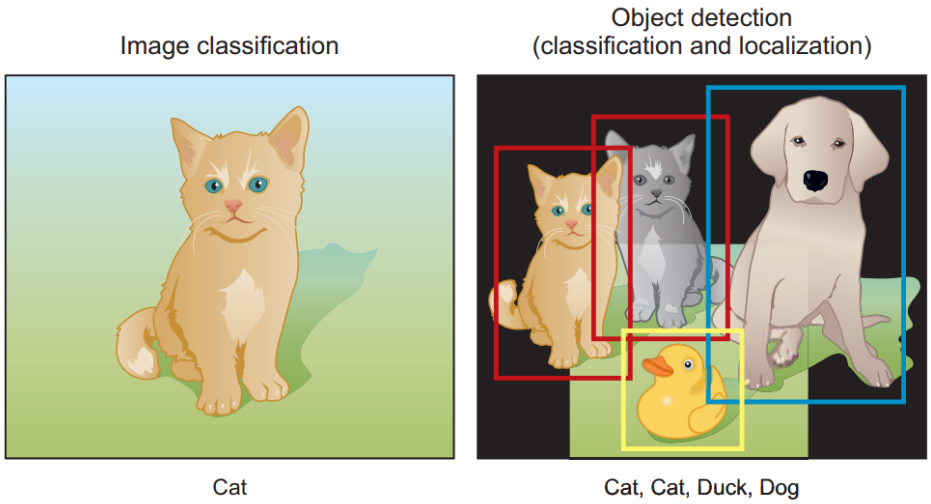

Figura 2.1: Exemplo de classificação (imagem da esquerda) e detecção de objetos (imagem da direita). Fonte (ELGENDY, 2020).

2.3.2 Segmentação Semântica

A segmentação semântica tem como objetivo a classificação da pixel a pixel em suas respectivas classes na imagem selecionada. Dessa forma, é possível obter uma máscara da imagem com as delimitações exatas de cada classe. Esse tipo de segmentação não leva em consideração a quantidade de objetos ou a qual instância desses objetos cada pixel pertence, como pode ser visto na Figura 2.2 em que cada classe é representada por uma única cor (pessoas em vermelho e carros em azul)

e não existe distinção entre as instâncias de cada classe. Uma das aplicações relevantes dessa técnica é no desenvolvimento de carros autônomos para distinguir partes da imagem que são pista, prédios e árvores.

Figura 2.2: Exemplo de segmentação semântica. Fonte (GéRON, 2019) .

2.3.3 Segmentação de Instâncias

A segmentação de instâncias [2] busca unir a detecção de objetos e a segmentação semântica, que foram explicadas anteriormente. Nela os objetos e suas respectivas *bounding boxes* são localizados e, para cada instância, é realizada a segmentação semântica, gerando uma máscara do objeto detectado naquela instância. Dessa forma proporcionando um maior discernimento acerca dos objetos de estudo na imagem. Um exemplo dessa técnica pode ser observado na Figura 2.3, na qual é realizada a segmentação de instância de diversos objetos, como carros, semáforos e pedestres. Note que cada instância encontrada é representada em uma cor diferente para expressar a diferenciação entre cada uma das ocorrências dos objetos.

2.4 Anotações COCO

O Microsoft Common Objects in Context (COCO), disponível publicamente em (LIN et al., 2014), consiste em uma base de dados de larga escala que busca proporcionar avanços na área de visão computacional, especialmente em detectar e classificar objetos em contextos complexos e variados.

Ao longo do tempo o COCO teve grande relevância em testar métricas de novos modelos de *machine learning*, e comumente é utilizado como base de testes

[2] Essa foi a técnica usada de forma majoritária no trabalho.

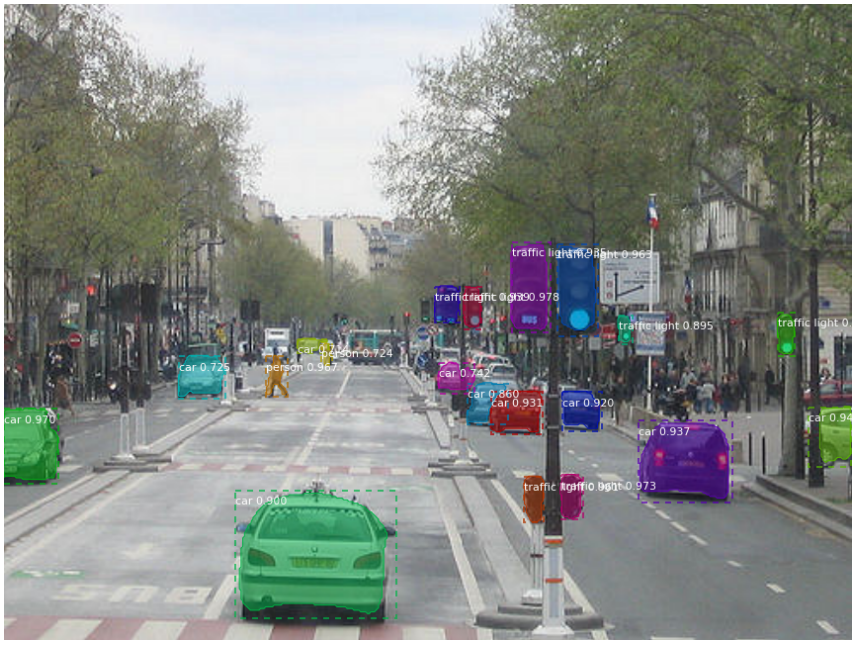

Figura 2.3: Exemplo de segmentação de instâncias. Fonte (matterport, 2022) .

primária de novos estudos, como por exemplo o Mask R-CNN (HE et al., 2017) e o Yolact++ (BOLYA et al., 2019a) utilizam resultados obtidos nessa base para embasar a eficácia de suas arquiteturas. Por esse motivo, muitos modelos mais recentes, como os citados anteriormente, utilizam o formato de anotações COCO como padrão para análise. No caso deste trabalho o Yolact (Seção 2.6) utiliza esse tipo de padrão.

O formato padrão de anotações, que contém todas as informações acerca da classe e da segmentação dos objetos, no contexto de tarefas de segmentação de instâncias segue o seguinte modelo [3]:

```
Estrutura Geral
{
    "info": info // Informacoes gerais da base de dados,
    "images": [image] // Lista com os nomes das imagens,
    "annotations": [annotation] // Lista de anotacoes de cada imagem,
    "licenses": [license] // Licenca de uso da base
}

Estrutura annotations
{
```

[3]Todas as modificações descritas na Seção 3.3 foram baseadas no padrão COCO

```
    "id": int, // Identificador da anotacao
    "image_id": int, // Identificador da imagem
    "category_id": int,  // Classe da imagem
    "segmentation": RLE or [polygon], // Coordenadas da segmentacao
    "area": float, // Area da segmentacao
    "bbox": [x,y,width,height], // Coordenadas do objeto
    "iscrowd": 0 or 1, // Indica de objeto esta em um grupo
}
```

2.5 Mask R-CNN

O modelo Mask R-CNN, proposto em (HE et al., 2017), incrementa modelos anteriores para resolver problemas de segmentação de instância, gerando máscaras de alta qualidade com pouco custo computacional adicional. O Mask R-CNN tem grande parte de sua arquitetura (Figura 2.4) derivada do Faster R-CNN (REN et al., 2015), adicionando uma ramificação extra para o cálculo das máscaras de cada instância.

Arquitetura

A base do modelo segue os passos da Faster R-CNN, na qual a imagem inteira passa por uma rede neural convolucional a fim de que seja extraído uma matriz de features (*feature map*). Em seguida são executadas duas etapas:

- **Region Proposal Network**: nesse passo o *feature map* passa por uma Fully Convolutional Neural Network (FCNN) que prediz regiões de interesse. Na Figura 2.4, essa etapa é representada pelo retângulo a esquerda;
- **Bounding Box, Classe e Máscara**: na segunda etapa, as regiões são processadas por uma etapa denominada *RoI Align*, na qual as informações relevantes são mantidas ao máximo. Por fim, como no Faster R-CNN, são identificadas as *bounding boxes* e a respectiva classe da área escolhida. Em adição, o modelo insere uma terceira camada, que para cada região de interesse, prediz uma máscara para o objeto. Na Figura 2.4, essa etapa é representada pelos dois retângulos menores da direita e o paralelepípedo acima deles;

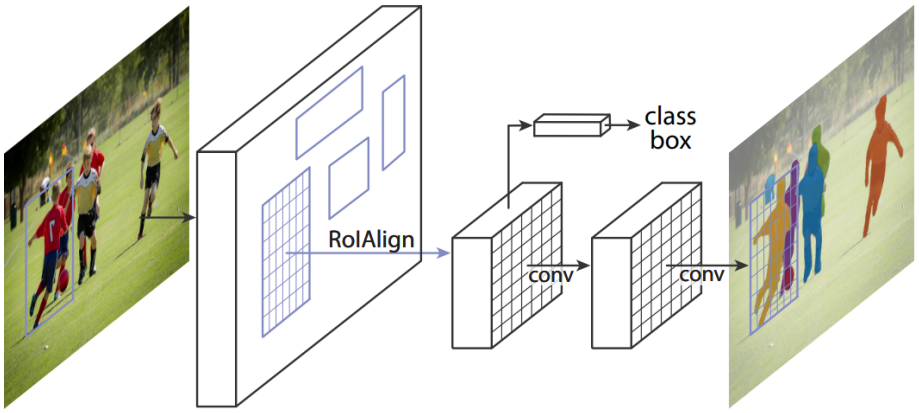

Figura 2.4: Framework Mask R-CNN. Fonte (HE et al., 2017) .

2.6 Yolact++ (You Only Look at Coeficients)

O Yolact (BOLYA et al., 2019b) e o Yolact++ (BOLYA et al., 2019a) trazem como principal avanço a possibilidade da segmentação de instâncias em tempo real (> 30 fps). De acordo com os autores, o modelo foi capaz de performar com rapidez muito maior que outras arquiteturas, como o Mask R-CNN (HE et al., 2017) citado anteriormente. O modelo, diferentemente do Mask R-CNN, realiza a detecção de objetos em apenas um passo, sem a necessidade de *pooling* nas áreas de interesse extraídas.

Arquitetura

A arquitetura do Yolact (figura 2.5) pode ser explicada em quatro etapas: extração de *features*, gerador de protótipos, cálculo de coeficientes e construção da máscara.

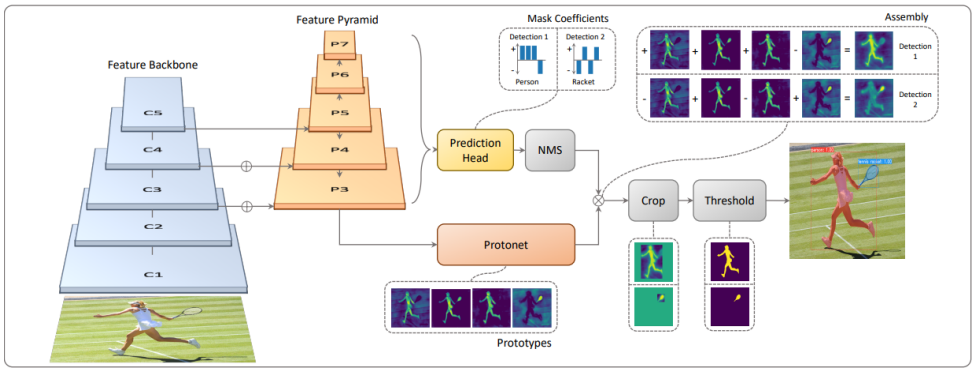

Figura 2.5: Arquitetura Yolact. Fonte (BOLYA et al., 2019b) .

Extração de features

Para a extração de características, a imagem original passa por um *feature backbone* já conhecido, que é o ResNet-101 seguido de uma *feature pyramid network*. Dessa forma obtendo máscaras mais robustas e de melhor resolução ao final do processo (Figura 2.6).

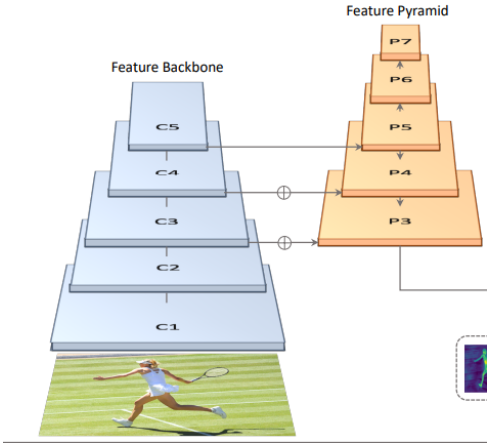

Figura 2.6: Backbone e FPN. Fonte (BOLYA et al., 2019b) .

Geração de protótipos

O gerador de protótipos prediz um conjunto de k máscaras protótipo para a imagem original. Essa *protonet* consiste em uma FCNN (Figura 2.7) com duas camadas de convolução 3×3 e uma final 1×1 para gerar uma saída de k predições de máscara. Nesse estágio nenhuma função de perda é levada em consideração pela rede.

Cálculo de coeficientes

A partir dos protótipos gerados no passo anterior, para cada um dos k protótipos é calculado um coeficiente para a confiabilidade da máscara. Isso é feito adicionando uma camada adicional à parte de classificação e detecção de objeto (Figura 2.8).

Construção da máscara

Ao final, é feita uma simples combinação linear entre os coeficientes de máscara e os protótipos. As operações são feitas a partir de uma multiplicação de matrizes e uma sigmóide:

$$M = \sigma(PC^T) \tag{2.1}$$

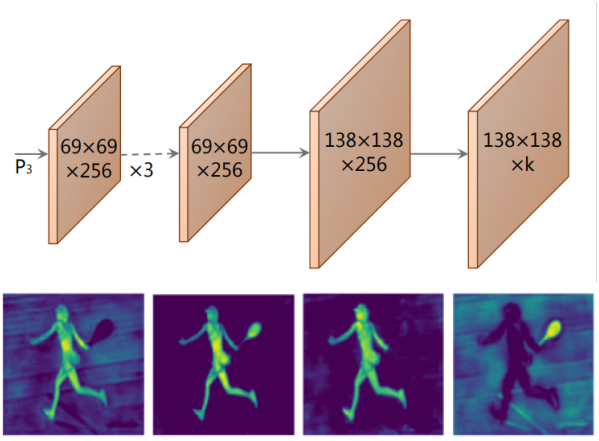

Figura 2.7: Na imagem superior a protonet, que gera as predições de protótipos e na imagem inferior um exemplo de protótipos gerados. Nesse caso $k = 4$. Fonte (BOLYA et al., 2019b) .

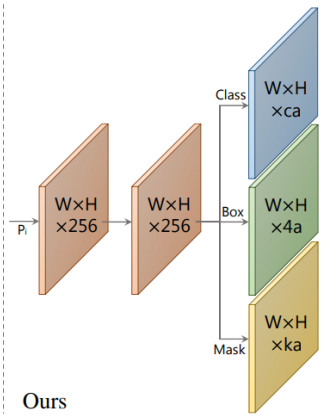

Figura 2.8: Rede para calcular os coeficientes para cada máscara. Fonte (BOLYA et al., 2019b) .

onde P é a matriz de protótipos, C a matriz de coeficientes, M a máscara gerada e σ a função sigmóide. Essa operação é exemplificada na Figura 2.9, na qual as imagens representam as máscaras protótipo e os símbolos de subtração e adição os coeficientes.

2.7 Métricas de avaliação utilizadas

Para avaliar o desempenho de um modelo é preciso escolher métricas que formalizem de maneira adequada os resultados. Neste trabalho, a principal métrica utili-

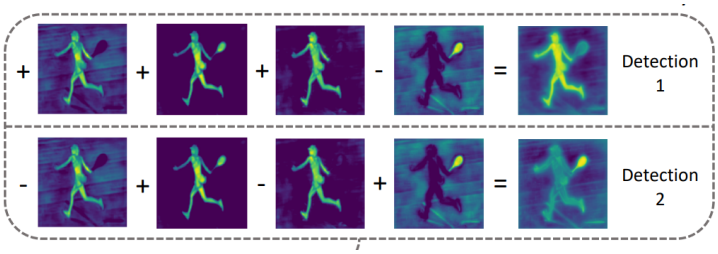

Figura 2.9: Construção das máscaras a partir dos coeficientes e protótipos. Fonte (BOLYA et al., 2019b) .

zada foi a de *mean average precision (mAP)*, mas para entendê-la são necessários alguns conceitos chave descritos a seguir.

Intersection Over Union (IoU)

Essa métrica mede a taxa de coincidência da área da predição e da área verdadeira do objeto de interesse. Também conhecida como índice de Jaccard, essa métrica vai de 0 a 1, sendo o menor valor sem nenhuma interseção e o maior quando as áreas se alinham perfeitamente. O cálculo é feito a partir da Equação 2.2 e pode ser visualizado na figura 2.10, na qual o retângulo vermelho corresponde a base verdade e o retângulo em azul a *bouding box* predita. A partir delas o cálculo do IoU é exemplificado com as áreas de união e interseção em azul.

A equação a seguir representa o cálculo de IoU, onde A corresponde a área verdade da posição do objeto e B a área predita:

$$IoU\ (A, B) = \frac{|A\ \cap B|}{|A\ \cup\ B|} \tag{2.2}$$

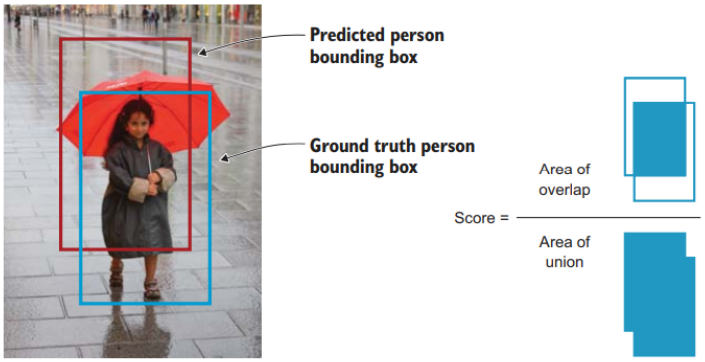

Figura 2.10: Cálculo de IoU para predição de uma *bounding box*. Fonte (ELGENDY, 2020) .

Matriz de confusão

A matriz de confusão ilustra uma visão geral de como as predições do modelo se comparam aos dados verdadeiros a partir de um limite definido, que também pode ser definido como *threshold*. Os resultados podem ser divididos em quatro categorias:

- **True Positives (TP)**: uma predição correta, com o IoU com valor superior a um limite mínimo definido;
- **True Negatives (TN)** uma predição correta, onde o modelo não detecta nenhum objeto e esta é a realidade;
- **False Positives (FP)**: quando a previsão realizada possui um IoU com a verdade menor que o limite mínimo definido.
- **False Negatives (FN)**: quando o modelo não detecta um objeto, mas na realidade ele está presente na imagem.

Precisão e Revocação

A precisão mede o quão acurados são os resultados. Do total de previsões, quantas foram corretas:

$$Precisão = \frac{TP}{TP + FP} \tag{2.3}$$

A revocação mede a capacidade do modelo de encontrar predições verdadeiras:

$$Revocação = \frac{TP}{TP + FN} \tag{2.4}$$

Mean Average Precision (mAP)

A partir das medidas de precisão e revocação é possível construir um gráfico que relaciona as duas. A área abaixo da curva desse gráfico é denominada precisão média (*average precision* AP). Por fim, a média do cálculo de AP para todas as classes é denominado *mean average precision*. Ela vai de 0 a 100 e é uma das métricas mais utilizadas para avaliar modelos de detecção de objetos e segmentação (KISHIDA, 2005).

Capítulo 3

Materiais e Métodos

Os recursos de software e hardware escolhidos para o desenvolvimento deste trabalho são descritos em detalhes nesse capítulo, bem como as fontes de dados utilizadas e as manipulações realizadas para que se adequassem ao objetivo proposto. Ainda, os passos incrementais para a coleta de resultados e o modelo proposto estão descritos abaixo[1].

A proposta do trabalho foi dividida de acordo com as bases de dados utilizadas e estão listadas a seguir.

3.1 Configurações de Hardware

No desenvolvimento de técnicas de Aprendizagem de Máquina, é possível utilizar os recursos obtidos com a *Central Process Unit* (CPU) ou com a *Graphics Processing Unit* (GPU). A primeira tendo ótimo desempenho em cálculos matemáticos, porém com limitações em executar diversos processos ao mesmo tempo, dessa forma, sendo mais indicada para tarefas específicas e de difícil paralelização. A segunda, por sua vez tem destaque em operações computacionais que podem ser divididas em diversas partes menores e executadas em paralelo. Desta forma, no caso de aplicações de aprendizagem profunda, a GPU se torna a opção mais viável para utilização.

3.1.1 Configuração Google Colaboratory

O ambiente utilizado para rodar os modelos escolhidos para o trabalho foi o Google Colaboratory[2] e para uma maior disponibilidade de recursos foi contratado o

[1] Os códigos desenvolvidos para cada etapa do projeto estão disponíveis no Github do autor e podem ser acessados em <https://github.com/guilodron>

[2] Colab (2021)

plano PRO da ferramenta. A configuração de hardware disponibilizada possuiu uma Graphics Processing Unit *(GPU)* NVIDIA Tesla P100 com 16 GB de memória, uma *Central Process Unit* (CPU) *Intel* ® *Xeon* ® 2.30 GHz, um espaço de disco de 167 GB e 13 GB de memória RAM. A ferramenta escolhida, em sua versão Pro, contou com uma disponibilidade maior de tempo de sessão para a execução das atividades, todavia ainda mostrou limitações ao rodar os modelos por tempo prolongado, o que resultou, em vezes, na perda do progresso obtido. Em conjunto com o Colab, foi utilizado o Google Drive para o armazenamento dos resultados obtidos, uma vez que os arquivos do ambiente remoto são removidos ao fim da execução da máquina.

3.1.2 Configuração Local

Para pequenos testes e para a maior parte da manipulação das bases de dados, foi utilizada uma máquina pessoal que se mostrou mais prática no armazenamento e livre manipulação dos arquivos. O computador possui uma CPU AMD Ryzen 5 3600 3.60GHZ, uma GPU NVIDIA GeForce RTX 2060 e 16GB de RAM.

3.2 Configurações de *Software*

Para garantir a produtividade do desenvolvimento é importante que as ferramentas de *software* escolhidas sejam eficientes e possuam um bom suporte e documentação. Neste trabalho buscou-se utilizar as linguagens de programação e bibliotecas mais competitivas no mercado, que estão detalhadas a seguir.

3.2.1 Ambiente

Nos últimos anos, a linguagem Python teve um grande crescimento de popularidade e hoje em dia se configura como uma das linguagens mais utilizadas no mercado, fato que é refletido pela colocação em primeiro lugar no índice TIOBE[3]. Além disso, grande parte da comunidade de Aprendizado de Máquina, bem como o artigo base utilizado para este estudo (TASSIS; TOZZI; KROHLING, 2021), utilizam essa linguagem de programação. Dessa forma, o Python foi escolhido como ferramenta de programação para o desenvolvimento das atividades. No ambiente remoto (Seção 3.1) a versão da linguagem utilizada foi a 3.7.13 e localmente a versão utilizada foi a 3.9.12.

[3]TIOBE... (2022)

Para organizar os códigos foi utilizado os cadernos Jupyter. A principal motivação para essa organização foi a capacidade dos cadernos de armazenarem as saídas de cada comando executado e permitir adicionar explicações visualmente agradáveis para descrever o que está sendo incrementado em cada etapa. O google colaboratory[4] já possui um suporte nativo a ferramenta, já para o desenvolvimento local foi utilizado o editor de texto Visual Studio Code[5], que é altamente customizável e de livre acesso, não se limitando a linguagem python.

Também foi utilizado o Git[6] para o versionamento de código e disponibilização dos códigos, imagens e anotações geradas durante o projeto.

3.2.2 Bibliotecas

Devido a popularidade da linguagem Python e seu uso na comunidade de desenvolvimento de Aprendizado de máquina, discutidos anteriormente, existem diversas bibliotecas que abstraem a implementação de tarefas comuns. Para as operações, visualização e armazenamento de imagens foram utilizadass as bibliotecas OpenCV[7] na versão 4.5.5 e Numpy[8] na versão 1.21.5.

Para a exibição de planilhas de dados e das métricas foram utilizadas as bibliotecas Pandas[9] na versão 1.4.3 e Matplotlib[10] na versão 3.5.1, tendo em vista que ambos funcionam bem visualmente com a abordagem de cadernos Jupyter.

Por fim, a implementação do modelo Yolact++ escolhida foi a disponibilizada por Bolya publicamente[11]. A biblioteca é implementada utilizando o Pytorch [12] na versão 1.0.1 e disponibiliza as funcionalidades necessárias para o treinamento, visualização e teste das bases de dados escolhidas com grande praticidade.

3.3 Bases de Dados

As bases de dados são componentes essencias para o desenvolvimento científico e a validação de novos estudos. Em geral, são coleções de imagens previamente classificadas por especialistas e divididas em grupos de treinamento, teste e validação.

[4]Colab (2021)
[5]VSCODE (2022)
[6]Git (2022)
[7]OpenCV (2022)
[8]Harris et al. (2020)
[9]team (2020)
[10]Hunter (2007)
[11]Bolya et al. (2019a)
[12]Paszke et al. (2019)

Tabela 3.1: Número de imagens de cada classe na base RoCoLe.

Classe	Quantidade de imagens
*H*ealthy	791
Red spider mite	167
Rust level 1	344
Rust level 2	166
Rust level 3	62
Rust level 4	30

Para ter uma melhor análise sobre o desempenho do modelo, foram escolhidas algumas bases de dados disponíveis publicamente e que se adequavam aos propósitos do estudo. Algumas alterações foram feitas para ajustá-las ao contexto de segmentação de instâncias.

3.3.1 RoCole

A base de dados RoCoLe (PARRAGA-ALAVA et al., 2019a), disponível publicamente, contém imagens de folhas de café para tarefas de segmentação de objetos, classificação binária e de mais de uma classe. As imagens foram tiradas utilizando uma máquina fotográfica de 5 MP em uma distância de 2 – 3 cm, levando em consideração folhas de uma área de plantação de 390 cafeeiros.

As imagens foram obtidas em condições reais de ambiente, com diferentes faixas de iluminação, temperatura e outras plantas ao fundo. No total, são 1560 imagens com anotações e classificações entre folhas saudáveis e doentes para classificação binária. Para a classificação multiclasse são divididas em seis classes, que correspondem aos sintomas apresentado na folha, de acordo com a Tabela 3.1 e ilustrado na Figura 3.1.

Modificações

O RoCoLe fornece apenas duas classes para tarefas de segmentação de instâncias: healthy (saudável) e unhealthy (afetada por sintoma). A fim de segmentar as folhas e ao mesmo tempo classificar qual doença está afetando-a, foi realizado um mapeamento das classes com seus respectivos objetos, dessa forma gerando um novo conjunto de anotações, como na figura 3.2, onde em destaque está a segmentação de uma folha e em texto a classificação do sintoma mapeado.

As modificações e o código estão disponíveis em Rodrigues (2022). A base de dados modificada foi composta por 1490 imagens, pelo fato de algumas terem sido perdidas no mapeamento das respectivas classes. Estas foram divididas em 30 e 70

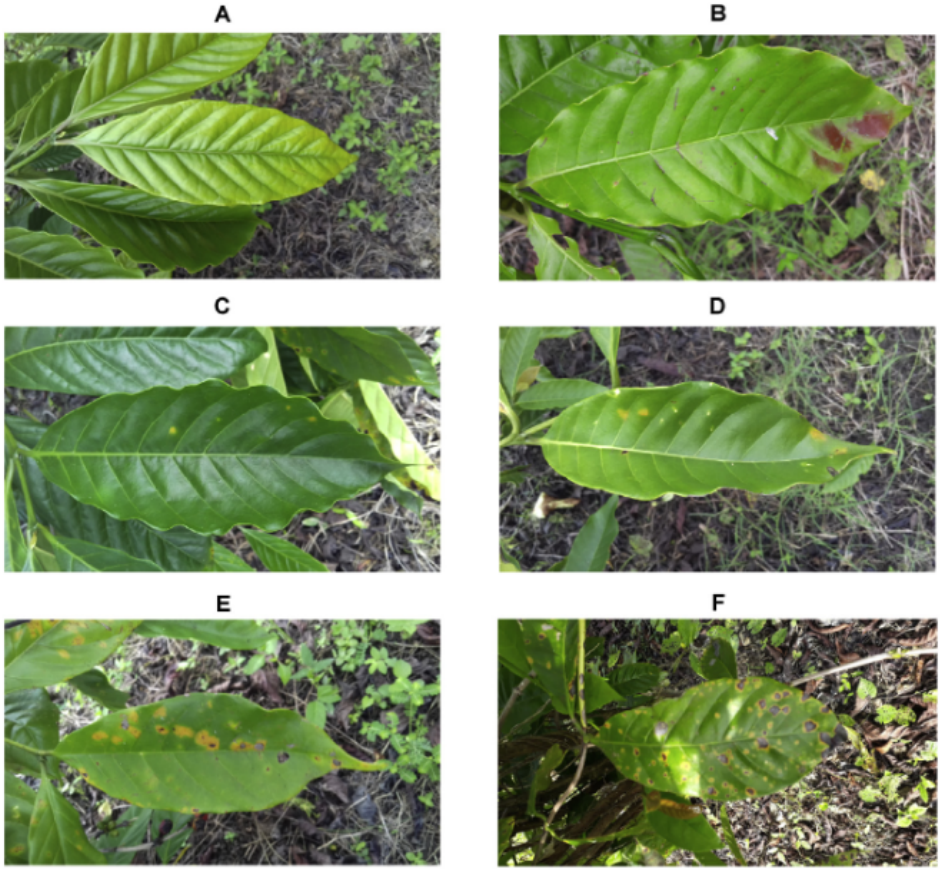

Figura 3.1: Extraída do artigo original (PARRAGA-ALAVA et al., 2019a) contendo os diferentes tipos de classe da base de dados. A) healthy. B) Red Spider Mite. C) Rust level 1. D) Rust level 2. E) Rust level 3. F) Rust level 4.

porcento para teste e treinamento, respectivamente. Essa nova base gerada foi a utilizada durante o trabalho.

3.3.2 Bracol

A base de dados Bracol (ESGARIO; KROHLING; VENTURA, 2019) contém imagens de folhas de café saudáveis e afetadas por quatro diferentes tipos de doença. As imagens foram coletadas por meio de smartphones (ASUS Zenfone 2, Xiaomi Redmi 5A, Xiaomi S2, Galaxy S8 and Iphone 6S) na região do Espirio Santo, Brasil.

Cada imagem do dataset é composta por uma folha em um fundo branco e a partir dessas foram gerados três conjuntos de dados:

Leaf Dataset: É formado pelo total de 1747 imagens de folhas, com a classificação acerca da existência de uma doença ou não e sua severidade feitas com o

Figura 3.2: Imagem com anotação após o mapeamento das classes.

Tabela 3.2: Número de imagens de cada classe no Semantic Segmantation Dataset do Bracol.

Classe	Quantidade de imagens
*H*ealthy	100
Leaf miner	117
Rust	119
Leaf Spot	121
Cercospora	33

auxílio de um especialista. A divisão de classes e os sintomas predominantes de cada imagens estão contidos em uma planilha 'dataset.csv'.

Sympton Dataset: Composto por fatias isoladas de cada instância de doença nas imagens do Leaf Dataset de modo que apenas um estado de saúde é apresentado por imagem. No total ele é composto por 2147 imagens e é dividido em pastas de acordo com o sintoma observado. Este se encaixa em problemas de classificação de imagens.

Semantic Segmentation Dataset: Contém um conjunto de 490 imagens de folhas selecionadas do Leaf Dataset, sendo que, para cada uma existe uma máscara representando a área afetada por algum dos quatro sintomas. Ele é dividido em três subpastas: treinamento, validação e teste. A proporção de imagens com os sintomas predominantes podem ser vistos na Tabela 3.2 e um exemplo é ilustrado na Figura 3.3, onde à esquerda são representadas as imagens originais das folhas e à direita as respectivas máscaras.

Para os propósitos deste trabalho apenas foi utilizado o Semantic Segmentation Dataset. Todavia, algumas alterações tiveram que ser feitas para transformá-lo em

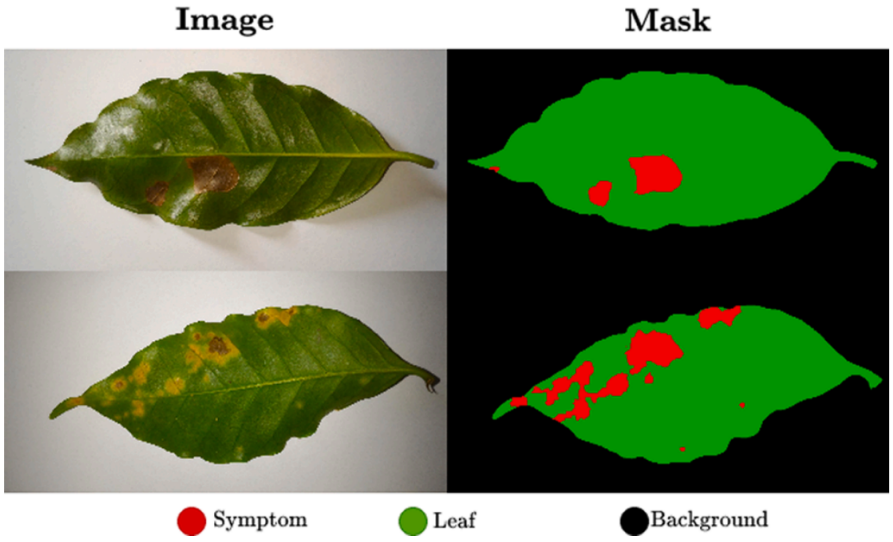

Figura 3.3: Exemplos de imagens e suas respectivas máscaras no Semantic Segmantation Dataset do Bracol. Fonte (ESGARIO; KROHLING; VENTURA, 2019).

uma base de dados com suporte a segmentação de instâncias. Como descrito a seguir.

Modificações

A fim de adaptar o Bracol para encaixá-lo no contexto de segmentação de instâncias, algumas modificações tiveram que ser feitas, gerando um novo conjunto de dados e anotações. O processo foi dividido em 3 etapas:

- Primeiro, utilizando a planilha fornecida pelo Leaf Dataset, foram isoladas as imagens afetadas por apenas um sintoma, dessa forma garantindo a classificação correta de cada instância. Do total das imagens isoladas foram separadas 43 imagens para teste, 42 para validação e 338 para treinamento;
- Em seguida, utilizando a máscara de cada imagem, utilizando a biblioteca OpenCV[13] foi extraída a parte afetada (em vermelho na Figura 3.3) da folha. A partir do resultado encontrado, para cada elemento isolado na imagem resultante, foram armazenadas informações de área, posição e contorno. Esses elementos representam uma instância individual de um sintoma;
- Por fim, as informações coletadas no passo anterior foram mapeadas em um arquivo de anotações no formato JSON seguindo o padrão COCO;

[13](OPENCV, 2022)

Cada etapa das modificações realizadas é exemplificada na Figura 3.4.

O novo dataset gerado com as anotações e o código utilizado estão disponíveis no repositório Rodrigues (2022) e podem ser utilizados de forma gratuita e livre, desde que seja feita a citação deste trabalho.

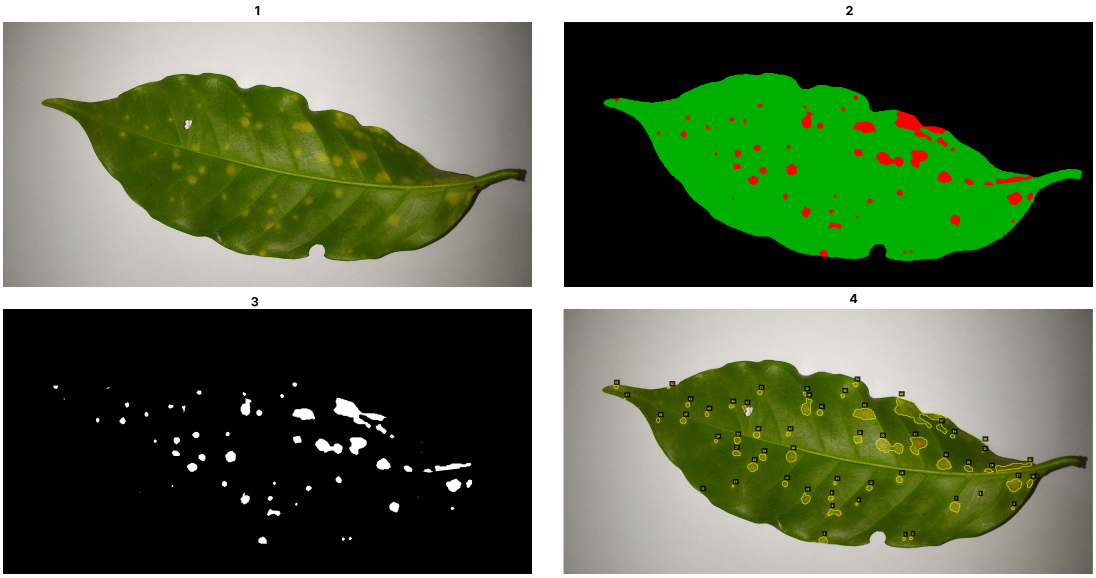

Figura 3.4: Etapas da alteração da base de dados Bracol. 1) Imagem da folha original 2) Máscara com sintomas em vermelho 3) Sintomas isolados em branco 4) Imagem com as anotações das instâncias individuais.

3.3.3 Bracot

A base de dados BRACOT (TASSIS; TOZZI; KROHLING, 2021), disponível publicamente, possui imagens de partes de pés de café arábica com folhas afetadas por doenças comuns desse tipo de plantação. A coleta foi realizada de forma a manter a paisagem natural e aumentar a variabilidade de cenários para o treinamento. No total são 300 imagens registradas com um dispositivo galaxy S8 e delas foram anotadas, por um especialista, 1662 instâncias individuais de folhas de café.

O conjunto é direcionado para tarefas de segmentação de instâncias de folhas de café, não fornecendo informações acerca da classificação das doenças ou a área dos sintomas para as instâncias de folha. Um exemplo dos dados providos pode ser visto na Figura 3.5.

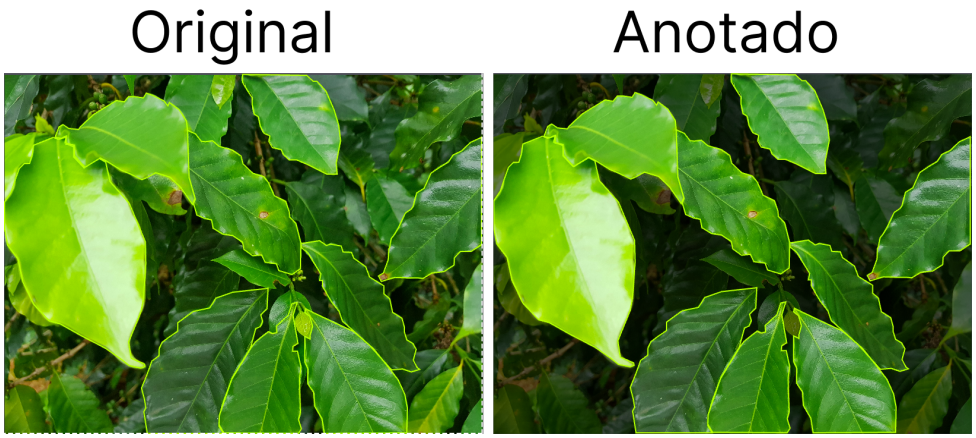

Figura 3.5: Imagem original e instâncias de folhas anotadas.

3.4 Aplicação do modelo nas bases de dados definidas

O contexto prático do estudo foi dividido de acordo com as bases de dados expostas anteriormente. Para cada uma, o Yolact++ foi treinado individualmente em momentos distintos e seus resultados armazenados para análise qualitativa das segmentações e classificações. Ao final, foi proposto uma conjunção de dois modelos treinados nas bases Bracol, apresentada na Seção 3.3.2 e Bracot, apresentada na Seção 3.3.3, para segmentar e analisar folhas individuais na base Bracot. Nesta seção são descritos em detalhes todos os passos executados para o treinamento do modelo, bem como explicações para as escolhas realizadas.

3.4.1 Treinamento na base RoCoLe

O treinamento do modelo Yolact++ foi feito no ambiente google colaboratory[14] utilizando a base de dados RoCoLe em sua versão modificada para contemplar a segmentação das folhas juntamente com a classificação de seus sintomas. Esse passo foi importante para a comparação dos resultados obtidos com um dos artigos utilizados como base para o desenvolvimento deste estudo (TASSIS; TOZZI; KROHLING, 2021).

A base modificada ainda passou por um ajuste de escala por meio da ferramenta Roboflow[15], isso foi feito para conseguir melhores tempos de treinamento e evitar a interrupção do programa por falta de memória alocada. Ainda assim, os impactos

[14] Colab (2021)
[15] Roboflow (2022)

da mudança realizada não se mostraram negativos nas métricas de avaliação e nas imagens finais geradas pelo modelo, como será apresentado na seção de resultados.

As imagens foram redimensionadas para 416×416 pixeis utilizando a plataforma Roboflow[16], citada anteriormente, e dividido em 1041 imagens para treinamento e 449 para validação. O modelo Yolact++ executou durante 72868 iterações, que corresponderam a 560 epochs[17] e a métrica utilizada para avaliação foi a de *mean average precision (mAP)*, pela comparabilidade com a literatura utilizada e por ter uma implementação já nativa da biblioteca empregada.

3.4.2 Treinamento na base Bracot

A avaliação do modelo foi realizada no ambiente google colaboratory[18] utilizando a base de dados original, apresentada na Seção 3.3.3. Em adição, o conjunto de imagens também passou por um redimensionamento com a ferramenta Roboflow (2022), uma vez que a grande resolução das imagens originais causou alertas de memória no ambiente remoto utilizado.

A base de dados foi redimensionado para 416x416 pixeis, como na seção anterior e dividido em 240 imagens para treinamento e 60 para validação, dessa forma mantendo sua distribuição original. O modelo Yolact++ executou durante 5338 iterações, que corresponderam a 177 epochs e a métrica utilizada para avaliação também foi a de *mean average precision (mAP)*, pelos motivos citados na subseção anterior.

3.4.3 Treinamento na base Bracol

Nesta etapa, para o treinamento do Yolact++, foi utilizado o novo dataset gerado a partir do Bracol, contendo as anotações individuais de cada sintoma em todas as folhas, como descrito na Subseção 3.3.2. No processo de ajuste do dataset, algumas imagens foram perdidas pelo fato de serem afetadas por mais de um tipo de sintoma (Seção 3.3.2, dessa forma, o conjunto de imagens foi dividido em 338 para treinamento, 43 para teste e 42 para validação, sendo que o último grupo foi separado para observar os resultados qualitativos das máscaras geradas, não sendo levado em conta nas funções de perda e ajustes do modelo.

[16] Técnicas de preprocessamento da ferramenta podem ser acessadas em <https://docs.roboflow.com/image-transformations/image-preprocessing>

[17] O número de epochs corresponde ao número de passagens que o algoritmo deu em todas as imagens da base de dados.

[18] Colab (2021)

Para gerar os resultados, o Yolact++ executou no ambiente google colaboratory [19] por 10000 iterações, que corresponderam a 312 epochs. Os pesos do treinamento e as imagens geradas foram armazenadas para a utilização no *pipeline* proposto apresentado posteriormente neste trabalho.

3.4.4 *Pipeline* de extração de sintomas proposto

Tendo em vista os resultados obtidos durante o treinamento com as bases Bracol e Bracot, foi proposto um *pipeline*, que tem como objetivo a segmentação de instâncias de forma automática em todas as folhas detectadas extraídas do dataset Bracot, uma representação do fluxo proposto pode ser visto na Figura 3.6.

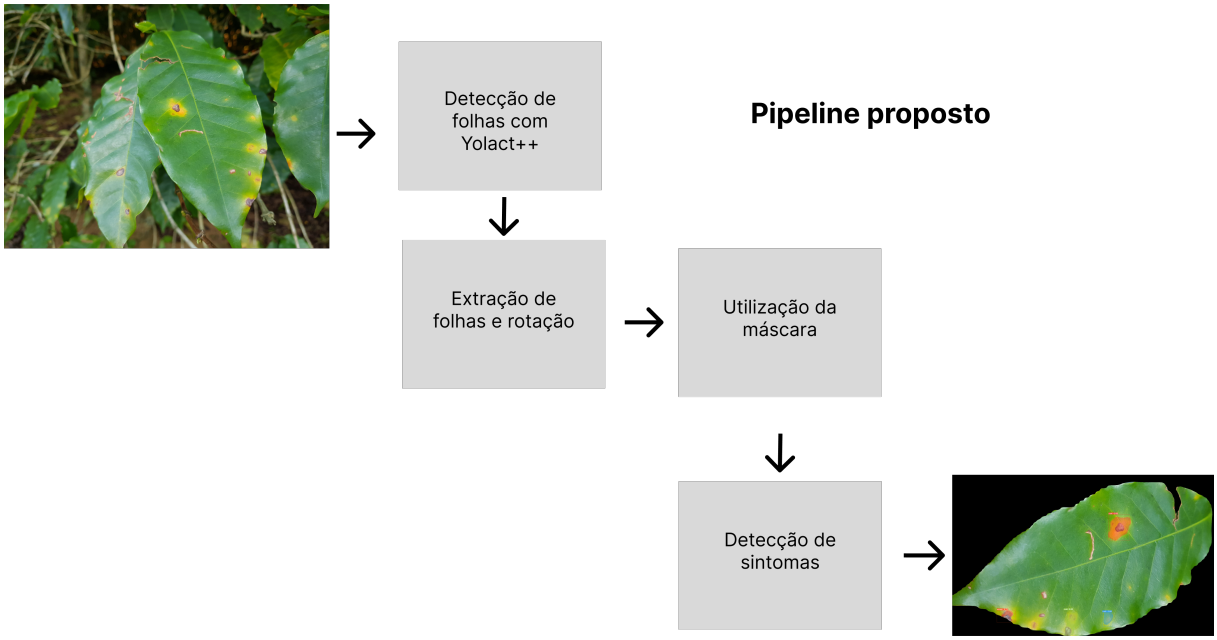

Figura 3.6: Estrutura do pipeline.

O fluxo implementado pode ser dividido em 4 etapas simples:

- **Detecção de folhas com Yolact**: inicialmente o modelo Yolact++ treinado para o dataset Bracot é utilizado para realizar a segmentação de instâncias de todas as folhas da imagem. Feito isso, um arquivo com as previsões é gerado contendo informações sobre as *bounding boxes* e máscaras encontradas, juntamente com a taxa de confiabilidade de cada uma.
- **Extração de folhas e rotação**: Com as previsões feitas pelo modelo é traçado um *treshold* mínimo na confiabilidade para que o programa trate as coorde-

[19] Colab (2021)

nadas como uma possível folha. O utilizado no estudo foi de 80%, pois através de testes com diferentes valores esse gerou os melhores resultados. Ao selecionar as folhas com maior confiança, a *bounding box* predita é utilizada para cortar a região da folha da imagem original. Em seguida é checado se o segmento extraído possui orientação vertical ou horizontal, caso o primeiro caso seja verdadeiro é feita uma rotação de 90 graus para que todas as imagens possuam orientação horizontal. Isso é feito para que o novo conjunto se assemelhe ao Bracol, que será utilizado nos próximos passos.

- **Utilização da máscara**: A máscara extraída para a folha selecionada é utilizada para remover o fundo, que contém informações desnecessárias para a análise. Para esse passo é feita uma simples operação AND em todos os pixeis para eliminar os que não fazem parte da máscara aplicada.
- **Detecção de sintomas**: O modelo treinado com a base Bracol é utilizado para a segmentação de instâncias de possíveis sintomas na folha escolhida.

Após a conclusão de todas as etapas, o fluxo proposto gera uma imagem contendo apenas um folha com suas respectivas predições de sintomas. O processo é repetido para todas as folhas de todas as imagens. Por não possuir um conjunto de anotações analisado por um especialista, a análise da eficácia do *pipeline* teve de ser feita de forma qualitativa a partir da quantidade visual de sintomas segmentados com sucesso. Os resultados e possíveis melhorias são discutidos no capítulo seguinte.

Capítulo 4

Resultados e Discussão

Os resultados obtidos em cada etapa do estudo são detalhados em profundidade neste capítulo. Além disso, um discussão sobre a performance do treinamento em comparação com o artigo de Tassis, Tozzi e Krohling (TASSIS; TOZZI; KROHLING, 2021), utilizado como base, será apresentada para cada seção, bem como possíveis melhorias e limitações do projeto. Para evitar repetição desenecessária, durante a apresentação dos resultados o termo *bounding box* foi reduzido para "bbox"e a segmentação para "mask".

4.1 Treinamento na base RoCoLe

Os resultados na base RoCoLe correspondem ao treinamento durante 34784 iterações, avaliados sobre a métrica de *mean average precision (mAP)*. O artigo base (TASSIS; TOZZI; KROHLING, 2021) também foi levado em consideração na análise dos resultados.

O modelo Yolact++ obteve um mAP total para *bounding box* de 43.12 e 44.76 para segmentação. O que mostrou evolução, quando comparado com o artigo base, que obteve 42.50 e 39.30 para *bounding box* e segmentação, respectivamente. Ademais, o artigo base contou com apenas a segmentação de folhas, sem levar em conta a classificação dos sintomas obtidos. Um comparativo mais detalhado dos resultados é representado na Tabela 4.1.

Também foi avaliado o mAP para cada classe individualmente. As classes que obtiveram melhores resultados foram "Healthy", com valores de mAP de 82.17 para segmentação e 76.48 para *bounding box*, seguida da classe "Rust Level 1", com valores de mAP de 54.89 e 54.60 para os mesmos objetivos. Os resultados detalhados em cada threshold de IoU podem ser observados na Tabela 4.2 e exemplos de resultados podem ser vistos na Figura 4.1, na qual à esquerda estão as imagens com

Tabela 4.1: Comparativo dos resultados do artigo base e do Yolact++.

	Yolact++(BOLYA et al., 2019a)		**Tassis**(TASSIS; TOZZI; KROHLING, 2021)	
	Segmentação	*Bounding Box*	Segmentação	*Bounding Box*
mAP	**44.76**	**43.12**	42.50	39.30
$mAP_{.50}$	52.61	**56.97**	**54.60**	55.20
$mAP_{.75}$	48.67	**48.85**	**48.70**	48.60

as anotações do especialista em destaque e à direita, em marrom, a segmentação predita pelo modelo.

Figura 4.1: Resultados do treinamento na base RoCoLe. A direita a imagem gerada pelo Yolact e a esquerda a imagem original com as anotações do especialista..

Os resultados deixaram evidente que o desbalanceamento do número de imagens entre as classes da base de dados (Tabela 3.1) influenciaram diretamente os resultados obtidos. Classes com menor número de imagens tiveram um resultado pior, tanto ao detectar *bounding boxes*, quanto na segmentação das máscaras. Uma relação entre a quantidade de imagens e o desempenho pode ser observada, nos

Tabela 4.2: Resultados individuais de mAP para cada classe no dataset RoCoLe.

Healthy											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	**76.4**	**93.5**	93.5	92.2	91.3	90.6	**90.4**	85.9	77.2	44.6	5.2
mask	**82.1**	**93.5**	93.5	93.1	93.1	93.1	**92.3**	91.8	87.3	65.7	17.9
Red Spider Mite											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	43.8	61.9	61.9	61.9	58.0	58.0	49.7	47.98	27.4	11.0	0.33
mask	48.0	59.6	59.9	59.6	57.0	54.8	50.9	48.3	44.6	35.2	10.1
Rust Level 1											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	**54.6**	**65.6**	65.6	65.6	65.6	65.6	**65.6**	64.1	56.9	29.9	1.25
mask	**54.8**	**61.9**	61.9	61.9	61.9	60.8	**60.1**	60.1	60.1	47.9	11.6
Rust Level 2											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	38.0	48.0	45.0	45.0	45.0	45.0	44.2	42.9	41.6	22.8	0.16
mask	38.2	41.5	41.5	41.5	41.5	41.5	41.5	41.5	40.7	35.6	14.6
Rust Level 3											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	26.5	31.7	31.7	31.7	30.1	30.1	30.1	30.1	25.0	24.6	0.16
mask	31.8	33.9	33.9	33.9	33.9	33.9	33.9	33.9	33.9	29.8	17.2
Rust Level 4											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	19.2	40.9	24.7	24.7	24.7	24.7	12.9	12.9	12.9	11.1	2.38
mask	13.4	24.9	13.0	13.0	13.0	13.0	13.0	13.0	13.0	13.0	4.55

Tabela 4.3: Comparativo dos resultados do artigo base e do Yolact++ para o dataset Bracot.

	Yolact++(BOLYA et al., 2019a)		**Tassis**(TASSIS; TOZZI; KROHLING, 2021)	
	Segmentação	*Bounding Box*	Segmentação	*Bounding Box*
mAP	62.23	54.08	**73.98**	**70.70**
$mAP_{.50}$	75.66	77.02	**81.41**	**82.22**
$mAP_{.75}$	69.85	68.56	**80.36**	**80.37**

Tabela 4.4: Resultados para diferentes thresholds de IoU no dataset bracot.

Folhas											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	54.0	77.0	76.8	76.1	75.2	71.7	68.5	55.7	31.6	7.58	0.23
mask	62.2	75.6	75.6	75.5	73.2	71.8	69.8	67.2	69.3	43.8	10.1

gráficos da Figura 4.2, tanto para o mAP de *bounding box*, quanto para o de segmentação.

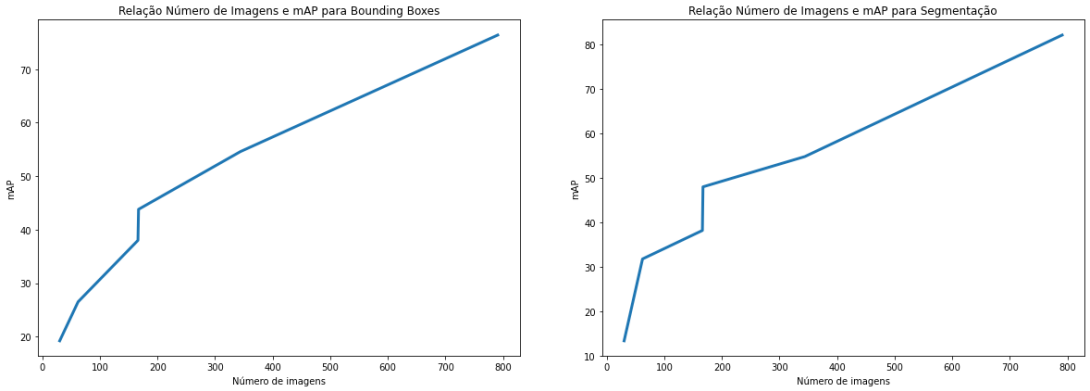

Figura 4.2: Gráficos com a relação entre número de imagens de resultados para mAP.

4.2 Treinamento na base Bracot

Os resultados do treinamento na base Bracot correspondem a 5338 iterações. O mAP obtido durante o treinamento se mostrou inferior ao artigo base (TASSIS; TOZZI; KROHLING, 2021), atingindo 54.08 para *bounding box* e 62.23 para segmentação das folhas. Em contrapartida Tassis, Tozzi e Krohling obtiveram 70.70 para *bounding box* e 73.98 para segmentação das folhas. O comparativo entre os resultados pode ser observado na Tabela 4.3, e os resultados completos para cada threshold de IoU na Tabela 4.4.

O desempenho pior no modelo Yolact++ foi de acordo com o esperado no início do projeto, uma vez que o modelo perde um pouco de precisão em detrimento da agilidade na análise das imagens e na obtenção de máscaras de maior qualidade. Essa diferença de qualidade pôde ser observada nessa base de dados e está representada na Figura 4.3

Figura 4.3: Diferença de qualidade das máscaras em comparação às geradas pelo Mask R-CNN (HE et al., 2017) no artigo base. À esquerda a imagem retirada do artigo base e à direita a imagem gerada pelo Yolact++ neste trabalho.

Apesar do conjunto usado para o treinamento ter sido redimensionado para um tamanho menor do que o original, os resultados nas imagens originais permaneceram com alta resolução e riqueza de detalhes. Exemplos de saída do modelo podem ser vistos na Figura 4.4.

4.3 Treinamento na base Bracol

Os resultados apresentados para a base de dados Bracol correspondem ao treinamento durante 10 mil iterações. Para o estudo, foi utilizado o novo conjunto de dados gerado para segmentação de instâncias explicado na Subseção **??**. O artigo base (TASSIS; TOZZI; KROHLING, 2021) não realizou testes no sentido de segmentação de instâncias, apenas apresentando resultados de segmentação semântica. Os autores obtiveram uma taxa de 94.25% de IoU médio para segmentação dos sintomas.

Para o novo dataset, o Yolact++ obteve um mAP total de 49.78 para detecção de *bounding boxes* e 50.52 para segmentação das máscaras. Como os sintomas foram levados em consideração no novo conjunto, foi possível calcular o desempenho individual para cada sintoma. Os resultados completos podem ser observados na Tabela 4.5.

As classes "miner"e "phoma"se destacaram com resultados superiores às demais. Isso se dá por apresentarem características mais marcantes, que se ressal-

Figura 4.4: Exemplos de resultados obtidos no dataset Bracot, à esquerda, em realce as imagens com as anotações do especialista e à direita as predições do modelo Yolact++ separando diferenciando cada instância por uma cor diferente.

tam quando comparadas a coloração e disposição da folha. Outro fator considerado para o pior desempenho das demais classes é o fato de possuírem dimensão menor que as de melhor precisão. Por esse motivo, é possível que algumas instâncias dessas classes tenham sido descartadas na etapa de extração de contornos na criação da base de dados modificada.

As imagens geradas no conjunto de teste tiveram uma boa qualidade na segmentação obtida, uma vez que visualmente cobriram grande parte dos sintomas detectados e obtiveram um bom alinhamento quando comparadas com as imagens anotadas pelo especialista. Exemplos de saídas nesse conjunto em comparação com os esperados podem ser observados na Figura 4.5, na qual podem ser observadas as folhas com as anotações corretas nas colunas "verdade"e os resultados obtidos pelo modelo Yolact++ na coluna "resultado".

Tabela 4.5: Resultados para diferentes thresholds de IoU no dataset bracol.

Miner											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	**64.9**	86.4	86.4	86.4	86.4	86.4	**76.2**	72.4	62.2	5.9	0.0
mask	**70.9**	86.4	86.4	86.4	86.4	86.4	**82.3**	80.2	72.7	41.5	0.0
Rust											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	21.9	53.7	44.5	39.0	32.9	25.6	13.5	7.8	1.8	0.7	0.0
mask	22.2	50.7	47.5	39.0	36.0	23.0	18.1	6.8	1.2	0.2	0.0
Phoma											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	**71.0**	94.8	94.8	94.8	94.8	94.8	**75.0**	70.3	62.9	27.2	0.19
mask	**70.2**	90.8	86.2	86.2	86.2	86.2	**81.4**	81.4	69.3	32.2	1.98
Cercospora											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	41.2	55.6	55.6	55.6	55.6	55.6	55.6	55.6	18.3	4.16	0.0
mask	38.6	55.7	55.7	55.7	55.7	55.7	55.7	20.7	20.7	10.4	0.0
Total											
	all	.50	.55	.60	.65	.70	.75	.80	.85	.90	.95
bbox	**49.7**	72.7	70.3	69.0	67.4	65.6	**55.1**	51.5	36.3	9.52	0.05
mask	**50.5**	70.9	69.0	66.8	66.1	62.8	**59.4**	47.3	41.0	21.1	0.50

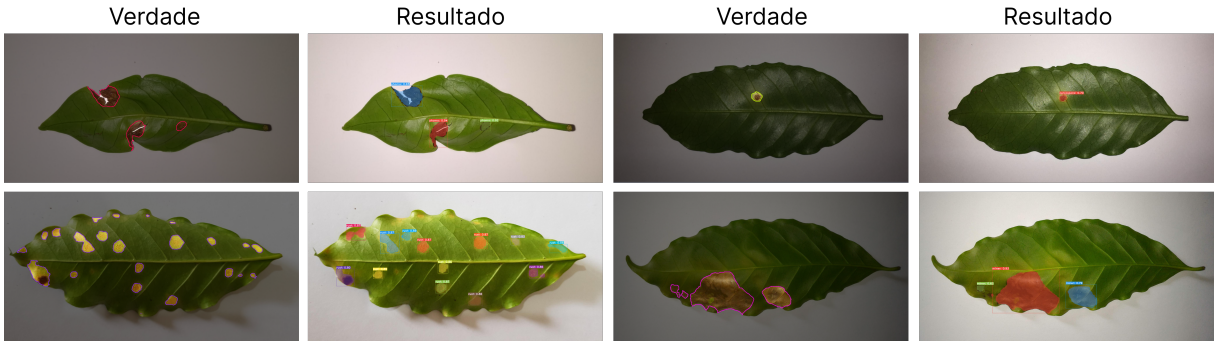

Figura 4.5: Exemplos de resultados obtidos no dataset Bracol.

4.4 Pipeline de extração de sintomas proposto

Pela falta de anotações prévias e o desconhecimento técnico acerca da classificação das doenças, a análise dos resultados obtidos pelo fluxo proposto foi feita de modo qualitativo. Dessa forma, o fluxo proposto na Subseção 3.4.4 foi capaz de realizar a segmentação de instâncias da maioria dos sintomas aparentes. Uma das imagens obtidas (imagem de entrada do *pipeline* na Figura 3.6) foi selecionada para ilustrar todos os passos descritos.

O algoritmo desenvolvido foi capaz de extrair, sem apresentar falhas ou perda

de informação, todas as instâncias de folhas obtidas durante a avaliação dos resultados na base de dados Bracot, em seguida, orientou corretamente as folhas individuais no sentido horizontal, bem como retirou o fundo com informação não relevante. Como pode ser observado na Figura 4.6, que ilustra a imagem obtida pelo primeiro passo do *pipeline*, em seguida uma de suas folhas extraída e por fim essa mesma folha rotacionada e com o fundo removido. Por fim o modelo treinado no dataset Bracol modificado foi capaz de segmentar os sintomas apresentados nas folhas extraídas. Como ilustrado na figura 4.7.

Figura 4.6: Extração e rotação da folha segmentada.

Figura 4.7: Segmentação dos sintomas na folha extraída.

Ao analisar as imagens que o modelo detectou sintomas em áreas erradas, ou que se encontravam fora da folha, foi detectado que durante a extração da máscara da folha, a escolha do fundo preto, gerado com a operação descrita na Subseção 3.4.4 se mostrou equivocada. Uma vez que se assemelhou a instâncias da classe "phoma", que possui a coloração mais escura e, no conjunto Bracol, comumente estava localizada nas extremidades da folha. Para contornar tal defeito, a cor de fundo poderia ser substituída por uma que se assemelhasse àquela do conjunto de treinamento. Exemplos de erro do pipeline podem ser vistos na Figura 4.8, que

ilustra predições em que o modelo Yolact++ segmentou áreas de doença em áreas que não fazem parte das folhas analisadas.

Figura 4.8: Imagens em que o modelo detectou sintomas fora da área da folha.

Capítulo 5

Conclusões

A detecção automatizada de pestes e doenças em plantações de café traz grande benefícios à qualidade e à eficiência das colheitas. Devido aos avanços do aprendizado de máquina e do aprendizado profundo, esse cenário está cada vez mais facilitado, uma vez que frequentemente são publicadas novas bases de dados com suporte a classificação e segmentação de doenças comuns.

Considerando os modelos capazer de realizar a segmentação de instâncias, o modelo Yolact++ testado teve desempenho satisfatório em todos os datasets em que foi aplicado. Ao comparar os resultados obtidos com o artigo base (TASSIS; TOZZI; KROHLING, 2021) utilizado, o Yolact++ teve performance semelhante ao treinamento com o Mask R-CNN, se destacando pela agilidade no processamento das imagens e na qualidade das máscaras geradas.

Além da eficiência do modelo escolhido para este trabalho, também é possível atestar a importância de uma padronização nos dados de entrada, tendo o formato COCO (LIN et al., 2014) como uma boa alternativa para armazenar informações das imagens de forma organizada e de fácil processamento. Ademais, a transformação das bases de dados para comportar a tarefa de segmentação de instâncias se mostrou essencial. As bases geradas a partir do RoCoLe e do Bracol podem ser base para continuação deste trabalho, ou para estudos futuros na área.

Outro ponto evidenciado pelo estudo foi o do impacto da distribuição das classes entre os conjuntos de treino e validação no resultado final. Datasets desbalanciados, como o RoCoLe, ou com poucas imagens para treino influenciaram negativamente o desempenho do modelo. Para contornar esse tipo de obstáculo, tem-se a opção de manipular as entradas, a fim de gerar novos dados, ou de montar um dataset artificial que simule condições reais do contexto estudado. Possíveis técnicas e métodos para o tratamento de bases de dados desbalanceadas pode ser encontrado em (KRAWCZYK, 2016).

Portanto, foi possível entender, a partir dos esforços realizados, que uma análise inicial dos dados antes do treinamento pode ser fator chave para obter bons resultados. Com isso, uma grande parte dos esforços deve ser dedicado à preparar os dados antes mesmo de escolher os modelos para treinamento.

A partir do trabalho realizado, ficam como objetivos futuros a ampliação das bases de dados modificadas, bem como o aumento dos dados de forma artificial. Além disso, o estudo de posssíveis modificações no Yolact++ (HUANG et al., 2021) para que sua performance não custe de maneira pesarosa a precisão do treinamento.

Referências

BOLYA, D.; ZHOU, C.; XIAO, F.; LEE, Y. J. YOLACT++: better real-time instance segmentation. *CoRR*, abs/1912.06218, 2019. Disponível em: <http://arxiv.org/abs/1912.06218>.

BOLYA, D.; ZHOU, C.; XIAO, F.; LEE, Y. J. YOLACT: real-time instance segmentation. *CoRR*, abs/1904.02689, 2019. Disponível em: <http://arxiv.org/abs/1904.02689>.

CHOLLET, F. *Deep Learning With Python*. 20 Baldwin Road: Manning Publications, 2018. ISBN 9781617294433.

COLAB, G. *Colaboratory - Frequently Asked Questions*. Google, 2021. Último acesso em 17 de setembro de 2021. Disponível em: <https://research.google.com/colaboratory/faq.html>.

ELGENDY, M. *Deep Learning for Vision Systems*. 20 Baldwin Road: Manning Publications, 2020. ISBN 9781617296192.

ESGARIO, J. G. M.; KROHLING, R. A.; VENTURA, J. A. *Deep Learning for Classification and Severity Estimation of Coffee Leaf Biotic Stress*. arXiv, 2019. Disponível em: <https://arxiv.org/abs/1907.11561>.

GIT. *Git*. 2022. Disponível em: <https://git-scm.com/>.

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

GéRON, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc, 2019. ISBN 9781492032649.

HARRIS, C. R.; MILLMAN, K. J.; WALT, S. J. van der; GOMMERS, R.; VIRTANEN, P.; COURNAPEAU, D.; WIESER, E.; TAYLOR, J.; BERG, S.; SMITH, N. J.; KERN, R.; PICUS, M.; HOYER, S.; KERKWIJK, M. H. van; BRETT, M.; HALDANE, A.; RÍO, J. F. del; WIEBE, M.; PETERSON, P.; GÉRARD-MARCHANT, P.; SHEPPARD, K.; REDDY, T.; WECKESSER, W.; ABBASI, H.; GOHLKE, C.; OLIPHANT, T. E. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>.

HE, K.; GKIOXARI, G.; DOLLÁR, P.; GIRSHICK, R. B. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. Disponível em: <http://arxiv.org/abs/1703.06870>.

HUANG, M.; XU, G.; LI, J.; HUANG, J. A method for segmenting disease lesions of maize leaves in real time using attention yolact++. *Agriculture*, v. 11, n. 12, 2021. ISSN 2077-0472. Disponível em: <https://www.mdpi.com/2077-0472/11/12/1216>.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, IEEE COMPUTER SOC, v. 9, n. 3, p. 90–95, 2007.

KISHIDA, K. Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments. v. 2005, 09 2005.

KRAWCZYK, B. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, IEEE COMPUTER SOC, 2016.

LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. Disponível em: <http://arxiv.org/abs/1405.0312>.

matterport. *Mask R-CNN for Object Detection and Segmentation*. 2022. Último acesso em 01 de setembro de 2022. Disponível em: <https://github.com/matterport/Mask_RCNN>.

MITCHELL, T. M. *Machine Learning*. New York: McGraw-Hill, 1997. ISBN 978-0-07-042807-2.

OPENCV. *OpenCV*. 2022. Disponível em: <https://opencv.org/>.

PARRAGA-ALAVA, J.; CUSME, K.; LOOR, A.; SANTANDER, E. Rocole: A robusta coffee leaf images dataset for evaluation of machine learning based methods in plant diseases recognition. *Data in Brief*, v. 25, p. 104414, 2019. ISSN 2352-3409. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352340919307693>.

PARRAGA-ALAVA, J.; CUSME, K.; LOOR, A.; SANTANDER, E. *RoCoLe: A robusta coffee leaf images dataset for evaluation of machine learning based methods in plant diseases recognition*. 2019. 104414 p. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2352340919307693>.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; DESMAISON, A.; KOPF, A.; YANG, E.; DEVITO, Z.; RAISON, M.; TEJANI, A.; CHILAMKURTHY, S.; STEINER, B.; FANG, L.; BAI, J.; CHINTALA, S. Pytorch: An imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019. p. 8024–8035. Disponível em: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv*, 2018.

REN, S.; HE, K.; GIRSHICK, R. B.; SUN, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. Disponível em: <http://arxiv.org/abs/1506.01497>.

Roboflow. *Roboflow*. 2022. Último acesso em 5 de setembro de 2022. Disponível em: <https://roboflow.com/>.

RODRIGUES, G. *Repositório dos códigos desenvolvidos*. 2022. Disponível em: <https://github.com/guilodron>.

SANTOS, T.; BARBEDO, J.; KOENIGKAN, L.; SOUZA, K. de; TERNES, S. Visão computacional aplicada na agricultura. In: _____. [S.l.: s.n.], 2020. p. 146–164. ISBN 978-65-86056-37-2.

SANTOS, T. T.; de Souza, L. L.; dos Santos, A. A.; AVILA, S. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, v. 170, p. 105247, 2020. ISSN 0168-1699. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168169919315765>.

TASSIS, L. M.; TOZZI, J. E.; KROHLING, R. A. A deep learning approach combining instance and semantic segmentation to identify diseases and pests of coffee leaves from in-field images. *Computers and Electronics in Agriculture*, v. 186, p. 106191, 2021. ISSN 0168-1699. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168169921002088>.

TEAM, T. pandas development. *pandas-dev/pandas: Pandas*. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>.

TIOBE Index. 2022. Disponível em: <https://www.tiobe.com/tiobe-index/>.

VSCODE. *Visual Studio Code*. 2022. Disponível em: <https://code.visualstudio.com/>.