



ALYSSON ALVES FERNANDES

**IDENTIFICAÇÃO DO ESTÁDIO DE MATURAÇÃO DE FRUTOS
DE CAFÉ COM A UTILIZAÇÃO DE VISÃO COMPUTACIONAL**

LAVRAS – MG

2024

ALYSSON ALVES FERNANDES

**IDENTIFICAÇÃO DO ESTÁDIO DE MATURAÇÃO DE FRUTOS DE CAFÉ COM A
UTILIZAÇÃO DE VISÃO COMPUTACIONAL**

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós Graduação em Engenharia de Sistemas e Automação, área de concentração em Sistemas Inteligentes, para a obtenção do título de Mestre..

Prof. Dr. Danton Diego Ferreira
Orientador

Dr. Luiz de Gonzaga Ferreira Júnior
Coorientador

**LAVRAS – MG
2024**

**Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca
Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).**

Fernandes, Alysson Alves.

Identificação do Estádio de Maturação de Frutos de Café com a
Utilização de Visão Computacional / Alysson Alves Fernandes. -
2024.

77 p. : il.

Orientador(a): Danton Diego Ferreira.

Coorientador(a): Luiz de Gonzaga Ferreira Júnior.

Dissertação (mestrado acadêmico) - Universidade Federal de
Lavras, 2024.

Bibliografia.

1. Cafeicultura de precisão. 2. Visão computacional. 3. Deep
learning. I. Ferreira, Danton Diego. II. Júnior, Luiz de Gonzaga
Ferreira. III. Título.

ALYSSON ALVES FERNANDES

**IDENTIFICAÇÃO DO ESTÁDIO DE MATURAÇÃO DE FRUTOS DE CAFÉ COM A
UTILIZAÇÃO DE VISÃO COMPUTACIONAL**

**IDENTIFICATION OF THE MATURATION STAGE OF COFFEE FRUITS USING
COMPUTER VISION**

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós Graduação em Engenharia de Sistemas e Automação, área de concentração em Sistemas Inteligentes, para a obtenção do título de Mestre..

APROVADA em 12 de Julho de 2024.

Prof. Dr. Bruno Groenner Barbosa	UFLA
Prof. Dr. Danton Diego Ferreira	UFLA
Dr. Luiz de Gonzaga Ferreira Júnior	UFLA
Profa. Dra. Margarete Marin Lordelo Volpato	Epamig

Documento assinado digitalmente
 **DANTON DIEGO FERREIRA**
Data: 30/09/2024 10:45:00-0300
Verifique em <https://validar.iti.gov.br>

Prof. Dr. Danton Diego Ferreira
Orientador

Dr. Luiz de Gonzaga Ferreira Júnior
Coorientador

**LAVRAS – MG
2024**

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa de Minas Gerais (FAPEMIG).

RESUMO

Este trabalho propõe e apresenta os resultados de um sistema baseado em visão computacional para identificação de diferentes estádios de maturação de frutos de café. Esta identificação possui papel crucial no momento da colheita, pois o grau de maturação influencia diretamente na qualidade do produto final. O sistema foi desenvolvido utilizando a arquitetura cliente-servidor, no qual uma aplicação *mobile* (cliente) é responsável por capturar fotos e enviar para uma aplicação *web* (servidor), onde a imagem é processada por um modelo de *Deep Learning* (*Mask RCNN* ou *YOLOv8*). Após o processamento, as informações obtidas são devolvidas à aplicação *mobile* para visualização do usuário. Toda comunicação foi feita via protocolo HTTPS. Visando lidar com problemas de rede e de assincronia, em ambos os lados do sistema existe um banco de dados armazenando todas as fotos e informações. Foram obtidos resultados relacionados ao processamento das imagens, em três diferentes contextos: (i) para frutos verdes, verde-cana, cereja e passas/secas, (ii) para frutos verdes e maduros, e (iii) identificação de frutos (contagem) independente do estágio de maturação. O modelo de *Mask RCNN* apresentou o melhor desempenho quando os resultados foram comparados, apresentando um APE médio no contexto (iii) de 9,61%, no contexto (i) o menor APE obtido foi de 9,12% para a classe verde-cana e no contexto (ii) obtendo seu menor APE para classe maduro, no valor de 8,22%. O sistema *web* foi desenvolvido, tendo seu servidor implementado em Python, onde o modelo de *Deep Learning* com melhor desempenho (*Mask RCNN*) foi integrado. A aplicação *mobile* foi implementada com o uso do *framework* React Native, sendo capaz de enviar fotos e receber relatórios do servidor.

Palavras-chave: cafeicultura de precisão; visão computacional; reconhecimento de padrões; *machine learning*. *deep learning*.

ABSTRACT

This work proposes and presents the results of a system based on computer vision for identifying different maturation stages of coffee fruits. This identification plays a crucial role at the time of harvest, as the degree of maturation directly influences the quality of the final product. The system was developed using client-server architecture, in which a mobile application (client) is responsible for capturing photos and sending them to a web application (server), where the image is processed by a Deep Learning model (Mask RCNN or YOLOv8). After processing, the information obtained is returned to the mobile application for user viewing. All communication was done via HTTPS protocol. In order to deal with network and asynchrony problems, on both sides of the system there is a database storing all photos and information. Results related to image processing were obtained in three different contexts: (i) for green, sugarcane, cherry and raisin/dried fruits, (ii) for green and ripe fruits, and (iii) fruit identification (counting) regardless of the stage of maturation. The Mask RCNN model presented the best performance when the results were compared, presenting an average APE in context (iii) of 9.61%, in context (i) the lowest APE obtained was 9.12% for the green-cane class and in the context (ii) obtaining its lowest APE for the mature class, in the amount of 8.22%. The web system was developed, with its server implemented in Python, where the best performing Deep Learning model (Mask RCNN) was integrated. The mobile application was implemented using the React Native framework, being able to send photos and receive reports from the server.

Keywords: precision coffee farming; computacional vision; pattern recognition; machine learning; deep learning.

INDICADORES DE IMPACTO

Este trabalho apresenta impactos relevantes nas áreas de tecnologia, economia e meio ambiente ao propor uma solução baseada em visão computacional para a identificação dos estádios de maturação de frutos de café. O sistema desenvolvido, utilizando *Deep Learning*, visa otimizar o processo de colheita do café, o que pode gerar uma melhoria na qualidade do produto final ao identificar com precisão o momento adequado para a colheita. Com a identificação precisa dos estádios de maturação, pode-se reduzir a colheita de frutos verdes ou sobre-maduros, diminuindo o desperdício e aumentando a eficiência produtiva. Isso também resulta em impactos econômicos, pois a redução de perdas e a melhoria da qualidade do café podem aumentar o valor de mercado do produto. O trabalho está alinhado com os Objetivos de Desenvolvimento Sustentável (ODS) da ONU, especialmente com o ODS 12 (Consumo e Produção Responsáveis) e o ODS 9 (Indústria, Inovação e Infraestrutura), ao promover o uso de tecnologias inovadoras na agricultura. Além disso, o uso de tecnologias de visão computacional e inteligência artificial também reduz a dependência de avaliações subjetivas de especialistas, tornando o processo de tomada de decisão mais objetivo e baseado em dados. Esta solução tem potencial para ser implementada em diferentes regiões produtoras de café, beneficiando diretamente agricultores, cooperativas e empresas envolvidas na cadeia produtiva do café. O sistema pode ser expandido para outros cultivos agrícolas, promovendo a transformação digital no setor agroindustrial e gerando impactos positivos na produtividade e sustentabilidade a longo prazo.

IMPACT INDICATORS

This work presents significant impacts in the areas of technology, economy, and environment by proposing a computer vision-based solution for identifying the maturation stages of coffee fruits. The developed system, utilizing *Deep Learning*, aims to optimize the coffee harvesting process, which can improve the final product's quality by accurately identifying the appropriate harvest time. With precise identification of maturation stages, it is possible to reduce the harvesting of unripe or overripe fruits, decreasing waste and increasing production efficiency. This also results in economic impacts, as reducing losses and improving the quality of coffee can increase the market value of the product. The work aligns with the United Nations' Sustainable Development Goals (SDGs), particularly SDG 12 (Responsible Consumption and Production) and SDG 9 (Industry, Innovation, and Infrastructure), by promoting the use of innovative technologies in agriculture. Additionally, the use of computer vision and artificial intelligence technologies reduces the reliance on subjective expert evaluations, making the decision-making process more objective and data-driven. This solution has the potential to be implemented in different coffee-producing regions, directly benefiting farmers, cooperatives, and companies involved in the coffee production chain. The system can be expanded to other agricultural crops, fostering digital transformation in the agribusiness sector and generating long-term positive impacts on productivity and sustainability.

LISTA DE FIGURAS

Figura 2.1 – Diferentes estádios de maturação do fruto de café.	15
Figura 2.2 – Diferentes estádios de maturação do fruto de café e a mudança contínua de coloração.	15
Figura 2.3 – Tábua de amostragem.	16
Figura 2.4 – Operação de convolução.	18
Figura 2.5 – Agrupamento de máximos.	18
Figura 2.6 – Rede <i>U-net</i> exemplificando a amostragem ascendente.	19
Figura 2.7 – <i>Flatennig</i>	21
Figura 2.8 – Exemplo <i>U-Net</i>	22
Figura 2.9 – Detecção de objetos, segmentação semântica e de instância.	22
Figura 2.10 – Fluxograma completo da arquitetura <i>Mask RCNN</i>	23
Figura 2.11 – Extração do mapa de características.	23
Figura 2.12 – Extração das regiões de interesse.	24
Figura 2.13 – Alinhamento das regiões de interesse.	24
Figura 2.14 – Classificação e geração de caixas limitadoras e máscaras.	24
Figura 2.15 – Arquitetura base YOLO.	25
Figura 2.16 – Linha do tempo apresentando os modelos de YOLO.	26
Figura 2.17 – Agrupamento espacial YOLO.	27
Figura 2.18 – Segmentação de imagens <i>Mask RCNN</i>	27
Figura 2.19 – Arquitetura Cliente-Servidor do sistema desenvolvido.	29
Figura 3.1 – Fluxograma representando as etapas de desenvolvimento do trabalho.	33
Figura 3.2 – Algumas das imagens presentes na base de dados.	35
Figura 3.3 – Exemplo de <i>Data Augmentation</i> realizado.	36
Figura 3.4 – Avaliação e rotulação do especialista.	37
Figura 3.5 – Captura de tela do processo de rotulação de imagens utilizando o <i>software</i> <i>LabelMe</i>	37
Figura 3.6 – <i>Google Colab</i>	39

Figura 3.7 – Cliente <i>mobile</i>	40
Figura 3.8 – Servidor executando a arquitetura de processamento de imagens.	41
Figura 4.1 – <i>Boxplot</i> apresentando cada uma das métricas.	47
Figura 4.2 – Exemplo de possível padrão para captura de fotos.	49
Figura 4.3 – Comparação entre erros de contagem.	50
Figura 4.4 – Imagens onde os modelos obtiveram os melhores desempenhos com relação a contagem.	51
Figura 4.5 – Imagens onde os modelos obtiveram os piores desempenhos com relação a contagem.	52
Figura 4.6 – Imagem 1 rotulada pelos modelos propostos: (a) <i>Mask RCNN</i> e (b) YOLOv8. .	55
Figura 4.7 – Imagem 2 rotulada pelos modelos propostos: (a) <i>Mask RCNN</i> e (b) YOLOv8. .	56
Figura 4.8 – Imagem 3 rotulada pelos modelos propostos: (a) <i>Mask RCNN</i> e (b) YOLOv8. .	56
Figura 4.9 – Imagem 4 rotulada pelos modelos propostos: (a) <i>Mask RCNN</i> e (b) YOLOv8. .	57
Figura 4.10 – Imagem 5 rotulada pelos modelos propostos: (a) <i>Mask RCNN</i> e (b) YOLOv8. .	58
Figura 4.11 – Imagem 6 rotulada pelos modelos propostos: (a) <i>Mask RCNN</i> e (b) YOLOv8. .	58
Figura 4.12 – Imagem 7 rotulada pelos modelos propostos: (a) <i>Mask RCNN</i> e (b) YOLOv8. .	59
Figura 4.13 – Gráfico de erros por pares de classes.	61
Figura 4.14 – Padrão de imagens Bazame e Molin (2021).	63
Figura 4.15 – Padrão de imagens Ramos et al. (2017).	63
Figura 4.16 – <i>Logs</i> iniciais referentes ao funcionamento online do servidor.	64
Figura 4.17 – Tabela para armazenamento de imagens.	65
Figura 4.18 – Tabela para armazenamento de contagens.	65
Figura 4.19 – Telas presentes no aplicativo (Parte 1).	67
Figura 4.20 – Telas presentes no aplicativo (Parte 2).	68

LISTA DE TABELAS

Tabela 2.1 – Número de camadas e número de parâmetros (em milhões) de cada modelo. . .	28
Tabela 4.1 – Métricas <i>Mask RCNN</i>	44
Tabela 4.2 – Métricas YOLO.	45
Tabela 4.3 – Média das métricas.	46
Tabela 4.4 – p -valores obtidos pela análise de variância.	48
Tabela 4.5 – Resultado após aplicação do Teste F.	48
Tabela 4.6 – Contagem de frutos por imagem.	50
Tabela 4.7 – Tempos de processamento de cada imagem em milissegundos.	52
Tabela 4.8 – Análise da identificação do estágio de maturação.	54
Tabela 4.9 – Erros Médios para as informações de porcentagem tomando a referência do especialista.	55
Tabela 4.10 – Análise da identificação do grau de maturação por pares de classes.	60
Tabela 4.11 – Comparação entre APs.	62
Tabela 4.12 – Comparação entre APs.	62

SUMÁRIO

1	INTRODUÇÃO	11
2	REFERENCIAL TEÓRICO	14
2.1	Maturação do Café	14
2.2	Maturação do Café e Visão Computacional	16
2.3	Redes Neurais Convolucionais (CNN's)	17
2.4	Detecção de Objetos e Segmentação de Imagens	21
2.5	<i>Mask RCNN</i>	22
2.6	YOLOv8	25
2.7	<i>Mask RCNN</i> vs YOLOv8	26
2.8	Arquitetura Cliente-Servidor	28
2.9	Protocolo HTTP/HTTPS	29
2.10	<i>Data Augmentation</i>	30
2.11	<i>Transfer Learning</i>	31
3	METODOLOGIA PARA DESENVOLVIMENTO DA APLICAÇÃO	33
3.1	Base de dados	34
3.2	Pré-processamento	36
3.3	Treinamento dos Modelos	38
3.4	Aplicação <i>Mobile</i> (Cliente)	40
3.5	Aplicação <i>web</i> (Servidor)	41
3.6	Métricas para avaliação de desempenho e comparações	41
4	RESULTADOS	44
4.1	Contagem	49
4.2	Identificação do Estádio de Maturação	53
4.3	Comparações com outros trabalhos	61
4.4	Implementação do Sistema	64
4.4.1	Servidor	64
4.4.2	Aplicação <i>Mobile</i>	65

5	CONCLUSÃO	69
	REFERÊNCIAS	71

1 INTRODUÇÃO

Nos últimos anos, a tecnologia na agricultura tem experimentado um crescimento acelerado, impulsionando a produtividade e a sustentabilidade das culturas, sendo este novo cenário denominado por muitos como o Agro 4.0. A integração de métodos de visão computacional, desenvolvimento de sistemas avançados e uso de sensores modernos têm revolucionado a maneira como as culturas são cultivadas, colhidas e processadas. Essas tecnologias proporcionam maior precisão e eficiência, permitindo o monitoramento detalhado das plantações, a otimização do uso de recursos naturais e a melhoria na qualidade dos produtos. Sistemas integrados de gestão agrícola têm permitido aos produtores monitorar e controlar os aspectos de produção, utilizando *big data* e inteligência artificial para análise de grandes quantidades de informação visando a geração de *insights* valiosos para tomadas de decisão. Os sensores por sua vez, desempenham um papel fundamental na agricultura de precisão, sendo capazes de fornecer informações a respeito do solo (tais como umidade, temperatura, acidez), permitindo assim ajustes em tempo real de irrigação e na aplicação de fertilizantes, o que reduz desperdícios e minimiza impactos ambientais. Por fim, temos a visão computacional sendo aplicada no monitoramento de pragas e doenças (TASSIS; SOUZA; KROHLING, 2021), mapeamento de áreas para plantio (ZHAO et al., 2019), estimativas de produção (KLOMPENBURG; KASSAHUN; CATAL, 2020) e identificação de maturação de frutos (OLIVEIRA et al., 2016), tarefas essas que influenciam diretamente na produção, qualidade e preço dos produtos (FREITAS et al., 2024).

A visão computacional é uma área da inteligência artificial que busca automatizar tarefas que o sistema visual humano realiza. Com a evolução das técnicas de aprendizado de máquina, especialmente as redes neurais profundas (*deep learnings*), a visão computacional tornou-se uma ferramenta poderosa para a análise e interpretação de imagens em uma variedade de contextos, incluindo a agricultura. A *Deep Learning* é uma sub-área da aprendizagem de máquina, onde são estudadas e implementadas redes neurais artificiais com um grande número de camadas e de neurônios, o que conseqüentemente exige um maior poder de processamento dos computadores utilizados, mas também possibilita a obtenção de resultados substancialmente melhores. Essas redes são chamadas de Redes Neurais Convolucionais (*Convolutional Neural Networks - CNNs*), e

dentre os diversos tipos de redes existentes, são as mais utilizadas para processamento de imagens. As redes neurais convolucionais (*Convolutional Neural Networks* - *CNNs*) são a base para muitos dos avanços recentes em visão computacional, permitindo que computadores reconheçam padrões complexos em imagens e vídeos com alta precisão. A capacidade das CNNs de capturar hierarquias de características em diferentes níveis de abstração as torna especialmente eficazes para o processamento de imagens agrícolas. Por exemplo, na detecção de maturação de frutos, as CNNs podem aprender a distinguir entre diferentes estádios de maturação com base em variações sutis de cor, textura e forma. Com o uso de arquiteturas avançadas como *Mask R-CNN* e *YOLOv8*, é possível realizar tanto a detecção quanto a segmentação de objetos, permitindo uma análise mais detalhada e precisa das imagens.

Sobre o Agro no Brasil, pode-se destacar o café como um dos principais produtos, onde o país figura como o maior produtor e exportador de café do mundo (CONAB, 2024). Dados do MTE (Ministério do Trabalho e Emprego) apontam uma produção de 55,1 milhões de sacas em 2023. Em termos financeiros, o Brasil exportou mais de 7 bilhões de dólares em café, além do próprio mercado interno, capaz de movimentar alguns bilhões de reais. A área plantada de café no Brasil alcança quase dois milhões de hectares, com mais de 300 mil produtores, gerando aproximadamente 500 mil empregos no campo, indústria e comércio (MDS, 2024). Devido a tal importância na economia do país, a preocupação com a qualidade do café produzido sempre foi alvo de imensa atenção, sendo a qualidade e a produção, dois pontos que podem ser limitados por diversos fatores (MARIN et al., 2021). Dentre os fatores, pode-se destacar a maturação dos frutos colhidos. A maturação pode ocorrer em diferentes momentos ao longo de um determinado período, o que pode afetar a qualidade do produto, pois, frutos colhidos verdes ou sobre maduros podem causar na bebida um sabor adstringente e amargo.

Para evitar uma perda de qualidade devido a este fator, uma avaliação da lavoura deve ser realizada visando a obtenção das informações de quando e por qual área deve-se dar início à colheita, sendo o grau de maturação dos frutos um bom indicador. O grau de maturação do fruto pode ser determinado por meio de sua cor, de forma visual, ou por meio da utilização de colorímetros (COSTA et al., 2020). Porém, a avaliação visual, mesmo que feita por um especialista,

é subjetiva e está sujeita a erros humanos. O colorímetro, por sua vez, possui a desvantagem de ser capaz de avaliar apenas uma pequena parte da superfície dos frutos.

Uma opção a ser explorada para contornar as limitações dos métodos existentes de identificação do estágio de maturação do café é o uso de visão computacional, que tem como vantagem a possibilidade de uma oferta de medidas mais precisas e maior alcance. Neste trabalho, foram exploradas técnicas baseadas em *Deep Learning* para estudo e realização da tarefa de separação de frutos de café, conforme seus respectivos estádios de maturação. Foram implementadas e testadas duas arquiteturas de *Deep Learning*: *Mask RCNN* e *YOLOv8*.

Esta Dissertação tem como objetivo apresentar uma solução implementada utilizando um algoritmo de visão computacional para identificação de quatro estádios de maturação de frutos de café. A solução envolveu o desenvolvimento de um sistema composto por duas partes: um aplicativo *mobile* para captura e envio de fotos, e uma aplicação *web* para atuar como servidor, recebendo as fotos, realizando seu processamento e devolvendo os resultados para o aplicativo. Para alcançar este objetivo principal, têm-se os seguintes objetivos específicos:

- Contar e classificar frutos de café através do uso de visão computacional, segundo as seguintes classes (estádios de maturação): verde, verde-cana, cereja e passa/seco;
- Testar e analisar a capacidade de duas diferentes arquiteturas de *Deep Learning* (*Mask RCNN* e *YOLOv8*) para tal tarefa;
- Desenvolver um sistema composto por um aplicativo *mobile* e uma aplicação *web* (servidor) que permita a captura e análise de imagens, além de fornecer uma interface para visualização dos resultados.

O texto está dividido da seguinte forma: (i) referencial teórico, onde são apresentados alguns assuntos relevantes para o entendimento geral; (ii) uma metodologia de desenvolvimento, descrevendo como o sistema foi desenvolvido e como ele funciona; (iii) resultados, e por fim, (iv) as conclusões.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta as definições sobre a maturação do fruto de café e conceitos a respeito de arquiteturas de serviço *web* e temas inerentes ao processamento de imagens (tais como tipos de segmentação, redes neurais convolucionais, arquiteturas utilizadas e métricas de desempenho).

2.1 Maturação do Café

O desenvolvimento reprodutivo do cafeeiro pode ser representado por diferentes modelos, porém, independentemente do modelo, a máxima qualidade (e consequentemente o maior valor comercial) se potencializa quando os frutos são colhidos no estágio de maturação cereja. O conhecimento dos estádios de maturação é essencial para planejar o início da colheita e as ações subsequentes, e por isso tal informação deve ser monitorada. Durante a maturação, cinco fases (Figura 2.1) são descritas de acordo com as mudanças na coloração do fruto: M_1 para frutos de coloração verde, ou seja, sem evidências de alteração na cor; M_2 para frutos de coloração verde-cana, ou seja, frutos que já iniciaram a mudança de coloração; M_3 para frutos no estágio "cereja", de coloração vermelho-claro e maduros fisiologicamente; M_4 para frutos no estágio "passa", de coloração vermelho-escuro e com início de desidratação; M_5 para frutos secos, desidratados com coloração externa escura (MORAES, 2008; PEZZOPANE, 2008).

A Figura 2.2 apresenta a evolução da maturação no formato de uma escala de cores, onde fica evidente não haver uma fronteira clara, que distinga os estádios de maturação, pois as mudanças de coloração ocorrem de forma contínua, sendo este um dos maiores desafios para classificação, independente do método utilizado.

Uma das formas mais utilizadas atualmente para determinação do grau de maturação de uma lavoura é a amostragem utilizando uma tábua de 100 furos. Neste método, uma pequena quantidade de café é colhida, essa tábua de 100 furos é emergida de dentro da amostra, e a partir de uma avaliação visual, os estádios de maturação presentes na lavoura são estimados. A Figura 2.3 mostra essa tábua.

Figura 2.1 – Diferentes estádios de maturação do fruto de café.



Fonte: adaptado de Borém (2023).

Figura 2.2 – Diferentes estádios de maturação do fruto de café e a mudança contínua de coloração.



Fonte: Gonzaga Treinamentos e Consultoria Agrícola Ltda.

Apesar de usual, este método é limitado e o resultado obtido pode não representar com devida precisão a situação da lavoura, pois a amostra colhida para imersão da tábua não pode ser grande, para evitar, em caso de momento inoportuno para colheita, o desperdício dos frutos colhidos. Além disso, o uso da tábua não dispensa a avaliação de um especialista, que pode variar de uma amostra para outra e, principalmente, de um avaliador para outro, o que dificulta uma padronização na classificação da maturação dos frutos. Em contrapartida, métodos que utilizam visão computacional podem superar essas limitações, tornando o processo automático e mais preciso. Além disso, apresentam a vantagem de não requerer a colheita de amostras do fruto para avaliação da maturação, pois podem ser aplicados com os frutos ainda na planta. Com isso, uma maior área pode ser amostrada, requerendo menor esforço do usuário (especialista), que executará apenas a tarefa de captura de imagens.

Figura 2.3 – Tábua de amostragem.



Fonte: Gonzaga Treinamentos e Consultoria Agrícola Ltda.

2.2 Maturação do Café e Visão Computacional

Referente a aplicação de visão computacional na cultura cafeeira, destacam-se alguns trabalhos interessantes na literatura. Ramos et al. (2017) realizaram a contagem dos frutos ainda na planta utilizando técnicas clássicas (filtro de Canny e Morfologia Matemática), porém não obtiveram generalização, além de encontrar dificuldades com a determinação de bordas. Costa et al. (2020) realizaram a classificação de diferentes espécies de frutos de café a partir de um banco de dados montado utilizando características colorimétricas. Bazame et al. (2021) identificaram e contabilizaram os frutos a partir de vídeos obtidos do interior da calha de uma colhedora, utilizando a arquitetura YOLO v3 (REDMON; FARHADI, 2018).

Em Esgario et al. (2022), foi implementado um aplicativo de *smartphone*, no qual o agricultor poderia fotografar uma folha de café, e em instantes receber uma análise relacionada à detecção de doenças e anomalias. Ferreira Júnior et al. (2020) estudaram o deslocamento dos galhos da planta e sua interação dinâmica com a máquina durante a colheita, assim, unindo tais informações, junto a uma análise do estágio de maturação referente a cada área da lavoura, seria possível ajustar

a colhedora para realizar a derriça apenas de frutos em grau de maturação satisfatório, evitando assim a colheita de frutos verdes, pois a estágio de maturação dos frutos tem direta relação com a força de desprendimento dos frutos e também com eficiência da derriça (Ferreira Júnior et al., 2018).

2.3 Redes Neurais Convolucionais (CNN's)

As arquiteturas de *Deep Learning* utilizadas neste trabalho (*Mask RCNN* e *YOLOv8*), são baseadas em Redes Neurais Convolucionais (*Convolutional Neural Networks* - CNN's) (SICRE; JURIE, 2015), tornando assim importante o entendimento de como tais redes funcionam. Este tipo de rede é amplamente utilizada para processamento de imagens, pois possui a capacidade inerente de extração de características, o que faz com que longos trabalhos de pré-processamento possam ser dispensados. As CNN's, geralmente e de forma bastante abrangente, realizam três operações sobre os dados de entrada: convolução, agrupamento dos máximos e amostragem ascendente.

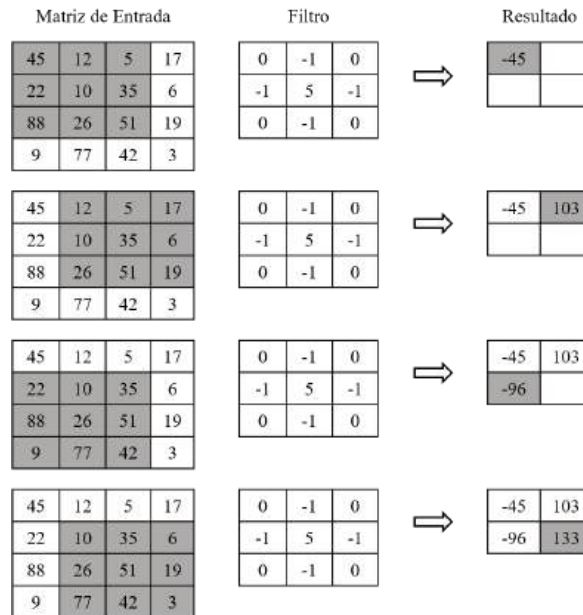
A operação de convolução utiliza filtros matriciais, que realizam multiplicações e somas através dos dados de entrada, e desta forma são gerados os mapas de características, já reduzindo as dimensões dos dados de entrada. A Figura 2.4 ilustra essa operação, na qual um filtro matricial (ilustrado na coluna central da imagem) 3×3 percorre os dados realizando uma multiplicação ponto a ponto e por fim a soma os resultados. De forma que na primeira linha da Figura 2.4, por exemplo, tem-se:

$$45x0 + 12x(-1) + 5x0 + 22x(-1) + 10x5 + 35x(-1) + 88x0 + \\ 26x(-1) + 51x0 = -45$$

A segunda operação, o agrupamento dos máximos, realiza uma nova redução de dimensões, utilizando um novo filtro que realiza a tarefa de gravar o maior valor existente no mapa de características, como pode ser visto na Figura 2.5.

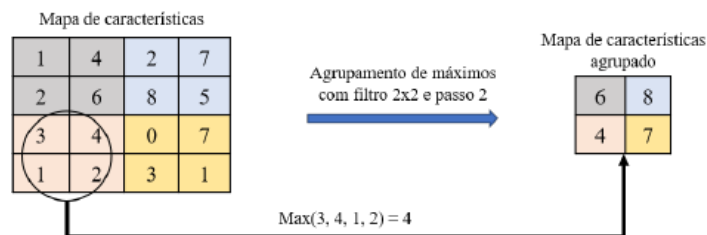
Com a realização das duas operações citadas, temos como resultado "O que" estamos identificando na imagem, porém não temos nenhuma informação sobre "Onde" esta característica se encontra na imagem (localização). Para obtenção desta informação, é realizada a operação de

Figura 2.4 – Operação de convolução.



Fonte: do Autor.

Figura 2.5 – Agrupamento de máximos.

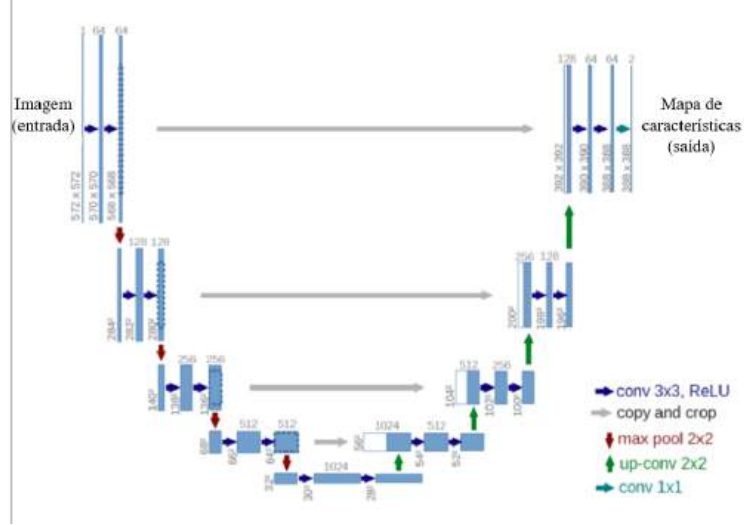


Fonte: do Autor.

amostragem ascendente, que leva a imagem novamente para dimensões mais altas, realizando o oposto do que é realizado nas operações de convolução. A Figura 2.6 apresenta a rede *U-net*, que realiza as 3 operações já citadas: convolução, agrupamento de máximos e amostragem ascendente (avaliando a imagem da esquerda para a direita).

Os passos anteriores descrevem o funcionamento de uma CNN de forma mais qualitativa, abrangente e simplificada. Uma abordagem mais detalhada do ponto de vista da matemática envolvida neste tipo de rede, também pode ser realizada. O modelo de CNN será abordado matematicamente com relação a **aprendizagem** e a **classificação**.

Figura 2.6 – Rede *U-net* exemplificando a amostragem ascendente.



Fonte: adaptado de Saha (2018).

A aprendizagem envolve as camadas de convolução e do agrupamento de máximos. A camada de convolução realiza a extração de características utilizando uma estrutura chamada de *kernel* (ou filtro) que percorre a imagem realizando multiplicações e somas. Suponha uma imagem **I** descrita por um tensor de dimensão $m_1 \times m_2 \times m_c$, onde:

- m_1 = altura da imagem
- m_2 = largura da imagem
- m_c = numero de canais

O *kernel* **K** a ser aplicado sobre essa imagem, também será um tensor, porém de dimensão $n_1 \times n_2 \times n_c$. O *kernel* deverá se movimentar sobre a imagem da esquerda para a direita realizando multiplicação entre **I** e **K**, e somando os resultados da multiplicação. A forma como *kernel* se movimenta, dependerá também do passo s determinado. Este passo ditará o número de colunas que o *kernel* deverá deslocar a cada iteração enquanto percorre a imagem. O tensor **F** resultante da multiplicação $\mathbf{I} \times \mathbf{K}$ possua dimensões $(m_1 - n_1 + 1) \times (m_2 - n_2 + 1) \times 1$, e pode ser observado na equação 2.1.

$$\mathbf{F}[i, j] = (\mathbf{I} * \mathbf{K})_{[i, j]} \quad (2.1)$$

A n -ésima informação do mapa de características gerado pela equação 2.1 é apresentada pela equação 2.2.

$$f[i, j] = \sum_x^{m_1} \sum_y^{m_2} \sum_z^{m_c} K_{[x,y,z]} I_{[i+x-1, j+y-1, z]} \quad (2.2)$$

Essa operação é realizada diversas vezes e utilizando diferentes filtros, para extração de diferentes características. O resultado obtido a partir dessas operações, na sequência é processado por uma função de ativação (ReLU, sigmoide, tangente hiperbólica, entre outras). Outro fator geralmente presente é o bias (b), assim como nas redes neurais convencionais. Para aplicação da função de ativação tem-se $\mathbf{C}(\text{Conv})$:

$$\mathbf{C} = \mathbf{F} + \mathbf{b}$$

$$\mathbf{C} = \mathbf{I} * \mathbf{K} + \mathbf{b}$$

$$\text{Conv}(\mathbf{I}, \mathbf{K}) = \phi_a(\mathbf{C}) = \phi_a(\mathbf{I} * \mathbf{K} + \mathbf{b})$$

onde ϕ_a representa a função de ativação. Uma das funções de ativação comumente utilizada é a ReLU, que elimina valores negativos, conforme a equação 2.3.

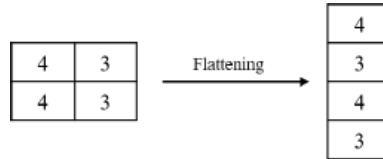
$$R(x) = \max(0, x) \quad (2.3)$$

Como uma última etapa da extração de características temos a etapa de agrupamento, que isola as características mais dominantes da imagem. Esse agrupamento visa reduzir dimensões, e pode ser feito em grades de dimensões pré-definidas. Os valores incluídos dentro de uma grade, deverão se tornar apenas um, e isso pode ser feito de diferentes formas, tais como: soma dos valores, média dos valores ou maior valor. ϕ_p representa a função que realizará o agrupamento, sendo aplicada segundo a equação 2.4, e obtendo $\mathbf{P}$.

$$\mathbf{P} = \phi_p(\mathbf{C}) \quad (2.4)$$

Com a finalização da extração de características, a etapa de classificação pode ser executada. Para isso, o tensor resultante é transformado em um vetor (Figura 2.7) através da operação de *flattening*.

Figura 2.7 – *Flatenning*.



Fonte: do Autor.

Após este processo, os dados entram na "camada totalmente conectada", que se trata de um *Multilayer Perceptron* (MLP), conforme:

$$\mathbf{X} = \sum_i w_i P_i + b$$

$$z = g(\mathbf{X})$$

onde g também representa uma função de ativação. Finalmente, como saída da camada totalmente conectada, uma classificação é obtida.

Uma arquitetura composta por CNN's poderá se utilizar de todo esse processo de diferentes formas, como por exemplo, obtendo os mapas de características e dividindo-os, para posterior envio de cada parte para a camada totalmente conectada, podendo assim classificar diversos objetos em uma mesma imagem. Existem ainda CNN's puramente convolucionais, que dispensam o uso da camada totalmente conectada (por exemplo, a rede *U-Net* apresentada na Figura 2.6). A saída visual de tais redes aparecem como "borrões" sobre os pontos de interesse na imagem, ou seja, o mapa de características é sobreposto na imagem (Figura 2.8).

2.4 Detecção de Objetos e Segmentação de Imagens

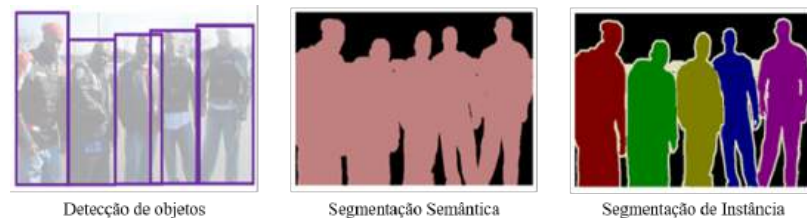
O processo de segmentação de imagens (WANG et al., 2018) pode ser definido como a extração de conjuntos de formas, contornos ou objetos presentes em uma imagem, enquanto a detecção de objetos não se preocupa em extrair as formas, e sim em apontar se em determinado

Figura 2.8 – Exemplo *U-Net*.

Fonte: do Autor.

ponto da imagem existe ou não um objeto alvo. Para este trabalho, cabe distinguir 3 tipos de tarefas realizadas em visão computacional: detecção de objetos, segmentação semântica e segmentação de instância. A segmentação semântica agrupa as instâncias de objetos, e a segmentação de instâncias separa todos os objetos individualmente (distingue os pixels de cada objeto). A Figura 2.9 ilustra de forma didática estes conceitos.

Figura 2.9 – Detecção de objetos, segmentação semântica e de instância.



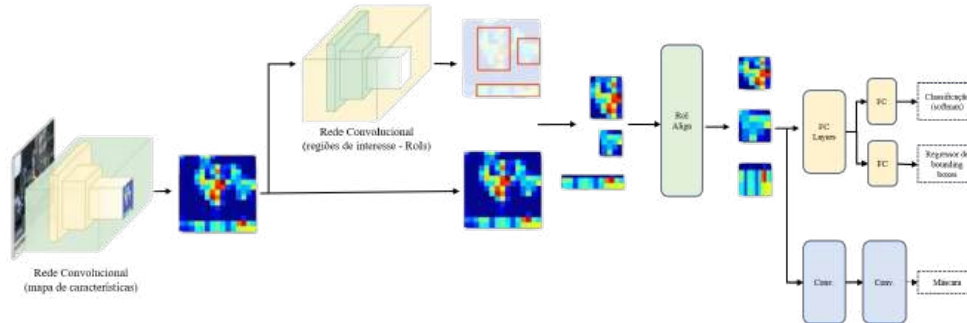
Fonte: adaptado de Lamba (2019).

2.5 *Mask RCNN*

O *Mask RCNN* (HE et al., 2017) é uma arquitetura desenvolvida em 2017, e atualmente é vista como uma das melhores arquiteturas para identificação de objetos e segmentção de imagens. A Figura 2.10 apresenta a arquitetura.

Algumas das principais etapas de processamento executadas pelo *Mask RCNN* serão expostas, visando elucidar seu funcionamento, tornando claras e distintas cada etapa de processamento.

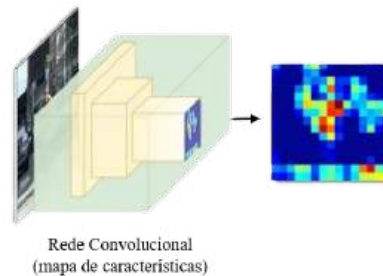
Figura 2.10 – Fluxograma completo da arquitetura *Mask RCNN*.



Fonte: adaptado de Hui (2018).

Primeiramente, são realizadas operações de convolução sobre as informações de entrada (pixels da imagem), visando a obtenção dos mapas de características, conforme Figura 2.11.

Figura 2.11 – Extração do mapa de características.



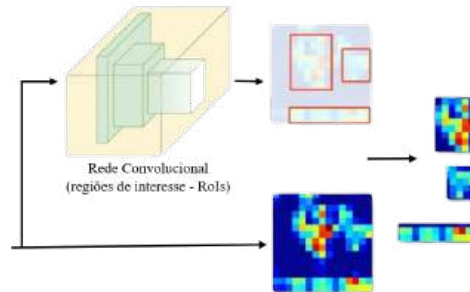
Fonte: adaptado de Hui (2018).

Em seguida, o mapa de características é dividido em regiões de interesse (*RoIs - Regions of Interest*) através da realização de novas operações de convolução e do agrupamento de máximos. O mapa de características intacto segue adiante também, conforme Figura 2.12.

Como próximo passo, tem-se o alinhamento das regiões de interesse com as suas respectivas posições originais no mapa de características, visando obter as informações de localização a respeito de cada região destacada, conforme Figura 2.13.

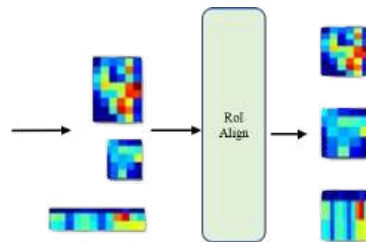
Por fim, camadas totalmente conectadas (neurônios convencionais), são responsáveis pela classificação de cada instância contida nas regiões de interesse, e também por gerar os limites para a plotagem da caixa limitadora sobre a região do mapa de características que contém a instância. Paralelamente, novas operações de convolução são aplicadas sobre as regiões de interesse, para gerar a máscara que deverá acompanhar o formato da instância identificada, conforme Figura 2.14.

Figura 2.12 – Extração das regiões de interesse.



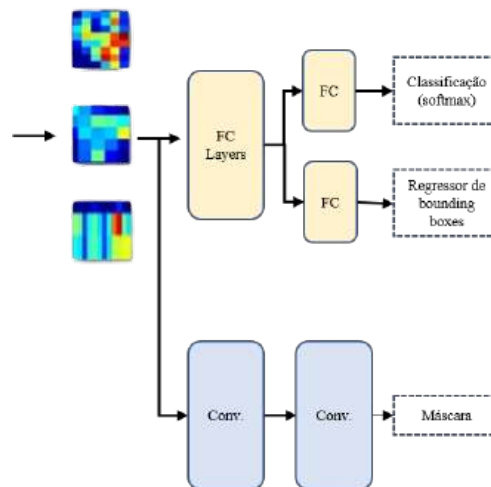
Fonte: adaptado de Hui (2018).

Figura 2.13 – Alinhamento das regiões de interesse.



Fonte: adaptado de Hui (2018).

Figura 2.14 – Classificação e geração de caixas limitadoras e máscaras.

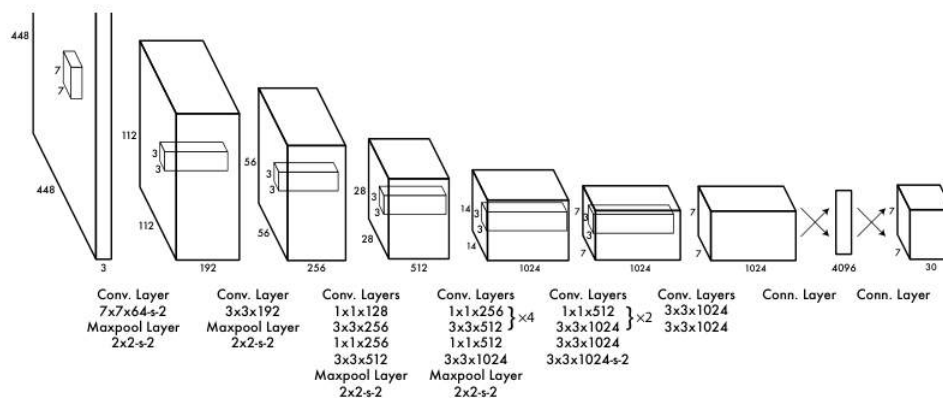


Fonte: adaptado de Hui (2018).

2.6 YOLOv8

A arquitetura YOLO (*You Only Look Once* - "Você só olha uma vez") (REDMON et al., 2016) foi projetada para detecção de objetos, sendo que para tal tarefa, uma rede neural convolucional é utilizada. Seu maior diferencial envolve o fato de ser uma arquitetura de passada única, que avalia cada área da imagem apenas uma vez (existe apenas uma rede neural profunda, e a imagem passa apenas uma vez por essa rede), o que consequentemente a torna uma das arquiteturas mais rápidas quando se trata de detecção de objetos. Essa arquitetura trata a detecção de objetos como um problema de regressão. Desde sua primeira versão (v1) até a versão atual (v8) diversas melhorias que incrementaram sua performance (velocidade e precisão de detecção) foram implementadas, porém a sua base de ser uma arquitetura de passada única sempre foi mantida. A Figura 2.15 apresenta a arquitetura base do YOLO, onde podemos observar um caminho totalmente linear, desde a entrada da imagem na rede, até as detecções obtidas na saída.

Figura 2.15 – Arquitetura base YOLO.

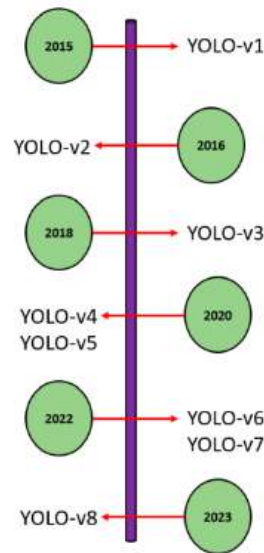


Fonte: Redmon et al. (2016).

Neste trabalho foi utilizada a versão 8 do YOLO, sendo esta a versão mais recente na data em que o sistema proposto começou a ser desenvolvido. A Figura 2.16 apresenta cada versão deste modelo e seu respectivo ano de implementação. As principais diferenças entre as versões envolvem principalmente melhorias de desempenho para detecção de objetos em imagens, vídeos e também detecção em tempo real. Além de melhorias com relação a detecção de pequenos objetos e também capacidade nativa de realizar segmentação de imagens, implementada em uma variante do YOLOv7

(WANG; BOCHKOVSKIY; LIAO, 2022) e aprimorada no YOLOv8 (JOCHER; CHAURASIA; QIU, 2023).

Figura 2.16 – Linha do tempo apresentando os modelos de YOLO.



Fonte: adaptado de Hussain (2023).

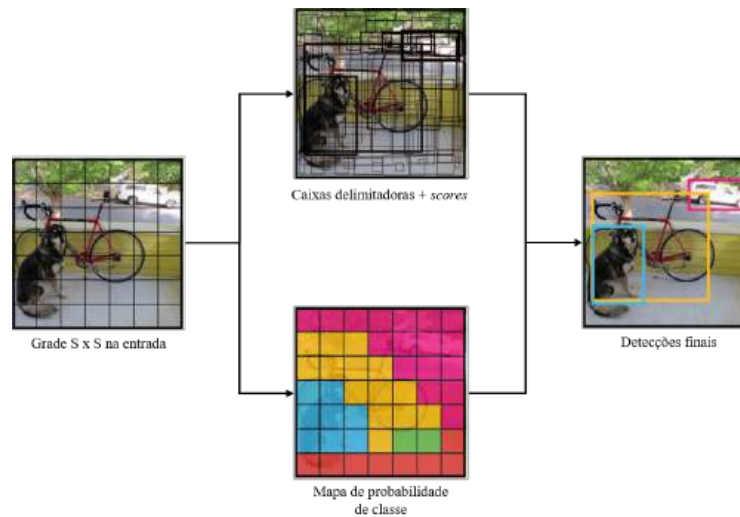
2.7 Mask RCNN vs YOLOv8

Apesar de ambas as arquiteturas possuírem o objetivo comum de processar imagens, existem diferenças em como tal tarefa é realizada e também na forma em que o resultado final é obtido, e tais diferenças, geram uma série de vantagens e desvantagens quando tais arquiteturas são comparadas. O *Mask RCNN* trabalha com a separação de regiões de interesse (RoI's), de forma que, uma rede convolucional faz a separação dessas regiões, e depois outra rede processa cada região de interesse buscando segmentar os objetos identificados. Enquanto o YOLO é do formato *Single Pass*, gerando um resultado que exige a passagem da imagem apenas uma vez pela rede, dessa forma ele identifica objetos, mas não executa segmentação.

O YOLO executa a identificação de objetos dividindo a imagem de entrada em grade, gerando e depois agrupando *bounding boxes*, de forma que um agrupamento espacial é realizado. Isso faz com que o YOLO seja ideal para aplicações em tempo real, pois tal fluxo de operação o torna

mais rápido, porém como desvantagem, o agrupamento espacial faz com que a arquitetura tenha mais dificuldade na identificação de objetos pequenos. A Figura 2.17 apresenta novamente o fluxo de funcionamento do YOLO.

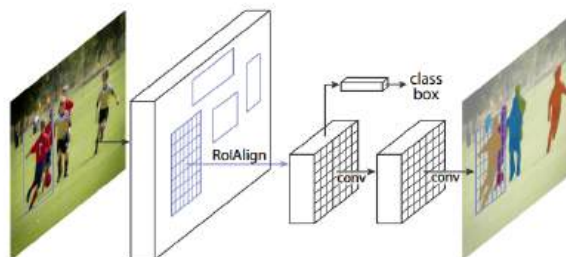
Figura 2.17 – Agrupamento espacial YOLO.



Fonte: adaptado de Redmon et al. (2016).

O *Mask RCNN* gera regiões de interesse e depois processa tais regiões, tendo assim a capacidade de segmentação de instâncias e sendo mais preciso na identificação de objetos pequenos. Porém, tal fluxo de operação faz com que seja uma arquitetura consideravelmente mais lenta, o que a torna mais limitada para aplicações em tempo real. A Figura 2.18 apresenta novamente o fluxo de funcionamento do *Mask RCNN*.

Figura 2.18 – Segmentação de imagens *Mask RCNN*.



Fonte: Wu et al. (2019).

A Tabela 2.1 apresenta um comparativo entre os modelos das arquiteturas exploradas neste trabalho, em questão do número de camadas e também do número de parâmetros.

Tabela 2.1 – Número de camadas e número de parâmetros (em milhões) de cada modelo.

Modelo	N.º de camadas	N.º de parâmetros
<i>Mask RCNN</i>	20	62.87 M
YOLOv8	74	11.12 M

Fonte: do Autor

O YOLOv8 além da detecção de objetos, também pode realizar segmentação de instâncias, porém neste trabalho ele será utilizado apenas para detecção de objetos, e será comparado aos resultados de segmentação obtidos pelo *Mask RCNN*. O tipo de contexto no qual ambos os modelos serão testados permite este tipo de comparação: um fruto de café detectado ou segmentado, deverá ser classificado e incluído em uma contagem, que posteriormente será confrontada com valores de referência já computados por um especialista da área de maturação de café. Estes modelos, na data atual, são considerados como estado da arte, com relação a detecção de objetos (YOLOv8) e segmentação de instâncias (*Mask RCNN*). A detecção de objetos é uma tarefa consideravelmente menos exigente com relação a custo computacional, sendo assim, em caso de resultados semelhantes, quando e comparada a segmentação no contexto da aplicação, poderá ser considerada como de maior viabilidade de implementação, já que durante a implementação de sistemas deste tipo, mesmo não se tratando de um fator crítico, a otimização no uso de recursos computacionais sempre deve ser considerada.

2.8 Arquitetura Cliente-Servidor

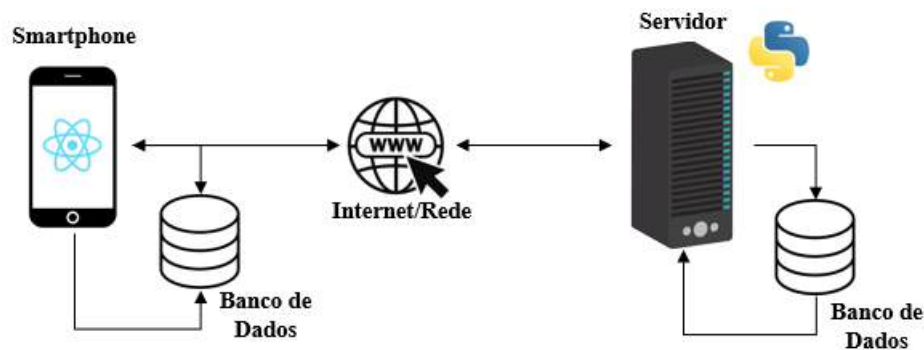
Arquitetura cliente-servidor ou modelo cliente-servidor (HARRINGTON, 2016) é uma arquitetura de rede na qual o processamento da informação é dividido em módulos ou processos distintos. Existe um processo que é responsável pela manutenção da informação (servidores) e outro responsável pela obtenção dos dados (clientes).

Os clientes solicitam um determinado serviço, através do envio de uma mensagem ao servidor. Enquanto o processo servidor está trabalhando a solicitação, o cliente está livre para realizar

outras tarefas. Os servidores oferecem serviços a processos usuários, ou seja, executam a tarefa solicitada e enviam uma resposta ao cliente que se traduz nos dados solicitados. A Figura 2.19 mostra um esquema deste tipo de arquitetura, e que representa o sistema proposto para o projeto.

Citando um exemplo de como funcionam as comunicações cliente/servidor, temos o uso de um navegador para acessar um site do lado do servidor, no qual o usuário ou cliente insere a URL. O servidor DNS procura o endereço IP do servidor da *web* e o fornece ao navegador. O navegador gera uma solicitação HTTP ou HTTPS e o servidor, como produtor, envia os arquivos. O cliente, como consumidor, os recebe e, normalmente, envia solicitações de acompanhamento (quando utilizado o protocolo TCP-IP).

Figura 2.19 – Arquitetura Cliente-Servidor do sistema desenvolvido.



Fonte: do Autor.

Este tipo de arquitetura de rede possui o ponto positivo de que todas as tarefas mais pesadas podem ser centralizadas no servidor, poupando desta forma um maior investimento de recursos para os clientes. O servidor *web* pode ser desde uma máquina conectada a internet dentro de um escritório, a uma máquina virtual presente em alguma plataforma provedora de serviços em nuvem, como *Amazon Web Services*, *Vltar* ou *Microsoft Azure*.

2.9 Protocolo HTTP/HTTPS

O protocolo HTTP (*Hypertext Transfer Protocol*) (KUROSE; ROSS, 2010) nada mais é do que um tipo de protocolo de comunicação utilizado para transferência de informações via *web*, se

tratando de um protocolo do tipo cliente-servidor, onde as requisições de informações são realizadas pelo cliente. Um documento completo é reconstruído a partir dos diferentes sub-documentos obtidos, como por exemplo texto, descrição do *layout* de uma tela, imagens, vídeos ou scripts.

Clientes e servidores se comunicam trocando mensagens individuais (ao contrário de um fluxo de dados). As mensagens enviadas pelo cliente, geralmente um navegador da *web*, são chamadas de solicitações (*requests*), ou também requisições, e as mensagens enviadas pelo servidor como resposta são chamadas de respostas (*responses*).

Desta forma, ao acessar um *link*, está sendo solicitado ao servidor que envie para o nosso navegador todas as informações contidas naquela página a ser acessada, desde arquivos HTML a arquivos de mídia e de estilo (CSS). O protocolo HTTPS nada mais é do que o protocolo HTTP, porém nele, as informações são encriptadas durante seu tráfego, o tornando assim, mais seguro.

2.10 Data Augmentation

Para obtenção de um bom desempenho, os modelos de *Machine Learning* normalmente exigem grandes quantidades de dados para seu treinamento, sendo que, quanto mais complexa a tarefa de classificação, mais dados se fazem necessários, pois os dados devem retratar da melhor forma possível o ambiente onde o modelo será aplicado. Em muitas aplicações reais, a obtenção de grandes quantidades de dados pode ser difícil, custosa ou levar um considerável tempo. Visando solucionar tais dificuldades, pode-se a partir dos dados já existentes, gerar novos dados contendo pequenas variações ou deslocamentos, como por exemplo, no processamento de imagens, uma imagem pode ser recortada, rotacionada, sofrer alterações de luminosidade e brilho; e a imagem resultante pode ser inserida no conjunto de dados, aumentando assim sua dimensão. Este tipo de solução recebe o nome de *Data Augmentation* (MUMUNI, 2022) e pode compreender, no caso de imagens, desde operações simples como as já citadas, até a utilização de IA's para geração de novas imagens.

Os exemplos de possíveis modificações a serem realizadas em uma imagem citadas no parágrafo anterior se enquadram na categorias de **Transformações Geométricas** e **Ajustes de Cor**.

Expandindo um pouco mais os tipos de modificações, outras categorias também podem ser utilizadas:

- **Adição de ruídos:** adição de ruído gaussiano na imagem ou ruído sal e pimenta (adição de pixels pretos e brancos de forma aleatória);
- **Transformações Espaciais Avançadas:** aplicações de distorções de perspectiva ou expansão e contração na imagem;
- **Mixagem de Imagens:**
 - *Mixup*: combinação de duas imagens de forma linear;
 - *Cutmix*: recorte de parte de uma imagem para posterior colagem sobre outra imagem;
 - *Cutout*: criação de buracos em uma imagem a partir do recorte de determinadas partes;
- **Combinações de técnicas:** utilização de várias categorais de técnicas aplicadas conjuntamente para geração de novos dados.

Com relação ao uso de IA para *Data Augmentation*, tal tarefa pode ser realizada por redes GANs (*Generative Adversarial Networks*) (BISWAS et al., 2023), onde uma rede geradora é responsável pela geração de imagens e uma rede discriminadora avalia se uma amostra de dados é real (extraída do *dataset* de treinamento) ou falsa (proveniente da rede geradora). Como exemplos de GAN's que podem ser utilizadas para tal finalidade, tem-se: *Deep Convolutional GAN* (KORA; RAVULA, 2020), *StyleGAN* (FAWAKHERJI et al., 2024) e *CycleGAN* (YANG et al., 2024).

2.11 Transfer Learning

O *Transfer learning* (aprendizado por transferência) é uma técnica em aprendizado de máquina onde um modelo desenvolvido para uma tarefa é reutilizado como ponto de partida para um modelo em uma tarefa diferente, mas relacionada (ZHOU, 2021). Em vez de treinar um modelo do zero, o *Transfer Learning* (TL) aproveita os conhecimentos já adquiridos por um modelo previamente treinado, acelerando o processo de treinamento e melhorando o desempenho quando os dados disponíveis são limitados. Essa técnica é especialmente útil em áreas como visão computa-

cional e processamento de linguagem natural, onde grandes conjuntos de dados e recursos computacionais são frequentemente necessários. Existem diversas formas para realizar o aprendizado por transferência, sendo as principais:

- **Extração de Características:** as camadas iniciais de um modelo pré-treinado são utilizadas como um extrator de características fixo, e as camadas totalmente conectadas são treinadas especificamente para especificamente para uma nova tarefa;
- ***Fine-Tuning*:** envolve o ajuste dos parâmetros de um modelo já treinado para utilização em um *dataset* semelhante;
- **Transferência via Redes Congeladas:** todo o modelo pré-treinado é congelado, e apenas as camadas de saída são treinadas;
- **Camadas Personalizadas:** adição de novas camadas em um modelo pré-treinado, mantendo as camadas originais intactas;
- ***Multitask Learning*:** treinamento de um modelo em várias tarefas simultâneas, onde uma ou mais tarefas irão se beneficiar do aprendizado compartilhado;
- ***Zero-shot Learning*:** permite que o modelo reconheça classes não vistas durante o treinamento, baseando-se em características aprendidas e em descrições semânticas das classes;
- ***Knowledge Distillation*:** envolve a transferência do conhecimento de um modelo grande e complexo (professor) para um modelo menor e mais simples (aluno), permitindo que o modelo menor herde parte do poder de predição do modelo maior.

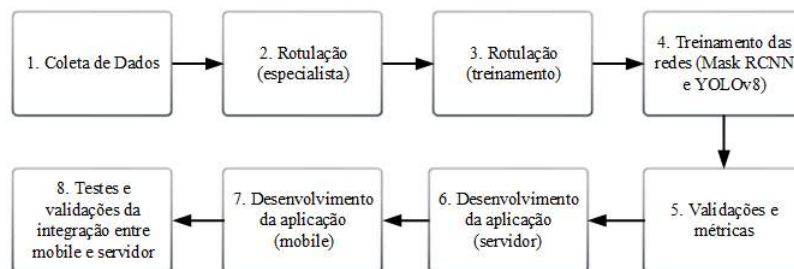
Neste trabalho o TL foi utilizado para ambas as *Deep Learnings* testadas. Foram aproveitados e refinados (*Fine-Tuning*) para a tarefa de detecção e classificação de frutos de café os modelos pré-treinados com o conjunto de dados COCO ¹(*Common Objects in Context*). Se tratando este de um conjunto de dados de *benchmark* amplamente utilizado na comunidade de visão computacional. Ele contém imagens de cenas complexas contendo objetos do dia-a-dia em seus contextos naturais. O COCO é usado para treinar e avaliar modelos de detecção de objetos, segmentação de instâncias e legendagem de imagens. Modelos pré-treinados em COCO, como as CNN's populares, são frequentemente utilizados como base para TL devido à sua robustez e generalização em tarefas de visão computacional.

¹ Disponível em: <<https://cocodataset.org/#home>>

3 METODOLOGIA PARA DESENVOLVIMENTO DA APLICAÇÃO

Para implementação da aplicação, dois diferentes caminhos podem ser tomados, sendo um deles, a implementação onde todo o processamento de imagens e informações ocorreria no dispositivo *mobile*. Esta opção requer adaptação de modelos para funcionarem utilizando o processamento do *smartphone*, e ainda, outro ponto de inflexão a ser contornado, faz referência a limitação do poder de processamento encontrado em dispositivos *mobile*, necessitando que a aplicação seja extremamente otimizada. A outra solução, utilizando uma arquitetura cliente-servidor, na qual o dispositivo *mobile* assume o papel de cliente, e a aplicação desenvolvida teria as funções apenas de capturar imagens e enviar para um servidor, onde a imagem é processada, e o resultado é devolvido a aplicação *mobile*. Esta segunda opção foi escolhida para implementação do sistema proposto nessa Dissertação. A comunicação entre as aplicações foi realizada com a utilização do protocolo HTTP. Um modelo para segmentação de imagens para ser utilizado no servidor também foi obtido a partir do treinamento das arquiteturas com uma base de dados pré-estabelecida. A Figura 3.1 apresenta o fluxograma seguido durante o desenvolvimento do trabalho.

Figura 3.1 – Fluxograma representando as etapas de desenvolvimento do trabalho.



Fonte: do Autor.

A coleta de dados (Etapa 1) envolveu a aquisição de fotos em campo e também através do *Google Images* com objetivo de complementar a base (a base de dados é majoritariamente composta por fotos obtidas em campo). As etapas de rotulação dos dados (Etapas 2 e 3) compreendem a rotulação feita pelo especialista e a rotulação realizada pelo desenvolvedor, com vistas à obtenção do material necessário para o treinamento dos modelos propostos na Etapa 4 (*Mask RCNN* e *YOLOv8*). Visando validar o método proposto e comparar o desempenho das redes profundas empregadas, um conjunto de métricas foi calculado com base nas imagens de teste (Etapa 5). Tendo

escolhido o melhor modelo, deu-se início a implementação do sistema *web*. Primeiramente foi desenvolvida a parte do servidor (Etapa 6), na qual a melhor arquitetura (definida com base nas métricas calculadas) foi implementada. Após a obtenção de um servidor *web* capaz de receber e processar imagens, foi desenvolvida a aplicação *mobile*, capaz de capturar e enviar fotos para o servidor. Por fim, foram realizados testes de integração entre as partes do sistema, testes esses que envolveram todo o fluxo de informações proposto.

3.1 Base de dados

A base de dados utilizada para treinamento e teste dos modelos contou com total de 100 imagens para treinamento e 7 imagens para teste. As imagens que compõe a base envolvem imagens obtidas através do *Google Images* e também imagens cedidas pela empresa especializada nesta área: Gonzaga Treinamentos e Consultoria Agrícola Ltda. Todas as imagens possuem as dimensões de 640 pixels de comprimento e 640 pixels de largura, ou seja, dimensões padronizadas. Esta base contém imagens nas quais 4 estádios de maturação (verde, verde-cana, cereja e passa/seco) são representados, além de apresentar os frutos em diferentes contextos: frutos colhidos, frutos diretamente na planta e imagens contendo apenas um fruto em diferentes cores de fundo. Essa variação de contexto busca tornar o modelo de *Deep Learning* o mais abrangente possível. A Figura 3.2 apresenta algumas das imagens presentes na base de dados.

Importante destacar que parte das imagens que compõem esse banco foram obtidas em condições reais de trabalho no campo, durante operação de colheita do café, dessa forma, as imagens possuem luminosidade e impurezas variadas, conforme realidade do campo. Para obtenção das dimensões citadas anteriormente (100 imagens para treinamento e 7 para teste) foram utilizadas técnicas de *Data Augmentation*: recorte, ajuste de contraste ou brilho, rotação e mudanças de *background*. Inicialmente, a base de dados era composta por 61 imagens para treinamento e 7 imagens para teste, utilizando as técnicas citadas anteriormente foram geradas 39 novas imagens. A Figura 3.3 apresenta um exemplo, onde uma imagem foi dividida em 4 partes, e tais partes sofreram alterações de brilho e de contraste. Com relação a rotulação, foram rotuladas um total de 2855 instâncias, divididas se seguinte forma: 411 da classe verde, 299 da classe verde-cana, 1342 da classe cereja e

Figura 3.2 – Algumas das imagens presentes na base de dados.



Fonte: do Autor.

803 da classe passa/seco. Essa diferença entre a quantidade de instâncias de cada classe é devida a época de aquisição das fotos, que foram obtidas no cenário onde o sistema seria mais útil, durante a época de pré-colheita. A presença de tal desbalanceamento entre classes e os respectivos impactos também serão abordados na apresentação dos resultados.

Outra observação relevante a respeito da base de dados, é que se trata de uma base que evoluiu e se adaptou aos objetivos do trabalho. Sendo uma base inicialmente voltada a classificação de frutos já colhidos e espalhados sobre uma superfície. Porém, durante o desenvolvimento do trabalho e a ocorrência de maior contato com situações reais, foi identificada a necessidade e conveniência de classificação de frutos ainda nos ramos, sem a necessidade de sua colheita. E a partir da detecção desta nova dor, novas imagens que contemplam este novo contexto foram rotuladas e adicionadas a base. As 7 imagens de teste utilizadas contemplaram o contexto dos frutos ainda nos ramos da planta.

Figura 3.3 – Exemplo de *Data Augmentation* realizado.



Fonte: do Autor.

3.2 Pré-processamento

Após a coleta, cada imagem foi avaliada por um especialista, que rotulou segundo um determinado padrão, a classificação de cada fruto a ser considerado nas imagens, e também em uma planilha sintetizou quantos frutos, segundo cada estágio de maturação deveriam ser identificados pelos modelos propostos neste trabalho, conforme a Figura 3.4.

A primeira tarefa a ser realizada, agora pelo desenvolvedor, é a padronização das dimensões de todas as imagens. Em seguida, os frutos foram rotulados uma por uma em todas as imagens de treinamento, utilizando o *software* gratuito LabelMe¹. Este *software* gera arquivos de extensão .json, que são utilizados juntamente com as imagens, para treinamento das arquiteturas. A Figura 3.5 mostra o processo de rotulação sendo realizado.

Diversos autores costumam incluir na fase de pré-processamento, tratamentos que padronizam características das imagens (brilho, contraste, saturação, mudanças na escala de cores), visando tornar toda a base de dados o mais “ideal” possível, como por exemplo em Rodríguez et al. (2020), onde os autores aplicaram diferentes transformações de cores nas imagens antes da aplicação do método de identificação, ou ainda existem trabalhos onde as imagens são coletadas em ambientes controlados, como estufas. Este tipo de pré-processamento ou ambientes controlados, contribuem

¹ Disponível em: <<https://github.com/labelmeai/labelme>>

Figura 3.4 – Avaliação e rotulação do especialista.



(a) Rotulações de referência realizadas pelo especialista.

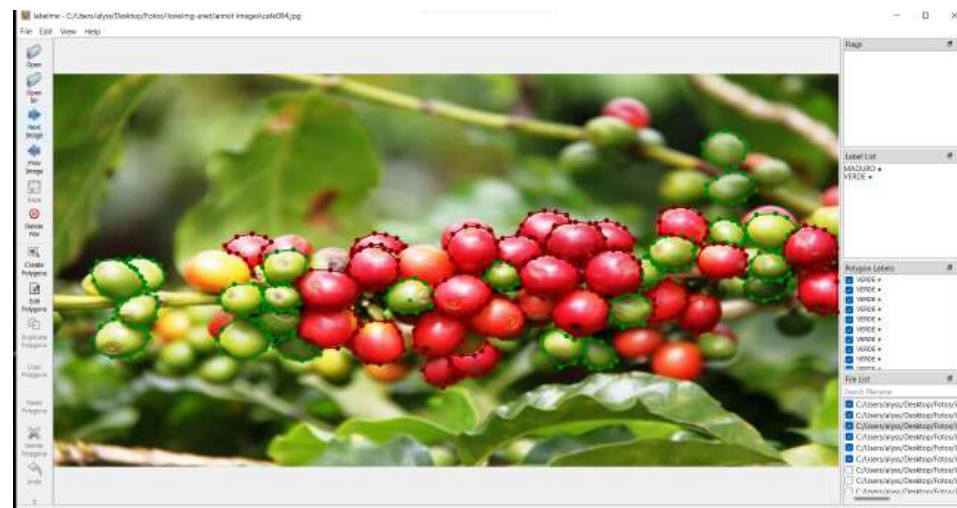
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1															
2	CONTAGEM TELA														
3	Repetições	Verde		Verde Cana		Cereja		Passa		Seco		Total Frutos			
4		Frutos	%	Frutos	%	Frutos	%	Frutos	%	Frutos	%				
5	Amostra 1														
6	Amostra 1.1	19	8,52	38	17,04	62	27,80	54	24,22	50	22,42	223			
7	Amostra 1.2	18	7,50	50	20,83	63	26,25	47	19,58	62	25,83	240			
8	Amostra 1.3	15	6,33	52	21,94	55	23,21	47	19,83	68	28,69	237			
9	Média	17,33	7,45	46,67	19,94	60	25,75	49,33	21,21	60	25,65	233,33			
10	Amostra 2														
11	Amostra 2.1	8	3,33	48	20,00	67	27,92	48	20,00	69	28,75	240			
12	Amostra 2.2	10	4,48	36	16,14	79	35,43	49	21,97	49	21,97	223			
13	Amostra 2.3	16	6,37	32	12,75	78	31,08	56	22,31	69	27,49	251			
14	Média	11,33	4,73	38,67	16,30	74,67	31,47	51,00	21,43	62,33	26,07	238			

Verde
Verde Cana
Cereja
Passa
Seco

(b) Síntese da contagem esperada em cada imagem.

Fonte: Gonzaga Treinamentos e Consultoria Agrícola Ltda.

Figura 3.5 – Captura de tela do processo de rotulação de imagens utilizando o *software* LabelMe



Fonte: do Autor.

de forma significativa para a performance do modelo e para a obtenção de bons resultados. No entanto, trazem consigo a necessidade de implementação dos mesmos tratamentos quando o modelo for implementado em uma aplicação real, caso contrário, sua performance poderá ser extremamente prejudicada. Neste trabalho, o caminho seguido foi diferente, pois as imagens não sofreram edições visando facilitar a tarefa dos modelos, o que pode dificultar a obtenção de bons resultados e também fazer com que amplas bases de dados sejam necessárias, porém ao mesmo tempo, dispensa a implementação de funcionalidades de pré-processamento no sistema de uso prático, e faz com que o modelo aprenda com imagens de características variadas (dentro de um **POP**² definido), estando assim plenamente preparado para uso em campo.

3.3 Treinamento dos Modelos

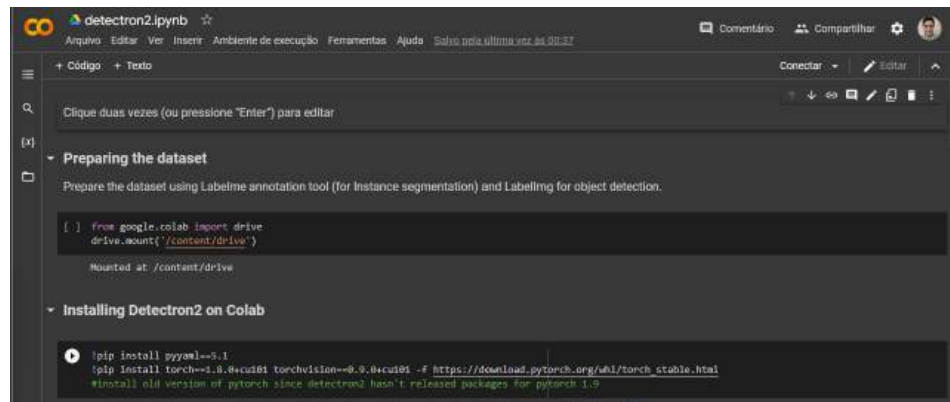
O processo de treinamento, frente aos passos que serão descritos, não diferem entre os modelos (*Mask RCNN* e *YOLOv8*), portanto o uso dos termos “arquitetura” e “modelo” fazem referência a ambas as redes.

Após o processo de rotulação dos frutos, cada imagem possui um arquivo de extensão .json associado; arquivo este que contém as coordenadas de cada instância rotulada em sua imagem correspondente, sendo esta combinação de arquivos, a nova composição do banco de dados. O *Mask RCNN* e o *YOLOv8* foram implementados com a utilização da linguagem *Python* (versão 3.10.9), no ambiente de desenvolvimento virtual *Google Colab* (vide Figura 3.6). Este ambiente foi escolhido por disponibilizar um alto poder computacional (possibilidade de utilização de GPUs), necessário para se ter um melhor tempo de processamento durante o treinamento, oferecendo uma GPU Tesla T4 de 15GB de VRAM, 12 GB de RAM, 80GB de espaço em disco e um processador Intel Xeon de 2,0GHz. Tal demanda de poder computacional se deve ao fato do grande número de operações realizadas pelas arquiteturas e também da biblioteca *PyTorch*, que permite a utilização de estruturas de dados complexas, denominadas tensores. Os tensores são estruturas de dados que carregam os dados a serem processados juntamente com todo o histórico de processamentos realizados sobre tais dados. No repositório do *Facebook* no *GitHub* encontram-se diversos modelos

² Procedimento Operacional Padrão

previamente treinados para detecção de objetos (os modelos COCO) . Após a realização de diversos testes, o modelo pré-treinado que apresentou melhor resultado após o treinamento de adequação para classificação de frutos de café, foi escolhido.

Figura 3.6 – *Google Colab*.



Fonte: do Autor.

Após o carregamento do banco de dados, a pasta que o contém deve ser informada à arquitetura, assim como as classes que foram rotuladas, o número de classes a serem identificadas, uma taxa de aprendizagem e o número máximo de épocas que o treinamento deverá durar. Ao fim do treinamento, um arquivo denominado “model_final.pth” (*Mask RCNN*) ou “best.pt” (YOLOv8) é gerado. Este arquivo contém todas as informações aprendidas pela arquitetura durante o treinamento, e deve ser utilizado durante o processo de testes com novas imagens. Como saída do processo de inferência (entrada de uma imagem na rede e processamento das previsões) de uma imagem quaisquer, tem-se três informações importantes: (i) quantidade de instâncias de cada classe detectada, (ii) a localização de cada instância, visando o desenho das *boundingboxes* sobre a imagem processada e (iii) os *scores* (valores entre 0 e 1 que indicam o quanto a arquitetura considerou determinada instância pertinente a uma classe específica). Durante o processo de inferência, o valor mínimo de *score* para que uma instância seja considerada pode ser ajustado.

O computador utilizado para realização de todos esses procedimentos foi um *notebook* Asus, modelo X515JF, processador Intel Core i5, 8GB de memória RAM e 256GB de armazenamento (SSD NVME). O *Google Colab* foi utilizado para facilitar e agilizar o processo de treinamento, sendo assim, vale ressaltar o fato de que ambos os modelos treinados podem reali-

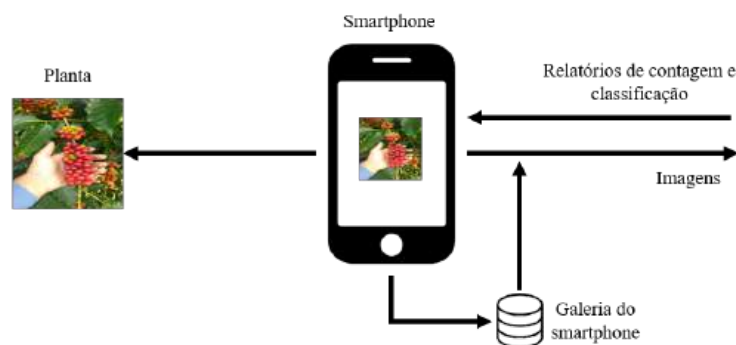
zar inferências sobre imagens sem a necessidade de que o *hardware* do servidor possua acesso a GPU's.

O uso de *Data Augmentation* e TL com os modelos COCO foram necessários devido ao banco de dados restrito para o desenvolvimento proposto nesta Dissertação. Com o sistema proposto implementado e sendo utilizado na prática, é esperado que a base de dados seja ampliada, abrindo assim a possibilidade de obtenção de um novo modelo a partir da nova base de dados, visando sempre melhorar a acurácia das redes neurais implementadas.

3.4 Aplicação *Mobile* (Cliente)

A aplicação *mobile* foi implementada com a utilização do *framework* React Native, baseado em Javascript. Esta aplicação possui um banco de dados local (implementado pela própria galeria do *smartphone*), no qual a imagem é armazenada, e posteriormente enviada para processamento. Esta implementação do banco de dados garantirá que todas as imagens capturadas sejam enviadas, pois sem um banco de dados, as imagens teriam que ser diretamente enviadas, e desta forma, caso haja problemas de conexão com o servidor, as imagens são perdidas. Enquanto, ao se utilizar um banco de dados, pode-se salvar todas as capturas e enviar uma por vez, sempre que a imagem anterior tenha sido enviada com sucesso. A Figura 3.7 ilustra essa parte do sistema.

Figura 3.7 – Cliente *mobile*.

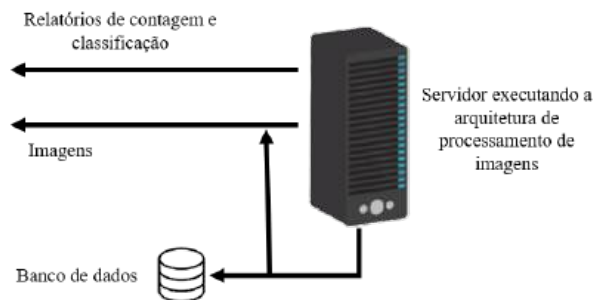


Fonte: do Autor.

3.5 Aplicação web (Servidor)

O servidor foi desenvolvido utilizando a linguagem Python. Desta forma, as arquiteturas, juntamente a modelos treinados, foram incorporadas ao servidor *web*, capaz de receber imagens, processá-las visando a identificação e classificação dos frutos, e por fim, realizar o envio dos resultados para a aplicação *mobile*. Assim como na aplicação *mobile*, pensando em questões de robustez, o servidor também contém um banco de dados, no qual as imagens recebidas são armazenadas, para posterior processamento. Logo após o processamento, os resultados são armazenados também no banco de dados, para posterior envio a aplicação *mobile*. Isto novamente visando evitar a perda de imagens e de informações devido a problemas de assincronia ou de comunicação via rede entre as aplicações (TANENBAUM, 2011). Além de robustez, o armazenamento dos resultados em banco possibilita a consulta de dados históricos. A Figura 3.8 ilustra esse lado do sistema.

Figura 3.8 – Servidor executando a arquitetura de processamento de imagens.



Fonte: do Autor.

3.6 Métricas para avaliação de desempenho e comparações

Com relação as métricas para avaliação de desempenho de cada modelo e posteriores comparações, foram calculadas métricas internas a cada modelo, tais como *Average Precision* - *AP*, *Mean Average Precision* - *MAP*, *Recall* e métricas relacionadas a contagem e identificação de frutos realizados pelas arquiteturas quando comparadas à avaliação feita pelo especialista.

A métrica *AP* mede a qualidade da classificação positiva do modelo, indicando a proporção de instâncias do tipo verdadeiro positivo, conforme a equação 3.1.

$$AP = \frac{TP}{TP + FP} \quad (3.1)$$

onde TP são os verdadeiro positivos e FP os falso positivos.

A MAP é calculada a partir dos valores de AP obtidos para cada classe, ou seja, é a média das APs obtidas, conforme a equação 3.2.

$$MAP = \frac{1}{N} \sum_{i=1}^N AP_i \quad (3.2)$$

onde N é o número de classes.

Na identificação de instâncias, pode-se também avaliar a magnitude da intersecção entre a área do rótulo (polígono desenhado durante a rotulação) e a área da instância predita pela arquitetura, em um valor que vai de 0 a 1 (ou de 0% a 100%). Este valor é denominado IoU (*Intersection over Union*, onde 0 indica que não houve intersecção e 1 indica uma intersecção total). A AP, geralmente considera valores de IoU variando de 0,50 a 0,95, em passos de 0,05. Tendo o conceito de IoU definido, serão apresentados resultados também para as métricas AP50 e AP75. Tais métricas são calculadas fixando a IoU em 0,50 ou 0,75 respectivamente, e considerando um verdadeiro positivo se a área de intersecção entre predição e rótulo for, respectivamente, 50% ou 75%.

O *Recall*, também conhecido como sensibilidade, mede a capacidade em encontrar todas as instâncias relevantes do problema, e é dado pela equação 3.3.

$$Recall = \frac{TP}{TP + FN} \quad (3.3)$$

onde FN representa a contagem de falsos negativos.

Para além de tais métricas, realizando comparações com as classificações realizadas pelo especialista, também serão obtidas métricas utilizando o erro percentual absoluto (APE - *Absolute Percentual Error*), dado pela equação 3.4.

$$APE = \left| \frac{CR - CE}{CE} \right| \times 100 \quad (3.4)$$

onde CR é a contagem de frutos realizada pelo modelo e CE representa a contagem fornecida pelo especialista.

Como método de validação cruzada foi utilizado o k -fold. Método no qual os dados são divididos em k subconjuntos e o modelo é treinado k vezes utilizando $k - 1$ folds para o treinamento, e o $fold$ restante para validação.

4 RESULTADOS

Utilizando a técnica de validação cruzada *k-fold* e levando em consideração a quantidade de imagens, foram obtidos 5 *folds* contendo 20 imagens cada. Realizando uma sequência de treinamentos, testes e validações para cada arquitetura de rede neural proposta, foram obtidos 5 modelos para cada uma (um modelo para cada *fold*). Os valores de *score* (limiar) definidos para o *Mask RCNN* e *Yolov8*, foram, respectivamente, 0,1 e 0,05. Estes limiares foram determinados empiricamente, onde foram testados valores entre 0,01 e 0,9 (variando em passos de 0,01) para cada uma das 7 imagens de teste. As Tabelas 4.1 e 4.2 apresentam as métricas de desempenho obtidas para os modelos de *Mask RCNN* e de *YOLOv8*, respectivamente.

Tabela 4.1 – Métricas *Mask RCNN*.

Fold	Classes	AP	AP50	AP75	MAP	Recall
1	Verde	0,416	0,599	0,591	0,485	0,784
	Verde-cana	0,430				
	Cereja	0,647				
	Passa/Seco	0,450				
2	Verde	0,590	0,673	0,672	0,562	0,887
	Verde-cana	0,620				
	Cereja	0,723				
	Passa/Seco	0,314				
3	Verde	0,592	0,701	0,696	0,597	0,885
	Verde-cana	0,581				
	Cereja	0,747				
	Passa/Seco	0,467				
4	Verde	0,700	0,759	0,759	0,638	0,892
	Verde-cana	0,740				
	Cereja	0,729				
	Passa/Seco	0,415				
5	Verde	0,681	0,685	0,685	0,549	0,765
	Verde-cana	0,471				
	Cereja	0,663				
	Passa/Seco	0,380				

Fonte: do Autor.

Tanto para o *Mask RCNN* quanto para o *YOLO*, o *fold* de melhor performance foi o de número 4. Em especial para a classe Passa/Seco houve uma maior quantidade de falsos posi-

Tabela 4.2 – Métricas YOLO.

Fold	Classes	AP	AP50	AP75	MAP	<i>Recall</i>
1	Verde	0,610	0,655	0.54	0.579	0,682
	Verde-cana	0,410				
	Cereja	0,638				
	Passa/Seco	0,659				
2	Verde	0,537	0,700	0.586	0.606	0,694
	Verde-cana	0,688				
	Cereja	0,803				
	Passa/Seco	0,395				
3	Verde	0,725	0,765	0.650	0.558	0,830
	Verde-cana	0,398				
	Cereja	0,649				
	Passa/Seco	0,461				
4	Verde	0,634	0,793	0.667	0.708	0,734
	Verde-cana	0,826				
	Cereja	0,683				
	Passa/Seco	0,691				
5	Verde	0,787	0,651	0.522	0,535	0,776
	Verde-cana	0,336				
	Cereja	0,675				
	Passa/Seco	0,343				

Fonte: do Autor.

vos (menores valores de AP). Para as demais classes, tal característica é menos evidente, porém presente, e se observada em conjunto com os valores de *Recall* que se apresentaram mais altos, fica evidente que os modelos funcionaram de forma a retornar mais falsos positivos e menos falsos negativos.

Pode-se ainda, em uma derradeira comparação envolvendo as métricas, calcular uma média geral para todos os *folds* conforme a Tabela 4.3.

Tabela 4.3 – Média das métricas.

Arquitetura	Classes	AP	AP50	AP75	MAP	<i>Recall</i>
<i>Mask RCNN</i>	Verde	$0,596 \pm 0,101$	$0,683 \pm 0,052$	$0,681 \pm 0,054$	$0,566 \pm 0,051$	$0,843 \pm 0,056$
	Verde-cana	$0,568 \pm 0,110$				
	Cereja	$0,702 \pm 0,039$				
	Passa/Seco	$0,405 \pm 0,055$				
YOLO	Verde	$0,659 \pm 0,088$	$0,713 \pm 0,057$	$0,593 \pm 0,058$	$0,598 \pm 0,060$	$0,743 \pm 0,055$
	Verde-cana	$0,532 \pm 0,191$				
	Cereja	$0,690 \pm 0,059$				
	Passa/Seco	$0,510 \pm 0,140$				

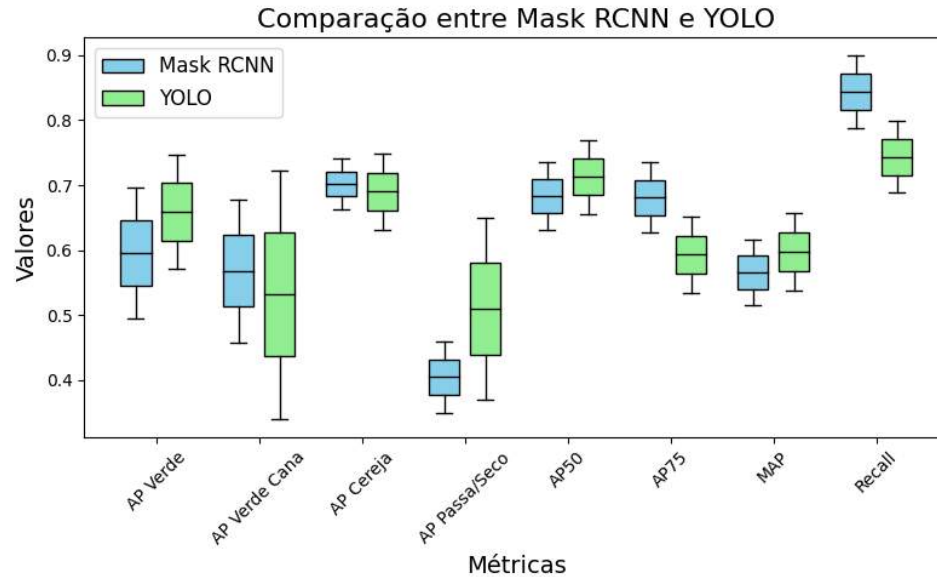
Fonte: do Autor.

Observando a Tabela 4.3 fica ainda mais claro a questão dos falsos positivos já comentada. Com a média de cada métrica, é possível observar que as classes Passa/Seco e Verde-cana foram as mais afetadas. Tal fato já era esperado, pois a classe Passa/Seco pode se confundir com a classe Cereja, e a classe Verde-cana, com as classe Verde. A linha de distinção entre esses pares de classes é tênue, ao ponto de que diferentes especialistas podem discordar quanto a distinção de frutos destes pares de classes.

A partir das informações sintetizadas pela Tabela 4.3, um gráfico do tipo *boxplot* foi obtido, conforme ilustra a Figura 4.1. Esse tipo de representação gráfica permite uma melhor visualização dos dados, tendo em vista o objetivo de definir qual dos métodos obteve o melhor desempenho.

Observando a Figura 4.1, fica evidente que apenas para o *recall* houve uma diferença mais significativa. Porém, é interessante realizar um teste estatístico para obtenção de valores concretos, além do que se pode observar qualitativamente.

Figura 4.1 – *Boxplot* apresentando cada uma das métricas.



Fonte: do Autor.

Utilizando o *software* Sisvar¹ e tomando os dados apresentados nas Tabelas 4.1 e 4.2, foi realizada uma análise de variância (ANOVA)(MONTGOMERY, 2020), cujos p -valores são apresentados na Tabela 4.4. A ANOVA é uma técnica estatística utilizada para comparação de médias, cujo objetivo é determinar se existem diferenças estatisticamente significativas entre médias. Como resultado da análise, tem-se a variabilidade total dos dados e também a estatística F (p -valores), que é dada pela razão entre a variabilidade entre *folds* e a variabilidade dentro dos *folds*. A partir dos p -valores, o Teste F (MONTGOMERY, 2020) pode ser aplicado considerando um determinado nível de significância, para assim aceitar ou rejeitar a hipótese estabelecida (geralmente chamada de H_0) de que as médias não são estatisticamente diferentes.

Considerando um Teste F ao nível de 5% de significância ($p > 0,05$), pode-se afirmar, estatisticamente, que o *Mask RCNN* obteve um desempenho superior ao YOLOv8 em relação à métrica *recall*, conforme a Tabela 4.5, onde as médias seguidas por mesma letra, para cada métrica, não diferem entre si ao nível de 5% de significância.

¹ Disponível em: <<https://des.ufla.br/~danielff/programas/sisvar.html>>

Tabela 4.4 – p -valores obtidos pela análise de variância.

Métrica	p -valor
AP Verde	0,374
AP Verde-cana	0,747
AP Cereja	0,740
AP passa/seco	0,202
AP50	0,468
AP75	0,057
MAP	0,455
Recall	0,035

Fonte: do Autor.

Tabela 4.5 – Resultado após aplicação do Teste F.

Rede	AP Verde	AP Verde-cana	AP Cereja	AP Passa/Seco	AP50	AP75	MAP	Recall
Média <i>Mask RCNN</i>	0,596 a	0,568 a	0,702 a	0,405 a	0,683 a	0,681 a	0,566 a	0,843 a
Média YOLO	0,659 a	0,532 a	0,690 a	0,510 a	0,713 a	0,593 a	0,598 a	0,743 b

Fonte: do Autor.

É importante ressaltar que a comparação entre métricas se apresenta como uma das possíveis formas de se realizar uma avaliação das arquiteturas, pois as métricas de identificação de objetos utilizadas envolvem principalmente uma comparação entre a área que deveria ser coberta pela segmentação do fruto, e a área que realmente foi coberta durante os testes. Como o principal objetivo do trabalho é realizar uma contagem e uma classificação do grau de maturação, o importante é que o modelo proposto identifique um fruto e sua maturação, independentemente se durante a segmentação, a área segmentada seja equivalente ao fruto como um todo. Portanto, uma avaliação dos modelos, segundo contagem e classificação dos frutos também foi realizada.

Para utilização da aplicação diretamente em campo, um POP deve ser delineado, considerando principalmente a forma como os frutos serão capturados pela foto. Para além disso, sabe-se também que uma foto não será capaz de capturar todos os frutos em um ramo, pois sobreposições entre frutos irão ocorrer, já que os frutos brotam no ramo acompanhando a disposição esférica do mesmo. Ainda que de forma primordial, com o objetivo de mostrar o potencial dos modelos obtidos, foi estabelecido através da consulta com um especialista, de que a forma ideal para captura de imagens, seria a realização de capturas por ramos isolados, conforme a Figura 4.2 ilustra.

Figura 4.2 – Exemplo de possível padrão para captura de fotos.



Fonte: Gonzaga Treinamentos e Consultoria Agrícola Ltda.

O campo de visão capturado pela foto é capaz de englobar aproximadamente $\frac{2}{3}$ de todos os frutos do ramo. Sendo assim, para uma estimativa mais precisa de produção da lavoura, a contagem realizada pela aplicação deverá ser multiplicada por $(1 + \frac{1}{3})$. As 7 imagens de teste seguem o padrão citado. Essas imagens foram processadas e os resultados são apresentados de forma detalhada nas próximas seções, em termos de contagem e identificação do estágio de maturação. Ressalta-se que, as referências (dados rotulados) para o cálculo de métricas de contagem e identificação foram definidas a partir da avaliação prévia realizada por um especialista.

4.1 Contagem

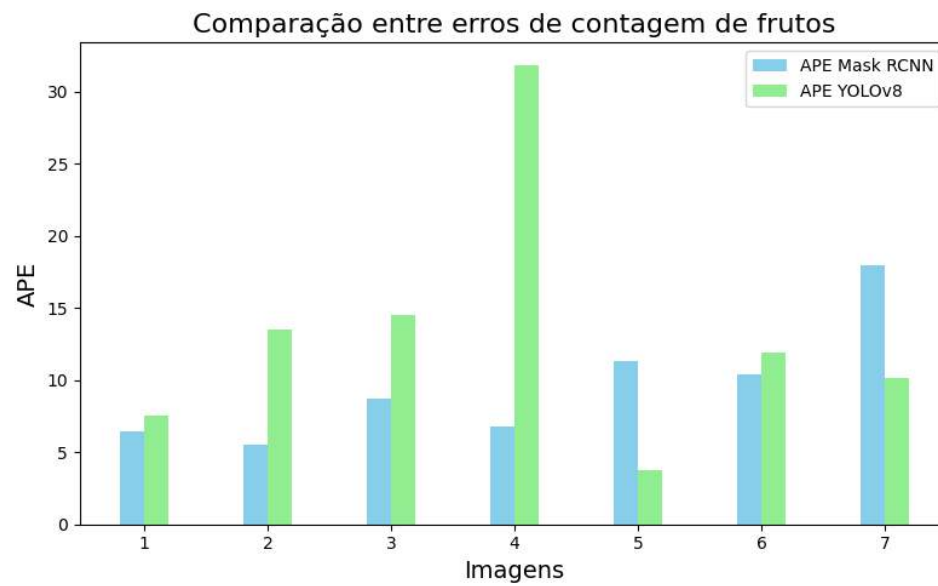
Todo o sistema proposto tem como principal objetivo a classificação dos frutos com relação a seus diferentes graus de maturação. Felizmente, o processo de classificação proposto traz como consequência a possibilidade de contagem dos frutos. No dia a dia do produtor, essa contagem pode contribuir para a estimativa de produção, de forma que o produtor possa capturar imagens de forma amostrada de sua lavoura, avaliar tais imagens observando a quantidade de frutos, e assim estimar a produção. Visando observar de forma geral a capacidade das arquiteturas em identificar instâncias e consequentemente contá-las, pode-se somar todos os valores referentes a cada classe, e construir uma tabela expondo o total de frutos identificados por imagem, frente ao total identificado pelo especialista. Este resultado é mostrado na Tabela 4.6, os erros de contagem são representados também no gráfico da Figura 4.3.

Tabela 4.6 – Contagem de frutos por imagem.

Imagem	Especialista	<i>Mask RCNN</i>	YOLOv8	APE <i>Mask RCNN</i> (%)	APE YOLOv8 (%)
1	93	87	86	6,45	7,53
2	126	119	109	5,56	13,49
3	69	75	59	8,70	14,49
4	93	98	64	5,38	31,18
5	53	59	51	11,32	3,77
6	67	74	75	10,45	11,94
7	138	151	115	17,97	10,16

Fonte: do Autor.

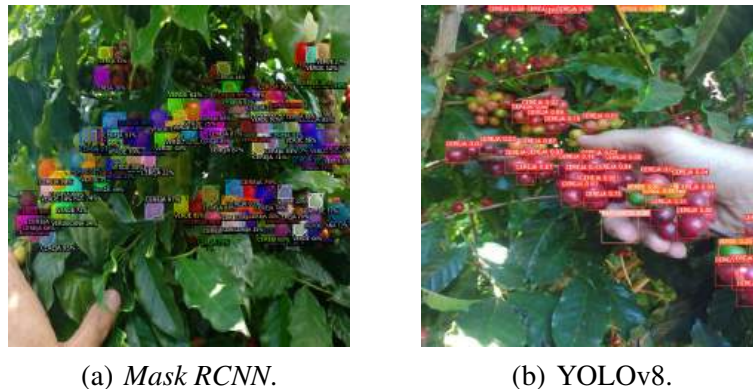
Figura 4.3 – Comparação entre erros de contagem.



Fonte: do Autor.

É interessante observar as diferentes performances dos modelos para cada imagem, com o objetivo de identificar pontos de facilidade e dificuldade de cada modelo ao realizar o processo de inferência. O *Mask RCNN* apresentou sua melhor performance para a imagem 4 (Figura 4.4(a)), sendo capaz de identificar praticamente todos os frutos em primeiro plano na imagem, e ignorando a maioria dos frutos em segundo plano, que realmente não devem entrar na contagem. O modelo de YOLOv8 apresentou melhores resultados para a imagem 5 (ilustrada na Figura 4.4(b)), na qual pode-se observar uma boa identificação dos frutos em primeiro plano na imagem, apesar de uma identificação errônea de um dos dedos do especialista que capturou a imagem, e também da não identificação de alguns frutos, em especial das classes verde e verde-cana.

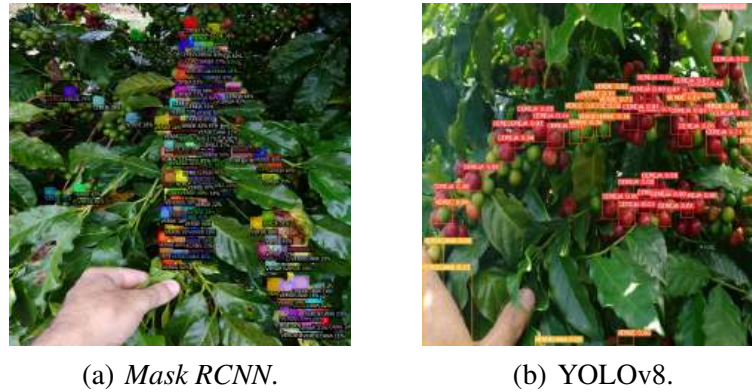
Figura 4.4 – Imagens onde os modelos obtiveram os melhores desempenhos com relação a contagem.



Fonte: do Autor.

Observando as imagens onde os modelos apresentaram seus piores desempenhos, tem-se que o *Mask RCNN* performou de forma menos satisfatória na imagem 7 (ilustrada na Figura 4.5(a)) e o YOLOv8 na imagem 4 (ilustrada na Figura 4.5(b)). Na imagem 7, o modelo contou uma maior quantidade de frutos se comparado ao valor de referência, devido ao fator de ter incluído na contagem uma maior quantidade de frutos que estavam contidos no segundo plano da imagem. Já na imagem 4, o modelo apresentou dificuldades novamente na identificação de frutos verde e verde-cana, dificuldade esta que se mostrou presente em todas as imagens onde o modelo de YOLOv8 foi testado, e que serão avaliadas de forma mais detalhada na próxima seção.

Figura 4.5 – Imagens onde os modelos obtiveram os piores desempenhos com relação a contagem.



Fonte: do Autor.

Pode-se observar também o tempo que cada arquitetura utilizou para processar cada imagem, conforme a Tabela 4.7. O *Mask RCNN* apresentou uma média de $198,14 \pm 11,04ms$, e o YOLOv8 uma média de $44,86 \pm 3,98ms$. Essa diferença entre os tempos pode ser facilmente entendida se a Tabela 2.1 for retomada, na qual fica visível o número de parâmetros consideravelmente maior do modelo de *Mask RCNN*, que possui aproximadamente 5,65 vezes mais parâmetros e sendo 4,42 vezes mais lento se comparado ao YOLOv8.

Tabela 4.7 – Tempos de processamento de cada imagem em milissegundos.

Imagem	<i>Mask RCNN</i>	YOLOv8
1	196	40
2	194	47
3	177	49
4	200	46
5	198	39
6	216	50
7	206	43

Fonte: do Autor

No geral, pode-se observar um melhor desempenho por parte do *Mask RCNN*, apesar de ter sido a arquitetura mais lenta, com um APE médio de 9,61%, enquanto o YOLOv8 apresentou um APE médio de 13,31%. As informações de tempo foram expostas justamente para possibilidade de observação deste *trade-off* entre performance na identificação dos frutos e velocidade de processamento.

4.2 Identificação do Estádio de Maturação

A Tabela 4.8 expõe os resultados numéricos obtidos juntamente aos resultados apontados pelo especialista em cada imagem no que se refere à identificação do grau de maturação do fruto de café. As colunas de nomenclatura “Valor” contém o número de frutos contados pelo especialista e pelos modelos propostos (*Mask RCNN* e *YOLOv8*); enquanto as colunas denominadas “%” apresentam o quanto cada classe representa frente ao total de frutos contados. Estas duas colunas (“Valor” e “%”) foram inseridas, pois a coluna “Valor” fornece informações com um viés no qual uma estimativa de produção pode ser realizada, enquanto a coluna “%” fornece informações que vão permitir estimar o estágio de maturação geral da lavoura. Nas duas últimas colunas da tabela são expostas informações de APE para cada modelo.

Observando de forma geral os resultados tem-se que a classe onde ambos os modelos obtiveram melhores resultados, foi a classe cereja, em contrapartida os resultados menos satisfatórios foram obtidos para a classe verde. Com relação aos dados apresentados pelas colunas “%”, foram calculados os valores de erro médio absoluto para cada classe, conforme Tabela 4.9. Observando os resultados neste contexto, pode-se concluir que ambos os modelos apresentaram um bom desempenho, com o maior valor de erro ocorrendo para o *YOLOv8*, sendo de apenas 5,59%.

Visando agora avaliar mais especificamente os resultados para contagem por classes, o APE (valores apresentados na Tabela 4.8) pode ser tomado como base para avaliação de qual modelo performou melhor na classificação de frutos, porém na avaliação realizada e apresentada, sua magnitude foi inversamente proporcional a quantidade de frutos, o que fez com que seu valor fosse alto em casos específicos, como por exemplo na Imagem 1, onde o modelo de *YOLOv8* identificou erroneamente 2 frutos verdes, enquanto o especialista identificou 0 frutos da classe verde, o que fez com que o APE assumisse um valor de 200%. Frente a isto, para uma avaliação mais assertiva, os resultados serão apresentados e avaliados segundo cada imagem. Para simplificação das análises a seguir, o modelo de *Mask RCNN* será tratado apenas como *Mask* e de *YOLOv8* como *YOLO*.

Na Imagem 1, ambos os modelos detectaram frutos verdes de forma errônea, enquanto apenas o *YOLO* detectou frutos verde-cana. Essas classes não foram identificadas pelo especialista, e em termos de contagem, o maior erro ocorreu quando o *YOLO* apresentou uma contagem de 3

Tabela 4.8 – Análise da identificação do estágio de maturação.

Imagem	Estádio	Especialista		<i>Mask RCNN</i>		YOLO		APE (%)	
		Valor	%	Valor	%	Valor	%	<i>Mask</i>	YOLO
1	Verde	0	0,0	1	1,15	2	2,33	100,0	200,0
	Verde-cana	0	0,0	0	0,0	3	3,49	0,0	300,0
	Cereja	81	87,10	72	82,76	76	88,37	11,11	6,17
	Passa/Seco	12	12,90	14	16,09	5	5,81	16,67	58,33
2	Verde	2	1,59	4	3,36	5	4,59	100,0	150,0
	Verde-cana	55	43,65	44	36,98	40	36,70	20,00	27,27
	Cereja	59	46,82	63	52,94	53	48,62	6,78	10,17
	Passa/Seco	10	7,94	8	6,72	11	10,09	20,00	10,00
3	Verde	0	0,0	1	1,33	1	1,70	100,0	100,0
	Verde-cana	0	0,0	0	0,0	0	0,0	0,0	0,0
	Cereja	63	91,30	72	96,00	58	98,30	14,29	7,94
	Passa/Seco	6	8,70	2	2,67	0	0,0	66,67	100,0
4	Verde	38	40,86	35	35,71	26	40,63	7,89	31,58
	Verde-cana	7	7,53	2	2,04	4	6,25	71,43	42,86
	Cereja	46	49,46	61	62,25	32	50,0	32,61	30,44
	Passa/Seco	2	2,15	0	0,0	2	3,13	200,0	0,0
5	Verde	4	7,55	9	15,25	6	11,76	125,0	50,00
	Verde-cana	3	5,66	1	1,70	1	1,96	66,67	66,67
	Cereja	46	86,80	49	83,05	43	84,31	6,52	6,52
	Passa/Seco	0	0,0	0	0,0	1	1,96	0,0	100,0
6	Verde	1	1,49	4	5,41	4	5,33	300,0	300,0
	Verde-cana	19	28,36	17	22,97	7	9,33	10,53	63,16
	Cereja	47	70,15	53	71,62	63	84,00	12,77	34,04
	Passa/Seco	0	0,0	0	0,0	1	1,33	0,0	100,0
7	Verde	90	65,22	83	54,97	56	48,70	7,78	37,78
	Verde-cana	6	4,35	25	16,56	9	7,83	316,67	50,00
	Cereja	42	30,44	43	28,48	49	42,60	2,38	16,67
	Passa/Seco	0	0,0	0	0,0	1	0,87	0,0	100,0

Fonte: do Autor.

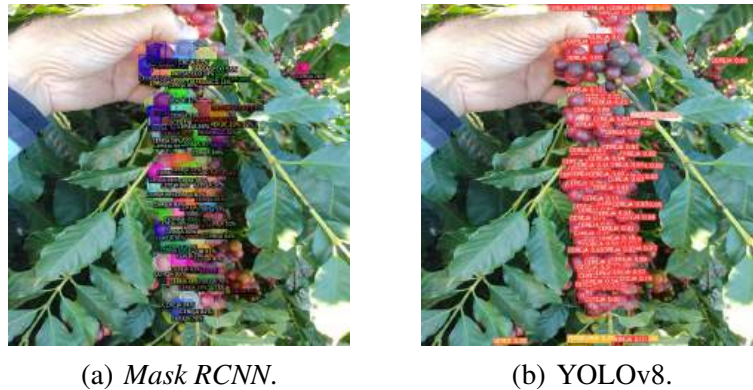
Tabela 4.9 – Erros Médios para as informações de porcentagem tomando a referência do especialista.

Estádio	Erro Médio “%” <i>Mask RCNN</i>	Erro Médio “%” YOLOv8
Verde	$4,47 \pm 3,22$	$4,54 \pm 5,04$
Verde-cana	$4,81 \pm 3,89$	$5,42 \pm 5,91$
Cereja	$5,02 \pm 3,50$	$5,59 \pm 5,09$
Passa/seco	$1,80 \pm 2,07$	$3,30 \pm 2,97$

Fonte: do Autor

frutos verde-cana. Quanto às classes cereja e passa/seco, o maior erro foi cometido pelo YOLO, ao detectar 7 frutos a menos do que a referência para a classe passa/seco. Fica evidente na Figura 4.6 (b), em sua parte superior central, parte dos frutos passa/seco não identificados.

Figura 4.6 – Imagem 1 rotulada pelos modelos propostos: (a) *Mask RCNN* e (b) YOLOv8.

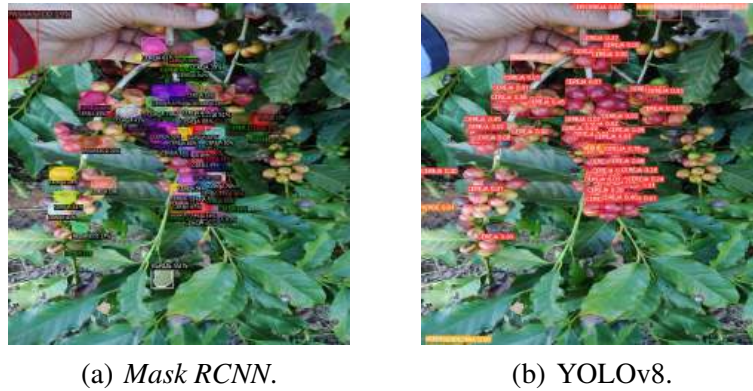


Fonte: do Autor.

Na Imagem 2 a maior dificuldade dos modelos ficou visível com relação a identificação de frutos verde-cana. Ambos os modelos contaram frutos verdes em quantidades superiores à correta, o que indica que frutos verde-cana foram tratados como verde, além do fato de que ambos os modelos apresentaram dificuldades para identificar frutos dos ramos laterais, frutos esses em sua maioria da classe verde-cana, conforme pode ser observado na Figura 4.7.

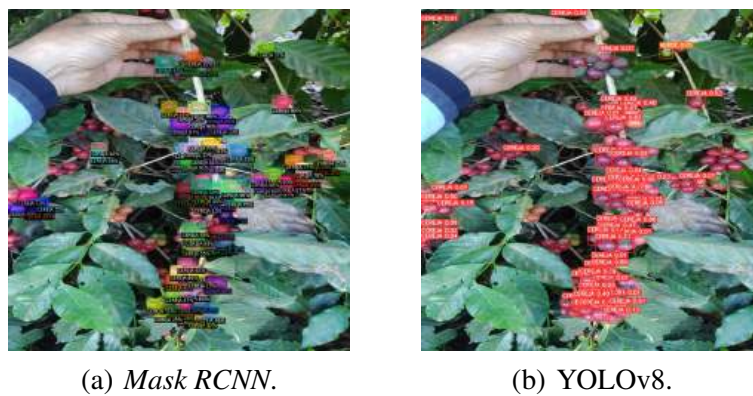
Na Imagem 3, mais uma vez as duas classes presentes foram cereja e passa/seco. A principal observação pode ser feita com relação aos frutos passa/seco não identificados pelo YOLO ou confundidos com a classe cereja pelo *Mask*, conforme a Figura 4.8.

Figura 4.7 – Imagem 2 rotulada pelos modelos propostos: (a) *Mask RCNN* e (b) YOLOv8.



Fonte: do Autor.

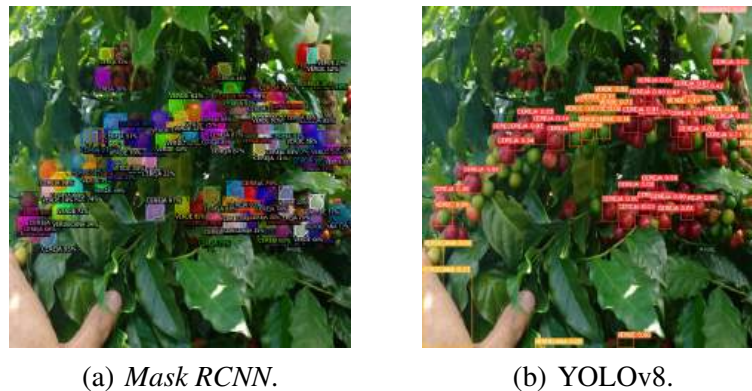
Figura 4.8 – Imagem 3 rotulada pelos modelos propostos: (a) *Mask RCNN* e (b) YOLOv8.



Fonte: do Autor.

A Imagem 4 apresenta uma boa mescla de frutos verdes e cereja. Observando a Figura 4.9, fica evidente a não identificação de parte dos frutos verdes pelo YOLO, enquanto como ponto negativo do *Mask*, pode-se observar a não identificação dos 2 frutos passa/seco observados pelo especialista, além de uma sobrecontagem de frutos cereja, devido ao fato da identificação não esperada de frutos presentes em segundo plano na imagem, apesar de como ponto positivo, ter apresentado uma excelente capacidade de identificação de frutos.

Figura 4.9 – Imagem 4 rotulada pelos modelos propostos: (a) *Mask RCNN* e (b) YOLOv8.



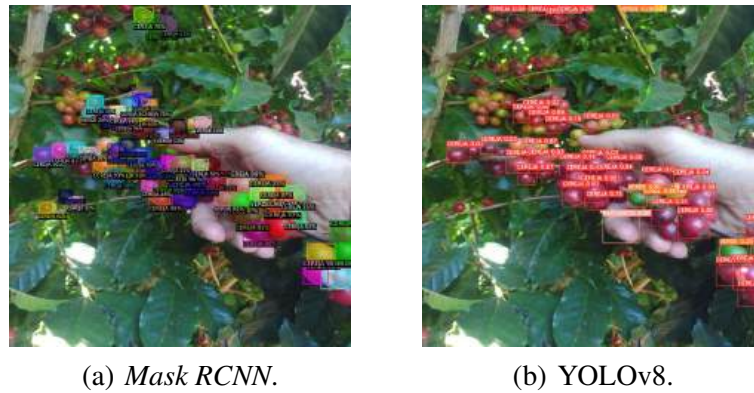
Fonte: do Autor.

Na Imagem 5 ambos os modelos realizaram uma sobrecontagem de frutos verdes, e uma subcontagem de frutos verde-cana, indicando uma possível confusão feita pelos modelos ao classificar estas classes. Para as demais classes, bons resultados foram obtidos, e o APE de 100% computado para o YOLO na classe passa/seco foi referente a identificação de um fruto, quando não deveria ser identificado nenhum. A Figura 4.10 mostra as imagens e as identificações realizadas pelos modelos.

Na Imagem 6 ambos os modelos realizaram uma sobredetecção de frutos verdes. Para as demais classes, o *Mask* apresentou um bom desempenho, enquanto o YOLO mais uma vez apresentou dificuldades na detecção de frutos verde-cana, como pode ser observado nos frutos da parte central superior da Figura 4.11(b).

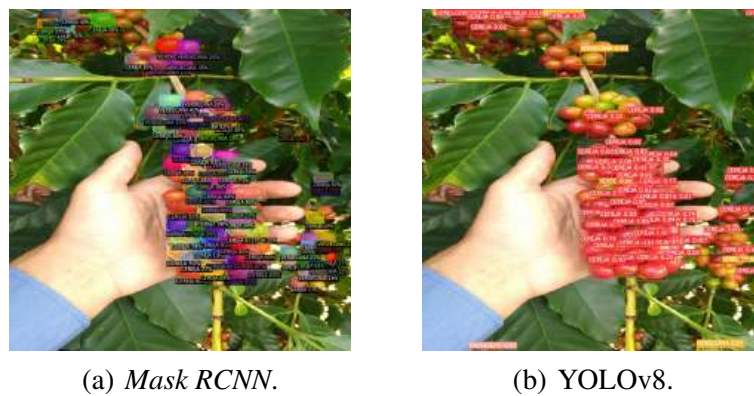
A Figura 4.12 apresenta a última imagem de teste utilizada. Esta imagem apresentou um bom equilíbrio entre as classes verde, verde-cana e cereja. Neste contexto o *Mask* realizou uma considerável sobrecontagem de frutos verde-cana, evidenciando a contagem de frutos em segundo

Figura 4.10 – Imagem 5 rotulada pelos modelos propostos: (a) *Mask RCNN* e (b) YOLOv8.



Fonte: do Autor.

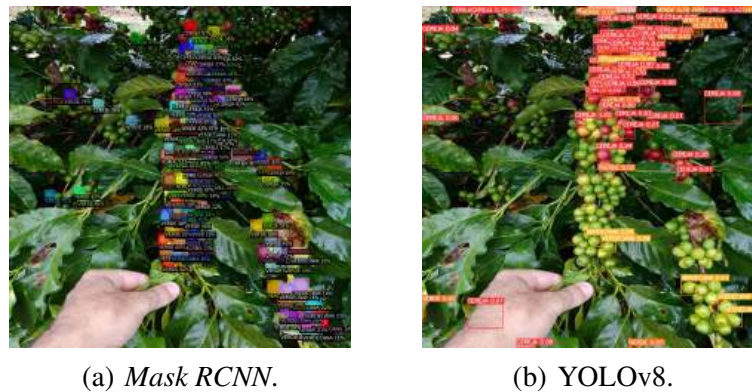
Figura 4.11 – Imagem 6 rotulada pelos modelos propostos: (a) *Mask RCNN* e (b) YOLOv8.



Fonte: do Autor.

plano e também uma confusão entre frutos verde e verde-cana, enquanto apresentou excelentes resultados para as demais classes. O YOLO por sua vez, e como pode ser visualizado na Figura 4.12 deixou de identificar uma grande quantidade de frutos verdes, enquanto para as demais classes também obteve uma boa performance.

Figura 4.12 – Imagem 7 rotulada pelos modelos propostos: (a) *Mask RCNN* e (b) YOLOv8.



Fonte: do Autor.

Por fim, visando utilizar os valores de APE, para avaliar qual o melhor modelo, tem-se para o *Mask RCNN* em questão de APE médio por classe de $105,81 \pm 90,54\%$ para classe verde, $69,33 \pm 104,67\%$ para classe verde-cana, $12,35 \pm 9,11\%$ para classe cereja e $43,33 \pm 67,59\%$ para a classe passa/seco. Para o modelo de YOLOv8 tem-se um APE médio por classe de $124,19 \pm 92,16\%$ para classe verde, $78,57 \pm 92,82\%$ para a classe verde-cana, $15,99 \pm 10,82\%$ para a classe cereja e $66,90 \pm 41,70\%$ para a classe passa/seco. O *Mask RCNN* obteve melhor desempenho em 3 das 4 classes. Ambos modelos, conforme pode ser observado nas figuras, apresentaram alguns problemas com relação a questões de fundo, principalmente a classe Verde, que em alguns momentos foi confundida com partes das folhas da própria planta. O desbalanceamento entre classes também deve ser levado em consideração, pois classes com menos instâncias rotuladas, apresentaram resultados menos satisfatórios.

Dentre as classes que os modelos podem ter dificuldade para realizar a distinção, tem-se: (i) verde e verde-cana (verdes), e (ii) cereja e passa/seco (maduros). Para melhor observar se essa dificuldade de fato ocorreu e se houve um número considerável de instâncias que foram confundidas, uma nova tabela (Tabela 4.10) foi construída, onde as classes foram observadas em pares: verde

e verde-cana foram unificadas, assim como cereja e passa/seco. A Figura 4.13 apresenta de forma gráfica os erros calculados para cada par de classes, segundo cada modelo.

Tabela 4.10 – Análise da identificação do grau de maturação por pares de classes.

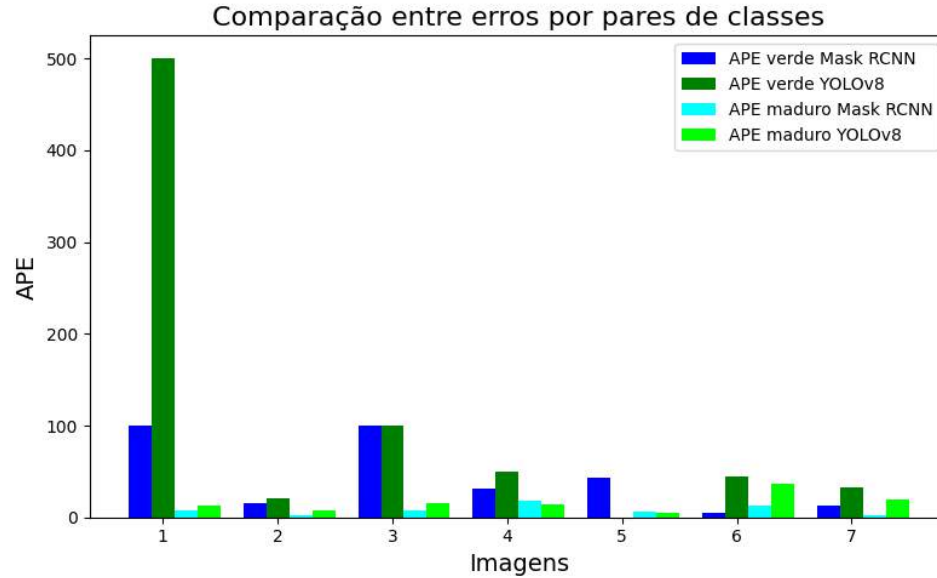
Imagem	Estádio	Especialista		<i>Mask RCNN</i>		YOLO		APE (%)	
		Valor	%	Valor	%	Valor	%	Mask	YOLO
1	Verde	0	0	1	1,15	5	5,81	100	500,0
	Maduro	93	100,0	86	98,85	81	84,19	7,53	12,90
2	Verde	57	45,24	48	40,34	45	41,28	15,79	21,05
	Maduro	69	54,76	71	59,66	64	58,72	2,90	7,25
3	Verde	0	0	1	1,33	1	1,70	100,0	100,0
	Maduro	69	100	74	98,77	58	98,30	7,25	15,94
4	Verde	45	48,39	37	37,76	30	46,88	17,78	33,33
	Maduro	48	51,61	61	62,24	34	53,12	27,08	29,17
5	Verde	7	13,21	10	16,95	7	13,73	42,86	0
	Maduro	46	86,79	49	83,05	44	86,27	6,52	4,35
6	Verde	20	29,85	21	28,38	11	14,67	5,00	45,00
	Maduro	47	70,15	53	71,62	64	85,33	12,77	36,17
7	Verde	96	69,57	108	71,52	65	56,52	12,50	32,29
	Maduro	42	30,43	43	28,42	50	43,48	2,38	19,04

Fonte: do Autor.

Com base nos dados apresentados na Tabela 4.10, tem-se para o *Mask RCNN*, em questão de erro médio por pares de classes, APE de $41,99 \pm 38,25\%$ para a classe verde e $9,49 \pm 7,85\%$ para a classe maduro. Para o modelo de YOLOv8 tem-se um erro médio por classe de $104,52 \pm 163,94\%$ para a classe verde e $17,83 \pm 10,61\%$ para a classe maduro. Pode-se observar uma diminuição considerável nos valores de APE, reforçando as colocações já feitas de que os modelos encontraram dificuldades para distinção das classes que foram agrupadas em pares nesta diferente forma de se avaliar os resultados.

Realizando uma análise geral com relação a classificação de instâncias, fica evidente que o modelo implementado com a arquitetura *Mask RCNN* apresentou melhor desempenho, o que vai de encontro às vantagens deste tipo de modelo, que envolvem um processamento mais prolongado

Figura 4.13 – Gráfico de erros por pares de classes.



Fonte: do Autor.

da imagem, no qual a imagem passa duas vezes por uma rede convolucional até ter as intâncias segmentadas, conforme discutido na Seção 2.6.

4.3 Comparações com outros trabalhos

Apesar de a abordagem de divisão de classes presente neste trabalho ser uma novidade na literatura, é possível através da junção de determinadas classes (ao juntar duas classes, a métrica referente será obtida através da média das métricas das classes unificadas), realizar algumas comparações com trabalhos apresentados por outros autores nesta mesma área de maturação de café e visão computacional. Como o modelo de *Mask RCNN* obteve os melhores resultados, suas melhores métricas (*fold 4*) foram tomadas como base para realização de comparações. Dentre os trabalhos presentes na literatura, pode-se realizar, através de adaptações, comparações com os trabalhos de Bazame e Molin (2021) e Ramos et al. (2017).

Bazame e Molin (2021) consideraram três classes: *unripe*, *ripe* e *overripe*. Para realização de uma comparação, a seguinte relação com as classes deste trabalho será considerada:

- *Unripe*: Verde e Verde-cana,

- *Ripe*: Cereja
- *Overripe*: Passa/Seco

Tabela 4.11 – Comparação entre APs.

Estádio	Autor	Bazame e Molin (2021)
<i>Unripe</i>	0,720	0.798
<i>Ripe</i>	0,729	0,840
<i>Overripe</i>	0,415	0,799

Fonte: do Autor

Ramos et al. (2017) consideraram 4 classes: *Unripe*, *Semi-ripe-1*, *Semi-ripe-2* e *Ripe and overripe*. Para realização de uma comparação, a seguinte relação com as classes deste trabalho será considerada:

- *Unripe*: Verde,
- *Semi-ripe-1*: Verde-cana,
- *Semi-ripe-2* e *Ripe and overripe*: Cereja e Passa/Seco

Tabela 4.12 – Comparação entre APs.

Estádio	Autor	Ramos et al. (2017)
<i>Unripe</i>	0,700	0.864
<i>Semi-ripe-1</i>	0,7409	1,00
<i>Semi-ripe-2</i> e <i>Ripe and overripe</i>	0,572	0,810

Fonte: do Autor

Observando as Tabelas 4.11 e 4.12, fica evidente a maior discrepância com relação a identificação das classes que envolveram a classe Passa/Seco, pois o método proposto neste trabalho apresentou um alto índice de falsos positivos. Com relação as demais classes, os resultados apresentados por outros autores foram superiores, porém em menores magnitudes. Como já discutido, o padrão das imagens utilizadas pode contribuir para os resultados de cada arquitetura. As Figuras 4.14 e 4.15 apresentam o padrão de fotos utilizadas pelos autores incluídos na comparação. Ambos os autores estabeleceram um POP mais robusto, padronizaram suas imagens e no caso de Ramos et al. (2017), as imagens utilizadas foram mais simples, envolvendo os frutos mais centralizados e maiores dentro da imagem, o que facilita o trabalho dos métodos aplicados.

Figura 4.14 – Padrão de imagens Bazame e Molin (2021).



Fonte: adaptado de Bazame e Molin (2021).

Figura 4.15 – Padrão de imagens Ramos et al. (2017).



Fonte: adaptado de Ramos et al. (2017).

4.4 Implementação do Sistema

O sistema *web* proposto, que utiliza a arquitetura de rede cliente servidor foi desenvolvido, de forma a contemplar todas as funcionalidades propostas. O sistema é apresentado a seguir, seguinte a divisão de servidor e cliente.

4.4.1 Servidor

Foi desenvolvido um servidor *web* para realização da tarefa de processamento de imagens. Este servidor possui rotas que podem ser acessadas via requisições HTTPS, que possuem as funções de receber imagens, retornar os resultados da classificação e gerar relatórios com base em datas. A Figura 4.16 mostra a saída do *prompt* do Windows ao rodar o servidor, onde pode-se observar seu endereço de funcionamento (endereço IP, juntamente a uma porta). A arquitetura utilizada no servidor para processamento de imagens foi o *Mask RCNN*, pois como já discutido anteriormente, foi o modelo que apresentou o melhor desempenho. Se trata de um modelo mais lento, porém o tempo de processamento não será um fator crítico no sistema implementado.

Figura 4.16 – *Logs* iniciais referentes ao funcionamento online do servidor.

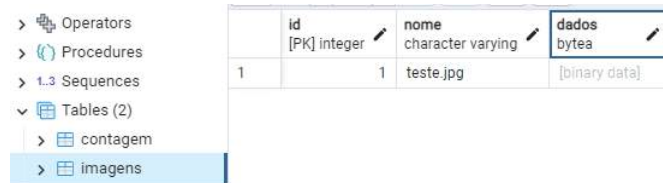
```
(detectron2) C:\Users\development\Desktop\api>python main.py
* Serving Flask app 'src.server.instance'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://10.0.0.111:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 657-959-822
```

Fonte: do Autor.

Além da funcionalidade de processamento de imagens, também foi implementada a funcionalidade de persistência de dados, realizado através do armazenamento das informações processados em um banco de dados. Neste banco foram criadas duas tabelas, sendo uma para armazenamento das imagens em formato binário e outra para armazenamento dos resultados de contagem, visando alimentar a funcionalidade de relatório da aplicação *mobile*.

A Figura 4.17 mostra a estrutura da tabela que deve armazenar imagens. Esta tabela possui uma coluna onde o nome da imagem é gravado e uma segunda coluna onde a imagem é gravada de forma binária.

Figura 4.17 – Tabela para armazenamento de imagens.

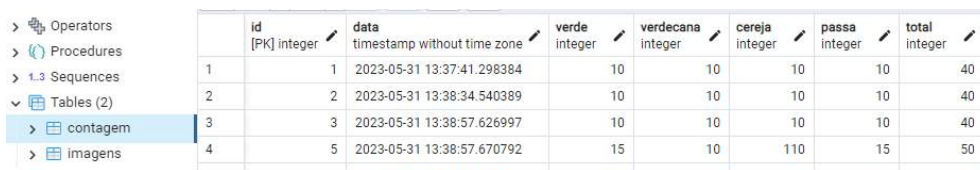


	id [PK] integer	nome character varying	dados bytea
1	1	teste.jpg	[binary data]

Fonte: do Autor.

A Figura 4.18 apresenta a estrutura da tabela responsável por armazenar os resultados de contagem por imagem. Esta tabela apresenta uma coluna para cada estágio de maturação identificado na imagem, e também uma coluna totalizadora. A coluna “data” é um dos pontos chaves desta tabela, pois através dela se faz possível a geração de relatórios na aplicação *mobile*, onde datas de referência para a pesquisa podem ser escolhidas, e assim o usuário pode acompanhar a evolução da lavoura ao longo do tempo.

Figura 4.18 – Tabela para armazenamento de contagens.



	id [PK] integer	data timestamp without time zone	verde integer	verdecana integer	cereja integer	passa integer	total integer
1	1	2023-05-31 13:37:41.298384	10	10	10	10	40
2	2	2023-05-31 13:38:34.540389	10	10	10	10	40
3	3	2023-05-31 13:38:57.626997	10	10	10	10	40
4	5	2023-05-31 13:38:57.670792	15	10	110	15	50

Fonte: do Autor.

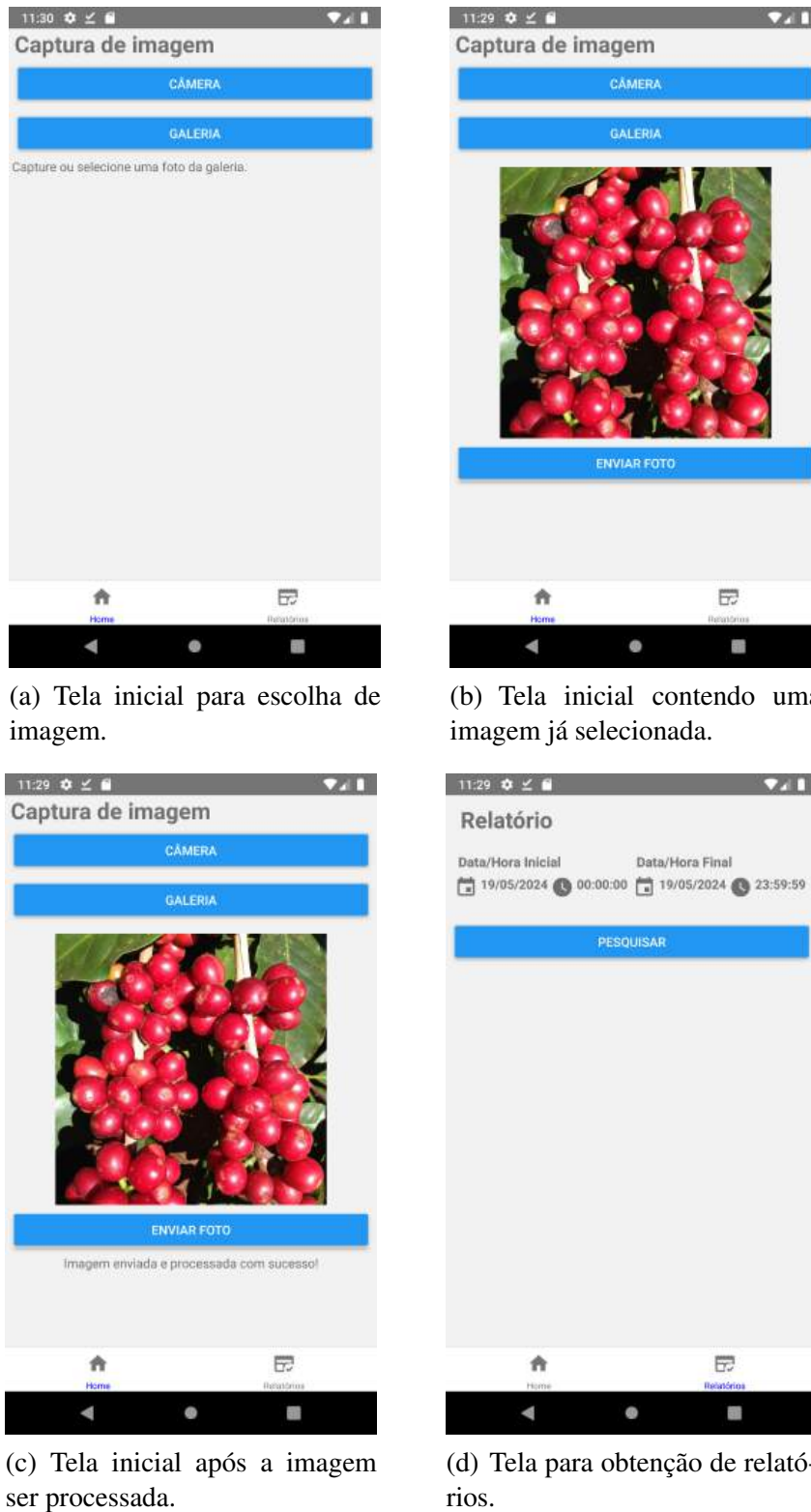
4.4.2 Aplicação Mobile

A aplicação *mobile* foi desenvolvida de forma a proporcionar as funcionalidades de: captura de imagem, busca de imagem na galeria, envio da imagem para o servidor, apresentação do resultado retornado pelo servidor e obtenção de relatórios referentes a contagens a partir de filtros de data inicial e final. O relatório retornado pelo aplicativo envolve um gráfico de pizza e uma tabela contendo as porcentagens de cada grau de maturação contido nas fotos englobadas pelo período de pesquisa.

A Figura 4.19 apresenta a primeira tela base da aplicação e suas variações, onde o usuário pode escolher a fonte de uma imagem e enviar para o servidor, visando a realização do processamento.

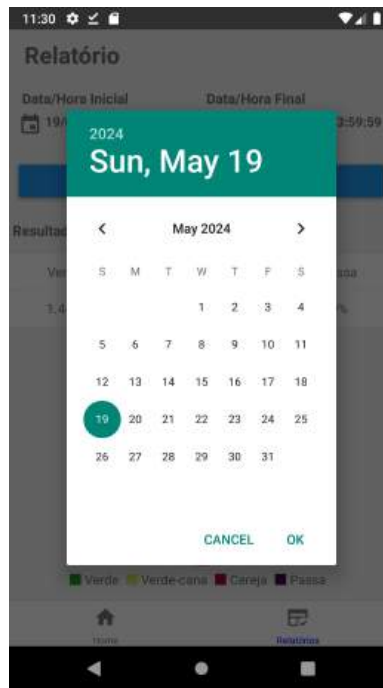
A Figura 4.20 apresenta a segunda tela da aplicação e suas variações, onde filtros de data inicial e final podem ser escolhidos para obtenção de relatórios referentes a contagem, gerados a partir das informações armazenadas no banco de dados do servidor. A aplicação fornecerá dados históricos através de suas consultas.

Figura 4.19 – Telas presentes no aplicativo (Parte 1).

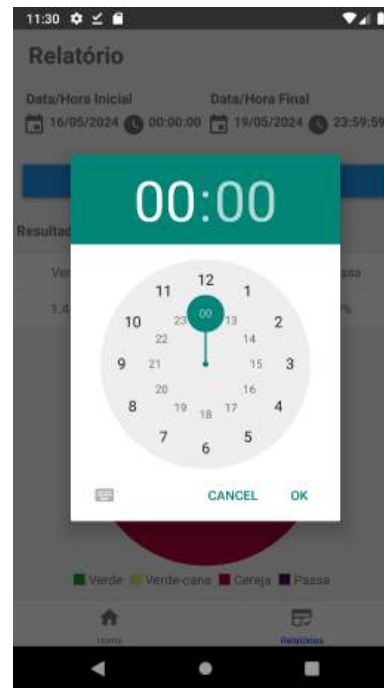


Fonte: do Autor.

Figura 4.20 – Telas presentes no aplicativo (Parte 2).



(a) Possibilidade da escolha de data para os filtros.



(b) Possibilidade da escolha de horas e minutos para os filtros.



(c) Relatório apresentado (exemplo 1).



(d) Relatório apresentado (exemplo 2).

Fonte: do Autor.

5 CONCLUSÃO

A conclusão está dividida em duas partes: (i) modelos de *Deep Learning* e seus respectivos desempenhos e melhorias, e (ii) o sistema cliente-servidor desenvolvido e suas possíveis melhorias e novas funcionalidades a serem exploradas futuramente. Com relação aos modelos, desempenhos satisfatórios foram obtidos no quesito de contagem de instâncias, com um APE médio de 9.61% para o *Mask RCNN* e 13.31% para o *YOLOv8*. Já no contexto de classificação de instâncias, os valores de APE foram consideravelmente mais altos, porém deve-se levar em consideração o fato de que nas imagens de teste, um erro de um fruto poderia gerar um APE de 100% no caso de falso positivo, se mostrando uma métrica bastante contundente para avaliação. Pode-se observar que os maiores valores de APE ocorreram para as classes verde e verde-cana, então uma ampliação da base de dados com foco em tais classes seria um ponto crucial a ser explorado (balanceamento da base de dados), para que os modelos aprendam de forma mais consistente a diferenciar tais classes (em algumas imagens folhas foram classificadas como frutos verdes). Sendo importante evidenciar que o banco de dados utilizado, assim como as imagens de teste, buscou dar maior relevância em sua composição, para imagens que retratem a situação real de campo, o que torna sua ampliação consideravelmente mais difícil, pois para cada foto, um especialista deve ir a campo, fazer o registro, e também uma avaliação onde ele contará e classificará os frutos segundo seu conhecimento e experiência. Em futuros trabalhos, considerando que o sistema atual foi desenvolvido de forma a armazenar em um banco de dados todas as imagens capturadas, modelos do tipo *Life Long Learning* (FLEMING, 2021) poderiam ser implementados para substituição dos modelos atuais, o que tornaria aprendizado contínuo e adaptativo, fazendo com que o sistema como um todo se tornasse mais robusto e preciso.

Um POP mais consistente também deve ser estabelecido, com vistas a alcançar uma maior padronização entre todas as imagens capturadas para treinamento e aperfeiçoamento dos modelos, assim como também para uso prático do sistema, servindo como referência para que futuros usuários possam usar a aplicação de forma mais eficaz. As comparações realizadas com trabalhos de outros autores evidenciaram que o uso de um POP deverá contribuir para obtenção de melhores resultados.

Com relação ao sistema cliente-servidor, tudo funcionou como o esperado e proposto, sendo este sistema capaz de capturar imagens, realizar seu processamento, armazenar os resultados e disponibiliza-los para acesso. Como futuras melhorias do sistema, podem ser incluídas novas telas na aplicação *mobile*, como uma tela para *login* de usuários, telas contendo instruções para melhor uso do sistema, tela para anotações e funcionalidades que sejam capazes de permitir a associação de áreas físicas de uma lavoura com a foto capturada e processada, o que resultaria em um novo filtro na tela de relatórios, no qual o usuário poderia realizar pesquisas levando em consideração áreas específicas de sua lavoura. Além dos frutos de café, novas funcionalidades de identificação, tais como a capacidade de analisar pragas e doenças a partir de folhas também poderiam ser incluídas no sistema, bastando para isso, o treinamento e adição de mais um modelo de *Deep Learning* treinado para tal tarefa, pois a forma como o sistema foi desenvolvido tornou fácil a inclusão de novas *features*.

E por fim, ainda tendo em vistas trabalhos futuros, uma implementação de um sistema que funcione *offline* também seria interessante, já que o sistema atual depende de uma conexão de internet entre cliente e servidor. Dessa forma o processamento seria realizado diretamente na aplicação *mobile*, dispensando a necessidade de conexão via internet, mas também não a removendo, pois o armazenamento de dados históricos no servidor é indispensável quando uma persistência de dados robusta está em questão. Desta forma os dados seriam sincronizados sempre que o dispositivo *mobile* obtivesse acesso a internet.

REFERÊNCIAS

- BAZAME, H. C.; MOLIN, J. P. **Quantification and classification of coffee fruits with computer vision**. Tese (Doutorado) — University of São Paulo, 2021. Disponível em: <<https://www.teses.usp.br/teses/disponiveis/11/11152/tde-14122021-160103/pt-br.php>>.
- BAZAME, H. C. et al. Detection, classification, and mapping of coffee fruits during harvest with computer vision. **Computers and electronics in agriculture**, Elsevier B.V, Amsterdam, v. 183, p. 106066, 2021. ISSN 0168-1699.
- BISWAS, A. et al. **Generative Adversarial Networks for Data Augmentation**. 2023. Disponível em: <<https://arxiv.org/abs/2306.02019>>.
- BORÉM, F. M. **Tecnologia pós-colheita e qualidade de cafés especiais**. [S.l.]: Editora UFLA, 2023.
- CONAB. **Conab - Análise do Mercado de Café**. 2024. Disponível em: <https://www.conab.gov.br/info-agro/analises-do-mercado-agropecuário-e-extrativista/analises-do-mercado/historico-de-conjunturas-de-cafe/item/download/36498_ee45f2f81b7584209430f1e9ed3a7793>.
- COSTA, A. G. et al. Classification of robusta coffee fruits at different maturation stages using colorimetric characteristics. **Engenharia Agrícola**, Associação Brasileira de Engenharia Agrícola, v. 40, n. 4, p. 518–525, 2020. ISSN 0100-6916.
- ESGARIO, J. G. et al. An app to assist farmers in the identification of diseases and pests of coffee leaves using deep learning. **Information processing in agriculture**, Elsevier B.V, v. 9, n. 1, p. 38–47, 2022. ISSN 2214-3173.
- FAWAKHERJI, M. et al. Shape and style gan-based multispectral data augmentation for crop/weed segmentation in precision farming. **Crop Protection**, Elsevier BV, v. 184, p. 106848, out. 2024. ISSN 0261-2194. Disponível em: <<http://dx.doi.org/10.1016/j.cropro.2024.106848>>.
- Ferreira Júnior, L. d. G. et al. Characterization of the coffee fruit detachment force in crop subjected to mechanized harvesting. **Coffee Science - ISSN 1984-3909**, v. 13, n. 1, p. 71–79, May 2018. Disponível em: <<https://coffeescience.ufla.br/index.php/Coffeescience/article/view/1379>>.
- Ferreira Júnior, L. d. G. et al. Dynamic behavior of coffee tree branches during mechanical harvest. **Computers and electronics in agriculture**, Elsevier B.V, Amsterdam, v. 173, p. 105415, 2020. ISSN 0168-1699.
- FLEMING, T. Models of Lifelong Learning: An Overview. In: **The Oxford Handbook of Lifelong Learning**. Oxford University Press, 2021. ISBN 9780197506707. Disponível em: <<https://doi.org/10.1093/oxfordhnb/9780197506707.013.3>>.
- FREITAS, V. V. et al. Coffee: A comprehensive overview of origin, market, and the quality process. **Trends in Food Science Technology**, v. 146, p. 104411, 2024. ISSN 0924-2244. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0924224424000876>>.

HARRINGTON, J. L. Chapter 1 - the database environment. In: HARRINGTON, J. L. (Ed.). **Relational Database Design and Implementation (Fourth Edition)**. Fourth edition. Boston: Morgan Kaufmann, 2016. p. 3–26. ISBN 978-0-12-804399-8. Disponível em: <<https://www.sciencedirect.com/science/article/pii/B9780128043998000016>>.

HE, K. et al. Mask r-cnn. In: **2017 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017. p. 2980–2988.

HUI, J. **Image segmentation with Mask R-CNN**. 2018. <https://jonathan-hui.medium.com/image-segmentation-with-mask-r-cnn-eb6d793272>.

HUSSAIN, M. Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. **Machines**, v. 11, n. 7, 2023. ISSN 2075-1702. Disponível em: <<https://www.mdpi.com/2075-1702/11/7/677>>.

JOCHER, G.; CHAURASIA, A.; QIU, J. **YOLO by Ultralytics**. 2023. Url<https://github.com/ultralytics/ultralytics>.

KLOMPENBURG, T. van; KASSAHUN, A.; CATAL, C. Crop yield prediction using machine learning: A systematic literature review. **Computers and Electronics in Agriculture**, v. 177, p. 105709, 2020. ISSN 0168-1699. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0168169920302301>>.

KORA, S.; RAVULA, S. Evaluation of deep convolutional generative adversarial networks for data augmentation of chest x-ray images. **Future Internet**, v. 13, p. 8, 12 2020.

KUROSE, J.; ROSS, K. **Redes de computadores e a internet: uma abordagem top-down**. Pearson, 2010. ISBN 9788588639973. Disponível em: <<https://books.google.com.br/books?id=raZtQwAACAAJ>>.

LAMBA. **Understanding Semantic Segmentation with UNET**. 2019. Url<https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>.

MARIN, D. B. et al. Unmanned aerial vehicle to evaluate frost damage in coffee plants. **Precision Agriculture**, p. 1–16, 2021.

MDS, M. do Desenvolvimento do B. **Pacto pelo trabalho decente na produção de café no Brasil ganha novas adesões**. 2024. Url<https://www.gov.br/mds/pt-br/noticias-e-conteudos/desenvolvimento-social/noticias-desenvolvimento-social/pacto-pelo-trabalho-decente-na-producao-de-cafe-no-brasil-ganha-novas-adesoes>.

MONTGOMERY, D. C. **Design and Analysis of Experiments**. 10th. ed. [S.l.]: John Wiley & Sons, 2020. ISBN 978-1119492443.

MORAES, H. M. Escala fenológica detalhada da fase reprodutiva de coffea arabica. **Bragantia**, v. 67, n. 1, p. 257–260, 2008.

MUMUNI, F. M. A. Data augmentation: A comprehensive survey of modern approaches. v. 16, p. 100258–100258, 2022.

OLIVEIRA, E. M. de et al. A computer vision system for coffee beans classification based on computational intelligence techniques. **Journal of Food Engineering**, v. 171, p. 22–27, 2016. ISSN 0260-8774. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0260877415300108>>.

PEZZOPANE, J. R. M. Escala para avaliação de estádios fenológicos do cafeeiro arábica. **Bragantia**, v. 62, n. 3, p. 499–505, 2008.

RAMOS, P. et al. Automatic fruit count on coffee branches using computer vision. **Computers and electronics in agriculture**, Elsevier B.V, Amsterdam, v. 137, p. 9–22, 2017. ISSN 0168-1699.

REDMON, J. et al. You only look once: Unified, real-time object detection. **arXiv.org**, Cornell University Library, arXiv.org, Ithaca, 2016. ISSN 2331-8422.

REDMON, J.; FARHADI, A. **YOLOv3: An Incremental Improvement**. 2018. Disponível em: <<https://arxiv.org/abs/1804.02767>>.

RODRÍGUEZ, J. P. et al. A computer vision system for automatic cherry beans detection on coffee trees. **Pattern recognition letters**, Elsevier B.V, Amsterdam, v. 136, p. 142–153, 2020. ISSN 0167-8655.

SAHA. **A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way**. 2018. Url<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

SICRE, R.; JURIE, F. Discriminative part model for visual recognition. **Computer vision and image understanding**, Elsevier Inc, SAN DIEGO, v. 141, p. 28–37, 2015. ISSN 1077-3142.

TANENBAUM, A. S. **Redes de Computadores**. [S.l.]: Pearson, 2011.

TASSIS, L. M.; SOUZA, J. E. Tozzi de; 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**, Elsevier B.V, Amsterdam, v. 186, p. 106191, 2021. ISSN 0168-1699.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. **YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors**. 2022. Disponível em: <<https://arxiv.org/abs/2207.02696>>.

WANG, G. et al. Interactive medical image segmentation using deep learning with image-specific fine tuning. **IEEE transactions on medical imaging**, IEEE, United States, v. 37, n. 7, p. 1562–1573, 2018. ISSN 0278-0062.

WU, Y. et al. **Detectron2**. 2019. <<https://github.com/facebookresearch/detectron2>>.

YANG, Y. et al. CycleGAN-based data augmentation for subgrade disease detection in gpr images with yolov5. **Electronics**, v. 13, n. 5, 2024. ISSN 2079-9292. Disponível em: <<https://www.mdpi.com/2079-9292/13/5/830>>.

ZHAO, X. et al. Use of unmanned aerial vehicle imagery and deep learning unet to extract rice lodging. **Sensors (Basel, Switzerland)**, MDPI AG, Switzerland, v. 19, n. 18, p. 3859, 2019. ISSN 1424-8220.

ZHOU, Y. Transfer learning and its application research. v. 1920, n. 1, p. 012058–012058, 2021.