UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

WILLIANA LUZIA SOUSA LEITE

ANÁLISE DE VIABILIDADE DO USO DE APRENDIZAGEM PROFUNDA PARA DETECÇÃO DE FRUTOS DE ACEROLA EM IMAGENS

RUSSAS

2020

WILLIANA LUZIA SOUSA LEITE

ANÁLISE DE VIABILIDADE DO USO DE APRENDIZAGEM PROFUNDA PARA DETECÇÃO DE FRUTOS DE ACEROLA EM IMAGENS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Prof. Ms. Tatiane Fernandes Figueiredo

RUSSAS

2020

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L556a Leite, Williana Luzia Sousa.
Análise de viabilidade do uso de aprendizagem profunda para detecção de frutos de acerola em imagens / Williana Luzia Sousa Leite. – 2020.
52 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2020.
Orientação: Prof. Me. Tatiane Fernandes Figueiredo.

1. Detecção de Frutos em Imagem. 2. Faster Region-based Convolutional Neural Network (Faster-RCNN). 3. Detecção de Objetos. 4. Aprendizado Profundo. 5. Acerola. I. Título.

CDD 005.1

WILLIANA LUZIA SOUSA LEITE

ANÁLISE DE VIABILIDADE DO USO DE APRENDIZAGEM PROFUNDA PARA DETECÇÃO DE FRUTOS DE ACEROLA EM IMAGENS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em:

BANCA EXAMINADORA

Prof. Ms. Tatiane Fernandes Figueiredo (Orientadora)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bonfim Amaro Júnior
Universidade Federal do Ceará - UFC

Prof. Dr. Rafael Fernandes Ivo
Universidade Federal do Ceará - UFC

RESUMO

O Ceará é o segundo maior produtor de acerolas do Brasil com 14,32% da produção brasileira do fruto, em grande parte, voltada para exportação. A crescente demanda por acerolas verdes tem contribuído para impulsionar a produção do fruto em alguns estados brasileiros. O conhecimento sobre a produção de frutos em cultivos e seus respectivos estágios de maturação podem orientar fruticultores no gerenciamento de recursos de mão-de-obra, como também apoiar decisões para planejamentos futuros. Para desenvolver um sistema automatizado de colheita robótica para frutos de acerola, a principal e mais custosa tarefa a ser desenvolvida é a detecção dos frutos. Neste trabalho será desenvolvido um módulo de detecção de frutos de acerola, envolvendo as etapas desde a aquisição da base de dados até a detecção da fruta em imagens. Para o nosso melhor conhecimento, não há estudos específicos sobre a detecção de frutos da acerola através de imagens digitais. Os principais desafios encontrados se encontram na aquisição e formação da base de dados, onde através dos experimentos computacionais, foi observado que a qualidade e quantidade de informações na base de dados causa grandes impactos no desempenho do modelo. Para desenvolver o módulo de detecção de frutos foi utilizado a arquitetura da rede *Faster Region-based Convolutional Neural Network* (Faster-RCNN). Para avaliar o modelo foi utilizado a métrica *mean Average Precisison* (mAP), onde obteve mAP de 89,5% com dados de treinamento e 92,2% nos dados de teste, em ambos os casos foi utilizado o valor de 0,5 para *Intersection Over Union* (IoU).

Palavras-chave: Detecção de Frutos em Imagem. *Faster Region-based Convolutional Neural Network (Faster-RCNN)*. Detecção de Objetos. Aprendizado Profundo. Acerola.

ABSTRACT

Ceará is the second-largest producer of acerolas in Brazil with 14.32% of the Brazilian production of the fruit, mostly, turned to export. The growing demand for green acerolas has contributed to boosting fruit production in some Brazilian states. Knowledge about fruit production in crops and their respective maturation stages can guide fruit growers in the management of labour resources, as well as support decisions for future planning. To develop an automated robotic harvesting system for acerola fruits, the main and most costly task to be developed is the detection of fruits. In this work, an acerola fruit detection module will be developed, involving the steps from the acquisition of the database to the detection of the fruit in images. To our best knowledge, there are no specific studies on the detection of acerola fruits through digital images. The main challenges encountered are in the acquisition and formation of the database, where through computational experiments, it was observed that the quality and quantity of information in the database cause significant impacts on the performance of the model. To develop the fruit detection module, the Faster-RCNN network architecture was used. To evaluate the model, the gls mAP metric was used, where it obtained 89.5% mAP with training data and 92.2% on the test data, in both cases the value was used from 0.5 to IoU.

Keywords: Image Detection of Fruits. Faster Region-based Convolutional Neural Network (Faster-RCNN). Object Detection. Deep Leaning. Acerola.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	*Average Precisison*
API	*Application Programming Interface*
AUC	Área Sob a Curva ou *Area under curve*
CNN	Rede Neural Convolucional ou *Convolutional Neural Network*
DCNN	Redes Neurais Convolucionais Profundas ou *Deep Convolutional Neural Networks*
Fast-RCNN	*Fast Region-based Convolutional Neural Network*
Faster-RCNN	*Faster Region-based Convolutional Neural Network*
IoU	*Intersection Over Union*
mAP	*mean Average Precisison*
Mask-RCNN	*Mask Region-based Convolutional Neural Network*
MLP	Perceptron de Múltiplas Camadas ou *Multilayer Perceptron*
RCNN	*Regions-based Convolutional Neural Networks*
RNA	Redes Neurais Artificiais
RoI	*Region of Interest*
RPN	*Region Proposal Networks*
SVM	Máquina de Vetores de Suporte ou *Support Vector Machine*
VGG	*Visual Geometry Group*
XML	*Extensible Markup Language*
ZF-Net	*Zeiler and Fergus network*

SUMÁRIO

1 INTRODUÇÃO

Segundo Calgaro e Braga (2012), o Brasil é considerado o maior produtor, consumidor e exportador de acerolas no mundo. Entre os principais estados produtores, o Ceará ocupa segundo lugar com 14,32% da produção brasileira do fruto, em grande parte, voltada para exportação. A comercialização e exportação de frutos verdes da acerola vem conquistando espaço no mercado há alguns anos devido a sua alta concentração de ácido ascórbico, popularmente conhecido como vitamina C. Esta crescente demanda tem contribuído para impulsionar a produção do fruto em alguns estados brasileiros (RURAL, 2014). No entanto, embora o crescimento da produção de acerola tenha aumentado, as tecnologias para otimização da colheita ainda são inacessíveis para boa parte dos agricultores do país (SEBRAE, 2017).

O conhecimento sobre a produção de frutos em cultivos e seus respectivos estágios de maturação podem orientar fruticultores no gerenciamento de recursos de mão-de-obra, como também apoiar decisões para planejamentos futuros (KOIRALA *et al.*, 2019). Dentre as formas de obtenção de informações, a detecção precisa de frutos individuais baseadas em processamento digital de imagens é um dos principais componentes em sistemas automatizados de colheita robótica. Com o conhecimento preciso das localizações individuais das frutas no campo, é possível realizar estimativas e mapeamentos de produção, gerando informações relevantes para os produtores que consequentemente conseguem gerenciar melhor suas atividades (BARGOTI; UNDERWOOD, 2017a).

Para desenvolver um sistema automatizado de colheita robótica para frutos de acerola, a principal e mais custosa tarefa a ser desenvolvida é a detecção dos frutos (BARGOTI; UNDERWOOD, 2017a). Para o nosso melhor conhecimento, não há estudos específicos sobre a detecção de frutos da acerola através de imagens digitais, e por conseguinte, não se encontra na literatura uma base de dados com as imagens do fruto de acerola e suas respectivas marcações. Desta forma, o presente trabalho tem como foco a análise do uso de técnicas de processamento digital de imagens para detecção dos frutos de acerola em imagens retiradas do campo. O estudo realizado tem como objetivo mensurar o desempenho do modelo inteligente criado, considerando uma detecção minimamente aceitável, a partir de uma base de dados de imagens gerada em um ambiente não controlado.

Com exceção da acerola, alguns outros frutos tem se tornado objetos de pesquisa na área de processamento digital de imagens, havendo na literatura diversas abordagens para detecção de frutos em imagens do campo, dentre estes trabalhos, pode-se citar Jana *et al.* (2017),

Awate *et al.* (2015), Bargoti e Underwood (2017b), Yamamoto *et al.* (2014), Bargoti e Underwood (2017a), Koirala *et al.* (2019) que apresentam inúmeras técnicas de pré-processamento, com o intuito de minimizar o impacto da variabilidade das imagens. Algumas destas técnicas são utilizadas neste trabalho, havendo grande influência nos resultados obtidos pelo modelo proposto. A metodologia definida para este trabalho é dividida em 2 etapas, sendo a primeira de pré-processamento dos dados, onde recortes são realizados na imagem. Na segunda etapa, cada recorte é analisado separadamente, ou seja, cada imagem recortada é enviada para modelo inteligente responsável por detectar os objetos.

A estrutura deste trabalho encontra-se da seguinte forma. No Capítulo 2 é apresentado o objetivo geral e os objetivos específicos deste trabalho; o Capítulo 3 expõe os tópicos chaves da pesquisa, enquanto o Capítulo 4 apresenta os trabalhos que se assemelham em algum aspecto com este trabalho. O Capítulo 5 descreve os procedimentos utilizados para realizar a pesquisa e assim como seu desenvolvimento. Por fim, nos Capítulos 6 e 7 são apresentados respectivamente, os resultados obtidos na pesquisa, e uma conclusão sobre o estudo e possíveis melhorias da pesquisa.

2 OBJETIVOS

2.1 Objetivo geral

Analisar a viabilidade do uso de aprendizagem profunda (*deep learning*) para detecção de frutos de acerola em imagens.

2.2 Objetivos específicos

- Gerar uma base de dados de imagens de aceroleiras em parceria com a fazenda Meri Pobo Agropecuária;
- Tratar a base de dados de imagens de frutos de acerola e de aceroleiras;
- Explorar as arquiteturas de redes neurais artificiais, mais usuais na literatura, com o objetivo de encontrar o modelo com melhor desempenho em relação ao problema;
- Analisar a viabilidade do modelo proposto, investigando os resultados obtidos em comparação com os resultados reais.

3 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordados os conceitos fundamentais para compreensão da solução proposta neste trabalho. Nas Seções 3.1, 3.2 e 3.3 há uma breve explanação das respectivamente áreas de conhecimento.

3.1 Processamento Digital de Imagens

A área de Processamento Digital de Imagens é definida por Backes e Junior (2019) como a área de estudo dedicada a repassar para máquina a capacidade de processamento visual humana, envolvendo habilidades como separar regiões ou objetos de interesse em uma cena, capturar objetos e aplicar transformações na imagem. Ballard e Brown (1982) a define como a ciência que estuda e desenvolve tecnologias permitindo que máquinas enxerguem e extraiam características do meio.

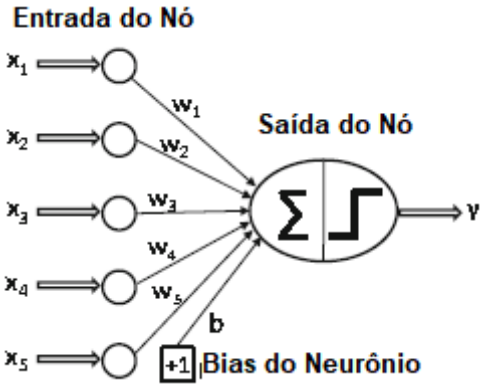
3.2 Aprendizagem de Máquina

Aprendizagem de Máquina ou *Machine Learning* é uma subárea da inteligência artificial e da ciência cognitiva que ao longo dos anos, com a evolução dos hardwares, vem ganhando notoriedade na literatura e em aplicações no mundo real (SKANSI, 2018). Em Aprendizagem de Máquina, os algoritmos buscam aprender e melhorar seu desempenho com a experiência, construindo modelos matemáticos baseados em dados amostrais (MITCHELL, 1997). Esta área se divide em 3 principais ramificações: aprendizado supervisionado, não supervisionado e aprendizagem por reforço. Neste trabalho, o modelo utilizado pertence a classe de aprendizado supervisionado. Nesta classe de algoritmos é necessário uma base de dados, onde para cada conjunto de dados que represente uma informação/recurso deve existir também um valor alvo que se deseja alcançar (SKIENA, 2017). Para criação de modelos através destes algoritmos, o conjunto de dados deve ser dividido, usualmente, em três conjuntos: treinamento, validação e teste. O conjunto de dados de treinamento são utilizados para a aprendizagem do modelo, enquanto o conjunto de validação e de teste são utilizados para avaliar o modelo. Normalmente, o conjunto de validação também é utilizado para o reajuste de parâmetros do modelo e o conjunto de teste é utilizado para avaliar o modelo final.

3.2.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA) são técnicas populares de aprendizagem de máquina que simulam o mecanismo de aprendizado dos organismos biológicos. Essas redes contêm unidades de computação chamadas neurônios, onde a partir de conexões com outros neurônios, geram algum aprendizado/conhecimento específico, seja reconhecimento de padrões, correlações em dados brutos, agrupamento de dados, classificações, dentre outras aplicações. O algoritmo *perceptron*, proposto por Rosenblatt (1961), é a representação mais simples de uma RNA, contendo uma única camada computacional, visto que a camada de entrada não é contabilizada por não haver nenhuma computação. Na Figura 1 há uma representação de sua arquitetura (AGGARWAL, 2018).

Figura 1 – Representação da arquitetura de um Perceptron com bias.

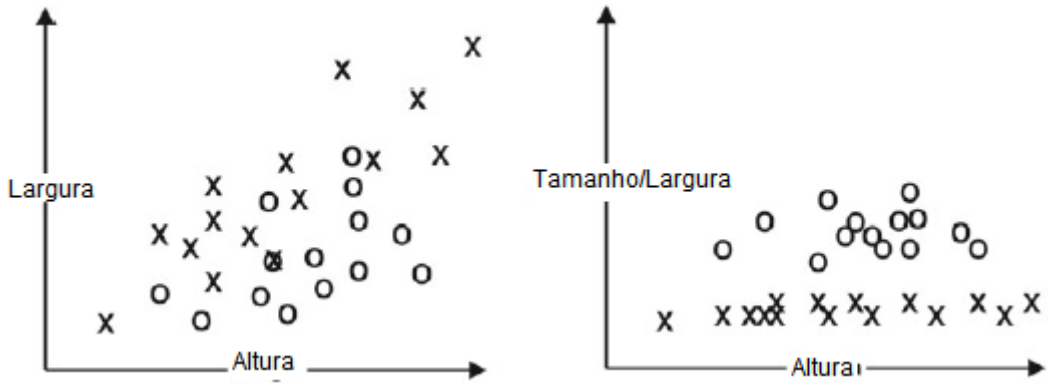


Fonte: Adaptado a partir de Aggarwal (2018).

O *perceptron*, por conter apenas uma camada, também pode ser denominado de neurônio. Um neurônio, como apresentado na Figura 1, é composto por entradas $x = \{x_1, ..., x_n\}$, pesos $w = \{w_1, ..., w_n\}$, *bias* b, função de ativação f e uma camada de saída y, o conhecimento da rede é representado pelos seus pesos que são ajustados de acordo com os erros encontrados.

O algoritmo *perceptron* por se tratar de um modelo simples só é capaz de convergir quando aplicado a dados linearmente separáveis, como por exemplo, as portas lógicas *AND* e *OR*. Para escapar dessas limitações o algoritmo Perceptron de Múltiplas Camadas ou *Multilayer Perceptron* (MLP) foi desenvolvido para conseguir lidar com dados não linearmente separáveis, como por exemplo, o famoso problema da porta lógica *XOR* (SKANSI, 2018). Na Figura 2, a esquerda, é apresentado um exemplo de dados não linearmente separáveis, ou seja, não existe um hiperplano que separe as classes *X* e *O* e a direita é apresentado um conjunto de dados linearmente separável, onde claramente as classes podem ser separadas por um hiperplano.

Figura 2 – Exemplo de dados respectivamente não linearmente e linearmente separáveis.



Fonte: Adaptado a partir de Skansi (2018).

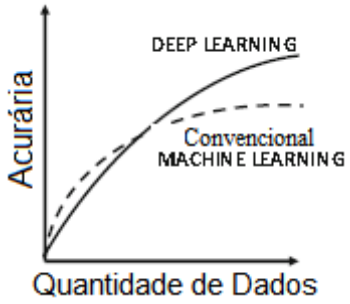
As redes MLP foram propostas por Rumelhart *et al.* (1985) e possuem uma camada de entrada, pelo menos uma camada intermediária, e uma camada de saída. (AGGARWAL, 2018). O aprendizado destas redes é realizado em fases, chamadas épocas, onde redes MLP do tipo *feedforward*, em que a informação se move em apenas uma direção, propagam os dados de entrada até a saída, calculando o erro e propagando-o da saída até a entrada. O cálculo do erro é feito sob uma função predefinida de erro ou *loss*. Todo o aprendizado dos pesos é feito automaticamente pelo algoritmo *backpropagation* que usa a programação dinâmica para atualizar os parâmetros da rede (AGGARWAL, 2018).

3.2.1.1 Deep Learning e as Redes Neurais Convolucionais

Aprendizagem Profunda ou *Deep Learning* é uma classe de aprendizagem de máquina que possui alto poder e flexibilidade ao representar dados com múltiplas dimensões (GOODFELLOW *et al.*, 2016). Em outras palavras, algoritmos que possuem arquitetura de *Deep Learning*, possuem pilhas de camadas em que progressivamente extraem diferentes níveis de características a cada camada. A existência de uma pilha de camadas, por outro lado, requer um conjunto de dados de treinamento maior, fator que influência diretamente no desempenho do modelo, como apresentado na Figura 3 (LESKOVEC *et al.*, 2020).

As Redes Neurais Convolucionais ou Convolutional Neural Network (CNN) foram um dos primeiros exemplos de sucesso na área de aprendizado profundo. Em LeCun *et al.* (1998), os autores produziram a primeira rede neural convolucional chamada LeNet-5, que obteve resultados significativos no conjunto de dados MNIST. Desde então, os êxitos atraentes das arquiteturas CNNs em concursos de classificação de imagens após o ano de 2011, trouxeram maiores atenções para a área (AGGARWAL, 2018). As CNNs, segundo Aggarwal (2018)

Figura 3 – Uma comparação ilustrativa da precisão de um algoritmo típico de aprendizado de máquina com um de *deep learning*.



Fonte: Adaptado a partir de Aggarwal (2018).

são redes que utilizam a operação convolucional em pelo menos uma de suas camada. Em contrapartida, Vargas *et al.* (2016) define as CNNs como uma variação das redes MLP, inspiradas no processo biológico de processamentos de dados visuais.

Segundo Araújo *et al.* (2017), uma das vantagens da utilização de uma CNN consiste na capacidade de extrair características relevantes, através do aprendizado de transformações na imagem. Outra vantagem está na dependência de menos parâmetros de ajustes em comparação com redes totalmente conectadas com o mesmo número de camadas ocultas. Vargas *et al.* (2016) cita como diferencial, uma CNN possibilitar a mistura de múltiplos mapas de características ao mesmo tempo, permitindo a extração de características cada vez mais complexas. Assim, as CNNs são capazes de sozinhas criarem filtros extremamente complexos aproveitando ao máximo as informações provenientes dos dados de treinamento.

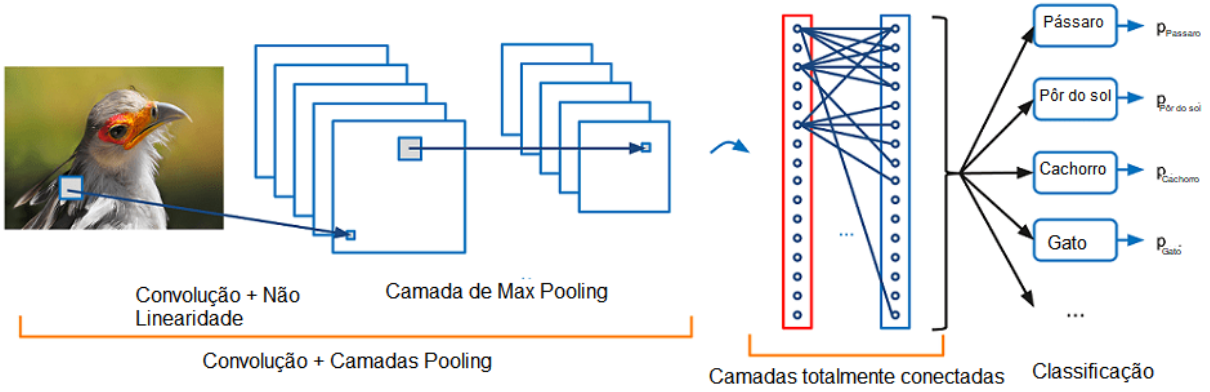
Uma CNN é composta de duas grandes etapas: a extração de características através das convoluções e a classificação. A arquitetura de uma típica CNN possui basicamente três tipos de camadas, as convolucionais, *pooling* e camadas totalmente conectadas. A camada de convolução contêm um conjunto de neurônios que são responsáveis por aplicar filtros em determinadas partes da imagem. Gonzalez e Woods (2009) define o processo de convolução da seguinte forma: dado uma imagem f e uma máscara w, a operação convolução matemática é definida a seguir como:

$$w(x,y) * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t).f(x-s,y-t).$$

Onde a e b são a metade da altura e largura da máscara w. De maneira simplificada, a convolução é o produto escalar entre a máscara e a vizinhança de cada pixel da imagem. A camada de *pooling*, tem como principal objetivo a redução de dimensionalidade dos dados na

rede, para tal, a operação mais utilizada é o *max-pooling*, onde as partes mais importantes da imagem são capturadas. A Figura 4 apresenta um exemplo genérico de CNN, onde estas etapas são denominadas *convolution + pooling layers*.

Figura 4 – Exemplo de CNN para classificação de espécies de animais, com camadas de convolução, *max pooling* e uma camada totalmente conectada. Sua saída é a classificação da imagem de entrada em alguma classe.



Fonte: Adaptado a partir de Academy (2019).

Na camada totalmente conectada, seu propósito é classificar a imagem de acordo com classes predeterminadas utilizando as características extraídas das camadas de convolução e de *pooling*, nesta camada todos os neurônios estão conectados entre si. Na Figura 4, há uma representação de uma CNN para classificação de imagens.

3.2.1.2 Aplicações das redes neurais convolucionais

A literatura apresenta uma infinidade de aplicações de redes neurais convolucionais nas mais diversas áreas de conhecimento. Neste trabalho será relevante para o entendimento das seções posteriores as aplicações de CNN como extratores de características. Uma CNN para extração de características consiste em uma rede sem as últimas camadas totalmente conectadas e portanto contém principalmente camadas de convolução. Redes com este propósito normalmente são utilizadas em conjunto com outros algoritmos para diferentes finalidades, como por exemplo, detecção de objetos (REN *et al.*, 2015a) e recuperação de imagem baseada em conteúdo (SHAH *et al.*, 2017). As aplicações de CNNs para classificação de imagens, ganhou notoriedade na literatura por conta dos bons resultados obtidos nas mais diversas competições (KAPUR, 2017). Devido a sua alta capacidade de aprendizado e extração de características, o método tem assumido o lugar de soluções usualmente conhecidas como, por exemplo, o algoritmo Máquina de Vetores de Suporte ou *Support Vector Machine* (SVM).

3.3 Detecção de Objetos em Imagens

Segundo Cyganek (2013), a detecção de objetos pode ser interpretada como a tarefa de detectar a presença ou ausência de um objeto específico em uma imagem. Caso a presença do objeto na imagem seja positiva, sua localização deve ser fornecida, assim como uma rotulação relacionada ao objeto. Esta última tarefa pode ser definida como um problema à parte de classificação. Em contrapartida, o autor Amit (2002) fornece uma definição mais genérica de que a detecção de objetos envolve as etapas desde identificar um local, até identificar e registrar componentes de uma classe de objeto específica em vários níveis de detalhes.

De acordo com Zhao *et al.* (2019) o problema da detecção de objetos em imagens pode ser modelado de duas formas com propostas baseadas em regiões ou baseadas em regressão. Na literatura existem diversos métodos eficientes que implementam estas duas abordagens mencionadas. Dentre eles, destaca-se o algoritmo do tipo baseado em região *Faster-RCNN* que será a abordagem de detecção de objetos utilizada para solucionar o problema proposto nesta pesquisa.

3.3.1 Redes neurais convolucionais baseada em regiões mais rápidas

A Rede neural convolucional baseada em regiões mais rápidas ou *Faster-RCNN*, é um *framework* de detecção de objetos baseada em Redes Neurais Convolucionais Profundas ou *Deep Convolutional Neural Networks* (DCNN), que faz parte da evolução da família de algoritmos para detecção de objetos: *Regions-based Convolutional Neural Networks* (RCNN). A primeira rede a ser apresentada desta família foi a RCNN, proposta por Girshick *et al.* (2014), que possui fundamentalmente 3 passos:

1. Execute uma pesquisa seletiva para gerar propostas de objetos;
2. Extraia as características de cada provável objeto utilizando uma CNN;
3. Envie as características extraídas para um SVM para classificar os objetos encontrados.

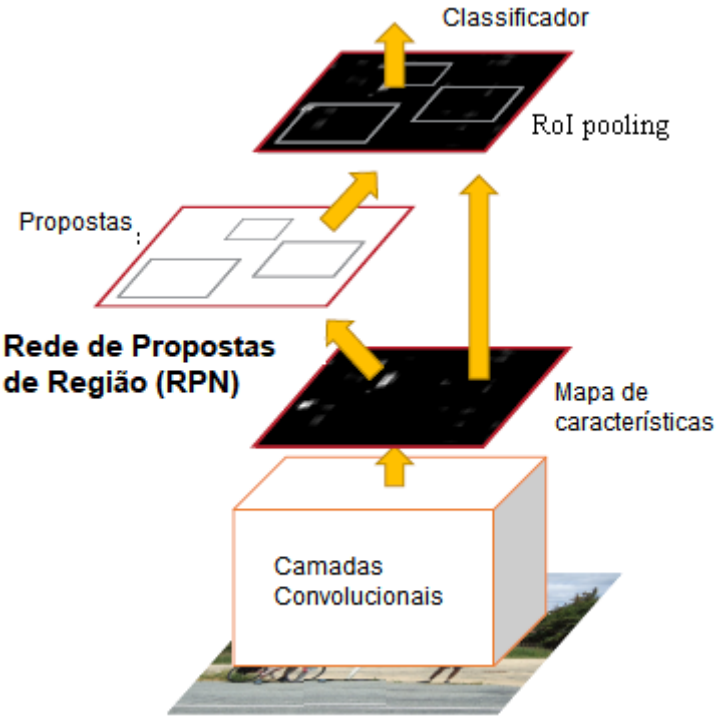
Ao longo da utilização da RCNN os autores Ren *et al.* (2015b) identificaram algumas desvantagens significativas na rede, a primeira delas encontra-se na utilização do método pesquisa seletiva, no qual é gerado aproximadamente 2.000 propostas de regiões, a segunda desvantagem é relacionada a extração de recurso, que é realizado por objeto identificado, tornando o processo altamente dispendioso pelo alto número de regiões propostas na pesquisa seletiva. Na tentativa

de solucionar as desvantagens apresentadas, Ren *et al.* (2015b) propôs como evolução o *Fast-RCNN*, onde algumas de suas principais mudanças tinham como foco a extração de recurso, que diferentemente da RCNN, era realizada apenas uma vez na imagem completa.

Com as mudanças, a *Fast-RCNN* se tornou muito mais eficiente que a RCNN, no entanto, como ainda utilizava o método de pesquisa seletiva para propor regiões continuava sendo um modelo consideravelmente lento. Após algumas evoluções dessa família de algoritmos, Ren *et al.* (2015a) propôs o modelo *Faster-RCNN* que substitui a pesquisa seletiva por uma RPN conseguindo uma alta eficiência nos testes de desempenho do modelo, e aproveitando as vantagens do compartilhamento de pesos entre a CNN para extração de recursos da RPN e a CNN para extração de recursos do detector *Fast-RCNN*.

Uma *Faster-RCNN* é dividida basicamente em 2 módulos, sendo o primeiro módulo composto por uma rede RPN e o segundo módulo composto pelo detector que utiliza as regiões propostas do módulo anterior para fazer as detecções. Na Figura 5 é apresentado um modelo geral de como a rede é estruturada.

Figura 5 – *Faster-RCNN* uma rede unificada para detecção de objetos.

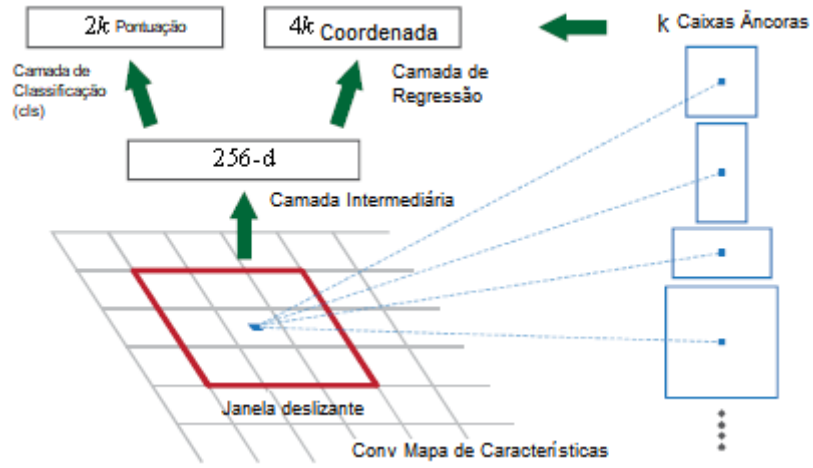


Fonte: Adaptado a partir de Ren *et al.* (2015a)

3.3.1.1 Region Proposal Networks

O primeiro módulo é responsável pela geração de propostas de regiões. Segundo Ren *et al.* (2015a), o RPN ao receber como entrada uma imagem de qualquer tamanho, sua saída deve ser um conjunto de propostas regiões. De acordo com os autores, a rede RPN funciona da seguinte forma: primeiro a imagem é enviada à uma CNN para extração de características, como resultado desta etapa um mapa de recursos é gerado. A partir do mapa de recursos o RPN aplica uma janela deslizante para cada ponto do mapa, onde para cada janela se propõe no máximo k objetos, para facilitar a compreensão. Na Figura 6 há uma representação de como a janela desliza sobre o mapa de recursos.

Figura 6 – Aplicação da janela deslizante em uma rede RPN.

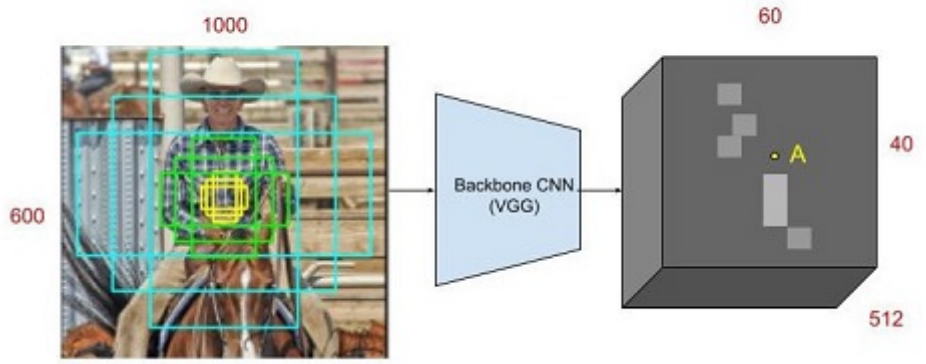


Fonte: Adaptado a partir de Ren *et al.* (2015a)

Para definir o tamanho das caixas delimitadoras, que são 4 coordenadas que formam os retângulos envolvendo um determinado objeto, o RPN utiliza o conceito de âncoras. As âncoras são caixas delimitadoras de referência, que por padrão possuem 3 escalas e 3 proporções, produzindo portanto $k = 9$ âncoras em cada posição de deslizamento. Na Figura 7 é apresentado 9 âncoras de 3 escalas e 3 proporções diferentes, geradas a partir do ponto A.

Dessa forma, por padrão, 9 objetos são propostos em cada janela, mas a rede ainda precisa verificar se em algumas destas propostas possuem realmente um objeto, como também ajustar a caixa delimitadora para o tamanho real do objeto. Para isso, cada janela deslizante é mapeada para um recurso de dimensão inferior (256-d para uma rede *Zeiler and Fergus network* (ZF-Net) e 512-d para uma rede VGG). Em seguida, este recurso de dimensão inferior é repassado para duas camadas totalmente conectadas: a camada para regressão das caixas

Figura 7 – Representação de 9 âncoras de 3 escalas e 3 proporções diferentes, geradas a partir do ponto A no mapa de recursos gerado pela rede desenvolvida e treinada pelo renomado VGG.



Fonte: Ananth (2019)

delimitadoras, representada na Figura 6 como *reg layer*, possuindo como saída 4k coordenadas, representando as coordenadas de cada caixa delimitadora, e a camada para classificação da caixa, representada na Figura 6 como *cls layer*, possuindo como saída 2k probabilidades, representando a probabilidade de um objeto estar ou não nas propostas.

Para calcular a função de perda ou *Loss*, que determina o erro entre a saída do algoritmo e o valor esperado, da RPN, a métrica IoU é utilizada, onde pode-se analisar o quão próxima da caixa delimitadora correta o modelo chegou. Dessa forma, o IoU é calculado como a relação entre a intersecção da caixa delimitadora prevista com a caixa delimitadora correta sobre a união das duas caixas delimitadoras, na Figura 8 há uma representação do cálculo.

Figura 8 – Cálculo da métrica IoU.

Fonte: Adaptado a partir de Rosebrock (2016)

Desta forma, o *Loss* da rede RPN é calculado utilizando um rótulo que assume 2 valores: positivo, quando a âncora(s) com maior(es) IoU se sobrepõem a uma caixa de

base ou quando a âncora(s) possuir/possuirem IoU > 0,7; ou não positivo, quando a âncora(s) possuir/possuirem IoU < 0,3. A função utilizada por Ren *et al.* (2015a) para calcular o *Loss* da rede *Faster-RCNN* é definida a seguir:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*).$$

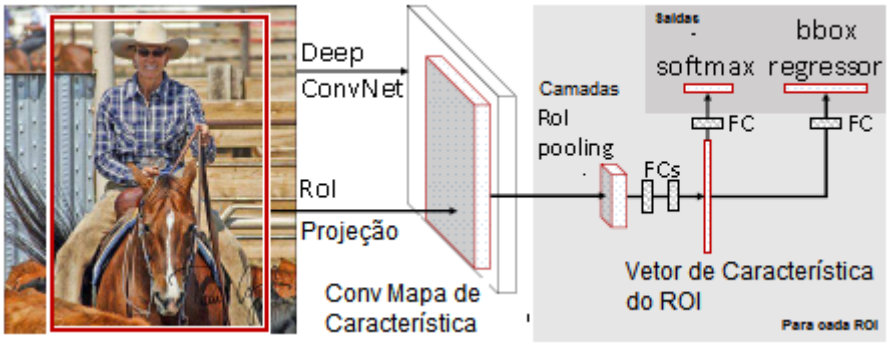
Onde, i é o índice de uma âncora em um mini lote e p_i é a probabilidade prevista de a âncora possuir um objeto. Desta forma, $p_i = 1$ se a âncora for positiva e 0 se a âncora for negativa. Sendo t_i um vetor que representa as 4 coordenadas: $[t_x, t_y, t_w, t_h]$ que parametrizam a caixa delimitadora prevista, sendo t_i^* o vetor que representa as coordenadas *ground-truth*. Estas coordenadas são responsáveis por refletir o resultado verdadeiro/correto que espera-se do modelo, associado a uma âncora positiva. Por fim, o termo L_{cls} corresponde ao log do *Loss* das duas classes: objeto e não-objetos, enquanto o termo $p_i^* L_{reg}$ é responsável por representar que o *Loss* da regressão só será calculado se a âncora for positiva, ou seja, se somente se $p_i^* = 1$. É importante mencionar, que as saídas das camadas de classificação *cls* e de regressão *reg*, nesta função, assumiram os valores de $\{p_i\}$ e $\{t_i\}$ respectivamente. Para normalizar estes dois termos são utilizados as variáveis: N_{cls} e N_{reg}, que correspondem aos valores padrões de tamanho do mini-lote e número de locais de ancoragem.

3.3.1.2 Detecção com Fast-RCNN

O módulo de detecção *Faster-RCNN* utiliza a rede *Fast-RCNN* para realizar a tarefa de detecção. De acordo com Ren *et al.* (2015b), o detector *Fast-RCNN* funciona da seguinte forma: dado um conjunto de propostas de regiões de uma determinada imagem, chamadas de *Region of Interest* (RoI), estas propostas são passadas para uma CNN, como apresentada na Seção 3.2.1.1, para extração de recursos/características da imagem. Os recursos extraídos são enviados para a camada de *RoI Pooling*, onde cada RoI é agrupada em um mapa de características de tamanho fixo. Esta etapa é essencial, pois as dimensões dos dados devem ser de tamanho fixo para se tornarem entradas adequadas para as próximas camadas. Após a aplicação da camada *RoI Pooling*, as informações são mapeadas para um vetor de recurso que servirá como entrada para duas camadas totalmente conectadas. Sendo a rede possuidora de dois vetores de saída por RoI: a primeira aplica a função *Softmax*, para obter a probabilidade de uma RoI pertencer a determinada classe e a segunda saída é responsável pela regressão das coordenadas das caixas

delimitadoras por classe. Na Figura 9 é apresentado a arquitetura de uma *Fast-RCNN*.

Figura 9 – Modelo da rede *Fast-RCNN*.



Fonte: Adaptado a partir de Ren *et al.* (2015a)

Como mencionado anteriormente, a rede *Fast-RCNN* possui duas saídas: a primeira trata-se da camada *cls* que possui uma distribuição de probabilidades discretas $p = (p_0, ..., p_K)$ por RoI, para cada $K+1$ categorias. Enquanto a segunda saída *reg* fornece os valores: $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$, sendo x e y valores centrais e w e h valores de largura e altura da caixa delimitadora. Segundo Ren *et al.* (2015b), a função a ser utilizada para o cálculo do *Loss* da rede é definida a seguir:

$$L(p,u,t^u,v) = L_{cls}(p,u) + \lambda[u \geq 1]L_{loc}(t^u,v)$$

Sendo u a representação da classe correta e v as coordenadas corretas do objeto, onde o termo $L_{cls}(p,u) = -logp_u$ se torna o log da classe correta u. Por fim, o terceiro termo L_{loc} é definido por uma tupla referente a caixa delimitadora correta para a classe u, onde $v = (v_x, v_y, v_w, v_h)$ é uma tupla que corresponde aos valores preditos $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ para a classe u.

3.3.1.3 União da RPN com a Fast-RCNN

Para unificar a rede através do compartilhamento de pesos das CNN de extração de recursos, os autores Ren *et al.* (2015a) realizaram diversos testes para encontrar o método que obtivesse melhor resultado. Por fim, a abordagem escolhida para as duas redes compartilharem pesos foi a abordagem treinamento alternado em quatro etapas.

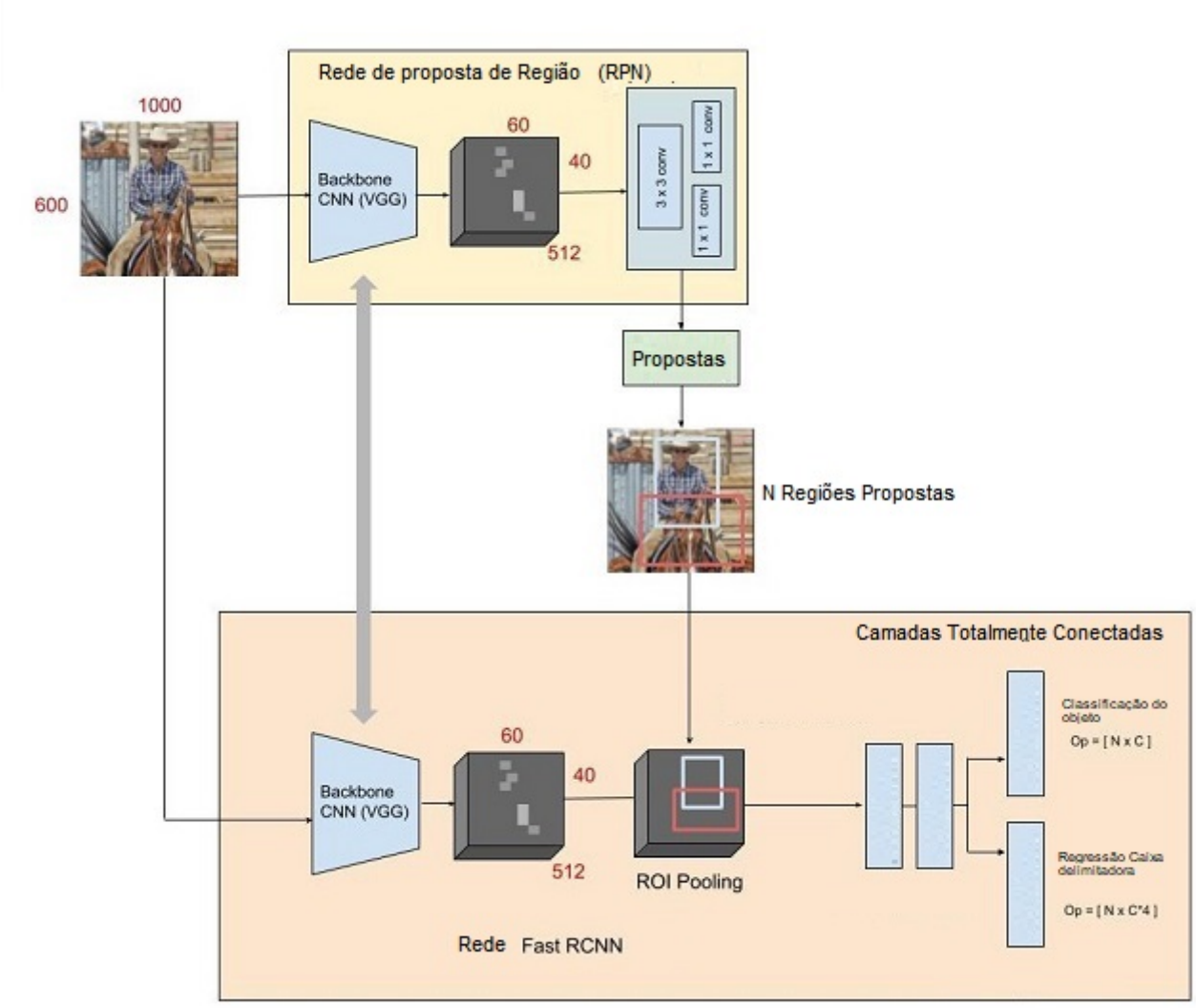
1. Na primeira etapa, a RPN é treinada independentemente, onde os pesos iniciais de sua

CNN para extração de características utiliza os pesos de um modelo pré-treinado pela ImageNet que em seguida, são ajustados para a tarefa de proposta da região;

2. Na segunda etapa, a rede de detecção *Fast-RCNN* é treinada usando as propostas geradas pela RPN da etapa 1. Essa rede de detecção também inicializa seus pesos usando o modelo pré-treinado pelo ImageNet. Nesse ponto, as duas redes não compartilham camadas convolucionais;
3. Na terceira etapa, o RPN é inicializado com pesos do *Fast-RCNN* e ajustado para a tarefa de proposta da região. Desta vez, os pesos das camadas compartilhadas são corrigidos e os pesos das camadas exclusivas do RPN são ajustados. Neste ponto, as duas redes compartilham camadas convolucionais;
4. Na ultima etapa, as camadas convolucionais compartilhadas permanecem fixas, e as camadas exclusivas do *Fast-RCNN* são ajustadas.

Através dessa abordagem de compartilhamento de pesos a rede *Faster-RCNN* se torna uma rede unificada para detecção de objetos. Na Figura 10 é apresentado com detalhes a arquitetura da rede *Faster-RCNN*.

Figura 10 – Arquitetura detalhada da *Faster-RCNN*, uma rede unificada para detecção de objetos.



Fonte: Adaptado a partir deAnanth (2019)

4 TRABALHOS RELACIONADOS

Este capítulo descreve os trabalhos da literatura mais relevantes para a contextualização do problema proposto nesta monografia. Na Seção 4.1 é apresentado o uso de um algoritmo de aprendizado profundo para resolver o problema da detecção precisa das frutas: manga, amêndoa e maçã. Na Seção 4.2 é mostrado o problema da detecção individual de tomates em plantas utilizando técnicas de processamento digital de imagens e aprendizado de máquina. Na Seção 4.3 e 4.4 é apresentado o problema da detecção de frutas e detecção de doenças em frutos a partir de imagens naturais, para tal, utiliza-se características de cor e textura. Por fim, a Seção 4.5 apresenta um comparativo entre as atividades realizadas no estado da arte e nesta monografia.

4.1 *Detecção profunda de frutas em pomares*

Segundo Bargoti e Underwood (2017a), a etapa de detecção dos frutos é imprescindível para a obtenção de bons resultados em aplicações que realizam a estimativa/mapeamento de produção e em tarefas de automação de colheitas na agricultura. Neste trabalho, os autores realizam a tarefa de detecção objetos para os seguintes frutos: manga, amêndoa e maçã. Para geração da base de dados as imagens foram retiradas da planta completa, onde os autores analisaram a quantidade de dados que são necessários para capturar a variabilidade de um conjunto de dados.

A base de dados de treinamento é composta por imagens do cultivo, onde cada imagem possui em torno de 100 à 1000 frutos. O primeiro desafio encontrado diz respeito ao tamanho das imagens e as limitações de hardware necessárias para processá-las. Para resolver o problema da alta dimensionalidade dos dados, utilizou-se uma técnica de detecção baseada em *patches*, apresentada em maiores detalhes na Seção 5.1.2. Os autores utilizaram o *framework Faster-RCNN*, fazendo um comparativo entre as redes VGG16 e ZF-Net para analisar qual apresentava o melhor desempenho. Para cada um dos três tipos de frutos estudados (maçã, amêndoa e manga), foram utilizadas, respectivamente, 726, 385 e 1154 imagens. Para contornar o baixo número de imagens disponíveis, os autores aplicaram técnicas para aumentar em mais de 50% o conjunto de dados de treinamento das frutas maçã e manga.

Os resultados foram analisados utilizando as métricas AP e *f1-score*. A rede que obteve melhor desempenho foi a VGG16, que resultou em um aumento considerável do *f1-score* em comparação com a ZF-Net. Para aumentar a precisão na detecção de frutos menores, os

autores treinaram uma série de limiares para IoU diferentes. As detecções de maçãs e mangas, atingiram AP com valores acima de 80%, já as amêndoas, que possuem o perfil mais semelhante ao propósito deste trabalho, sua AP pontuação máxima foi de pouco mais que 70%, em cerca de 100 épocas. O *f1-score* das frutas, maçã, manga e amêndoas correspondem respectivamente a: 90,4%, 90,8% e 77,5% . A baixa pontuação em *f1-score* e AP da amêndoa é justificável pela semelhança em cor e textura à folhagem, as quais, quando combinadas com imagens de baixa resolução, resultaram em um conjunto de dados difícil de rotular e realizar a detecção manualmente. Um dos principais desafios relatados pelos autores foi a formação da base de dados, visto que os dados são capturados em cenas ao ar livre, onde há variações significativas, devido à variabilidade em: condições de iluminação, distância da fruta, agrupamento de frutas, ponto de vista da câmera dentre outros aspectos.

4.2 *Detecção individual de tomates em plantas*

O problema da detecção individual de frutos é fundamental para a análise da safra, visto que predições e análises podem ser feitas com base nos resultados obtidos. Sendo assim, em Yamamoto *et al.* (2014) é proposto um método eficaz para a detecção individual de tomates jovens, maduros e imaturos em plantas. O método envolve três etapas: a segmentação baseada em pixels; a segmentação baseada em *blob* e a detecção individual das frutas.

Na etapa de segmentação baseada em pixels, as imagens foram rotuladas em *fruit*, *leaf*, *stem* e *background* e em seguida extraídas 15 características de cor. Na segunda etapa, a segmentação baseada em *blob* é aplicada gerando 3 classes: *single-fruits*, *multi-fruits* e *non-fruit*. As segmentações são realizadas com base em árvores de decisão que são geradas de acordo com as características encontradas. Na terceira etapa, o algoritmo *K-means* foi aplicado para determinar automaticamente o número de *clusters* com intuito de detectar os frutos individuais nos *blobs* identificados como pertencentes a classe *multi-fruits*.

O método desenvolvido, por utilizar o algoritmo *K-means*, não exige ajuste de limiares para detecção de frutas individuais, essa é uma vantagem importante, pois os limiares ideais sempre variam de imagem para imagem. A abordagem proposta obteve *recall* de 0,80 e *precision* de 0,88. Analisando os valores de *recall* e *precision* obtidos, nota-se que o *f1-score* do método proposto é inferior ao obtido no trabalho apresentado da seção anterior.

4.3 *Reconhecimento de frutas usando características de cor e textura*

O problema da *detecção de frutas em imagens naturais* é uma questão corrente na literatura, algumas razões como plano de fundo variado, ou casos em que a fruta é da mesma cor ou similar que elementos do plano de fundo, exemplificam os desafios encontrados na resolução do problema. Jana *et al.* (2017) apresentou um método de segmentação baseado em informações de cor e borda e uma classificação para os frutos maçã, pera asiática, pepino, manga, laranja, abacaxi, romã e morango.

O método consiste em: pré-processamento da imagem, extração de características, representação das características e classificação usando SVM. Na primeira etapa foi utilizado o algoritmo *GrabCut* que separa o plano de fundo da região de interesse. Na segunda etapa, foram analisadas a textura utilizando a matriz de nível de cinza de co-ocorrência e a cor. Nas últimas etapas, o modelo de SVM é treinado com base nas características extraídas. A técnica apresentada possui 83,33% de *precision* geral, demonstrando equilíbrio entre precisão e tempo em comparação a outra técnicas analisadas. No entanto, observar somente a métrica *precision*, não é a melhor abordagem para analisar o desempenho de um detector de frutas, visto que devemos levar em consideração os falsos negativos, neste caso, o ideal seria analisar as métricas *recall* e *f1-score*.

4.4 *Detecção de doenças em frutos usando características de cor e textura*

Para manter o nível de crescimento efetivo, rendimento das frutas e ainda mantê-las saudáveis, os agricultores precisam monitorar os frutos desde a colheita até o seu período de progresso. No entanto, o monitoramento manual não traz resultados precisos ou satisfatórios, pois existe a necessidade constante da opinião de especialistas, dito isso, Awate *et al.* (2015) propôs um sistema eficaz para detecção de doenças nos frutos uvas, maçã e romã, visando otimizar o monitoramento em questão.

A metodologia para treinamento e aprendizagem proposto no sistema é dividida em etapas. Na primeira e na segunda etapa são abordados a forma de aquisição das imagens, e o seu processo de segmentação, na qual o algoritmo *K-means* é utilizado para rotular os *pixels* da imagem. Na terceira etapa, utilizou-se para o processo de extração de característica o algoritmo *Surf* e como descritor local e detector a segmentação *blob*. Por fim, uma rede neural artificial é treinada como classificador, com as imagens pré-processadas e características extraídas. Como

resultado, a abordagem proposta pode apoiar o diagnóstico preciso de doenças em frutas com esforço computacional menor.

4.5 Comparativo entre atividades do estado da arte

A seguir a Tabela 1 apresenta as principais atividades realizadas no estado da arte em comparação com o que foi desenvolvido nesta pesquisa. A atividade destacada de vermelho representa o diferencial deste trabalho.

Tabela 1 – Comparação entre esta monografia e o estado da arte.

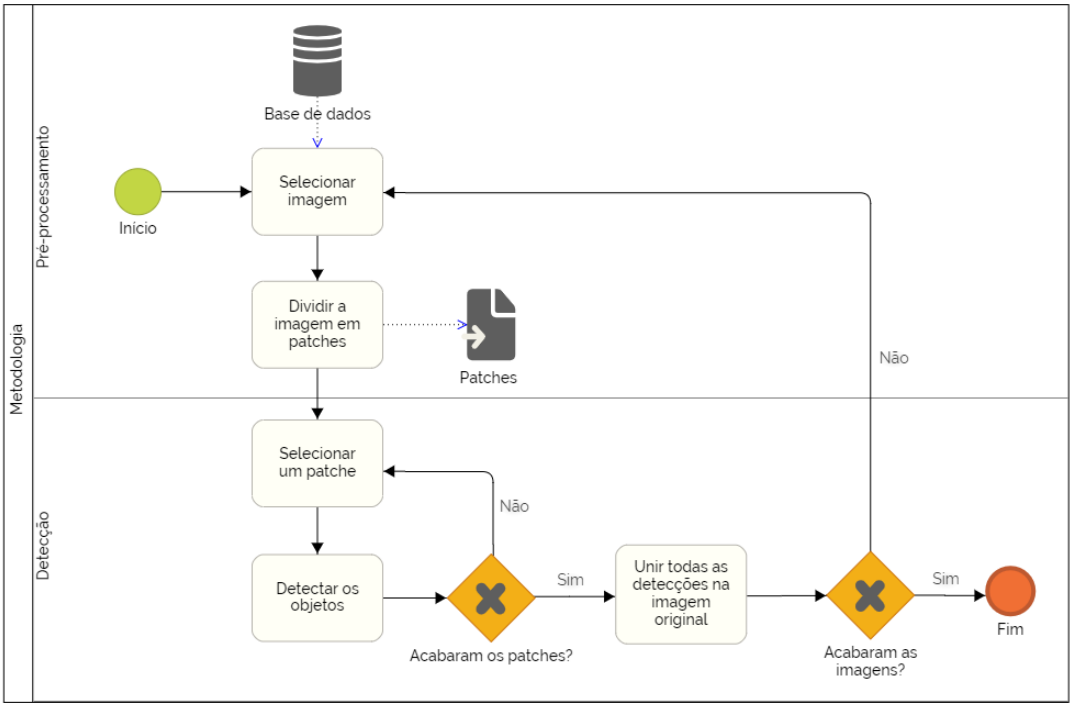
Lista de Atividades	Trabalhos Comparados				
	Bargoti e Underwood (2017a)	Yamamoto (2014)	Jana (2017)	Awate (2015)	TCC Williana
Criação e anotação de uma base de dados com frutos de acerola					X
Detecção de frutos usando *Deep Learning*	X				X
Detecção individual de frutos em imagens naturais	X	X	X		X
Detecção de frutos usando segmentação em imagens naturais		X	X	X	
Detecção de doença nos frutos em imagens naturais				X	

Fonte: Elaborado pelo Autor (2019).

5 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo serão abordados as fases necessárias para alcançar os objetivos gerais e específicos propostos. Todas as implementações foram desenvolvidas na linguagem de programação python, utilizando a plataforma de código aberto TensorFlow, a *Application Programming Interface* (API) de alto nível Keras e bibliotecas conhecidas como Numpy e OpenCV. Na Figura 11 há uma representação gráfica da metodologia descrita a seguir.

Figura 11 – Representação da metodologia proposta.



Fonte: Elaborado pelo Autor (2019).

5.1 Aquisição das imagens

A acerola *Malpighia emarginata DC* é um arbusto que se desenvolve em clima tropical e subtropical (NAGY *et al.*, 1980), seus frutos são ricos em ácido ascórbico, popularmente conhecido como Vitamina C. A aceroleira pode produzir ao mesmo tempo flores e frutos em diferentes estágios de formação (PONTES *et al.*, 2015). Para aquisição da base de dados de imagens utilizadas neste trabalho, os autores retiraram fotografias das aceroleiras na Fazenda Meri Pobo Agropecuária, em 3 lotes diferentes, com aparelho celular comum da marca *Iphone 5* com câmera de 8 *megapixels*.

Desta forma, foram obtidas 72 imagens de aceroleiras, subdividas em:

1. Lote 1: 12 imagens de frente e 12 imagens na parte detrás de cada aceroleira do lote;
2. Lote 2: 12 imagens de frente e 12 imagens na parte detrás de cada aceroleira do lote;
3. Lote 3: 12 imagens de frente e 12 imagens na parte detrás de cada aceroleira do lote.

Na Figura 12 há um exemplo de fotografia retirada de uma aceroleira completa, correspondente ao Item 1, Lote 1. Note que a figura está com tamanho reduzido, a imagem original possui dimensões de 3.024 *pixels* de largura por 4.032 *pixels* de altura.

Figura 12 – Imagem retirada de aceroleira situada na fazenda Meri Pobo Agropecuária com tamanho reduzido. Tamanho original de 3.024 x 4.032 *pixels*.

Fonte: Base de dados.

5.1.1 Desafios no processo de aquisição das imagens

Analisando os possíveis problemas que ocorreriam na formação da base de dados, seguindo o padrão proposto, nota-se dois principais desafios, sendo o primeiro, as dimensões mínimas que a imagem necessita para representar, sem perda significativa de qualidade, todos os

objetos presentes. O segundo desafio se encontra no tamanho dos objetos em relação a imagem completa da aceroleira. Como exemplificado na Figura 12, a imagem necessitou ser de alta resolução, para conseguir capturar todos os detalhes essenciais da aceroleira e seus pequenos frutos.

A complicação existente no processamento de imagens grandes com *Deep Learning*, está relacionado a capacidade de performance da máquina em que os modelos são executados. Redes CNNs e derivadas, interpretam imagens como múltiplos vetores ou matrizes cujas dimensões correspondem a altura x largura x profundidade da imagem, onde profundidade significa o número de canais que a imagem possui, dessa forma, utilizando a Figura 12 como exemplo, para representar esta imagem seria necessário processar uma matriz de $3.024 \times 4.032 \times 3 = 36.578.304$ posições.

5.1.2 Pré-processamento

Diante dos desafios mencionados, a solução que mais se adequou aos requisitos deste trabalho é a detecção baseada em *patches*, técnica amplamente utilizada na literatura (BARGOTI; UNDERWOOD, 2017a; XIA *et al.*, 2018; MENG *et al.*, 2017; KOVALEV *et al.*, 2016; CRESSON, 2018). A detecção baseada em *patches*, consiste basicamente em dividir a imagem em subimagens/*patches* de tamanho menor, para que o processamento seja realizado nas imagens menores em vez das imagens originais. Para este trabalho a divisão foi feita seguindo o formato de *grid*, com tamanho fixo de 512 *pixels* de largura e 512 *pixels* de altura, como apresentado na Figura 13.

Como pode ser observado, o recorte no formato *grid*, gera também imagens que não possuem o fruto da acerola, como por exemplo, céu, chão e outros ambientes do plano de fundo. Dessa forma, um trabalho manual se fez necessário para avaliar quais *patches* deveriam ser passados para etapa de marcação das imagens, visto que cada *patche* para fazer parte da base de dados deve necessariamente possuir ao menos uma acerola.

Figura 13 – Imagem retirada de aceroleira situada na fazenda Meri Pobo Agropecuária, separada em *patches* de 512x512.

Fonte: Base de dados.

Devido ao número reduzido de imagens capturadas, fez-se necessário a complementação da base de dados com outras imagens retiradas da Fazenda Meri Pobo e também imagens retiradas da internet. Para complementar as imagens retiradas na Fazenda Meri Pobo foi utilizada a técnica de divisão baseada em *patches*. Para obter mais imagens da internet foi utilizada as seguintes *strings* de busca no *Google Images*: acerola, *Barbados cherry* e *Malpighia Emarginata*.

Portanto, a constituição da base de dados completa foi realizada através de:

1. 166 imagens são resultantes da operação de divisão dos *patches* nas imagens retiradas na Fazenda Meri Pobo Agropecuária;

2. 238 imagens são decorrentes de buscas em mecanismo de pesquisa na web, como *Google Images*.

À vista disso, como resultado do processo de aquisição e pré-processamento das imagens a base de dados final é constituída por 404 imagens, com dimensões padronizadas em 512 *pixels* de largura e 512 *pixels* de altura.

Figura 14 – Imagens retiradas da base de dados.

Fonte: Fazenda Meri Pobo Agropecuária

Fonte: Google Images.

Na Figura 14, à esquerda, há um exemplo de imagem obtida na fazenda Meri Pobo Agropecuária, respectivo ao item 1 e na direita, há um exemplo de imagem obtida em páginas web, correspondente ao item 2.

5.2 Analisando a base de dados final

Apesar de todo o processamento aplicado as imagens, a base de dados ainda é considerada um desafio, por ter um ambiente não controlado das imagens, variando inclinação, presença ou não de sombras, iluminação, ruídos ou borrões, distanciamento da câmera, bem como o fato de que o objeto acerola em alguns estágios possui cor semelhante ao *background*, como também possui muitos objetos sobrepostos, como nos casos de cachos de acerolas, dificultando ainda mais o processo de detecção individual dos objetos. A Figura 15 mostra exemplos dessa variação.

Por fim, a base de dados final é divida em: 80% para treinamento e 20% para teste e validação. Apesar de não ser uma boa prática para modelos de *Deep Learning*, utilizar o mesmo conjunto de dados para validação e testes, é uma forma de manter uma quantidade razoável de

Figura 15 – Imagens diversificadas da base de dados.

Fonte: Base de dados.

Fonte: Base de dados.

Fonte: Base de dados.

Fonte: Base de dados.

imagens no conjunto de treinamento, dado as condições de possuir um conjunto pequeno de imagens disponíveis.

A marcação de todas as imagens da base de dados final, atividade responsável por delimitar as coordenadas de cada objeto de interesse na base de dados final, foi realizada utilizando a ferramenta de anotações gratuita e *open-source*: *Pychet Labeller*, desenvolvida em *python* e diponível no repositório https://github.com/acfr/pychetlabeller. As imagens foram marcadas em uma classe: acerola, gerando como resultado arquivos no formato *Extensible Markup Language* (XML), que representam informações básicas da imagem, como por exemplo, localização no sistema de arquivos, e as posições relativas a cada objeto encontrado, com as coordenadas $Xmin, Ymin, Xmax, Ymax$.

5.3 Detecção de objetos em aceroleiras

Para utilizar a abordagem de detecção de objetos *Faster-RCNN* descrita em detalhes na seção 3.3.1, foi utilizado o repositório *open source*: https://github.com/matterport/Mask_RCNN. Este repositório representa a implementação da rede *Mask Region-based Convolutional Neural Network* (Mask-RCNN), o estado da arte em detecção de objetos e segmentação de imagens, onde utiliza a rede *Faster-RCNN* para realizar a detecção de objetos e, logo após, segmenta cada objeto detectado, sendo assim, para utilizá-lo. Neste trabalho algumas modificações no código original do repositório foram necessárias, para que a rede ignorasse a tarefa de segmentação, mantendo apenas a detecção de objetos.

6 RESULTADOS

Este capítulo discute os resultados do método proposto no Capítulo 5, utilizando uma base de dados construída a partir de diversas fontes, conforme apresentado na Seção 5.2. Posteriormente os resultados serão comparados com alguns métodos de objetivos similares ao desta pesquisa presentes na literatura, como vistos no Capítulo 4.

Por se tratar de uma pesquisa que utiliza um modelo de *Deep learning* e portanto exige alto poder computacional, o *hardware* a ser utilizado para a implementação do modelo foi o serviço em nuvem *Google Colab Pro*, versão paga do serviço *Google Colab*, hospedado pelo *Google*. O *Google Colab* é um ambiente baseado no *Jupyter Notebook*, que não requer configuração para ser usado e permite que qualquer pessoa escreva e execute códigos na linguagem python através do navegador. Este serviço fornece um ambiente de trabalho em python com versões estáveis do *framework Tensorflow*, comumente utilizado em modelos de *Machine learning*. O *hardware* disponibilizado pela *Google* durante a implementação deste método, possui a seguinte configuração: processador da Intel(R) Xeon(R), com frequência base de 2.20GHz e dois núcleos, 25 GB de memória RAM e GPU Tesla modelo P100-PCIE com memória de 16GB. O *Google Colab Pro* apresenta algumas restrições sobre o uso da plataforma, as duas principais são: tempo de acesso e tempo limite de inatividade. O tempo de acesso limite no *Google Colab Pro* é de 24h seguidas e o tempo limite da tolerância de inatividade é de 90 minutos. Toda a implementação do método foi desenvolvida em linguagem python, através do ambiente de desenvolvimento *Colab Pro*, utilizando principalmente as bibliotecas OpenCV, Mrcnn, Numpy e Keras.

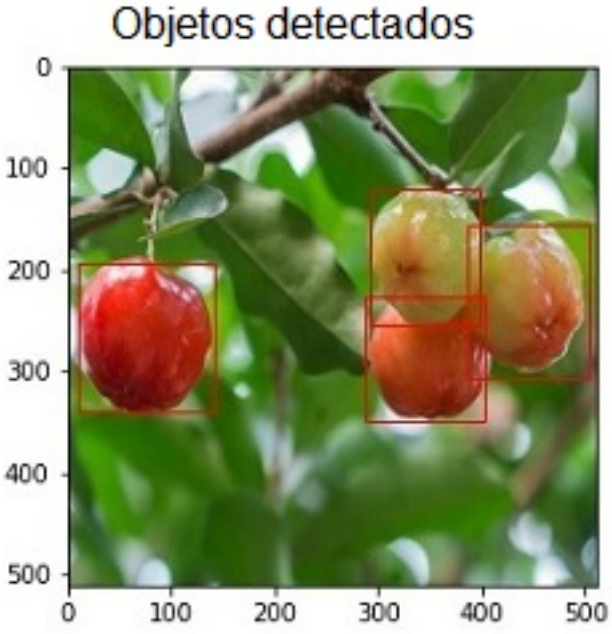
6.1 Detecção de objetos em aceroleiras

O objetivo dessa etapa, como já mencionado anteriormente, é localizar acerolas em uma imagem digital. Esse processo é realizado através da rede Mask-RCNN, utilizando apenas a parte da rede para a detecção de objetos. Para avaliar o modelo foram utilizadas 130 imagens com o máximo de diversificação possível dadas as limitações definidas no Capítulo 5. Na Figura 16 há um exemplo de resultado obtido na etapa de testes.

Como utilizado em Ren *et al.* (2015a), a métrica para validação do modelo deste trabalho também será a mAP, além de ser o estado da arte para avaliar modelos de detecção de objetos, foi a principal métrica adotada na competição de detecção de objetos promovido pela *Common Objects in Context*(COCO) e PASCAL Visual Object Classes(VOC)(DANIILIDIS

Figura 16 – Exemplo de detecção realizada em imagem de teste.

Fonte: Base de dados.



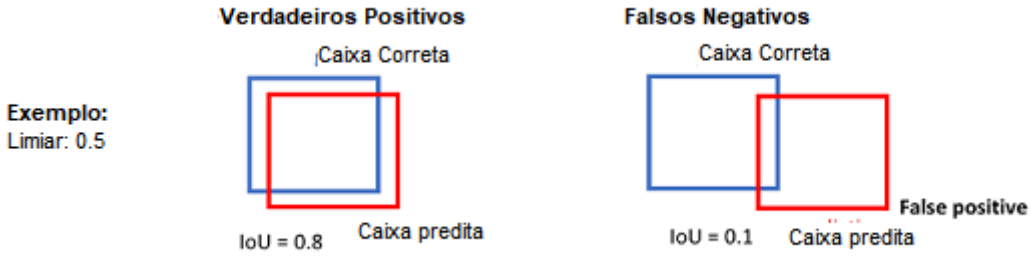
Fonte: Resultado do modelo.

et al., 2010). Para utilizar a métrica mAP, serão utilizados os conceitos de precisão, *recall* e AP. Dessa maneira, a precisão é responsável por medir a capacidade do modelo de predizer valores positivos como realmente verdadeiros e o *recall* responsável por medir a capacidade do modelo acertar quando avalia um exemplo como negativo. Já a métrica AP, corresponde a uma combinação das métricas precisão e *recall*.

Dado que tp corresponde a verdadeiros positivos, tn corresponde a verdadeiros negativos, fp corresponde a falso positivos e fn corresponde a falso negativos. Considerando tp quando a caixa delimitadora prevista obtém a pontuação IoU maior que determinado limiar, tornando-se um positivo verdadeiro, e fp quando a caixa delimitadora prevista não é associada

a uma *ground truth*, ou seja, a caixa que o modelo previu não existe, é um falso positivo, e fn como número de erros, isto é, quando existe *ground truth* que não foram previstas, na Figura 17 há um exemplo utilizando um limiar de 50%.

Figura 17 – Representação de como as variáveis tp, fn e fp se relacionam com a pontuação IoU, utilizando um limiar de 50%.



Fonte: Adaptado a partir de Jordan (2018)

Sendo assim, a precisão e o *recall* são definidos como nas seguintes equações.

$$precisão = \frac{tp}{tp+fp}$$

$$recall = \frac{tp}{tp+fn}$$

Uma maneira comum de combinar as métricas *recall* e precisão é plotando a curva de precisão-*recall*, onde os valores de *recall* assumem o eixo x e os valores de precisão assumem o eixo y. Na Figura 18 há um exemplo de análise da saída de um detector de objetos, onde na coluna *rank* representa a quantidade objetos detectados, na coluna Está correto? apresenta o valor *true* se a detecção possuir $IoU >= 0.5$, caso contrário *false*. Por fim, na Figura 19 é plotado a curva de precisão-*recall*.

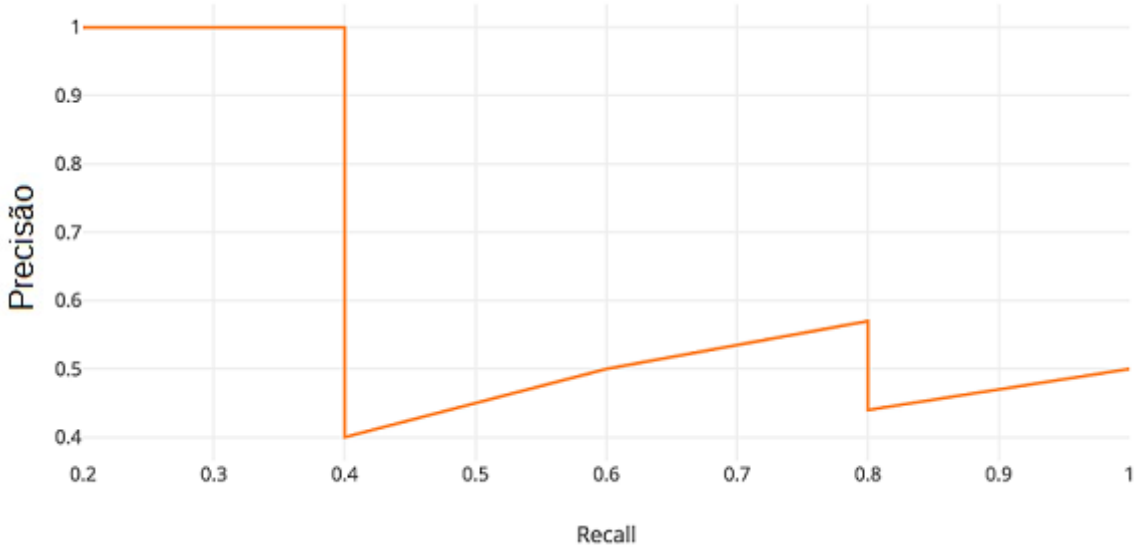
A medida que o modelo se depara com falsos positivos a curva começa a apresentar o padrão de *zigzag*, evidenciando as pequenas variações de precisão que ocorrem. Para suavizar a curva de precisão-*recall*, os valores de precisão são trocados pelo maior valor de precisão a direita em um determinado nível de *recall*, dessa forma a precisão interpolada p_{interp} é definida como:

Figura 18 – Análise das saídas de um detector de objetos.

Rank	Está correto?	Precisão	Recall
1	True	1.0 ↑	0.2 ↑
2	True	1.0 –	0.4 ↑
3	False	0.67 ↓	0.4 –
4	False	0.5 ↓	0.4 –
5	False	0.4 ↓	0.4 –
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑

Fonte: Adaptado a partir de Hui (2018)

Figura 19 – Representação da curva de precisão-*recall*.



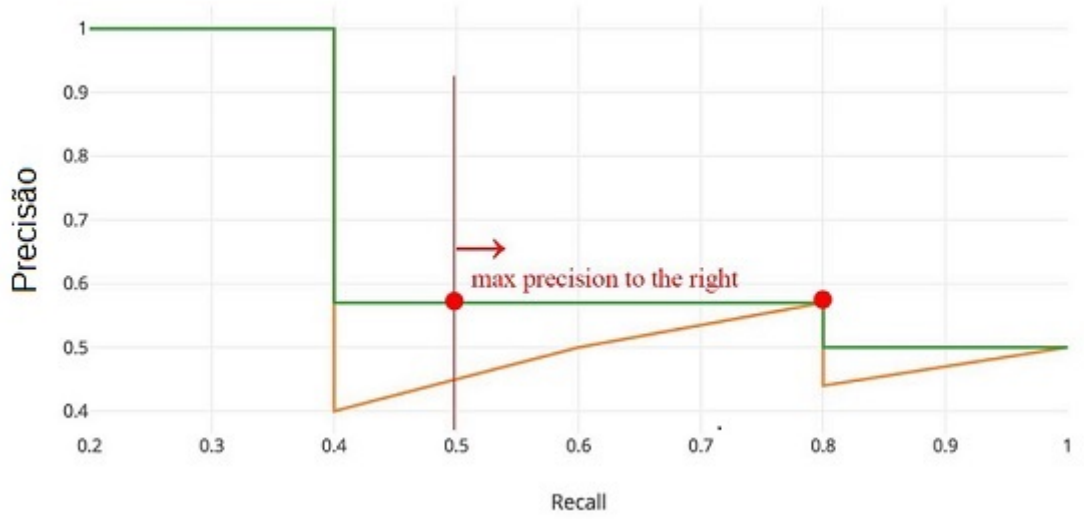
Fonte: Adaptado a partir de Hui (2018)

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$

Na Figura 20 há uma representação do resultado da interpolação na curva precisão-*recall* apresentada anteriormente. Embora a curva de precisão-*recall* possa ser utilizada para avaliar o desempenho de um detector, não é fácil utilizá-la para comparar diferentes detectores quando as curvas se cruzam.

Para encapsular as métricas de precisão e *recall* em um único valor numérico, a métrica AP é utilizada, onde basicamente é uma aproximação da área sob a curva de precisão-*recall*(YILMAZ; ASLAM, 2006; PENG, 2019), podendo ser definida como:

Figura 20 – Representação da curva de precisão-*recall* com a interpolação.



Fonte: Adaptado a partir de Hui (2018)

$$AP = \int_0^1 p(r)\,dr$$

Variando de 0 a 1, correspondendo aos limites das variáveis precisão e *recall*, onde consequentemente a AP corresponderá a mesma variação de 0 a 1. Uma das formas de se calcular uma AP é através da Área Sob a Curva ou *Area under curve* (AUC) aplicada a uma curva de precisão-*recall* suavizada, onde sempre que for necessário calcular a p_{interp} então se calcula a AP, dessa forma o cálculo para obter o valor de uma AP é definido como:

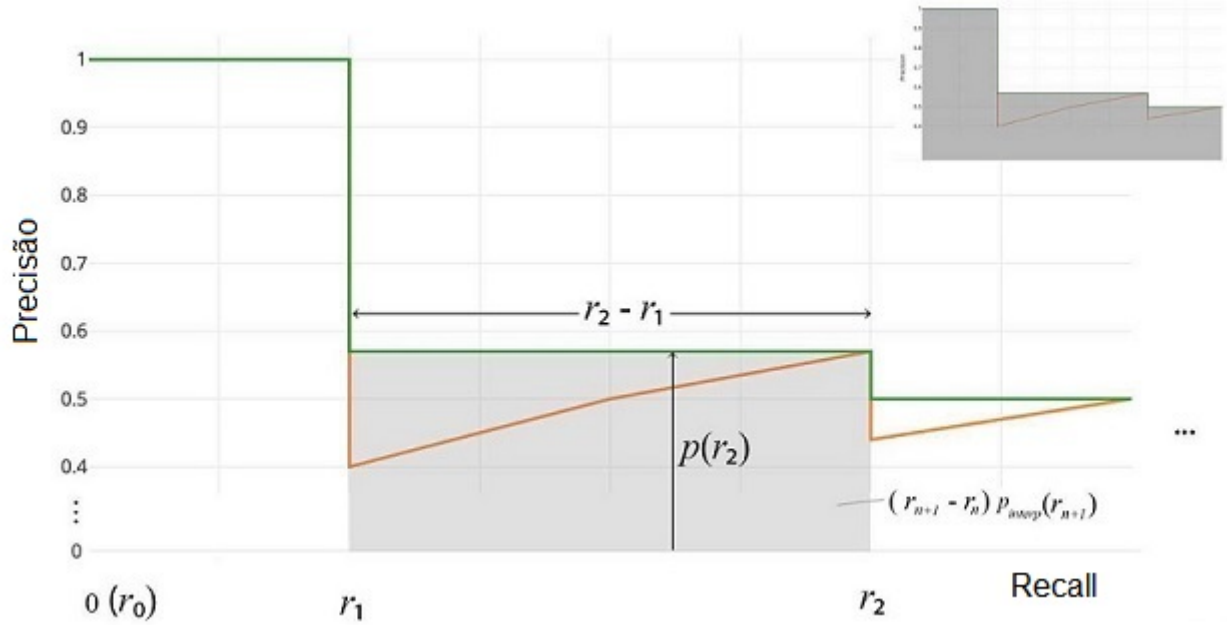
$$AP = \sum (r_{n+1} - r_n)\, p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

Na Figura 21 há uma representação simples de como a formula da AP pode ser aplicada aos exemplos anteriores. Como a área forma a representação de um retângulo, para calcular sua área basta multiplicar a largura $r_2 - r_1$ vezes a altura $p(r_2)$ que é obtida através da função p_{interp}. Por fim, após calcular a área de cada retângulo, para obter a AP, basta somar todos as N áreas encontradas. Esta forma de calcular a AP foi adotada em desafios da *Pascal VOC* a partir da versão de 2010(EVERINGHAM *et al.*, 2015).

A métrica mAP é definida como a média de todas as AP's calculadas, onde a AP é calculada para cada imagem, e para cada classe. Portanto, uma mAP é definida como:

Figura 21 – Representação do cálculo da APna curva de precisão-*recall* com a interpolação.



Fonte: Adaptado a partir de Hui (2018)

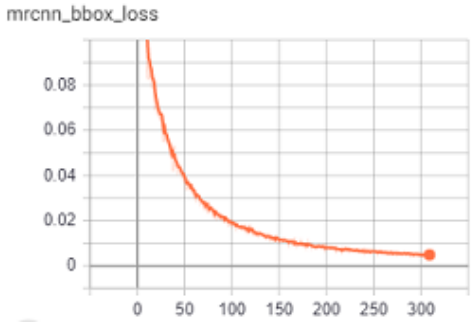
$$MAP = \frac{\sum_{n=1}^{N} AP(n)}{N}$$

Onde N é a quantidade total de dados a serem analisados.

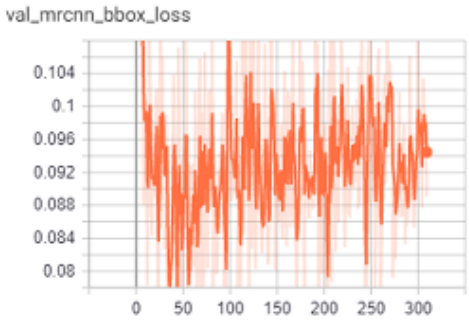
Como o *framework* utilizado para implementação deste modelo possui resultados além do exigido para este trabalho, que é a segmentação semântica, para avilá-lo durante o treinamento a variável *Loss* não pode ser considerada, visto que a mesma é um somatório dos *Loss* da detecção e segmentação. Sendo assim, para avaliar a evolução do modelo foi analisado as variáveis *mrcnn_bbox_Loss*, *mrcnn_class_Loss* e seus respectivos valores no conjunto de validação, representado pelas variáveis *val_mrcnn_bbox_Loss* e *val_mrcnn_class_Loss*. As variáveis que possuem *bbox* em seu nome, analisam a saída da rede com relação a caixa delimitadora.

O método proposto no Capítulo 5, foi treinado ao longo de 309 épocas, em aproximadamente 24h de execução, tempo limite permitido pelo *Google Colab Pro*. A evolução da aprendizagem do modelo é apresentado na Figura 22. Ao analisar o progresso de aprendizagem do *framework*, nota-se um comportamento estranho com os resultados das validações. Em Brownlee (2019), este comportamento é descrito como um conjunto de dados de validação que não fornece informações suficientes para avaliar a capacidade de generalização do modelo. Este comportamento pode ocorrer se o conjunto de validação possui poucos exemplos em comparação

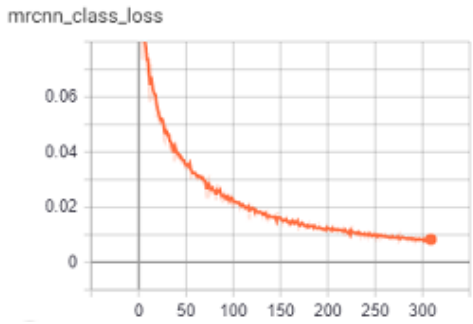
Figura 22 – Evolução da aprendizagem, *loss* por época.



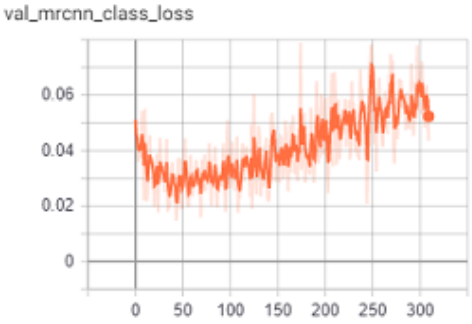
(a) Treinamento: caixas delimitadoras.



(b) Validação: caixas delimitadoras.



(c) Treinamento: classes.



(d) Validação: classes.

Fonte: Elaborado pelo Autor (2020).

com o conjunto de dados de treinamento.

Dessa forma, pode-se concluir, que à medida que a base de dados aumenta, consequentemente o modelo aumentará sua capacidade de generalização. Este comportamento foi observado ao longo dos treinamentos, visto que à medida que a base de dados aumentava, diminuía o padrão de *zigzag*, no conjunto de validação. Na Tabela 2, é apresentado os melhores resultados obtidos durante o treinamento do modelo.

Tabela 2 – Resultados da detecção de acerolas.

Conjunto	mAP	IoU
Treinamento	89,5,%	0.5
Teste	92,2%	0.5

Fonte: Elaborado pelo Autor (2020).

Estes resultados, apesar de terem potencial para alcançar valores ainda melhores, demostram um bom desempenho do detector. Analisando os resultados explanados na Seção 4.1, especificamente as detecções de amêndoas, onde obteve AP máxima de pouco mais que 70%, o modelo proposto nesta monografia obteve aproximadamente um acréscimo de 22,2% de AP, em comparação com os resultados de Bargoti e Underwood (2017a).

Analisando outras abordagens, como o modelo apresentado por Lin *et al.* (2020),

onde é apresentado a detecção de frutas esféricas ou cilíndricas, baseado em informações de cor, profundidade e forma, contasta-se que os resultados para a detecção dos frutos de pimentão, berinjela, e goiaba, apresentaram um mAP de 86,3%, 74,1% e 80,7% respectivamente. Dessa forma, pode-se observar que este trabalho atingiu mAP superior em todos os resultados, não só com o conjunto de testes, como também com o conjunto de treinamento.

7 CONCLUSÃO

Este capítulo discute as considerações e lições aprendidas a respeito da metodologia, criada neste trabalho, para detecção de frutos de acerola em imagens digitais, assim como os resultados obtidos.

7.1 Considerações gerais

Neste trabalho foi elaborada uma metodologia para detecção de frutos de acerola em imagens digitais de aceroleiras retiradas no campo. Para desenvolver a proposta foram analisadas diversas técnicas e trabalhos relacionados, visando minimizar os desafios encontrados neste tipo de aplicação, como exemplo, pode-se citar: o processamento de imagens de alta resolução, treinamento do modelo de *deep learning* com uma base de dados consideravelmente pequena, criada em um ambiente não controlado.

A metodologia proposta consiste na adaptação do *framework* Mask-RCNN, como descrito na Seção 5.3, para detecção de frutos de acerola em imagens digitais. Para tal, na etapa de pré-processamento das imagens, utilizou-se a técnica de detecção baseada em *patches*, buscando assim resolver o problema da alta resolução das fotos. Para reduzir o problema do baixo número de dados, mais imagens foram adquiridas através do buscador *Google Images*. Esse aumento de dados, permitiu que o modelo aprendesse mais, por ter acesso a mais exemplos do fruto de acerola.

A consequência da adquisição dessas imagens, realizadas de forma aleatória através de mecanismos de busca, é o aumento da complexidade dos dados durante a etapa de aprendizagem do modelo, pois desta forma, as imagens acabam tendo uma alta variabilidade, como por exemplo: distanciamento do fruto, condições de iluminação, dentre outros aspectos, como detalhados nas Seções 5.1.1 e 5.2. Apesar do aumento da complexidade do modelo, com a aquisição das novas imagens, pode-se afirmar que aprendizado do mesmo aumentou, de tal forma que esta consequência tornou-se irrelevante. Dito isso, pode-se concluir que o aumento da base de dados, ainda que sejam dados variáveis, refletiram no aumento de precisão do modelo. No entanto, essa variabilidade dos dados também resultou em um baixo desempenho de *recall*. Assim, pode-se inferir que para obter melhores resultados em precisão e *recall* paralelamente, a base de dados deve possuir mais imagens, sendo também interessante a possibilidade de testes com imagens retiradas em um ambiente controlado.

Por fim, pode-se concluir também que a principal dificuldade encontrada na realização desta pesquisa, se encontra na criação da base de dados. A atividade de aquisição das imagens demanda muito esforço físico e repetitivo, visto que após a captura de imagens ainda há a necessidade da aplicação das etapas de pré-processamento e marcação. Compreender na prática, o quanto é relevante para o aprendizado do modelo a forma como a base de dados é gerada é de suma importância para otimizações futuras da metodologia definida.

7.2 Trabalhos futuros

Espera-se em trabalhos futuros reformular a base de dados objetivando uma melhoria dos resultados obtidos com a métrica *recall*, assim como desenvolver um módulo de classificação de estágios de maturação da acerola. Desenvolver este módulo de classificação é uma tarefa igualmente dispendiosa quando se trata da criação da base de dados, visto que deve-se recolher exemplos para cada estágio de maturação. No entanto, uma vez que a base de dados esteja completa, o problema se torna um problema de classificação de imagens simples, onde uma CNN para classificação poderia ser aplicada, ou até mesmo adaptar alguma CNN conhecida na literatura.

Por fim, também espera-se em trabalho futuros desenvolver um módulo de análise e predição de colheita para frutos de acerola. Esta etapa pode ser facilmente desenvolvida por se tratar de um fruto, como apresentado em Pontes *et al.* (2015), que não apresenta muitas alterações fenológicas ligadas à questões climáticas, possuindo um período fixo de 3 dias para transação de um estágio para outro de maturação. O uso deste tipo de previsões poderia auxiliar fruticultores na tomada de decisão sobre o melhor momento de colheita em fazendas.

REFERÊNCIAS

ACADEMY, D. S. As 10 principais arquiteturas de redes neurais. In: ____. **Deep Learning Book**. [s.n.], 2019. cap. 10. Disponível em: <http://www.deeplearningbook.com.br>.

AGGARWAL, C. C. Neural networks and deep learning. **Springer**, Springer, v. 10, p. 978–3, 2018.

AMIT, Y. **2D object detection and recognition: Models, algorithms, and networks**. [S.l.]: MIT Press, 2002.

ANANTH, S. **Faster RCNN for object detection, a technical paper summary**. 2019. Disponível em: https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46. Acessado: 01 jan. 2020.

ARAÚJO, F. H.; CARNEIRO, A. C.; SILVA, R. R.; MEDEIROS, F. N.; USHIZIMA, D. M. Redes neurais convolucionais com tensorflow: Teoria e prática. **SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos**, Sociedade Brasileira de Computação, v. 1, p. 382–406, 2017.

AWATE, A.; DESHMANKAR, D.; AMRUTKAR, G.; BAGUL, U.; SONAVANE, S. Fruit disease detection using color, texture analysis and ann. In: IEEE. **2015 International Conference on Green Computing and Internet of Things (ICGCIOT)**. [S.l.], 2015. p. 970–975.

BACKES, A. R.; JUNIOR, J. J. d. M. S. **Introdução à visão computacional usando Matlab**. [S.l.]: Alta Books Editora, 2019.

BALLARD, D. H.; BROWN, C. M. **Computer vision**. [S.l.]: Prentice Hall, 1982.

BARGOTI, S.; UNDERWOOD, J. Deep fruit detection in orchards. In: IEEE. **2017 IEEE International Conference on Robotics and Automation (ICRA)**. [S.l.], 2017. p. 3626–3633.

BARGOTI, S.; UNDERWOOD, J. P. Image segmentation for fruit detection and yield estimation in apple orchards. **Journal of Field Robotics**, Wiley Online Library, v. 34, n. 6, p. 1039–1060, 2017.

BROWNLEE, J. **How to use Learning Curves to Diagnose Machine Learning Model Performance**. 2019. Disponível em: <https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/>. Acessado: 25 mai. 2020.

CALGARO, M.; BRAGA, M. A cultura da acerola. **Área de Informação da Sede-Col Criar Plantar ABC 500P/500R Saber (INFOTECA-E)**, Brasília, DF: Embrapa, 2012., 2012.

CRESSON, R. A framework for remote sensing images processing using deep learning techniques. **IEEE Geoscience and Remote Sensing Letters**, IEEE, v. 16, n. 1, p. 25–29, 2018.

CYGANEK, B. **Object detection and recognition in digital images: theory and practice**. [S.l.]: John Wiley & Sons, 2013.

DANIILIDIS, K.; MARAGOS, P.; PARAGIOS, N. **Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings**. [S.l.]: Springer, 2010. v. 6315.

EVERINGHAM, M.; ESLAMI, S. A.; GOOL, L. V.; WILLIAMS, C. K.; WINN, J.; ZISSERMAN, A. The pascal visual object classes challenge: A retrospective. **International journal of computer vision**, Springer, v. 111, n. 1, p. 98–136, 2015.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2014. p. 580–587.

GONZALEZ, R. C.; WOODS, R. C. **Processamento digital de imagens .** [S.l.]: Pearson Educación, 2009.

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

HUI, J. **mean Average Precision for Object Detection**. 2018. Disponível em: <https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173>. Acessado: 14 mai. 2020.

JANA, S.; BASAK, S.; PAREKH, R. Automatic fruit recognition from natural images using color and texture features. In: IEEE. **2017 Devices for Integrated Circuit (DevIC)**. [S.l.], 2017. p. 620–624.

JORDAN, J. **Evaluating image segmentation models.** 2018. Disponível em: <https://www.jeremyjordan.me/evaluating-image-segmentation-models/>. Acessado: 13 mai. 2020.

KAPUR, S. **Computer Vision with Python 3**. [S.l.]: Packt Publishing Ltd, 2017.

KOIRALA, A.; WALSH, K.; WANG, Z.; MCCARTHY, C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'mangoyolo'. **Precision Agriculture**, Springer, v. 20, n. 6, p. 1107–1135, 2019.

KOVALEV, V.; KALINOVSKY, A.; LIAUCHUK, V. Deep learning in big image data: Histology image classification for breast cancer diagnosis. In: **Big Data and Advanced Analytics, Proc. 2nd International Conference, BSUIR, Minsk**. [S.l.: s.n.], 2016. p. 44–53.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, Ieee, v. 86, n. 11, p. 2278–2324, 1998.

LESKOVEC, J.; RAJARAMAN, A.; ULLMAN, J. D. **Mining of massive data sets**. [S.l.]: Cambridge university press, 2020.

LIN, G.; TANG, Y.; ZOU, X.; XIONG, J.; FANG, Y. Color-, depth-, and shape-based 3d fruit detection. **Precision Agriculture**, Springer, v. 21, n. 1, p. 1–17, 2020.

MENG, Z.; FAN, X.; CHEN, X.; CHEN, M.; TONG, Y. Detecting small signs from large images. In: IEEE. **2017 IEEE International Conference on Information Reuse and Integration (IRI)**. [S.l.], 2017. p. 217–224.

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

NAGY, S. *et al.* **Tropical and subtropical fruits: composition, properties and uses.** [S.l.]: AVI Publishing Co., Inc., 1980.

PENG, G. Performance and accuracy analysis in object detection. 2019.

PONTES, A.; SOARES, F.; LIMA, F.; DINIZ, C. Uso do ciclo fenológico da aceroleira para padronizaÇÃo do ponto de colheita mecanizada. In: **XLIV Congresso Brasileiro de Engenharia Agrícola**. [S.l.: s.n.], 2015.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2015. p. 91–99.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2015. p. 91–99.

ROSEBROCK, A. **Intersection over Union for object detection**. 2016. Disponível em: <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>. Acessado: 01 mai. 2020.

ROSENBLATT, F. **Principles of neurodynamics. perceptrons and the theory of brain mechanisms**. [S.l.], 1961.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning internal representations by error propagation**. [S.l.], 1985.

RURAL, G. **No Ceará, cresce o cultivo da acerola orgânica para exportação**. 2014. Disponível em: <http://g1.globo.com/economia/agronegocios/noticia/2014/03/no-ceara-cresce-o-cultivo-da-acerola-organica-para-exportacao.html>. Acessado: 01 set. 2019.

SEBRAE, A. **Produtor rural está conectado à tecnologia da informação**. 2017. Disponível em: <https://revistapegn.globo.com/Tecnologia/noticia/2017/07/produtor-rural-esta-conectado-tecnologia-da-informacao.html>. Acessado: 01 set. 2019.

SHAH, A.; NASEEM, R.; IQBAL, S.; SHAH, M. A. *et al.* Improving cbir accuracy using convolutional neural network for feature extraction. In: IEEE. **2017 13th International Conference on Emerging Technologies (ICET)**. [S.l.], 2017. p. 1–5.

SKANSI, S. **Introduction to Deep Learning: from logical calculus to artificial intelligence**. [S.l.]: Springer, 2018.

SKIENA, S. S. **The data science design manual**. [S.l.]: Springer, 2017.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres. In: **Proceedings of the XXIX Conference on Graphics, Patterns and Images**. [S.l.: s.n.], 2016. p. 1–4.

XIA, G.-S.; BAI, X.; DING, J.; ZHU, Z.; BELONGIE, S.; LUO, J.; DATCU, M.; PELILLO, M.; ZHANG, L. Dota: A large-scale dataset for object detection in aerial images. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2018. p. 3974–3983.

YAMAMOTO, K.; GUO, W.; YOSHIOKA, Y.; NINOMIYA, S. On plant detection of intact tomato fruits using image analysis and machine learning methods. **Sensors**, Multidisciplinary Digital Publishing Institute, v. 14, n. 7, p. 12191–12206, 2014.

YILMAZ, E.; ASLAM, J. A. Estimating average precision with incomplete and imperfect judgments. In: **Proceedings of the 15th ACM international conference on Information and knowledge management**. [S.l.: s.n.], 2006. p. 102–111.

ZHAO, Z.-Q.; ZHENG, P.; XU, S.-t.; WU, X. Object detection with deep learning: A review. **IEEE transactions on neural networks and learning systems**, IEEE, v. 30, n. 11, p. 3212–3232, 2019.