UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DIEGO FALKOWSKI CARBONI

SISTEMA PARA IDENTIFICAÇÃO DE PLANTAS INVASORAS EM LAVOURAS COM DETECTORES DE OBJETOS APLICADOS A IMAGENS E VÍDEOS

PORTO ALEGRE

2021

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DIEGO FALKOWSKI CARBONI

SISTEMA PARA IDENTIFICAÇÃO DE PLANTAS INVASORAS EM LAVOURAS COM DETECTORES DE OBJETOS APLICADOS A IMAGENS E VÍDEOS

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como requisito para Graduação em Engenharia Elétrica.

Orientador: Prof. Dr. Valner João Brusamarello

PORTO ALEGRE

2021

DIEGO FALKOWSKI CARBONI

SISTEMA PARA IDENTIFICAÇÃO DE PLANTAS INVASORAS EM LAVOURAS COM DETECTORES DE OBJETOS APLICADOS A IMAGENS E VÍDEOS

Projeto de Diplomação apresentado ao Departamento de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como requisito para Graduação em Engenharia Elétrica.

Prof. Dr. Luiz Tiarajú dos Reis Loureiro
Coordenador do departamento de Engenharia Elétrica - UFRGS

Banca Examinadora:

Prof. Dr. Alexandre Balbinot
UFRGS

Prof. Dr. Marcelo Götz
UFRGS

Prof. Dr. Valner João Brusamarello
Prof. Orientador - UFRGS

Data de Aprovação: 21 de maio de 2021.

Agradecimentos

Ao Prof. Valner por ter possibilitado e auxiliado a realização deste trabalho. Ao Gabriel (Eirene Solutions) por ter disponibilizado o banco de dados. À minha irmã Laura, por ter se disponibilizado para fazer a revisão textual. Aos meus pais, pelo imenso apoio durante a realização deste trabalho.

Resumo

O constante aumento da demanda por alimentos exige que técnicas inovativas sejam aplicadas à agricultura. Este trabalho apresenta a metodologia de treinamento de um detector de objetos por meio de *deep learning* e avaliação por meio de métricas referência no campo da visão computacional para a detecção de plantas invasoras em meio a lavouras, com objetivo de ser utilizado em conjunto com um sistema de pulverização seletiva. Os resultados encontrados indicam que o sistema detector proposto com a utilização de modelos YOLO, apresentam AP@0.5 próximos à 0.7 na detecção de ervas daninhas, indicando alto potencial do modelo para este tipo de aplicação. Outro objetivo deste trabalho, a análise da viabilidade de uso dos modelos YOLO com o sistema computacional de baixo custo Raspberry PI 4 (4GB) por meio da velocidade de inferência em arquivos de imagem e vídeo, demonstrou que o Raspberry PI 4 não é adequado para aplicações em tempo real a 30 FPS. Apesar disso, os resultados obtidos demonstram que é possível a utilização do modelo YOLO com Raspberry PI 4 para visão computacional em aplicações onde seja necessário amostrar imagens em menos de um quadro por segundo (1 FPS).

Palavras-chave: Agricultura, soja, ervas daninhas, plantas invasoras, pulverização seletiva, visão computacional, detecção de objetos, *deep learning*, YOLO, CNN, Raspberry PI 4.

Abstract

The constant increase in demand for food requires innovative techniques being applied to agriculture. This study presents the methodology of training of an object detector using deep learning and it's evaluation by reference metrics in the field of computer vision to be used in the detection of weeds located in crop fields, in order to be used in conjunction with a selective spraying system. The results found indicate that the proposed detector system with the use of YOLO models, presents AP@0.5 close to 0.7 in the detection of weeds, indicating the model's high potential for this type of application. Another objective of this study, the analysis of feasibility of using YOLO models in a low-cost computer system, the Raspberry PI 4 (4GB), by inference speed testing in image and video files, demonstrated that the Raspberry PI 4 is not suitable for real-time applications at 30 FPS. However, the results demonstrate that it is possible to use the Raspberry PI 4 for computer vision applications where it is necessary to sample images in less than one frame per second (1 FPS).

Keywords: agriculture, soy, weeds, selective spraying system, computer vision, object detection, deep learning, YOLO, CNN, Raspberry PI 4.

Lista de Figuras

Lista de abreviaturas

RNA	Rede Neural Artificial
MLP	Multi Layer Perceptron
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
R-CNN	Region Convolutional Neural Network
FCN	Fully Convolutional Network
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
ReLU	Rectified Linear Unit
GPU	Graphics Processing Unit
ROI	Region Of Interest
AP	Average Precision
mAP	Mean Average Precision
IOU	Intersection Over Union
TPU	Tensor Processing Unit

Sumário

1 Introdução

A agricultura é uma das principais bases da economia do Brasil, sendo responsável por uma participação de 26,6% do PIB em 2020, segundo a Confederação Nacional de Agricultura (CNA, 2021). O Brasil figura entre os principais produtores e exportadores de produtos agropecuários e é esperado que se torne um dos maiores fornecedores de alimentos para o mundo (EMBRAPA, 2016) (RURAL, 2020).

Segundo o USDA[1] (USDA, 2021), é estimado que a demanda por alimentos irá aumentar entre 70% e 100% até 2050, e que para atender esta demanda, será necessário que a produção agrícola em países em desenvolvimento aumente em aproximadamente 100%.

Enquanto de um lado a demanda por alimentos continua aumentando, por outro, temos o problema da redução do espaço disponível para terrenos agrícolas, que simplesmente não é possível ser resolvido pelo desmatamento de biomas nativos, tendo em vista a perda de biodiversidade e mudanças climáticas que tem sido observados nos últimos anos. Por este motivo, novas técnicas de agricultura são necessárias para o contínuo sustento do crescimento da demanda por alimentos.

Em uma produção agrícola existem diversos cuidados que devem ser tomados para que a plantação se desenvolva da melhor forma possível, garantindo o aumento da produção. Alguns exemplos de fatores que prejudicam a produção agrícola são: pragas, fungos, clima desfavorável, solo pobre em nutrientes e ervas daninhas. As ervas daninhas são geralmente chamadas de inço, infestante ou invasora. O problema das ervas daninhas é bastante relevante visto que são agentes que competem com o cultivo por água, nutrientes e luz solar. Uma plantação infestada de invasoras tem sua produtividade comprometida, causando grandes prejuízos ao produtor rural. A principal forma de controle de ervas daninhas é por meio de agroquímicos conhecidos como herbicidas. O controle das ervas daninhas começa antes mesmo da semeadura, onde o herbicida é jogado na área de plantação através de veículos pulverizadores na etapa conhecida como pré-emergente. Isso prepara a área para que o cultivo possa se desenvolver sem a concorrência das invasoras. O controle também é feito após a emergência da plantação, onde algumas invasoras crescem juntamente com o cultivo. Esta etapa é conhecida como controle pós-emergente. A aplicação de defensivos agrícolas e herbicidas realizada por veículos pulverizadores é normalmente realizada de maneira não-seletiva, ou seja, o defensivo é espalhado sobre toda a plantação (inclusive sobre o cultivo) e torna-se possível devido ao uso de cultivo transgênico que resiste ao herbicida, removendo apenas as invasoras.

[1] United States Department of Agriculture (Departamento de Agricultura dos Estados Unidos)

Apesar da evolução no controle de ervas daninhas, algumas destas adquirem resistência contra os herbicidas e se tornam mais difíceis de controlar. Um dos maiores exemplos desta classe é a erva daninha conhecida como a buva [2]. A buva é uma planta daninha que tem causado grandes preocupações e prejuízos aos agricultores dos Estados de Rio Grande do Sul, Paraná, Santa Catarina e Mato Grosso do Sul, grandes regiões produtoras de soja, milho e trigo. No entanto, esta infestante também tem aparecido como problema em várias outras regiões do Brasil, como o cerrado, se espalhando com muita velocidade.

A resistência desenvolvida por estas invasoras, além da alta capacidade de proliferação, indicam que é necessário o uso de herbicidas em grande quantidade, causando prejuízos econômicos e exposição excessiva do consumidor aos agrotóxicos, que podem causar uma série de doenças — dependendo do produto que foi utilizado, do tempo de exposição e quantidade de produto absorvido pelo organismo. Por estes motivos, técnicas de pulverização seletiva são cada vez mais interessantes para o setor agrícola.

Justificado pela situação exposta, este trabalho apresenta o desenvolvimento de um sistema de detecção de ervas daninhas — sem a distinção entre espécies por classes — em uma plantação, por meio de visão computacional e com o uso de detecção de objetos e redes neurais convolucionais (CNNs)[3], assim como a avaliação do sistema. A avaliação deste sistema é realizada por métricas utilizadas na área de visão computacional, assim como a velocidade de detecção dos modelos analisados. Algoritmos estado da arte em detecção de objetos, como o EfficientDet (TAN; PANG; LE, 2019) e YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020), foram utilizados como base de estudos e comparação para esta aplicação. Além disso, é realizada a avaliação do uso dos modelos em um sistema computacional de baixo custo, o Raspberry PI 4. O principal motivo para esta avaliação é a análise da viabilidade de uso do equipamento para um sistema de pulverização seletiva. O sistema implementado poderá ser utilizado em conjunto com um sistema de pulverização seletiva, a ser instalado em pulverizadores agrícolas para realizar a aplicação localizada de agroquímicos.

[2] conyza bonariensis
[3] Convolutional Neural Networks

2 Fundamentação Teórica e Revisão Bibliográfica

2.1 Aprendizado de máquina

O aprendizado de máquinas — ou *machine learning* — é a ciência e técnica utilizada para melhorar um programa de computador, por meio de dados ou da experiência adquirida pelo próprio programa, sem a necessidade deste ser explicitamente programado, ou seja, automaticamente (SAMUEL, 1959).

Segundo Mitchell, "um programa de computador é dito aprender por experiência E, com respeito a alguma tarefa T e métrica de avaliação P, se sua avaliação em T, medida por P, melhora com a experiência E" (MITCHELL, 1997).

A literatura relacionada ao aprendizado de máquinas separa o campo em quatro diferentes tipos: aprendizado supervisionado, aprendizado não-supervisionado, aprendizado semi-supervisionado e aprendizagem por reforço. Os aprendizados de máquinas supervisionado e não-supervisionado estão relacionados ao uso de dados para a definição de modelos capazes de prever ou classificar padrões ocultos. A principal diferença entre os dois tipos de aprendizado é a maneira como os dados são fornecidos ao programa, onde o aprendizado supervisionado necessita um "professor", normalmente um rótulo indicando o comportamento dos dados, permitindo assim que o programa encontre sua própria maneira de relacionar os dados. O aprendizado não-supervisionado é tipicamente utilizado para encontrar padrões escondidos em um conjunto de dados, sem receber informações prévias sobre os dados. O aprendizado semi-supervisionado combina as técnicas de aprendizado não-supervisionado com aprendizado supervisionado de maneira a atacar o problema da necessidade de rótulos para grandes quantidades de dados no aprendizado supervisionado e tornar o aprendizado não-supervisionado mais eficiente por meio do uso de conjuntos de dados parcialmente rotulados. Já o aprendizado por reforço é completamente diferente dos outros tipos de aprendizado. Conforme descrito por Géron, "o sistema de aprendizado, ou agente, observa o ambiente, seleciona, realiza ações e, então, recebe recompensas ou penalidades em retorno" (GéRON, 2017). Resumidamente, é um processo de aprendizado por tentativa e erro onde o sistema deve encontrar a melhor estratégia para aprimorar-se.

2.2 Redes Neurais Artificiais

O *deep learning*[1] pode ser entendido como um subcampo do aprendizado de máquina que se preocupa com o treinamento de redes neurais artificiais (RNAs) com muitas camadas

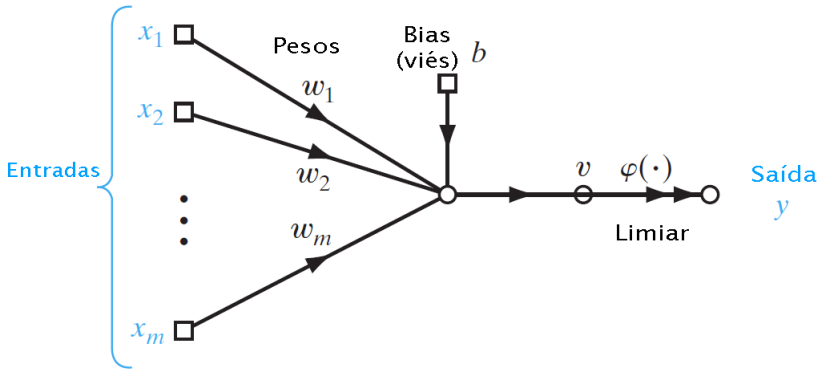
[1] aprendizagem profunda, em português

de modo eficiente e vem ganhando muita atenção recentemente (RASCHKA; MIRJALILI, 2019).

O termo rede neural artifical (RNA) refere-se a algoritmos computacionais que procuram representar o processamento de informações da mesma forma que em sistemas biológicos, que são feitos de cadeias muito complexas de neurônios interconectados. Historicamente, dois tipos de grupos de pesquisadores trabalharam com RNAs, um grupo motivado pela utilização dos modelos de RNAs desenvolvidos para obter maior compreensão do processo de aprendizagem biológico e outro grupo motivado em obter algoritmos capazes de realizar tarefas inéditas (MITCHELL, 1997).

O perceptron foi a primeira rede neural descrita algoritmicamente. O perceptron é a forma mais básica de rede neural, usada para classificar padrões linearmente separáveis. O algoritmo consiste de um único neurônio com pesos e vieses ajustáveis, conforme Figura 1, capaz de reduzir o erro encontrado em cada iteração a partir da atualização dos pesos (HAYKIN, 2009).

Figura 1 – Perceptron.



Fonte: Adaptado de (HAYKIN, 2009).

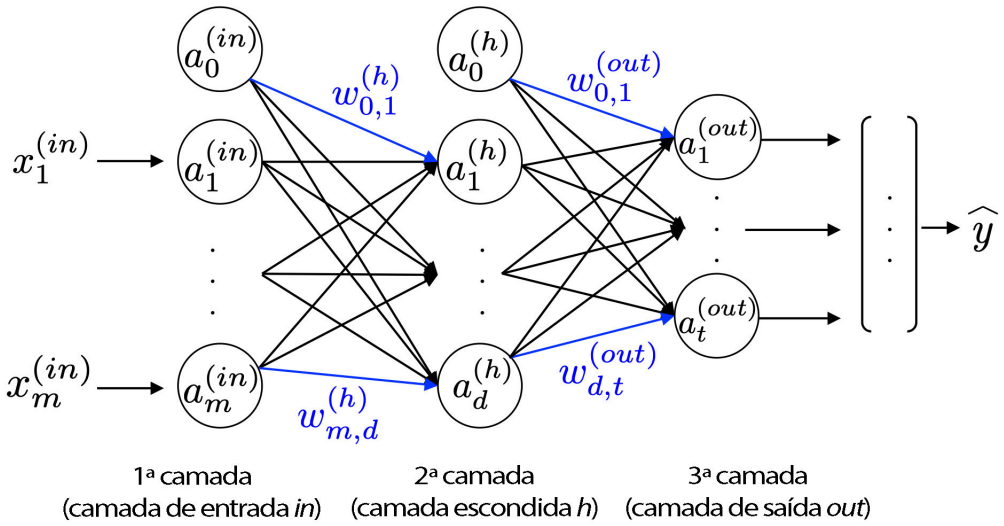
Segundo Mitchell, RNAs fornecem um método geral e prático para aprender funções com valores reais, valores discretos e com valores vetoriais a partir de exemplos. Uma das principais vantagens do uso de RNAs é a robustez a erros nos dados de treinamento (MITCHELL, 1997).

O exemplo mais típico de RNA é a rede neural *feedforward*, também conhecida por rede perceptron multicamadas (MLP)[2], composta por perceptrons totalmente conectados. O objetivo da rede *feedforward* é se aproximar para uma função f^* (GOODFELLOW; BENGIO; COURVILLE, 2016). Esse tipo de rede apareceu quando os pesquisadores descobriram as limitações teóricas que muitos modelos de camada única de RNAs tinham,

[2] *multi layer perceptron*

como apenas sendo capazes de aproximar funções que eram linearmente separáveis. Um exemplo de rede MLP com uma camada de entrada, uma camada escondida e uma camada de saída está apresentado na Figura 2. Caso uma rede MLP possua mais de uma camada escondida, esta rede passa a ser chamada de rede neural artificial profunda[3] (RASCHKA; MIRJALILI, 2019).

Figura 2 – Rede neural *feedforward* com uma camada.



Fonte: Adaptado de (RASCHKA; MIRJALILI, 2019).

O aumento de poder computacional e geração de dados nos últimos anos permitiram o treinamento de redes neurais artificiais em um tempo cada vez menor, e o desenvolvimento de novos algoritmos e técnicas capazes de resolver problemas complexos. Contudo, segundo Goodfellow, "a menos que novas tecnologias permitam um escalonamento mais rápido, as redes neurais artificiais não terão o mesmo número de neurônios que o cérebro humano até pelo menos 2050" (GOODFELLOW; BENGIO; COURVILLE, 2016). Isto significa que ainda há muito potencial a ser descoberto no campo do *deep learning*.

2.3 Redes Neurais Convolucionais

Uma rede neural convolucional (também conhecida por CNN, *convolutional neural network*) (LECUN *et al.*, 1989) é um tipo especializado de rede neural usada para o processamento dados que apresentam topologias similares a grades, como imagens, áudio, texto e vídeo. Segundo Goodfellow, "CNNs são simplesmente redes neurais que utilizam a operação de convolução na tarefa de extração de caraterísticas ao longo das camadas de uma rede" (GOODFELLOW; BENGIO; COURVILLE, 2016).

[3] *deep artificial neural network*

A convolução é uma operação matemática aplicada na análise de sinais e sistemas que permite verificar ou testar os comportamentos de sinais em domínios distintos. Uma importante propriedade diz que uma operação de convolução no domínio do tempo equivale a uma multiplicação no domínio da frequência. Assim, sabemos que o espectro de um sinal pode ser multiplicado por um filtro seletor de frequências. Desde que no domínio do tempo façamos a operação de convolução entre o sinal e a resposta ao impulso desse filtro. (INGLE; PROAKIS, 2012)

Em um sistema de computador, onde as operações são realizadas por meio de um sistema binário, todas as operações são realizadas de forma discreta. A definição de uma convolução no domínio discreto é dada pela Equação 1.

$$s[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{1}$$

Na terminologia de redes convolucionais, o primeiro sinal é o argumento de entrada e o segundo argumento é chamado de filtro ou *kernel*. O resultado da operação de convolução em CNNs é referido por mapa de características (*feature map*, em inglês). Em aplicações de aprendizado de máquina, a entrada é normalmente uma matriz multidimensional de dados, e o filtro uma matriz multidimensional de parâmetros que são adaptados pelo algoritmo. Estes argumentos são tipicamente referidos como tensores (GOODFELLOW; BENGIO; COURVILLE, 2016).

Para o exemplo de uma imagem I e filtro K bidimensionais no domínio discreto, uma única operação S de convolução para o ponto (i, j) é dada aplicando o filtro em um segmento de sobreposição da imagem (m, n), conforme Equação 2.

$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(m,n)K(i-m, j-n) \tag{2}$$

Ao invés de implementar a convolução, muitas bibliotecas de aprendizado de máquina utilizam a correlação cruzada[4], que realiza a mesma função que a convolução, porém sem inverter o *kernel*. A operação de correlação cruzada está apresentada na Equação 3. No contexto do aprendizado de máquina ambas operações são referidas por convolução. (GOODFELLOW; BENGIO; COURVILLE, 2016)

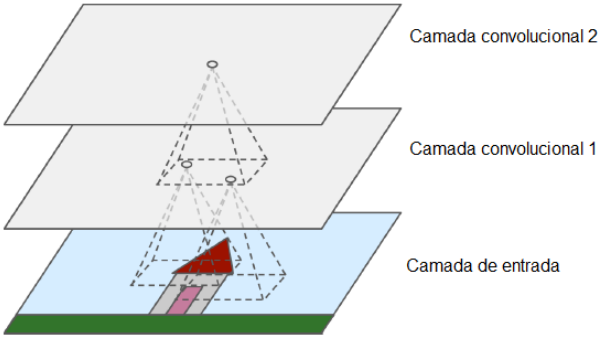
$$S(i,j) = (I * K)(i,j) = \sum_{m} \sum_{n} I(i+m, j+n)K(m,n) \tag{3}$$

As CNNs são capazes de aprender as características de dados que são mais relevantes para a tarefa selecionada. As camadas convolucionais tendem a extrair estas características, que então são passadas à camadas totalmente conectadas. A extração destas características

[4] *cross-correlation*

se dá por meio de campos receptivos (uma analogia ao campo receptivo do olho humano). Os pixels das camadas convolucionais não estão completamente conectados com a camada de entrada, como no caso dos perceptrons de múltiplas camadas, eles estão conectados com o campo receptivo que é deslizado sobre a imagem, um exemplo visual deste processo está apresentado na Figura 3 de maneira simplificada. Isto permite que a rede se concentre nas características de baixo nível, como por exemplo vértices ou bordas, reduzindo a dimensão dos dados analisados e consequentemente o custo computacional.

Figura 3 – Campos receptivos retangulares sobre uma imagem.

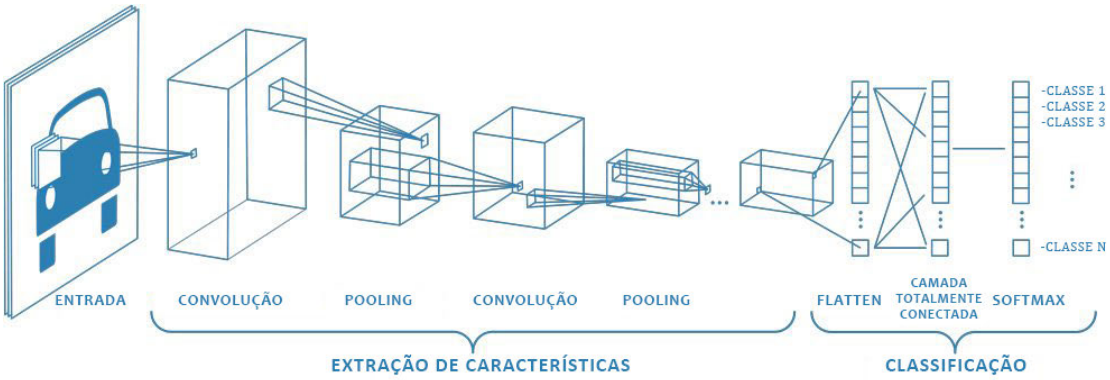


Fonte: Adaptado de (GéRON, 2017)

Existem muitas vantagens em usar um sistema convolucional para o processamento de imagens. Camadas convolucionais tendem a apresentar bom desempenho em tarefas relacionadas a análise de imagens, pois todas as unidades em um mapa de recursos compartilham os mesmos valores de peso. Isso significa que CNNs tem melhor escalabilidade para entradas com um grande volume de informações do que MLPs, reduzindo o número de parâmetros do sistema.

Tipicamente, arquiteturas CNN empilham algumas camadas convolucionais, seguidas de camadas de subamostragem (*pooling*), seguidas de mais camadas convolucionais e subamostragem, que então são passadas a camadas totalmente conectadas, que por sua vez, realizam a previsão. A imagem fica cada vez menor à medida que avança pela rede, mas também fica cada vez mais profunda (ou seja, com mais mapas de recursos) (GéRON, 2017). A Figura 4 apresenta uma arquitetura típica de CNN, onde uma imagem de entrada é fornecida à rede que extrai as características e então realiza a classificação da imagem.

Figura 4 – Arquitetura típica de CNN.



Fonte: Adaptado de (MATHWORKS, 2018)

2.4 ConvNets — um breve histórico

2.4.1 Bancos de dados

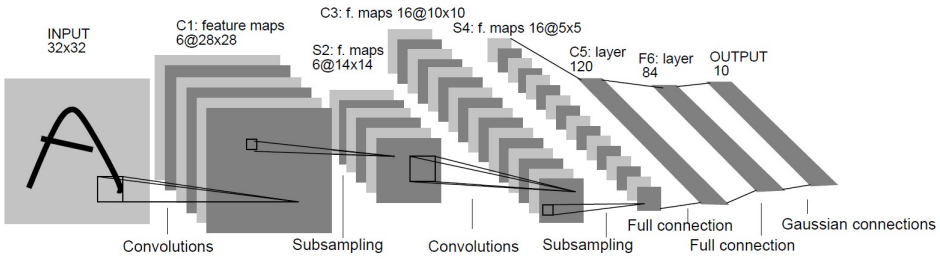
O desenvolvimento no campo da visão computacional vem sendo impulsionado por bancos de dados e suas respectivas competições como o *ImageNet Large Scale Visual Recognition Challenge (ILSVRV)* (RUSSAKOVSKY *et al.*, 2015), *PASCAL VOC* (EVERINGHAM *et al.*, 2012), *Google Open Images Dataset* (KUZNETSOVA *et al.*, 2020) e *COCO Dataset* (LIN *et al.*, 2014). Estes bancos de dados são grandes catalisadores de conhecimento, proporcionando aos pesquisadores uma plataforma com dados disponíveis abertamente e oportunidade de apresentação dos trabalhos. Nestas competições foram introduzidos modelos referência na área, como o AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) e VGG (SIMONYAN; ZISSERMAN, 2014). Além disso os bancos de dados são utilizados para o pré-treinamento de detectores de objetos, que é necessário para o modelo realizar a definição de o que é um objeto.

2.4.2 LeNet

Em *Backpropagation Applied to Handwritten Zip Code Recognition* (LECUN *et al.*, 1989) foi aplicado o primeiro o algoritmo de retro propagação (*backpropagation*) à aplicações práticas. Este algoritmo foi utilizado para treinar uma rede neural convolucional para ler números escritos à mão e foi aplicado com sucesso na identificação de números de CEP em cartas manuscritas. Em *Gradient-based learning applied to document recognition* (LECUN *et al.*, 1998) foram revisados vários métodos de reconhecimento de caracteres manuscritos em papel. Os métodos foram comparados e os resultados mostraram que a rede convolucional superou todos as outras técnicas, chegando a ser aplicado por vários bancos para reconhecer números escritos à mão em cheques em imagens de 32x32 pixels. Esta arquitetura proposta foi chamada de LeNet-5, apresentada no diagrama da Figura

5. A arquitetura LeNet-5 consiste em dois conjuntos de camadas convolucionais e da média de *pooling*, seguidas por uma camada convolucional achatada, depois duas camadas totalmente conectadas e, finalmente, um classificador *softmax*. A arquitetura é direta e simples de entender, por isso é um dos principais modelos utilizados na aprendizagem de redes neurais convolucionais.

Figura 5 – Arquitetura LeNet-5



Fonte: Adaptado de (LECUN *et al.*, 1989).

2.4.3 AlexNet

A AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) é uma CNN introduzida no ILSVRV-2012 (RUSSAKOVSKY *et al.*, 2015), ficando em primeiro lugar, com uma taxa de erro de 15,3%, muito inferior ao segundo colocado que teve uma taxa de erro de 26,2%. A AlexNet introduziu alguns conceitos fundamentais que vieram a ser aplicados em futuros trabalhos de concepção de CNNs, revivendo o interesse pelas redes neurais convolucionais no campo da visão computacional. Segundo o autor, a profundidade de camadas da rede foi essencial pela sua performance. Apesar do alto custo computacional, o uso de múltiplas GPUs tornou viável o treinamento da rede. Entre os conceitos fundamentais introduzidos pela AlexNet estão: empilhamento de camadas convolucionais, uso de ReLU (*Rectified Linear Units*) como função de ativação, treinamento com múltiplas GPUs em paralelo e divisão da carga computacional entre GPUs, *pooling overlapping*, aumento de dados (*data augmentation*) e camadas *dropout* (HINTON *et al.*, 2012).

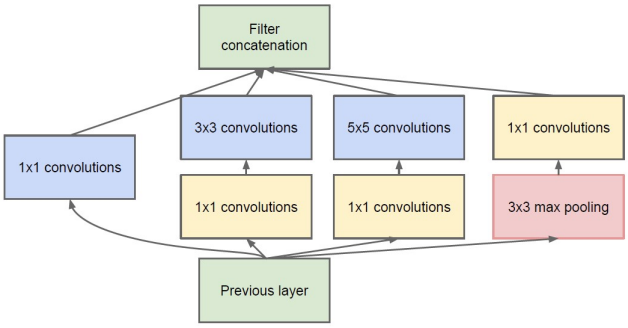
2.4.4 VGGNet e GoogLeNet

Conforme sugerido pelo trabalho desenvolvido em AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), o número de camadas de uma CNN poderia melhorar os resultados obtido por um modelo. Em VGG (SIMONYAN; ZISSERMAN, 2014), os autores investigam o efeito do aumento do número de camadas em um modelo. Os autores, inspirados em AlexNet, aumentaram o número de camadas convolucionais e reduziram o tamanho dos filtros convolucionais para 3x3. O resultado foi um modelo estado da arte, atingindo a primeira colocação (quesito "*Classification+localization*") no ILSVRV-2014

(RUSSAKOVSKY *et al.*, 2015) e a comprovação de que o modelo com maior número de camadas também apresentava desempenho estado da arte em outros bancos de dados.

O GoogLeNet (SZEGEDY *et al.*, 2014), desenvolvido para o ILSVRV-2014 (participando em paralelo com VGGNet), é um modelo de rede convolucional profunda de 22 camadas (ou 27 camadas, contabilizando as camadas de *pooling)*, desenvolvida por pesquisadores da empresa Google, capaz de resolver problemas de classificação de imagens e também de realizar detecção de objetos. A principal característica da GoogLeNet é o uso de blocos que realizam convoluções com diferentes tamanhos de filtros, chamada de *Inception* (Figura 6), empilhadas uma sobre as outras, reduzindo o custo computacional da extração de características e permitindo a implementação de um maior número de camadas. O GoogleNet atingiu primeira colocação em múltiplas categorias do ILSVRV-2014, inclusive em detecção de objetos, tornando-se modelo referência na área. Segundo Szegedy, "a maneira mais direta de melhorar o desempenho de redes neurais profundas é aumentar seu tamanho" (SZEGEDY *et al.*, 2014). O que pode ser observado na implementação de outros modelos de CNNs com alta performance.

Figura 6 – Módulo Inception

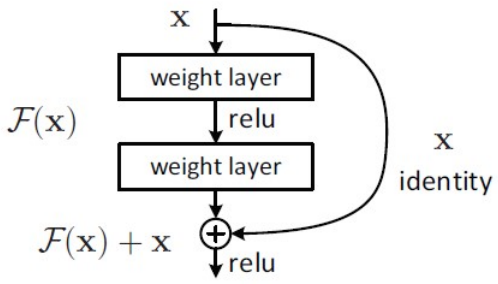


Fonte: (SZEGEDY *et al.*, 2014).

2.4.5 ResNet

O trabalho *Deep Residual Learning for Image Recognition* (HE *et al.*, 2015a), ganhador do ILSVRC-2015 (RUSSAKOVSKY *et al.*, 2015), parte do princípio que redes convolucionais mais profundas são mais difíceis de treinar. Utilizando o conceito de aprendizado residual, o trabalho demonstrou que é possível treinar redes 8 vezes maiores que a VGGNet, sem aumento de complexidade e custo computacional, além de apresentar evidência que redes com aumento da profundidade são mais fáceis de ser otimizadas e apresentam melhor precisão. Os blocos de aprendizado residual apresentam conexões com "atalhos", que simplesmente realizam o mapeamento de identidade, adicionando suas saídas às saídas das camadas empilhadas, conforme Figura 7. Desta forma, caso a função alvo está próxima à função identidade durante a inicialização do modelo, a velocidade de treino aumenta consideravelmente (GéRON, 2017).

Figura 7 – Bloco residual



Fonte: (HE *et al.*, 2015a).

2.5 Detecção de Objetos

2.5.1 Introdução

A tarefa de localizar e classificar objetos em uma imagem ou vídeo, por meio de visão computacional, é chamada de detecção de objetos. As técnicas de detecção de objetos mais importantes utilizam aprendizado de máquina, e podem ser divididas em dois grupos: as que usam redes neurais e as que não usam. Entre as que não usam redes neurais para detecção destacam-se os métodos: *Viola-Jones object detection framework* (VIOLA; JONES, 2001), *scale-invariant feature transform*(SIFT) (LOWE, 1999), *histogram oriented gradients*(HOG)(DALAL; TRIGGS, 2005) e *optical flow* (TURAGA; CHELLAPPA; VEERARAGHAVAN, 2010), entre outras.

Estes modelos, apesar de importantes para o desenvolvimento do estudo de visão computacional, cada vez mais estão sendo substituídos pelo uso de detectores que usam redes neurais. Os métodos com redes neurais são capazes de identificar as características automaticamente, após treinamento prévio, enquanto os outros requerem a definição das características a serem encontradas.

Entre os detectores de objetos com redes neurais artificiais, destacam-se os que são implementados com o uso de CNNs. Alguns dos principais detectores são: *Region Proposal CNN* (R-CNN) (GIRSHICK *et al.*, 2014), *Fast R-CNN* (GIRSHICK, 2015), *Faster R-CNN* (REN *et al.*, 2015), *Single Shot MultiBox Detector* (SSD) (LIU *et al.*, 2016), *You Only Look Once* (YOLO) (REDMON *et al.*, 2016), entre outros.

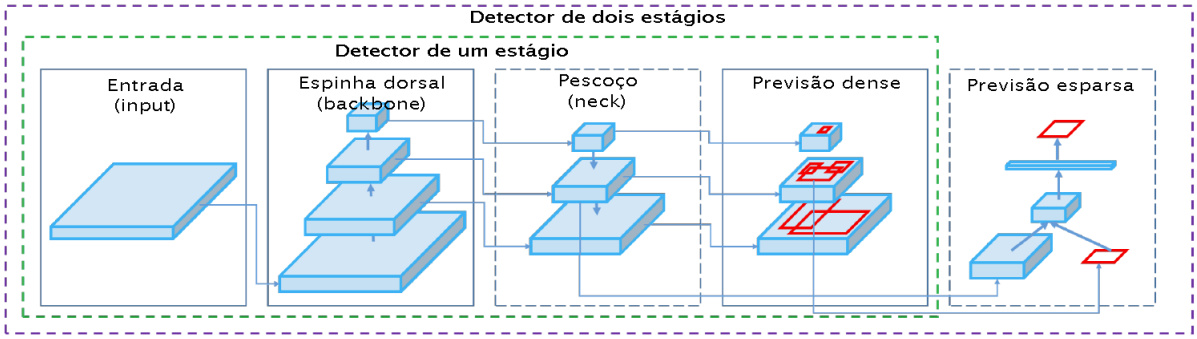
2.5.2 Construção dos modelos de detectores

Um detector de objetos é geralmente composto por duas partes: uma espinha dorsal (*backbone*) e uma cabeça (*head*). As espinhas dorsais utilizadas atualmente, são tipicamente redes convolucionais profundas, como a VGGNet ou ResNet, pré-treinados em um banco de dados como o Coco Dataset (LIN *et al.*, 2014) ou ImageNet (RUSSAKOVSKY *et al.*, 2015).

Já as cabeças, são categorizadas de duas formas, como detectores de objeto de um estágio, ou de dois estágios. Os modelos mais representativos para os detectores de dois estágios são as CNNs baseadas em regiões de interesse (R-CNNs), enquanto que, para os modelos de um estágio, os modelos mais representativos são o YOLO (REDMON *et al.*, 2015), SSD (LIU *et al.*, 2016) e RetinaNet (LIN *et al.*, 2017). Alguns trabalhos mais recentes começam a incluir algumas camadas entre as espinhas dorsais e a cabeça, recebendo o nome de pescoço (*neck*). O pescoço é utilizado para coletar mapas de características de diferentes estágios. Completando o detector, são adicionadas camadas totalmente conectadas, no caso do detector de um estágio, e mais camadas esparsas, no caso do detector de dois estágios. Estas camadas são responsáveis por realizar as previsões finais de classificação e de caixas delimitadoras (*bounding boxes*) do modelo. (BOCHKOVSKIY; WANG; LIAO, 2020)

Um diagrama originalmente apresentado em *YOLOv4* (BOCHKOVSKIY; WANG; LIAO, 2020) descreve os componentes de detectores de objetos de maneira simples, conforme Figura 8.

Figura 8 – Configuração de detectores de objetos



Fonte: Adaptado de (BOCHKOVSKIY; WANG; LIAO, 2020)

2.5.3 Saco de brindes e saco de especiais

Um detector pode ser previamente treinado offline, a fim de pré-definir os pesos do modelo. É interessante o uso de técnicas que possam tornar o modelo mais preciso nessa etapa de treinamento de maneira a não aumentar o custo computacional da inferência dos resultados. Estas técnicas são chamadas de saco de brindes (*bag of freebies*). As técnicas saco de brindes buscam reduzir o viés do modelo treinado, evitando o sobre-ajuste e melhorando a generalização do modelo. Algumas das principais técnicas saco de brindes são o aumento de dados, *dropout* (SRIVASTAVA *et al.*, 2014), *focal loss* (LIN *et al.*, 2017), *random erasing* (ZHONG *et al.*, 2017), entre outras.

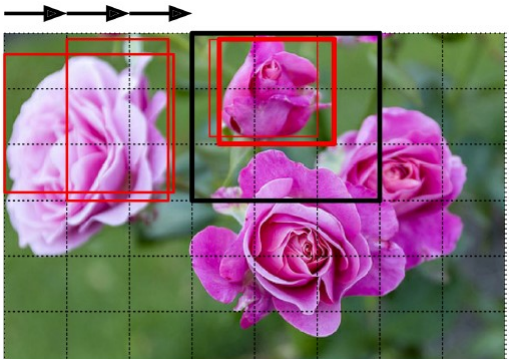
As técnicas chamadas de saco de especiais (*bag of specials*) podem ser aplicadas para aprimorar um modelo após o treinamento, durante a inferência do resultado, a contrapartida

de um custo computacional mais elevado. Este custo computacional, principalmente na aplicação de detector de objetos à vídeos, pode tornar o modelo inviável para o tipo de aplicação, portanto estas melhoras devem ter um custo computacional baixo e melhorar significativamente a precisão de inferência do detector de objetos. Em YOLOv3 (REDMON; FARHADI, 2018), foi aprimorado o módulo SPP (HE *et al.*, 2015b) para a concatenação das saídas de *max-pooling*. Com a adição deste módulo SPP, o YOLOv3-608 conseguiu atingir uma melhora de 2,7% em AP@0.5 no banco de dados MS COCO, a um custo computacional extra de 0,5%, justificando a sua utilização. Algumas destas técnicas podem ser utilizadas para aprimorar certos atributos do modelo, como o aumento do campo receptivo. Entre estas técnicas destacam-se: *Spatial Pyramid Pooling*(SPP) (HE *et al.*, 2015b), *Atrous Spatial Pyramid Pooling* (ASPP) (CHEN *et al.*, 2018) e *Receptive Field Block* (RFB) (LIU; HUANG; WANG, 2017).

2.5.4 Janela deslizante

Uma abordagem muito comum pelos primeiros trabalhos de detecção de objetos era escolher uma rede treinada para classificar um tipo de objeto em uma imagem e deslizar este classificador ao longo da imagem onde se deseja fazer a detecção. Este método é conhecido como janela deslizante ou *sliding window*. A Figura 9 exemplifica este processo.

Figura 9 – Detecção por janela deslizante



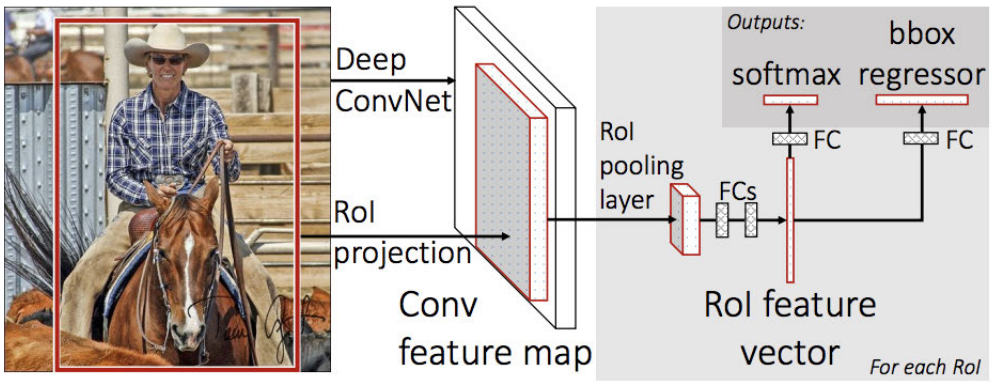
Fonte: (GéRON, 2017)

Como é possível observar no exemplo, a caixa preta (retângulo 3 × 3) representa a região da imagem analisada, que passa sobre uma imagem dividida em partes (um grid 6 × 8). Ao passar pela a imagem, a CNN identifica a flor e realiza a detecção, ao deslizar para a próxima seção da imagem, a CNN pode fazer a detecção do mesmo objeto, criando múltiplas caixas para o mesmo objeto. A técnica é simples de se compreender, entretanto ela pode se tornar extremamente custosa computacionalmente com múltiplas classes e tamanho das imagens, além de ser necessário o pós-processamento do resultado devido as múltiplas detecções do mesmo objeto (GéRON, 2017).

2.5.5 CNNs baseadas em regiões de interesse (R-CNNs)

Dada uma imagem de entrada, uma R-CNN aplica um ou mais algoritmos especializados em extrair regiões de interesse (ROI)[5], onde cada região de interesse pode ser definida por uma caixa delimitadora (*bounding box*). Entre os principais algoritmos utilizados para esta tarefa estão o busca seletiva, *objectness* e caixas de âncoras. Estes algoritmos buscam encontrar objetos na imagem, por meio da diferenciação entre algo que é objeto e o plano de fundo da imagem. Após a definição das regiões de interesse da imagem, um classificador é aplicado nas regiões encontradas para determinar a probabilidade de a ROI conter um objeto e a classe de objeto encontrado. A Figura 10 descreve de maneira simplificada este processo.

Figura 10 – Fast R-CNN



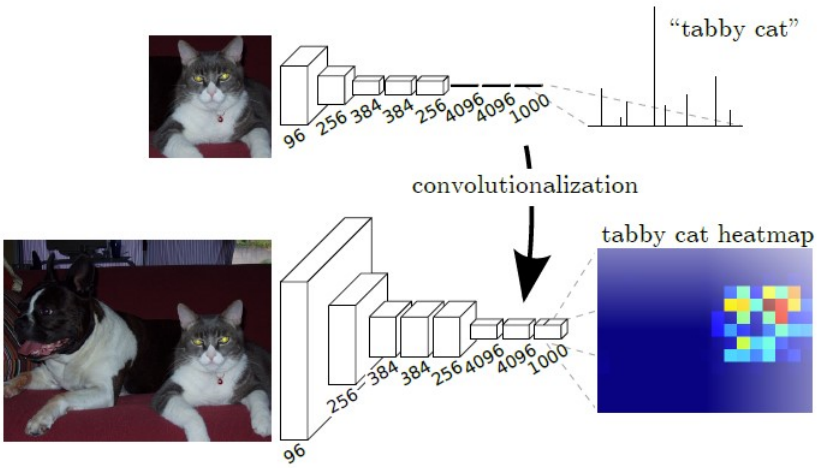
Fonte: (GIRSHICK, 2015)

2.5.6 Redes totalmente convolucionais (FCN)

O trabalho apresentado por (LONG; SHELHAMER; DARRELL, 2014) introduziu o conceito de rede totalmente convolucional (*fully convolutional network*)(FCN), buscando a construção de "redes totalmente convolucionais que tomam uma entrada arbitrária e produzem uma saída de tamanho correspondente com aprendizado e inferência eficientes". As FCN substituem as camadas totalmente conectadas por camadas convolucionais, permitindo o fornecimento de uma entrada de qualquer tamanho, em oposição à rede totalmente conectada, que necessita uma entrada de tamanho específico. (GéRON, 2017). O resultado de saída desta rede é um mapa de características de pixels, conforme exemplo da Figura 11, que compara a saída de uma rede totalmente conectada com substituição pela rede FCN. Este tipo de rede permite o treinamento pixel a pixel da rede, permitindo a solução de problemas de segmentação de objetos e possibilitando a reformulação do problema de detecção de objetos para um problema de regressão, conforme apresentado no detector de objetos YOLO (REDMON *et al.*, 2015).

[5] do inglês, *regions of interest*

Figura 11 – Comparação da saída da rede totalmente conectada com FCN



Fonte: (LONG; SHELHAMER; DARRELL, 2014)

2.6 Métricas de avaliação

Em um sistema de detecção de objetos existem dois tipos de respostas, a posição e a classificação das imagens. Estas respostas são independentes e avaliadas por intervalos de confiança. Segundo Padilla, existe uma certa falta de consenso nas métricas de avaliação utilizadas em problemas de detecção de objetos apresentados na literatura e competições onde são avaliados os modelos (Padilla; Netto; da Silva, 2020). Nas competições, é normalmente utilizada a relação de precisão e *recall* com a precisão média (AP)[6] como métrica de classificação. A combinação destas métricas são usadas para a obtenção de outras métricas, como a média da precisão média, conhecida por mAP[7]. Já a métrica utilizada para avaliação de posição é a interseção sobre união (IOU)[8], que avalia a relação da área sobreposta entre a caixa do objeto obtida pelo modelo e sua respectiva caixa rotulada, e a área da união entre as caixas. Como os banco de dados de competições são a base para os mais recentes trabalhos, onde os trabalhos também são avaliados, é interessante manter as mesmas métricas estabelecidas para avaliação e comparação dos modelos. Alguns exemplos de métricas de avaliação usado por modelos em bancos de dados estão apresentadas na Tabela 1.

[6] *average precision*
[7] *mean average precision*
[8] *intersection over union*

Tabela 1 – Métricas utilizadas na avaliação de detectores de objetos.

Modelo	Banco de dados avaliado	Métricas utilizadas
EfficientDet	COCO	AP@[.5:.05:.95], AP@.50, AP@.75
Fast R-CNN	VOC 2007,2010,2012	AP, mAP (IOU=.50)
Faster R-CNN	COCO	AP@[.5:.05:.95]; AP@.50
RetinaNet	COCO	AP@[.5:.05:.95]; AP@.50, AP@.75, APS, APM, APL
SSD	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; APS; APM; APL; AR1; AR10; AR100; ARS; ARM; ARL
Yolo v2	VOC 2007,2012	AP; mAP (IOU=.50)
Yolo v4	COCO	AP@[.5:.05:.95]; AP@.50; AP@.75; APS; APM; APL
Yolo v5	COCO	AP@[.5:.05:.95]; AP@.50

Fonte: Adaptado de (PADILLA *et al.*, 2021)

A precisão é a habilidade do modelo de identificar somente os objetos de interesse. É a porcentagem das previsões corretas positivas, dada pela Equação 4.

$$Precisão = \frac{TP}{TP + FP} = \frac{TP}{todas\ as\ detecções} \tag{4}$$

Onde *TP* é o número de verdadeiros positivos e FP é o número de falsos positivos obtidos na avaliação do modelo.

O *recall*[9] é a habilidade em encontrar todos os casos relevantes (todas as caixas delimitadoras). É a porcentagem de verdadeiros positivos detectados entre todas as caixas delimitadoras, dado pela Equação 5.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{todas\ caixas\ delimitadoras} \tag{5}$$

Onde *TP* é o número de verdadeiros positivos e FN é o número de falsos negativos obtidos na avaliação do modelo.

Um dos problemas em simplesmente usar as métricas de precisão e *recall* para avaliação do modelo, é o fato de que a curva da relação entre precisão e *recall* contém seções onde ambos os parâmetros aumentam ao mesmo tempo, especialmente com valores de *recall* baixos. Portanto, para obter uma melhor ideia da performance do modelo, é interessante computar a precisão para diferentes valores fixos de *recall*, isto é chamado de precisão média (AP) (Padilla; Netto; da Silva, 2020). A AP pode ser calculada pela área sob a curva de *Precisão x Recall*, normalmente calculada pela interpolação de 11 pontos

[9] também conhecido por revocação ou sensibilidade em português

entre 0 e 1, conforme Equação 6. Quando existem múltiplas classes na mesma imagem, é calculada a média da AP de cada classe, a média da precisão média (mAP).

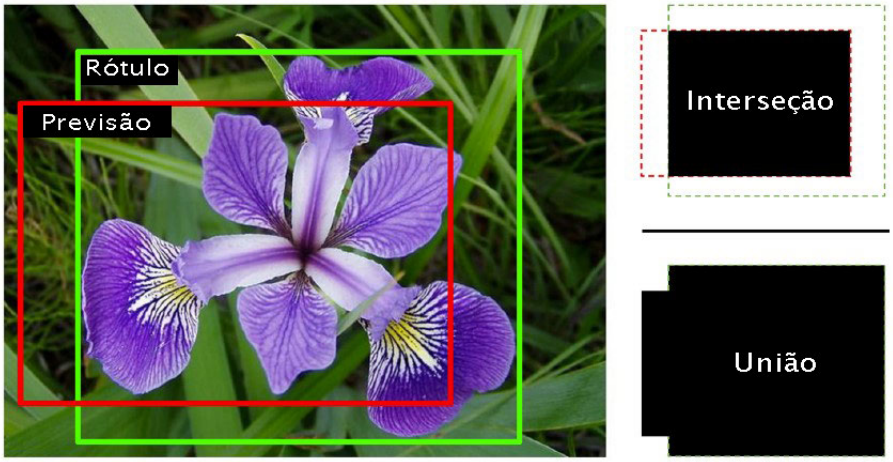
$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \rho_{interp(r)} \tag{6}$$

A métrica de avaliação da posição do retângulo delimitador do detetor de objetos, IOU, pode ser definida pela Equação 7. A Figura 12 apresenta um exemplo visual desta métrica.

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{7}$$

Onde B_p é a caixa delimitadora estimada e B_{gt} é a caixa delimitadora rotulada.

Figura 12 – Métrica interseção sobre união (IOU)



Fonte: Adaptado de (GéRON, 2017)

Uma abordagem é definir um valor de IOU como correto, somente se for superior a 0,5. Assim é possível combinar IOU e mAP para avaliar a classificação das imagens para diferentes valores de IOU, a notação para esta métrica combinada é geralmente definida como mAP@0.5 (Padilla; Netto; da Silva, 2020). Na competição PASCAL VOC, por exemplo, é utilizada esta notação.

2.7 YOLO — You Only Look Once

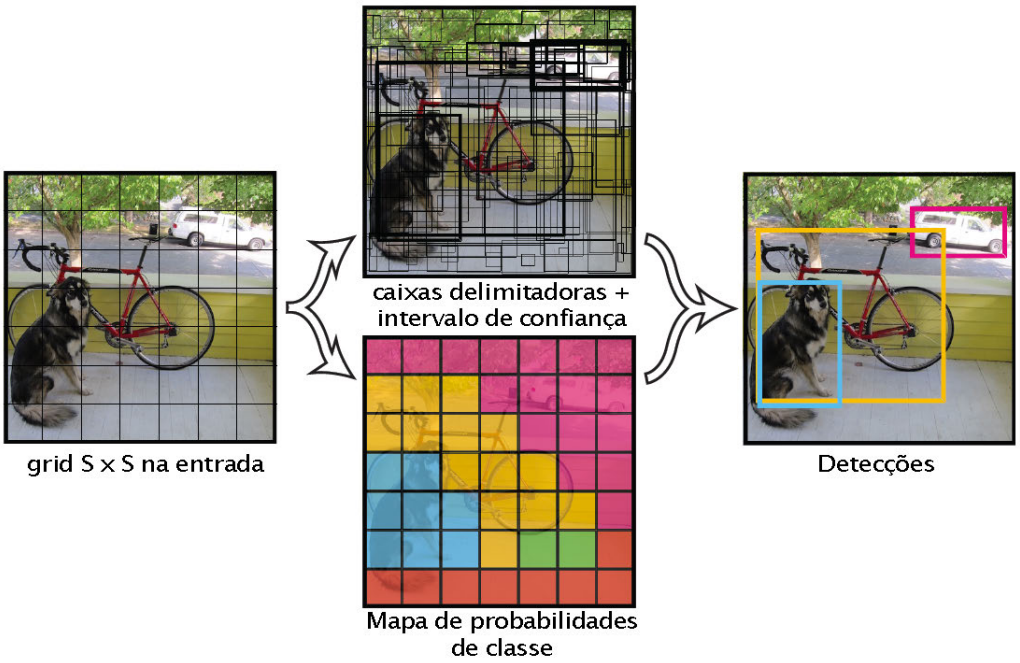
2.7.1 YOLO

YOLO (REDMON *et al.*, 2015) é uma arquitetura para detecção de objetos rápida e precisa, capaz de processamento de imagens em tempo real (para vídeos com taxas de quadros regulares). A arquitetura original *YOLO — You Only Look Once*, foi apresentada em 2015 e subsequentemente aprimorada nos trabalhos, YOLOv2 (REDMON; FARHADI,

2017), YOLOv3 (REDMON; FARHADI, 2018) e YOLOv4 (BOCHKOVSKIY; WANG; LIAO, 2020). Originalmente implementada na linguagem C++ e CUDA (REDMON, 2013–2016), o YOLO aborda a detecção de objetos como um problema de regressão para caixas delimitadoras e suas classes associadas, extraídas diretamente dos pixels da imagem, diferentemente das R-CNNs. A arquitetura YOLO foi inspirada pelo modelo de classificação de imagens GoogLeNet (SZEGEDY *et al.*, 2014), possuindo 24 camadas convolucionais e 2 camadas completamente conectadas. Diferentemente dos módulos de incepção utilizados em GoogLeNet, o YOLO utiliza redução 1x1 das camadas, seguidos por camas convolucionais 3x3, conforme proposto em *Focal Loss for Dense Object Detection* (LIN; CHEN; YAN, 2013).

Previamente ao primeiro modelo YOLO, os trabalhos apresentados buscavam reutilizar modelos de classificação para realizar a detecção. O objetivo do YOLO foi criar um modelo de detector de objetos por regressão diretamente dos pixels de uma imagem, capaz de processar imagens em tempo real. Para alcançar seus objetivos o YOLO unifica os componentes de detecção de objeto e utiliza as características detectadas para prever cada *bounding box*, realizando este trabalho de maneira simultânea para todas as *bounding boxes*. Para realizar isso, o YOLO divide a imagem de entrada em uma grade bidimensional de SxS e prevê B caixas delimitadoras, o intervalo de confiança para essas caixas e as probabilidades C da classe, conforme Figura 13.

Figura 13 – Divisão da imagem e previsão pelo YOLO



Fonte: (REDMON *et al.*, 2015)

Cada caixa delimitadora consiste em 5 previsões: x, y, w, h e intervalo de confiança, onde x e y representam o centro da caixa, e, w e h representam comprimento e altura

previstos. O intervalo de confiança representa o valor de IOU estimado pelo modelo. Além disso, cada célula da grade que contém uma *bounding box*, estima a probabilidade de classe C, $P(Class_i|Object)$ do objeto detectado. Somente é estimado um grupo de probabilidades por grade, independentemente do número de objetos detectados. No momento do teste, todas estimativas condicionais são multiplicadas e então é obtido a estimativa para a classe do objeto, conforme Equação 8.

$$P(Class_i|Object) * P(Object) * IOU_{pred}^{truth} = P(Class_i) * IOU_{pred}^{truth} \tag{8}$$

2.7.2 YOLOv2

Em YOLOv2 (REDMON; FARHADI, 2017) algumas técnicas foram adicionadas para melhorar a precisão do modelo. Normalização de lote, maior resolução do classificador e caixas de âncoras convolucionais foram aplicadas para aumentar o desempenho. Além disso, a espinha dorsal do modelo (*backbone*) responsável pela classificação das imagens foi aprimorada, passando a ser chamada de Darknet-19 (com 19 camadas convolucionais e 5 camadas de *max-pooling*). Então o modelo foi treinado utilizando os bancos de dados COCO (LIN *et al.*, 2014) e ImageNet (RUSSAKOVSKY *et al.*, 2015) simultaneamente, por meio de um modelo de classificação hierárquico nomeado *WordTree* (REDMON; FARHADI, 2017), capaz de prever probabilidades não só para uma classe, mas para todo o conjunto de sinônimos de classe. Isto potencializou a extração de características de um banco de dados para outro, diminuindo o viés e melhorando a precisão obtida pelo modelo durante o treinamento. As técnicas implementadas foram capazes de melhorar o mAP de 57,9% (YOLOv1) para 73,4% (YOLOv2) nos bancos de dados PASCAL VOC 2007+2012.

2.7.3 YOLOv3

Em YOLOv3 (REDMON; FARHADI, 2018), a principal mudança foi a implementação de um novo extrator de características, utilizando uma combinação do YOLOv2, Darknet-19 (REDMON; FARHADI, 2017) e ResNet (HE *et al.*, 2015a). O novo modelo passou a ser chamado de Darknet-53, devido ao uso de 53 camadas convolucionais no modelo, a estrutura do modelo está apresentada na Figura 14. As melhorias propostas pelo novo YOLO tornaram a precisão do modelo similar a outros modelos estado da arte, entretanto, com um tempo de inferência significativamente menor.

Figura 14 – Darknet-53

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fonte: (REDMON; FARHADI, 2018)

2.7.4 YOLOv4

Após preocupações com possíveis problemas éticos, Redmon decidiu parar de desenvolver o YOLO, que passou a ser aprimorado por (BOCHKOVSKIY; WANG; LIAO, 2020) desenvolvendo o YOLOv4 e seus variantes. Os objetivos do modelo YOLO em YOLOv4 continuaram sendo os mesmos, um modelo rápido, preciso e capaz de ser rapidamente treinado.

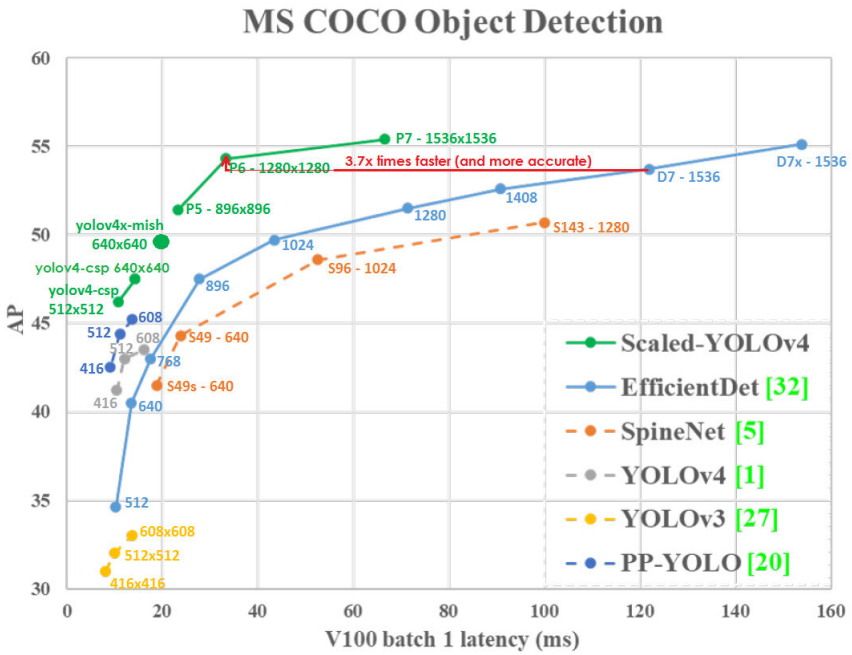
A principal alteração no modelo YOLOv4 foi a melhoria da espinha dorsal do modelo, Darknet-53, aprimorada com as técnicas utilizadas em *CSPNet* (WANG *et al.*, 2019), passando a se chamar de CSPDarknet53. Adicionalmente, foram feitas melhorias para o aumento do campo receptivo da CSPDarknet53, sendo combinado um bloco SPP (HE *et al.*, 2015b) e utilizado PANet (*Path Aggregation Network*) (LIU *et al.*, 2018) como método agregador de parâmetros (o pescoço do modelo). A cabeça do modelo continua sendo o detector YOLOv3. Outras melhorias inspiradas pelos trabalhos mais recentes do campo de visão computacional foram implementadas, como regularização *DropBlock* (GHIASI; LIN; LE, 2018), aumento de dados com mosaico (BOCHKOVSKIY; WANG; LIAO, 2020), função de ativação *Mish* (MISRA, 2019), entre outras (BOCHKOVSKIY; WANG; LIAO, 2020).

O trabalho *Scaled-YOLOv4* (WANG; BOCHKOVSKIY; LIAO, 2020) demonstra a escalabilidade do uso de redes neurais baseadas em CSP (WANG *et al.*, 2019). A rede *Scaled-YOLOv4* modifica a profundidade, comprimento, resolução e até a estrutura da rede baseada em CSP, demonstrando desempenho de estado da arte com 55.5% AP (73.4%

AP50) para o COCO Dataset (LIN *et al.*, 2014) com uma velocidade de inferência de aproximadamente 16 FPS com uma GPU Tesla V100.

A Figura 15 demonstra o desempenho dos modelos YOLOv4, comparados com outros modelos estado da arte.

Figura 15 – Desempenho do YOLOv4

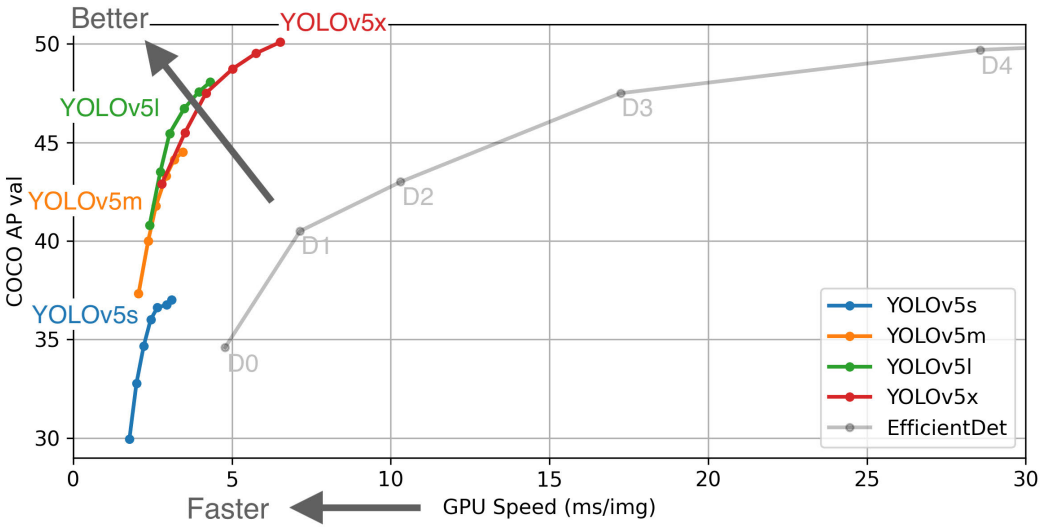


Fonte: (BOCHKOVSKIY; WANG; LIAO, 2020), (ALEXEYAB, 2021).

2.7.5 YOLOv5

Outras implementações do YOLO como o *Ultralitycs YOLOv5* (JOCHER *et al.*, 2021) — baseado no YOLOv4, que utiliza da ferramenta PyTorch (Python), em vez de Darknet (C e CUDA) — foram desenvolvidas por pesquisadores independentes. Os resultados do YOLOv5 avaliados no banco de dados COCO(LIN *et al.*, 2014) são similares aos encontrados pelo YOLOv4. A Figura 16 mostra a comparação do YOLOv5 com o detector EfficientDet (TAN; PANG; LE, 2019). A vantagem no uso de PyTorch é que o *framework* é mais intuitivo e possui um suporte mais abrangente para *deep learning*.

Figura 16 – Desempenho do YOLOv5



Fonte: (JOCHER *et al.*, 2021)

2.7.6 tiny-YOLO

Por ser um modelo que não somente busca a precisão, mas também a velocidade, foram desenvolvidas versões compactas do modelo YOLO chamadas de tiny-YOLO ou YOLO-tiny. Estes modelos são versões reduzidas do modelo YOLO, com um menor número de camadas, e apresentam pior desempenho, porém o treinamento é realizado de forma mais rápida e o modelo possui maior velocidade de inferência, por possuir menos parâmetros, proporcionando um excelente custo benefício.

2.8 Trabalhos Similares

Esta seção visa apresentar os conceitos que vêm sendo utilizados recentemente em trabalhos nacionais e internacionais, a fim de estabelecer um contexto com os trabalhos estado da arte.

O trabalho *Weed detection in soybean crops using ConvNets* (FERREIRA *et al.*, 2017), assim como neste presente trabalho, busca realizar a detecção de ervas daninhas presentes em uma plantação de soja, em Campo Grande, MS, Brasil, por meio do uso de redes neurais convolucionais. As imagens de 4000 x 3000 pixels da plantação foram coletadas por meio de um *drone* e então separadas por meio de "superpixels" onde então foi realizado o treinamento utilizando a topologia de rede AlexNet(KRIZHEVSKY; SUTSKEVER; HINTON, 2012), elaborada no software CaffeNet (JIA *et al.*, 2014). Os resultados obtidos pelos autores foram uma precisão superior a 98% e precisão média de 99,5% na classificação das imagens de superpixels, de quatro classes, extraídas das fotos da plantação. Dados de tempo de pré-processamento e inferência não foram expostos no trabalho.

O trabalho *Redes neurais convolutivas aplicadas à detecção de ervas daninhas* (ROSA, 2019), realizado na Universidade Federal do Rio Grande do Sul, UFRGS, expõe um detector de ervas daninhas por segmentação de imagem. O banco de dados utilizado possui imagens de uma plantação de cenouras com fotos tiradas à uma posição diretamente vertical em relação ao solo. As imagens foram treinadas em duas redes convolucionais, uma rede FCN e uma rede SegNet. A métrica de avaliação utilizada para o resultado da segmentação foi a "*Class Accuracy*", a qual foi avaliada com valores superiores a 75%. Nota-se que foi utilizado a segmentação de imagens neste trabalho. A utilização desta técnica parece não ser interessante para a detecção de plantas invasoras e aplicação de herbicidas uma vez que a precisão obtida na segmentação da imagem é desperdiçada pelo método de aplicação do herbicida (jatos em formato de cone ou filetes).

O trabalho *Farm land weed detection with region-based deep convolutional neural networks* (SARKER; KIM, 2019) avalia a performance do uso da rede convolucional ResNet (HE *et al.*, 2015a) para a detecção e classificação de ervas daninhas em uma plantação de cebolas. Utilizando a rede ResNet-101 como espinha dorsal para arquiteturas de R-CNNs foi obtido mAP (média da precisão média) próxima a 80%, enquanto o modelo comparado, VGG-16 (SIMONYAN; ZISSERMAN, 2014), apresentou um mAP de aproximadamente 57%

O trabalho *Weed detection in canola fields using maximum likelihood classification and deep convolutional neural network* (ASAD; BAIS, 2020) utiliza segmentação de objetos para detecção de ervas daninhas em uma plantação de canola. O trabalho utiliza as últimas CNNs estado da arte, VGG e ResNet, como espinha dorsal para a UNET e SegNet. As métricas de avaliação utilizadas no trabalho, *Mean Intersection Over Union* (MIOU) e

Frequency Weighted Intersection Over Union(FWIOU) apresentam valores de 82,88% e 98,69%, respectivamente. Os resultados obtidos, mais uma vez, expõem o grande potencial na utilização das CNNs para a detecção de ervas daninhas. Entretanto, o tempo de inferência da classificação não foi analisado, impedindo verificação da possibilidade do uso em uma tarefa em tempo real.

O trabalho *Weed Identification using Convolutional Neural Network and Convolutional Neural Network Architectures* (GOTHAI *et al.*, 2020) utiliza o banco de dados *Plant Seedlings Dataset V2* (GISELSSON *et al.*, 2017) para o treinamento de três diferentes modelos de redes neurais convolucionais: VGG-16, ZFNet e AlextNet. Os resultados encontrados indicam precisão superior a 90% na classificação de diferentes classes de ervas daninha, indicando, mais uma vez, a eficácia do uso de redes convolucionais para a classificação de ervas daninhas.

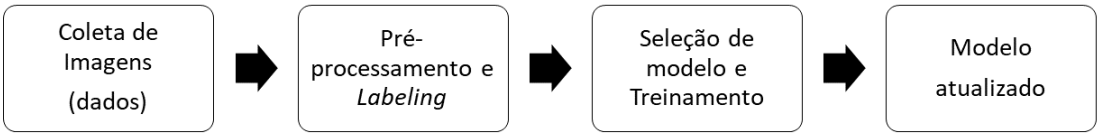
O trabalho *Deep convolutional neural networks for image-based Convolvulus sepium detection in sugar beet fields* (GAO *et al.*, 2020) realiza a detecção de ervas daninhas (Convolvulus sepium) em uma plantação de beterrabas por meio da arquitetura YOLOv3. Os resultados apresentados indicam um desempenho de mAP@0.5 superior a 75%, com tempo de inferência aproximado de 6,5 milisegundos. Utilizando o mesmo hardware usado pelo autor do trabalho – uma Titan X, é estimado que se obtenha uma taxa de quadros por segundo superior a 150, permitindo a aplicação do modelo utilizado em tempo real.

3 Metodologia

3.1 Planejamento

Este trabalho propõe o desenvolvimento e avaliação de um sistema de detecção por visão computacional com o objetivo de localizar ervas daninhas. Para a construção deste sistema é proposto um detector de objetos baseado em redes neurais convolucionais. A construção deste sistema de detecção pode ser dividida em etapas, iniciando pela coleta de dados e formação de um banco de dados de imagens pré-processadas e rotuladas. Em seguida, passando para o processamento por aprendizado de máquina destes dados em ambiente de programação e obtendo como resultado um arquivo de pesos atualizados do modelo de aprendizado de máquina, o principal componente do sistema detector de objetos. O diagrama da Figura 17 descreve simplificadamente as etapas deste processo.

Figura 17 – Diagrama de etapas do projeto



Fonte: Autoria própria.

A partir de revisão bibliográfica e recentes trabalhos na área de visão computacional, foram selecionados modelos de detectores de objetos baseados em CNNs a serem implementados para a detecção de ervas daninhas e posteriormente comparados, a fim de verificar a adequação destes modelos para o propósito deste trabalho. A escolha dos modelos de detector de objetos foi realizada após análise de fatores como precisão média e velocidade de detecção (FPS)[1], conforme revisão bibliográfica. Entre os modelos escolhidos, o YOLO apresentou melhores métricas exigidas para o cumprimento dos objetivos deste trabalho. Outros detectores de objetos foram utilizados por motivo de comparação de resultados. Uma lista completa dos modelos estudados neste trabalho, *framework* utilizado e suas respectivas bases de código podem ser vistos na Tabela 2.

[1] quadros por segundo, ou *frames per second*, em inglês

Tabela 2 – Modelos utilizados para o trabalho.

Modelo	*Framework*	Referência do código
FasterRCNN-ResNet50	PyTorch (1.8.1)	(PYTORCH, 2021)
EfficientDet0	Tensorflow (2.4.1)	(TENSORFLOW, 2021)
SSD-MobileNetV2	Tensorflow (2.4.1)	(TENSORFLOW, 2021)
RetinaNet	Tensorflow (2.4.1)	(TENSORFLOW, 2021)
YOLOv4-tiny	Darknet	(ALEXEYAB, 2021)
YOLOv4	Darknet	(ALEXEYAB, 2021)
YOLOv5s	PyTorch (1.8.1)	(ULTRALYTICS, 2021)
YOLOv5m	PyTorch (1.8.1)	(ULTRALYTICS, 2021)
YOLOv5x	PyTorch (1.8.1)	(ULTRALYTICS, 2021)

Fonte: Autoria própria.

Outro objetivo deste trabalho, é a avaliação da utilização do modelo treinado para a detecção de ervas daninhas em imagens e vídeos, utilizando um hardware de baixo custo Raspberry Pi 4, a fim de verificar a viabilidade de sua implementação em um projeto de sistema de pulverização com auxílio de visão computacional e análise de vídeo em tempo real à 30 FPS.

3.2 Descrição do Banco de dados e Processamento de dados

Para a realização deste trabalho foi utilizado um banco de dados de fotos de ervas daninhas detectadas em meio a plantações de soja. As imagens originais do banco de dados utilizado para este projeto foram produzidas e fornecidas pela empresa Eirene Solutions. Este banco de dados é composto por duas partes: 92 fotos de diferentes pontos de uma plantação de soja coletadas por um operador de câmera fotográfica e 220 quadros retirados de vídeos de um operador de câmera caminhando sobre plantações de soja e milho. As fotos do banco de dados passarão a ser referidas simplesmente como fotos e os quadros retirado dos vídeos passarão a ser referidos como quadros. Amostras do banco de dados em questão estão apresentadas nas Figuras 18 e 19.

Figura 18 – Amostras das fotos coletadas

Fonte: Autoria própria.

Figura 19 – Amostras dos quadros coletados

Fonte: Autoria própria.

As fotos originais do banco de dados apresentam dimensão 640x480 (4:3) com 4 canais (RGBA). Enquanto os quadros, foram retirados de vídeos FULL HD (1920x1080), com 29,98 quadros por segundo e compressão H.264 com *bit rate constante* (CBR) de 20 Mbits. Pelo fato dos vídeos apresentarem compressão, é importante atestar que a qualidade da imagem dos quadros obtidos está parcialmente degradada, apresentando imagens pixeladas e, em alguns casos, com artefatos. Além disso, como o operador da câmera estava em movimento durante a gravação do vídeo, passando por plantas em diferentes posições, existe perda de foco em parte dos quadros.

Sistemas de pulverizadores seletivos normalmente são equipados com um grande número de câmeras, pois as câmeras de menor resolução apresentam menor custo econômico. A Figura 20 exibe um sistema pulverizador com as câmeras da barra direita demarcadas por retângulos verdes. No sistema Eirene Savefarm, por exemplo, as câmeras utilizadas são conectadas à sistemas Raspberry Pi 3, que além de coletarem as imagens, são responsáveis pelos atuadores do pulverizador. Deste modo, as imagens foram pré-processadas a fim de se obter somente a área central da figura, visto que esta é a área que será analisada e pulverizada pelo sistema de pulverização. Portanto foi aplicado "cropping" nas dimensões vertical e horizontal das fotos, transformando as fotos em quadrados com resolução de 480x480 pixels. O mesmo processo foi realizado com os quadros, o que de certa forma, diminui a influência da lente da câmera nas imagens. Além disso, foram aplicadas as técnicas de aumento de dados de *flip* horizontal e giro 90 graus, multiplicando o número de imagens do conjunto de treinamento banco de dados. O resultado final deste pré-processamento aplicado pode ser visto nas amostras do banco de dados apresentadas na Figura 21.

Figura 20 – Pulverizador agrícola equipado com sistema pulverizador seletivo

Fonte: Adaptado de (EIRENE, 2021).

Figura 21 – Amostra de imagens do banco de dados pré-processado.

Fonte: Autoria própria.

Por final, as imagens foram separadas entre conjuntos de treino e teste, onde o conjunto de teste foi separado em duas partes: teste e teste visual. As imagens foram separadas para os conjuntos de treino, teste e teste visual na proporção 7:2:1, sendo esta proporção selecionada empiricamente de maneira a obter o maior número de imagens, para redução de variância durante o treinamento, porém com imagens suficientes para a avaliação dos modelos por métricas e avaliação visual das métricas utilizadas após o treinamento. Como foram aplicadas técnicas de aumento de dados, as imagens de treinamento foram multiplicadas, aumentando o tamanho do conjunto de treinamento de 213 imagens para 547. Obviamente, a quantidade de imagens dos conjuntos de teste e teste visual permaneceram inalteradas.

3.3 Rotulação das imagens

O processo de rotulação ou *labeling* das imagens, consiste na marcação de caixas delimitadoras (*bounding boxes*) ao redor dos objetos de interesse, as ervas daninhas, no caso deste trabalho. Elas servem como referência para o processo de aprendizado de máquina supervisionado, onde são realizadas de uma à dezenas de marcações por imagem. Para a realização deste processo, foi utilizado o software *open-source* Microsoft VoTT 2.2.0

(MICROSOFT, 2021). O resultado deste processo está apresentado na Figura 22, onde foram marcados os rótulos da Figura 18.

Figura 22 – Banco de dados com rótulos/*labels*

Fonte: Autoria própria.

Os dados extraídos pelo processo de rotulação são todas as caixas delimitadoras da imagem — dadas pela posição de dois pontos de vértices opostos — e a classificação do objeto demarcado pela caixa.

É importante notar que cada implementação de detector de objetos utiliza um modelo de dados de entrada diferente. No caso do YOLOv4, implementado em Darknet (ALEXEYAB, 2021), a entrada deve ser fornecida por um arquivo de texto com o caminho do arquivo de imagem, separado por todas as *bounding boxes* — contendo *xmin, ymin, xmax, ymax* e classe — presentes na imagem, separadas por espaço. Já os modelos implementados em Tensorflow (TENSORFLOW, 2021), requerem um arquivo *TFRecords* de entrada, por exemplo.

3.4 Descrição dos experimentos, modelo treinado e arquiteturas propostas

3.4.1 Rotina de treinamento

Para a realização destes experimentos, foram realizados treinamentos dos modelos no ambiente de programação Google Colab, que utiliza a linguagem de programação iPython. A escolha do Google Colab como ambiente de treinamento, foi realizada pela integração do ambiente com as bibliotecas relacionadas a aprendizado de máquina e *deep learning*, além do fornecimento de uma GPU Tesla T4 (NVIDIA, 2019) gratuitamente, acelerando significativamente o treinamento se comparado ao treinamento sem uso de GPU.

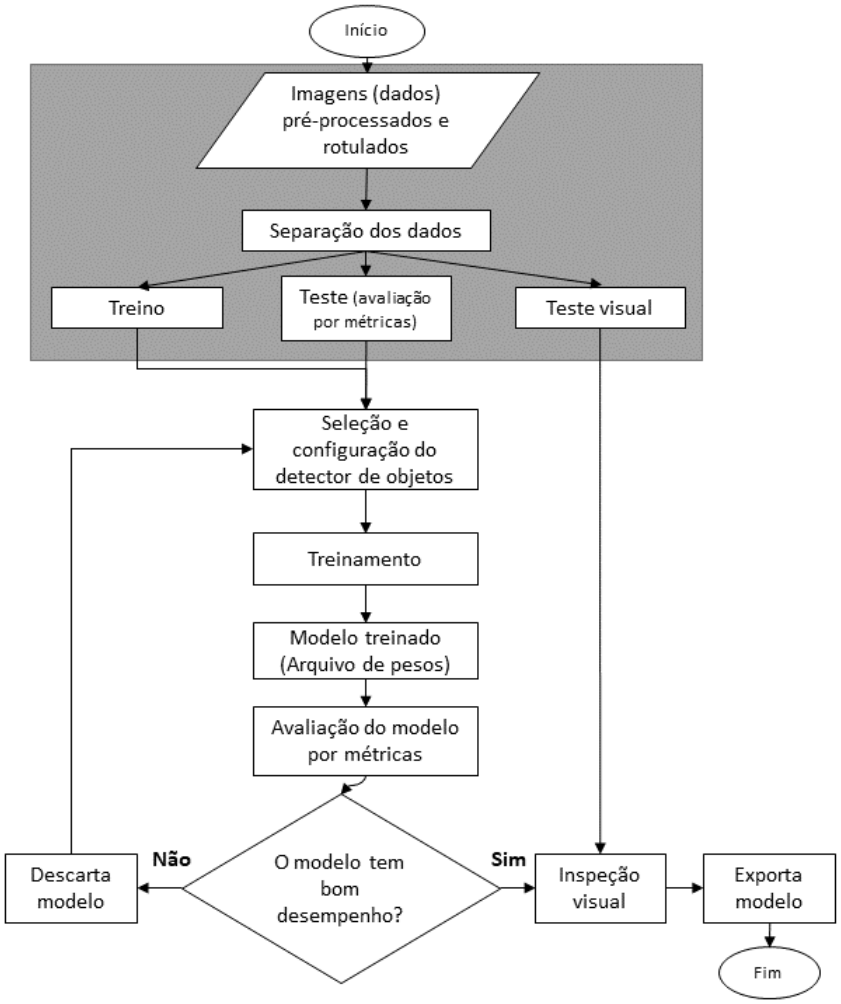
O *pipeline* de treinamento, apresentado na Figura 23, é composto por duas partes distintas, o fornecimento do banco de dados ao modelo, e a configuração e execução do treino do modelo. Estas partes podem ser consideradas independentes, uma vez que obtido o banco de dados é possível realizar seu fornecimento ao modelo da maneira adequada e, então, não será novamente alterado pelo programa. Isto permite a utilização do banco de dados para múltiplos modelos de detectores de objeto sem muita dificuldade. A configuração dos modelos utilizados foi realizada individualmente, visto as diferentes implementações utilizadas. Estas configurações estão apresentadas nas subseções seguintes, referentes a cada experimento realizado.

3.4.2 Avaliação preliminar

Para uma avaliação inicial dos modelos de detectores de objetos encontrados na literatura, foram testados os modelos e as configurações apresentados na Tabela 3. Em todos modelos foram utilizadas somente as fotos do banco de dados apresentado na subseção 3.2, para realizar o treinamento dos modelos de maneira mais ágil.

As fotos foram separadas em conjuntos de treino, teste e teste visual. A separação dos conjuntos foi realizada, primeiramente, com a separação prévia de 8 imagens de teste para inspeção visual, de maneira a manter uma quantidade suficiente de fotos para os outros conjuntos durante o treinamento. Por sua vez, os outros conjuntos foram separados em uma proporção 80%:20%, totalizando 68 e 17 imagens, para o conjunto de treino e teste, respectivamente. As avaliações dos modelos por meio de métricas foram realizadas no conjunto de teste durante o treinamento.

Figura 23 – Fluxograma



Fonte: Autoria própria.

Tabela 3 – Configuração dos modelos para avaliação preliminar

Modelo	Batches/Épocas	Batch Size	Outros
FasterRCNN-ResNet50	50	2	lr-steps 16 22
EfficientDet0	4000	16	-
SSD-MobileNetV2	4000	16	-
RetinaNet	4000	8	-
YOLOv4-tiny	4000	32	subdivisions=8
YOLOv4	4000	32	subdivisions=8
YOLOv5s	500	32	img_size=480
YOLOv5m	500	32	img_size= 480
YOLOv5x	500	32	img_size= 480

Fonte: Autoria própria.

3.4.3 Avaliação em vídeo

Após avaliação preliminar, foram realizados novos treinamentos e avaliações dos modelos com o banco de dados completo, composto de fotos e quadros de vídeos, totalizando 639 imagens separadas na proporção 70%:20%:10%, para os conjuntos de treino, teste e teste visual, respectivamente. A avaliação dos modelos foi realizada durante o treinamento no conjunto de teste e posteriormente ao treinamento por meio de inspeção visual no conjunto de teste visual. Os modelos utilizados neste experimento foram selecionados conforme desempenho apresentado. Esta seleção de modelos foi realizada devido ao longo tempo necessário para o treinamento de cada modelo no banco de dados completo, que demora múltiplas horas. Os modelos testados e suas configurações estão apresentados na Tabela 4.

Tabela 4 – Configuração dos modelos para avaliação completa

Modelo	Batches/Épocas	Batch Size	Outros
EfficientDet0	4000	16	-
YOLOv4-tiny	4000	32	subdivisions=8
YOLOv4	4000	32	subdivisions=8
YOLOv5s	500	32	img_size=480
YOLOv5m	500	32	img_size= 480
YOLOv5x	500	32	img_size= 480

Fonte: Autoria própria.

É importante notar que é esperado um resultado menos preciso no banco de dados com quadros de vídeo. Isto acontece por que os quadros apresentam tipos diferentes de plantas invasoras, aumentando a variabilidade dos objetos detectados. Ademais, os quadros de vídeos apresentam compressão, maior perda de foco e lente de câmera diferente (com abertura mais ampla), introduzindo mais fatores que afetam o treinamento em relação as fotos.

3.5 Avaliação do uso dos modelos no Raspberry Pi 4

O Raspberry Pi 4 é um computador de baixo custo que apresenta entrada de rede, entradas USB, saída de vídeo e outras utilidades, permitindo a sua integração a variados tipos de projetos. Este hardware vem sendo utilizado inclusive em aplicações de visão computacional, por possuir um módulo de câmera de baixo custo. Um exemplo de utilização de sistemas de baixo custo com visão computacional são sistemas de pulverização seletiva que detectam regiões de interesse sem aprendizado de máquina.

Com o objetivo de realizar a avaliação da utilização de detector de objetos com redes convolucionais em ervas daninhas, foi adquirido e utilizado um Raspberry Pi 4 (versão 4GB). O sistema operacional do equipamento, o *Ubuntu Desktop 20.10*, foi instalado em um cartão de memória SD *Sandisk 64GB Extreme*.

Após a conclusão dos treinamentos dos modelos conforme descrito nas subseções anteriores, todas as versões dos modelos YOLO treinados, foram transferidos para o disco do Raspberry Pi 4 para avaliação, por terem apresentados os melhores desempenhos em velocidade de inferência.

A avaliação do tempo de inferência de cada modelo se deu pela repetição de 5 inferências de cada modelo, por meio da execução dos seus respectivos *scripts* de detecção (que calculam automaticamente a velocidade das inferências), aplicadas em uma amostra de imagem de dimensões 640x480 do conjunto de testes. O cálculo da média destas 5 inferências por modelo, foi usado para obter um valor de quadros por segundo (FPS) estimado para utilização do hardware em tempo real. É importante destacar que foi realizada uma inferência previamente às inferências medidas devido ao carregamento dos modelos à memória do sistema computacional.

4 Resultados e Discussões

4.1 Avaliação de resultados preliminares

Utilizando a metodologia descrita na Seção 3 deste trabalho, foram rodadas as rotinas de programação no ambiente virtual Google Colab[1] para os modelos descritos. Os resultados estão apresentados na Tabela 5. Um gráfico destes resultados está apresentado na Figura 24, para facilitar a comparação entre modelos.

Tabela 5 – Resultados preliminares

Modelo	AP@0.5	Tempo médio de inferência	FPS Estimado
FasterRCNN-ResNet50	0,548	136ms	7
EfficientDet0	0,723	27ms	37
SSD-MobileNetV2	0,522	18ms	56
RetinaNet	0,496	56ms	18
YOLOv4-tiny	0,677	5ms	200
YOLOv4	0,680	34ms	29
YOLOv5s	0,681	8ms	125
YOLOv5m	0,763	17ms	59
YOLOv5x	0,619	24ms	42

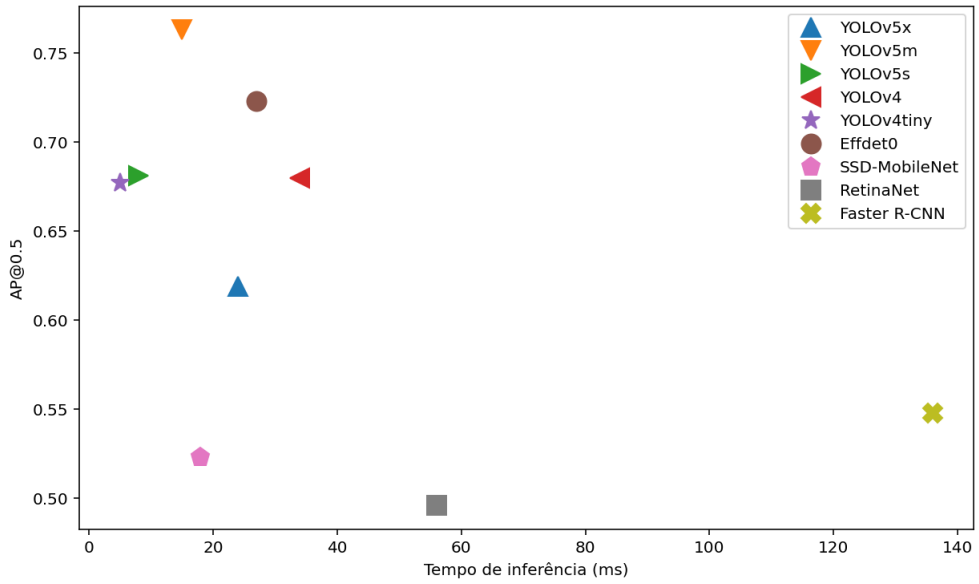
Fonte: Autoria própria.

Onde AP@0.5 refere-se a precisão média de cada modelo com IOU de 50%, o tempo médio aproximado de inferência é o tempo calculado para realizar a detecção em uma imagem 640x480 pixels e o FPS estimado é dado pelo inverso do tempo médio aproximado de inferência em uma GPU Tesla T4.

Analisando os resultados apresentados na Tabela 5 e Figura 24, é possível observar que os modelos YOLO e EfficientDet0 apresentam os melhores desempenhos de AP@0.5 e tempo de inferência, estando acima e a esquerda do gráfico. O modelo YOLOv5m apresentou o melhor AP@0.5 e uma boa velocidade de inferência, enquanto os modelos tiny-YOLOv4 e YOLOv5s, que possuem número reduzido de camadas, apresentaram as melhores velocidades de inferência. O desempenho AP@0.5 destes modelos ficou similar ao de suas variantes completas. O modelo Faster R-CNN apresentou o pior desempenho de velocidade de inferência, o que de certa forma, já era esperado. Enquanto isso, o modelo SSD-MobileNetv2 originalmente desenvolvido para dispositivos móveis embarcados, apresentou boa velocidade, ao custo de um AP@0.5 inferior aos modelos YOLO.

[1] com o uso de GPU Tesla T4 (NVIDIA, 2019), verificado pelo uso do comando *!nvidia-smi*

Figura 24 – Resultados: AP@0.5 por Tempo de inferência dos modelos



Fonte: Autoria própria.

Enfim, a avaliação preliminar dos modelos indica que o YOLOv4, YOLOv5 e EfficientDet0 devem ser o modelos preferidos ao realizar detecção de invasoras, tendo em vista o desempenho apresentado, tanto de AP@0.5 como de velocidade de inferência.

Uma amostra dos resultados obtidos, aplicados ao conjunto de teste utilizado, está apresentada na Figura 25, onde os *labels* estão marcados em azul e a detecção está marcada em rosa. Observando os resultados, é possível notar que não existe presença de falsos positivos nas detecções, mesmo nas situações onde havia área de plantação não trabalhada[2] que é similar a planta invasora. Entretanto o modelo YOLOv4 apresentado na figura, apresentou falsos negativos. Deste modo, é interessante o aumento do banco de dados de treinamento, para aprimoramento da generalização do modelo, que ficou condicionado à quantidade de imagens disponibilizadas[3]. Além do mais, as imagens também são dependentes da sazonalidade da cultura, podendo ocasionar diferença de desempenho no modelo treinado em imagens de plantações em diferentes estados. Assim, tem-se plena consciência que o tamanho do banco de dados deve aumentar para fornecer maior robustez ao sistema detector.

[2] este tipo de vegetação não foi treinado
[3] fornecidas pela empresa Eirene Solutions

Figura 25 – Resultados YOLOv4 no banco de dados de teste (Os *labels* estão em azul)

Fonte: Autoria própria.

4.2 Avaliação de resultados no banco de dados com fotos e quadros

4.2.1 Avaliação por métricas

Com a conclusão das avaliações preliminares dos modelos, foram realizados os treinamentos para o banco de dados composto por fotos e quadros de vídeos, conforme seção (3.4.3). Os resultados estão apresentados na Tabela 6 e Figura 26. Estes resultados foram obtidos pelas avaliações dos conjuntos de teste, realizadas conjuntamente aos treinamentos dos modelos.

Tabela 6 – Resultados (banco de dados completo)

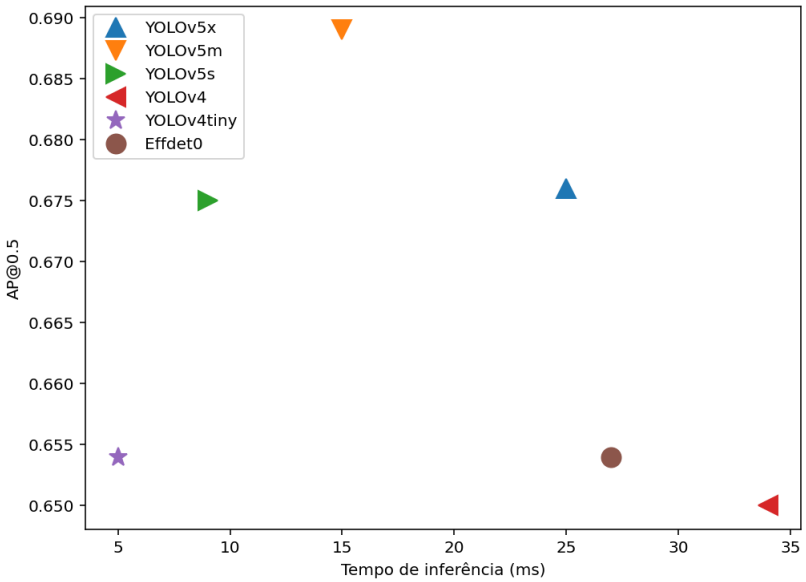
Modelo	AP@0.5	AP@0.5:0.95	Tempo médio de inferência	FPS estimado
EfficientDet0	0,654	0,326	27ms	37
YOLOv4-tiny	0,654	(-)	5ms	200
YOLOv4	0,650	(-)	34ms	29
YOLOv5s	0,675	0,331	9ms	111
YOLOv5m	0,689	0,366	15ms	67
YOLOv5x	0,676	0,363	25ms	40

Fonte: Autoria própria.

Onde AP@0.5 refere-se a precisão média de cada modelo com IOU de 50%, AP@0.5:0.95 refere-se a AP calculada para os intervalos de IOU de 50% à 95%,o tempo médio aproximado de inferência é o tempo calculado para realizar a detecção em uma imagem 640x480 pixels e o FPS estimado é dado pelo inverso do tempo médio aproximado de inferência em uma GPU Tesla T4.

Analisando os resultados apresentados na Tabela 6, é possível observar uma menor diferença entre os resultados de AP@0.5 dos modelos. Isto se deve a seleção dos modelos e ao conjunto de teste conter um maior número de imagens. Entretanto, a velocidade de inferência das imagens permaneceu praticamente inalterada para todos os casos, indicando que esta característica de desempenho é independente ao banco de dados, dependendo principalmente do hardware utilizado. Comparando os modelos testados, o YOLOv5m apresentou, novamente, melhor desempenho de AP@0.5, assim como uma boa velocidade de inferência, próxima a 60 FPS com a GPU Tesla T4. Os modelos tiny-YOLOv4 e YOLOv5s, com número reduzido de camadas, apresentaram desempenho em linha com os modelos completos, indicando que estes modelos são adequados ao banco de dados utilizado e podem ser uma alternativa para o uso com custo computacional limitado. O modelo YOLOv5x apresentou perda de desempenho em relação ao modelo YOLOv5m, indicando que o aumento do número de parâmetros possa ter prejudicado o modelo nesta tarefa, ou que os modelos com menos parâmetros tenham algum tipo de vantagem.

Figura 26 – Resultados: AP@0.5 por Tempo de inferência dos modelos



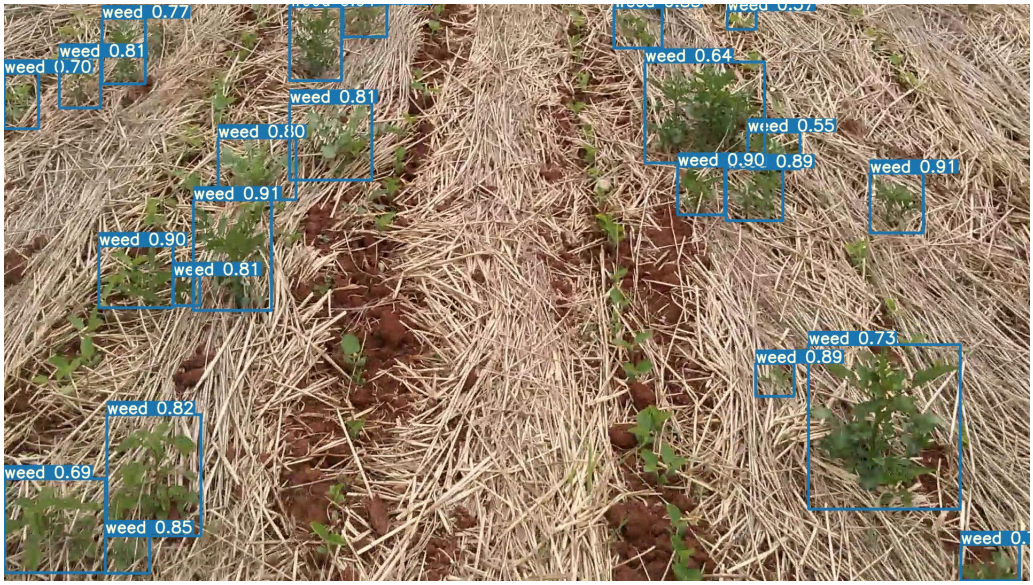
Fonte: Autoria própria.

É importante destacar que os vídeos apresentam compressão. Por este motivo era esperado que os modelos treinados apresentassem menor desempenho, o que pode ser observado na comparação dos resultados, entretanto a diferença de desempenho foi relativamente pequena. Apesar disso, os quadros selecionados e rotulados durante a elaboração do banco de dados possuem um certo viés relacionado ao operador que marcou as caixas delimitadoras, somente para os quadros onde foi possível a identificação das plantas invasoras, ou seja, os quadros com menos perda de foco e efeitos de compressão, entre outros. Deste modo, o desempenho dos modelos analisados pode apresentar divergência quando aplicados em vídeos de outros bancos de dados.

4.2.2 Avaliação visual

Um vídeo de um operador de câmera andando sobre uma plantação de soja contendo plantas invasoras foi utilizado para avaliar o modelo de melhor desempenho, o YOLOv5m. Neste vídeo, é possível observar que o modelo consegue se adaptar aos problemas mencionados anteriormente, como a compressão do vídeo. Portanto, é possível concluir que os modelos YOLO são capazes de serem treinados para realizar a detecção em vídeos com compressão, demonstrando alto potencial para seu uso em aplicações práticas. Um dos quadros deste vídeo, com as detecções apresentadas em azul, está exibido na Figura 27.

Figura 27 – Quadro de vídeo com detecções do YOLOv5m



Fonte: Autoria própria.

Outro ponto interessante é que o vídeo processado apresenta dimensões FULL HD (1920x1080 pixels), enquanto o modelo foi treinado somente na parte central dos quadros do banco de dados, "cropada" para a resolução 480x480 pixels. Mesmo com a imagem de entrada do modelo apresentando resolução diferente das imagens treinadas, o modelo foi capaz de generalizar a detecção de invasoras mesmo para áreas das bordas da imagem que apresentam pequenas deformações devido a lente, além de ângulo de visão e distâncias diferentes em relação ao centro da imagem.

4.3 Tempo de inferência em um Raspberry PI 4

Uma vez que o modelo YOLO apresentou menor velocidade de inferência, o mesmo foi testado em um Raspberry PI 4. Esse teste foi feito com a intenção de obter o tempo de inferência com uma imagem do banco de dados de teste (640x480 pixels). Assim, é possível fazer uma análise de viabilidade do uso de um hardware de baixo custo para visão computacional em tempo real, bem como obter o máximo número de quadros por segundos (FPS) nesse processamento. Os resultados obtidos estão apresentados na Tabela 7.

Tabela 7 – Resultados de inferência em Raspberry Pi 4

Modelo	Tempo de inferência	FPS Estimado
YOLOv4-tiny	3,5s	0,3
YOLOv4	33,5s	0,03
YOLOv5s	2,0s	0,5
YOLOv5m	4,5s	0,2
YOLOv5x	12,0s	0,08

Fonte: Autoria própria.

Os resultados obtidos demonstram que o uso de um hardware de baixo custo como o Raspberry PI 4 para o uso em aplicações de visão computacional em tempo real à 30 FPS — com os modelos utilizados e seus respectivos *frameworks* e implementações — não é viável para esta resolução de imagem sem a utilização de um acelerador. O melhor resultado foi obtido com o modelo YOLOv5s, com um FPS de 0,5 quadros por segundo. Por este motivo, foi realizado a inferência do modelo em um vídeo de 1 minuto, com aproximadamente 1700 quadros, para verificar a funcionalidade do modelo em um arquivo de vídeo, que normalmente é realizada de maneira mais rápida, devido ao carregamento do modelo na memória do sistema computacional. O tempo para conclusão da inferência no vídeo foi de aproximadamente 48 minutos, com uma taxa de processamento de aproximadamente 0,6 quadros por segundo, indicando que o processamento de imagens e vídeo apresentam velocidades semelhantes.

Apesar dos resultados obtidos indicarem que o uso do Raspberry PI 4 seria extremamente limitado, o experimento comprovou que é possível a utilização de um modelo treinado em um sistema computacional de baixo custo. A redução da resolução a ser inferido pelo modelo seria uma solução para o aumento da velocidade de processamento do vídeo, ao custo de um pior desempenho de precisão no modelo — que por sua vez, poderia ser reduzido diminuindo a área de captura da imagem. Além disso, sistemas de baixo custo dedicados a aprendizado de máquina, como os aceleradores *Google Coral* e *Intel Neural Compute Stick*, podem ser integrados[4] ao Raspberry PI 4 para acelerar o processo de inferência (ANTONINI *et al.*, 2019). De acordo com o melhor resultado obtido neste experimento, para que a velocidade de inferência seja suficiente para aproveitar todos os quadros adquiridos por uma câmera (30 FPS), é estimado que uma unidade aceleradora (TPU)[5] deva aumentá-la em 60 vezes. Outros *frameworks* como o *TensorFlow Lite (TFLite)*, que realiza a conversão das variáveis de ponto flutuante dos modelos para valores

[4] por meio de conexão USB

[5] *tensor processing unit*

inteiros 8 bits, também são uma alternativa para aumentar a velocidade de inferência ao custo de um pior desempenho em precisão média.

4.4 Resolução da imagem

O procedimento apresentado para a avaliação do tempo de inferência no Raspberry PI 4 foi repetido para várias resoluções de imagens com o intuito de avaliar a interferência das mesmas nos resultados dos modelos. Os resultados obtidos com os tempos de inferência de uma mesma imagem convertida para diferentes resoluções para os modelos tiny-YOLOv4 e YOLOv5s, estão apresentados nas Tabelas 8 e 9, respectivamente. A análise destes resultados indicam comportamentos diferentes para os modelos, enquanto o tiny-YOLOv4 diminui sua velocidade de processamento de pixels com o aumento da resolução, o modelo YOLOv5 aumenta sua velocidade, saturando nas resoluções mais altas testadas. Entretanto, ambos os modelos apresentaram velocidade de inferência similares nas resoluções mais baixas, ficando próximas de 1 segundo, o que não seria suficiente para a detecção em um vídeo em tempo real com 30 FPS.

Tabela 8 – Resultados da inferência em Raspberry PI 4 com tiny-YOLO4

Resolução	Tempo de inferência	Pixels/Tempo
192x192	0,7s	52663
256x256	1,3s	50412
384x384	3,0s	49152
512x512	5,4s	48545
640x640	8,7s	47080
960x960	20,4s	45176
1280x1280	37,1s	44162

Fonte: Autoria própria.

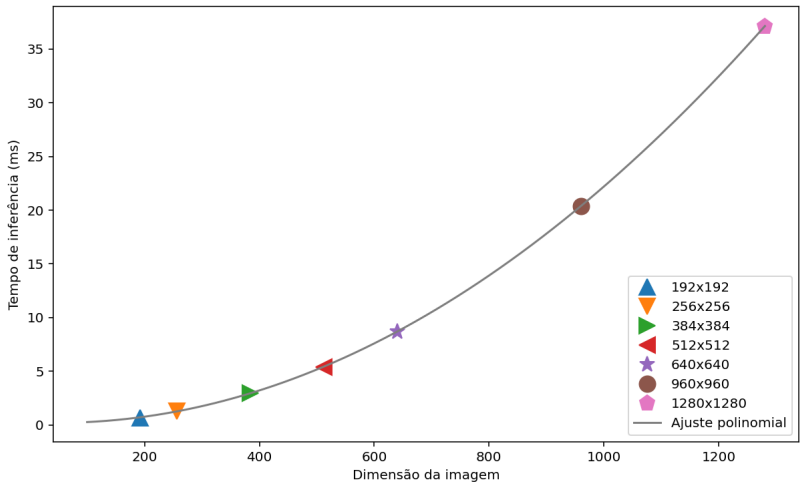
Tabela 9 – Resultados da inferência em Raspberry PI 4 com YOLO5s

Resolução	Tempo de inferência	Pixels/Tempo
192x192	0,7s	52663
256x256	0,8s	81920
384x384	1,6s	90726
480x480	2,0s	115200
640x640	2,8s	146286
1280x736	5,2s	181169
1600x928	8,1s	183309
1920x1088	11,3s	184864

Fonte: Autoria própria.

As Figuras 28 e 29 apresentam a relação do tempo de inferência de detecção com a dimensão da imagem processada, dada pela raíz do número total de pixels da imagem.

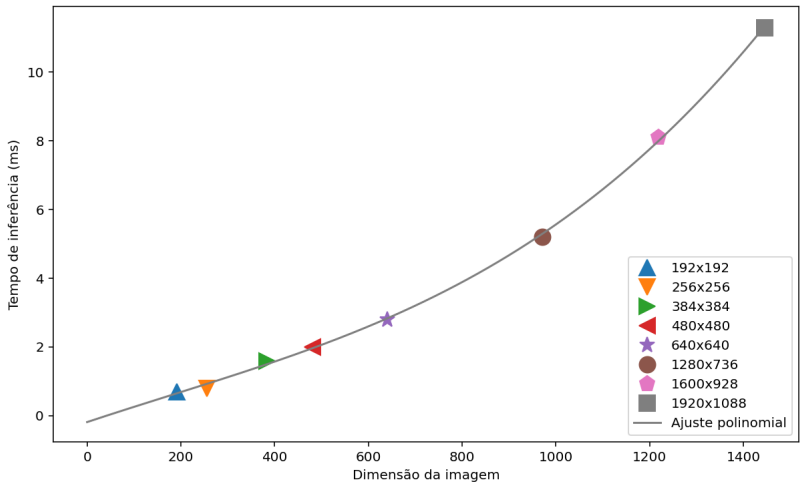
Figura 28 – Resultados tiny-YOLOv4



Fonte: Autoria própria.

Esta relação indica que a velocidade de inferência pode ser estimada por uma função linear em um intervalo pequeno, entretanto tende a ser não linear. Isto pode ser útil em casos onde seja necessário estimar um ganho de velocidade sem a necessidade de realização de inferências. Evidentemente, o processamento do hardware utilizado é o principal fator determinante do tempo de inferência das detecções.

Figura 29 – Resultados YOLOv5s



Fonte: Autoria própria.

5 Conclusão

A utilização de sistemas de visão computacional com inteligência artificial para a agricultura ainda está em desenvolvimento. São poucas as empresas utilizando estas técnicas, e muitas vezes em caráter de testes. A metodologia apresentada neste trabalho define as etapas de projeto, assim como o desenvolvimento de um detector de objetos a partir de modelos pré-treinados. Usando esta metodologia para o processamento das imagens coletadas, foram encontrados excelentes resultados para o sistema proposto, considerando todas as espécies de plantas invasoras como parte de uma mesma classe.

Os resultados obtidos, avaliados pelas métricas descritas no trabalho, demonstram um desempenho de AP@0.5 próximo a 70%. Isto significa que um veículo pulverizador com visão computacional seria capaz de eliminar grande parte das ervas daninhas encontradas nas plantações de soja, com economia significativa do uso de herbicidas, trazendo benefícios econômicos e biológicos. Portanto, é possível concluir, que no presente estado, já é possível a utilização dos modelos de detectores de objetos analisados para a detecção de ervas daninhas, com excelente desempenho.

Apesar do banco de dados conter um número relativamente pequeno de imagens, a utilização de quadros de vídeo com compressão no treinamento demonstrou que os modelos YOLO e EfficientDet são capazes de generalizar ruídos em imagens, evidenciando a robustez da técnica da aprendizagem de máquinas, assim como a praticidade para a realização da coleta de dados, que pode ser realizada pela gravação de vídeos comprimidos e sem a necessidade de estabilização da câmera, por exemplo.

Em relação ao uso do detector de objetos em tempo real por meio de um sistema Raspberry PI 4, ainda existem barreiras em relação a capacidade deste hardware. A análise do tempo de inferência de todos os modelos YOLO testados — inclusive os modelos com menor número de camadas, que exibem menor custo computacional — indicou que o sistema é incapaz de detectar as ervas daninhas com velocidade superior à um quadro e meio por segundo (1.5 FPS), mesmo com resoluções de imagem extremamente baixas para a aplicação desejada. Todavia, foi comprovado que este sistema é capaz de utilizar os detectores de objetos analisados e seria possível sua utilização em projetos que necessitem uma velocidade de amostragem de imagens menor. A combinação deste sistema com aceleradores de processamento para aprendizado de máquina, como o Google Coral e Intel NCS2, pode ser uma solução para os problemas de desempenho apresentados.

Em suma, os resultados validados sistematicamente com métricas compatíveis com a literatura, confirmam a viabilidade técnica a respeito da aplicação de detectores de objetos implementados com CNNs para o propósito de detecção de ervas daninhas.

6 Sugestões para trabalhos futuros

Tendo em vista as presentes discussões apresentadas neste trabalho, surge a necessidade da realização de novos estudos relacionados aos problemas apresentados. Em relação ao uso do detector de objetos para a detecção de ervas daninhas em lavouras, é necessário que seja realizado a avaliação do mesmo com o uso de um banco de dados maior. O conjunto de imagens utilizado neste trabalho, apesar de já ter apresentado excelentes resultados, foi relativamente pequeno se comparado aos demais bancos de dados utilizados na área da visão computacional. A utilização de um banco de dados maior proporcionaria maior generalização do modelo para diferentes estágios da plantação, condições climáticas e outras variáveis relacionadas a implantação deste sistema em veículos de pulverização.

A utilização deste detector de objetos com veículos de pulverização deverá ser reavaliada no local de implementação, visto que estes sistemas, compostos por câmeras, veículo e pulverizador, apresentam características diferentes às do banco de dados utilizado. Fatores como tipo de lente de câmera, posição em relação ao solo, velocidade do veículo e até estabilização do veículo acabaram não sendo incorporados pelo sistema proposto neste trabalho, porém, os resultados sugerem fortemente que isso poderá ser aplicado ao modelo, com o banco de dados adequado. Além disso, o sistema de controle para pulverização seletiva deverá ser integrado ao detector, para a avaliação das dinâmicas do processo.

Em relação ao uso destes modelos com o dispositivo Raspberry Pi 4, é interessante a realização de uma análise deste sistema em conjunto com aceleradores computacionais disponíveis no mercado, como o Google Coral e Intel Neural Compute Stick. Além disso outros sistemas computacionais de baixo custo como o Nvidia Jetson apresentam grande potencial para a aplicação em tempo real e podem também ser avaliados. Outra possível alternativa seria a troca dos sistemas de baixo custo, que se beneficiam da utilização de câmeras de baixa resolução e valor econômico, por um sistema de processamento central com um poder computacional mais elevado. Este sistema poderia ser integrado à câmeras de maior resolução e ângulo de visão, visto os resultados obtidos neste trabalho, diminuindo o número de câmeras necessárias no sistema em troca da necessidade de um sistema computacional de valor econômico maior e, obviamente, o desafio de integrar as respostas obtidas pelo sistema de detecção ao sistema de controle.

Referências Bibliográficas

ALEXEYAB. *AlexeyAB/darknet*. 2021. Disponível em: <https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects>.

ANTONINI, M.; VU, T. H.; MIN, C.; MONTANARI, A.; MATHUR, A.; KAWSAR, F. Resource characterisation of personal-scale sensing models on edge accelerators. In: . New York, NY, USA: Association for Computing Machinery, 2019. (AIChallengeIoT'19), p. 49–55. ISBN 9781450370134. Disponível em: <https://doi.org/10.1145/3363347.3363363>.

ASAD, M. H.; BAIS, A. Weed detection in canola fields using maximum likelihood classification and deep convolutional neural network. *Information Processing in Agriculture*, v. 7, n. 4, p. 535–545, Dec 2020. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2214317319302355>.

BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. Disponível em: <https://arxiv.org/abs/2004.10934>.

CHEN, L.-C.; PAPANDREOU, G.; KOKKINOS, I.; MURPHY, K.; YUILLE, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 40, n. 4, p. 834–848, 2018.

CNA. 2021. Acessado em 22/05/2021. Disponível em: <https://www.cnabrasil.org.br/boletins/pib-do-agronegocio-alcanca-participacao-de-26-6-no-pib-brasileiro-em-2020>.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.: s.n.], 2005. v. 1, p. 886–893 vol. 1.

EIRENE, S. 2021. Acessado em 22/05/2021. Disponível em: <https://www.savefarm.com.br/savefarm>.

EMBRAPA, F. O. d. N. U. p. A. e. A. 2016. Acessado em 22/05/2021. Disponível em: <https://www.embrapa.br/busca-de-noticias/-/noticia/16666472/brasil-sera-um-dos-maiores-exportadores-de-alimentos-preve-fao>.

EVERINGHAM, M.; GOOL, L. V.; WILLIAMS, C. K. I.; WINN, J.; ZISSERMAN, A. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. 2012. Http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

FERREIRA, A. dos S.; FREITAS, D. M.; SILVA, G. Gonçalves da; PISTORI, H.; FOLHES, M. T. Weed detection in soybean crops using convnets. *Computers and Electronics in Agriculture*, v. 143, p. 314–324, Dec 2017. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0168169917301977>.

GAO, J.; FRENCH, A. P.; POUND, M. P.; HE, Y.; PRIDMORE, T. P.; PIETERS, J. G. Deep convolutional neural networks for image-based convolvulus sepium detection in sugar beet fields. *Plant Methods*, v. 16, n. 1, Mar 2020. Disponível em: <https://plantmethods.biomedcentral.com/articles/10.1186/s13007-020-00570-z>.

GHIASI, G.; LIN, T.-Y.; LE, Q. V. *DropBlock: A regularization method for convolutional networks*. 2018. Disponível em: <https://arxiv.org/abs/1810.12890>.

GIRSHICK, R. Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. Disponível em: <https://ieeexplore.ieee.org/document/7410526>.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2014. Disponível em: <https://ieeexplore.ieee.org/document/6909475>.

GISELSSON, T. M.; JøRGENSEN, R. N.; JENSEN, P. K.; DYRMANN, M.; MIDTIBY, H. S. *A Public Image Database for Benchmark of Plant Seedling Classification Algorithms*. 2017. Disponível em: <https://arxiv.org/abs/1711.05458>.

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

GOTHAI, E.; NATESAN, P.; AISHWARIYA, S.; AARTHY, T.; SINGH, G. B. Weed identification using convolutional neural network and convolutional neural network architectures. *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, Mar 2020. Disponível em: <https://ieeexplore.ieee.org/document/9076532>.

GéRON, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2017. ISBN 1491962291. Disponível em: <https://www.xarg.org/ref/a/1491962291/>.

HAYKIN, S. S. *Neural networks and learning machines*. 3rd ed. ed. New York: Prentice Hall, 2009. OCLC: ocn237325326. ISBN 9780131471399.

HE, K.; ZHANG, X.; REN, S.; SUN, J. *Deep Residual Learning for Image Recognition*. 2015. Disponível em: <https://arxiv.org/abs/1512.03385>.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 37, n. 9, p. 1904–1916, 2015.

HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. R. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. Disponível em: <https://arxiv.org/abs/1207.0580>.

INGLE, V. K.; PROAKIS, J. G. *Digital signal processing using MATLAB*. 3rd ed. ed. Stamford, Conn: Cengage Learning, 2012. OCLC: ocn700516252. ISBN 9781111427375.

JIA, Y.; SHELHAMER, E.; DONAHUE, J.; KARAYEV, S.; LONG, J.; GIRSHICK, R.; GUADARRAMA, S.; DARRELL, T. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

JOCHER, G.; STOKEN, A.; BOROVEC, J.; NANOCODE012; CHRISTOPHERSTAN; CHANGYU, L.; LAUGHING; TKIANAI; YXNONG; HOGAN, A.; AL. et. ultralytics/yolov5 v4.0. *Zenodo*, Jan 2021. Disponível em: <https://zenodo.org/record/4418161>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. Curran Associates, Inc., v. 25, 2012. Disponível em: <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>.

KUZNETSOVA, A.; ROM, H.; ALLDRIN, N.; UIJLINGS, J.; KRASIN, I.; PONT-TUSET, J.; KAMALI, S.; POPOV, S.; MALLOCI, M.; KOLESNIKOV, A.; AL. et. The open images dataset v4. *International Journal of Computer Vision*, v. 128, n. 7, p. 1956–1981, Mar 2020. Disponível em: <https://arxiv.org/abs/1811.00982>.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, Dec 1989. Disponível em: <https://ieeexplore.ieee.org/document/6795724>.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998. Disponível em: <https://ieeexplore.ieee.org/document/726791>.

LIN, M.; CHEN, Q.; YAN, S. *Network In Network*. 2013. Disponível em: <https://arxiv.org/abs/1312.4400>.

LIN, T.; MAIRE, M.; BELONGIE, S. J.; BOURDEV, L. D.; GIRSHICK, R. B.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. arxiv 2014. *arXiv preprint arXiv:1405.0312*, 2014.

LIN, T.-Y.; GOYAL, P.; GIRSHICK, R.; HE, K.; DOLLáR, P. *Focal Loss for Dense Object Detection*. 2017. Disponível em: <https://arxiv.org/abs/1708.02002>.

LIU, S.; HUANG, D.; WANG, Y. *Receptive Field Block Net for Accurate and Fast Object Detection*. 2017. Disponível em: <https://arxiv.org/abs/1711.07767v3>.

LIU, S.; QI, L.; QIN, H.; SHI, J.; JIA, J. Path aggregation network for instance segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2018.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. Ssd: Single shot multibox detector. *Computer Vision – ECCV 2016*, p. 21–37, 2016. Disponível em: <https://arxiv.org/abs/1512.02325>.

LONG, J.; SHELHAMER, E.; DARRELL, T. *Fully Convolutional Networks for Semantic Segmentation*. 2014. Disponível em: <https://arxiv.org/abs/1411.4038>.

LOWE, D. Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. [S.l.: s.n.], 1999. v. 2, p. 1150–1157 vol.2.

MATHWORKS. *Convolutional Neural Network: 3 things you need to know*. 2018. Acessado em 22/05/2021. Disponível em: <https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html>.

MICROSOFT. *microsoft/VoTT*. 2021. Disponível em: <https://github.com/microsoft/VoTT>.

MISRA, D. *Mish: A Self Regularized Non-Monotonic Activation Function*. 2019. Disponível em: <https://arxiv.org/abs/1908.08681>.

MITCHELL, T. M. *Machine Learning*. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.

NVIDIA. *Tesla T4 Graphics Processing Unit*. Santa Clara, CA, USA: [s.n.], 2019. Acessado em 22/05/2021. Disponível em: <https://www.nvidia.com/en-us/data-center/tesla-t4/>.

Padilla, R.; Netto, S. L.; da Silva, E. A. B. A survey on performance metrics for object-detection algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.: s.n.], 2020. p. 237–242.

PADILLA, R.; PASSOS, W. L.; DIAS, T. L. B.; NETTO, S. L.; SILVA, E. A. B. da. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, v. 10, n. 3, 2021. ISSN 2079-9292. Disponível em: <https://www.mdpi.com/2079-9292/10/3/279>.

PYTORCH. *pytorch/vision*. 2021. Disponível em: <https://github.com/pytorch/vision/tree/master/references/detection>.

RASCHKA, S.; MIRJALILI, V. *Python Machine Learning, 3rd Ed.* 3. ed. Birmingham, UK: Packt Publishing, 2019. ISBN 978-1789955750.

REDMON, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. <http://pjreddie.com/darknet/>.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. *You Only Look Once: Unified, Real-Time Object Detection*. 2015.

REDMON, J.; DIVVALA, S.; GIRSHICK, R.; FARHADI, A. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. Disponível em: <https://ieeexplore.ieee.org/document/7780460>.

REDMON, J.; FARHADI, A. Yolo9000: Better, faster, stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. Disponível em: <https://ieeexplore.ieee.org/document/8100173>.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. Disponível em: <http://arxiv.org/abs/1804.02767>.

REN, S.; HE, K.; GIRSHICK, R.; SUN, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015. Disponível em: <https://arxiv.org/abs/1506.01497>.

ROSA, M. C. D. Redes neurais convolutivas aplicadas à detecção de ervas daninhas. *Ufrgs.br*, 2019. Disponível em: <https://lume.ufrgs.br/handle/10183/204423>.

RURAL, C. *"Brasil será o maior produtor de alimentos do mundo até 2025"*. 2020. Acessado em 22/05/2021. Disponível em: <https://www.canalrural.com.br/programas/informacao/rural-noticias/brasil-sera-o-maior-produtor-de-alimentos-do-mundo-ate-2025/>.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, v. 115, n. 3, p. 211–252, 2015.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, v. 3, n. 3, p. 210–229, Jul 1959. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5392560>.

SARKER, M. I.; KIM, H. *Farm land weed detection with region-based deep convolutional neural networks*. 2019. Disponível em: <https://arxiv.org/abs/1906.01885>.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. Disponível em: <https://arxiv.org/abs/1409.1556>.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. *arXiv.org*, 2014. Disponível em: <https://arxiv.org/abs/1409.4842>.

TAN, M.; PANG, R.; LE, Q. V. *EfficientDet: Scalable and Efficient Object Detection*. 2019. Disponível em: <https://arxiv.org/abs/1911.09070>.

TENSORFLOW. *tensorflow/models*. 2021. Disponível em: <https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md>.

TURAGA, P.; CHELLAPPA, R.; VEERARAGHAVAN, A. Advances in video-based human activity analysis. *Advances in Computers*, p. 237–290, 2010. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0065245810800075>.

ULTRALYTICS. *ultralytics/yolov5*. 2021. Disponível em: <https://github.com/ultralytics/yolov5>.

USDA. *Food Security*. 2021. Acessado em 22/05/2021. Disponível em: <https://www.usda.gov/topics/food-and-nutrition/food-security>.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* [S.l.: s.n.], 2001. v. 1, p. I–I.

WANG, C.-Y.; BOCHKOVSKIY, A.; LIAO, H.-Y. M. Scaled-yolov4: Scaling cross stage partial network. *arXiv.org*, 2020. Disponível em: <https://arxiv.org/abs/2011.08036>.

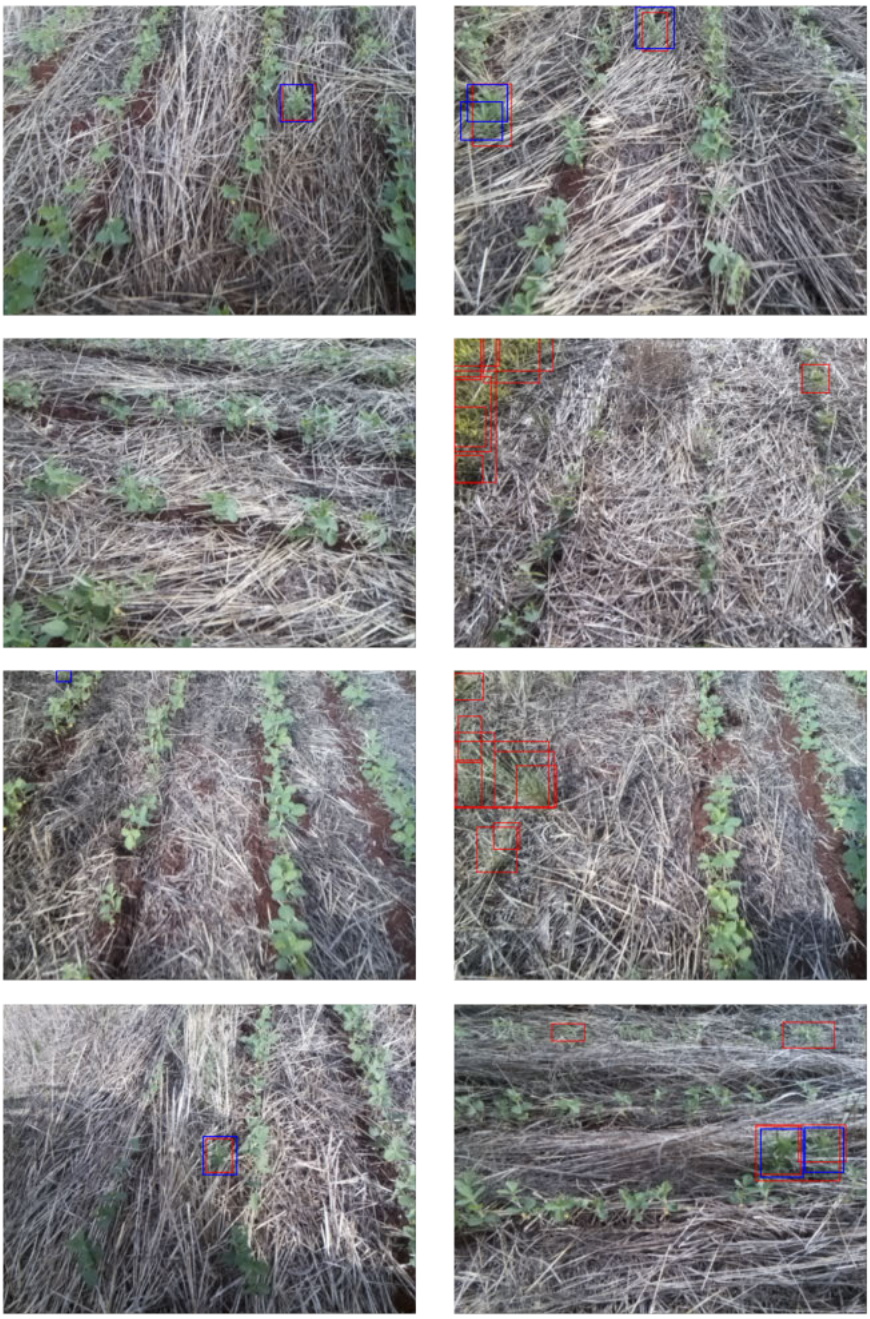
WANG, C.-Y.; LIAO, H.-Y. M.; YEH, I.-H.; WU, Y.-H.; CHEN, P.-Y.; HSIEH, J.-W. *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*. 2019. Disponível em: <https://arxiv.org/abs/1911.11929>.

ZHONG, Z.; ZHENG, L.; KANG, G.; LI, S.; YANG, Y. *Random Erasing Data Augmentation*. 2017. Disponível em: <https://arxiv.org/abs/1708.04896>.

Apêndices

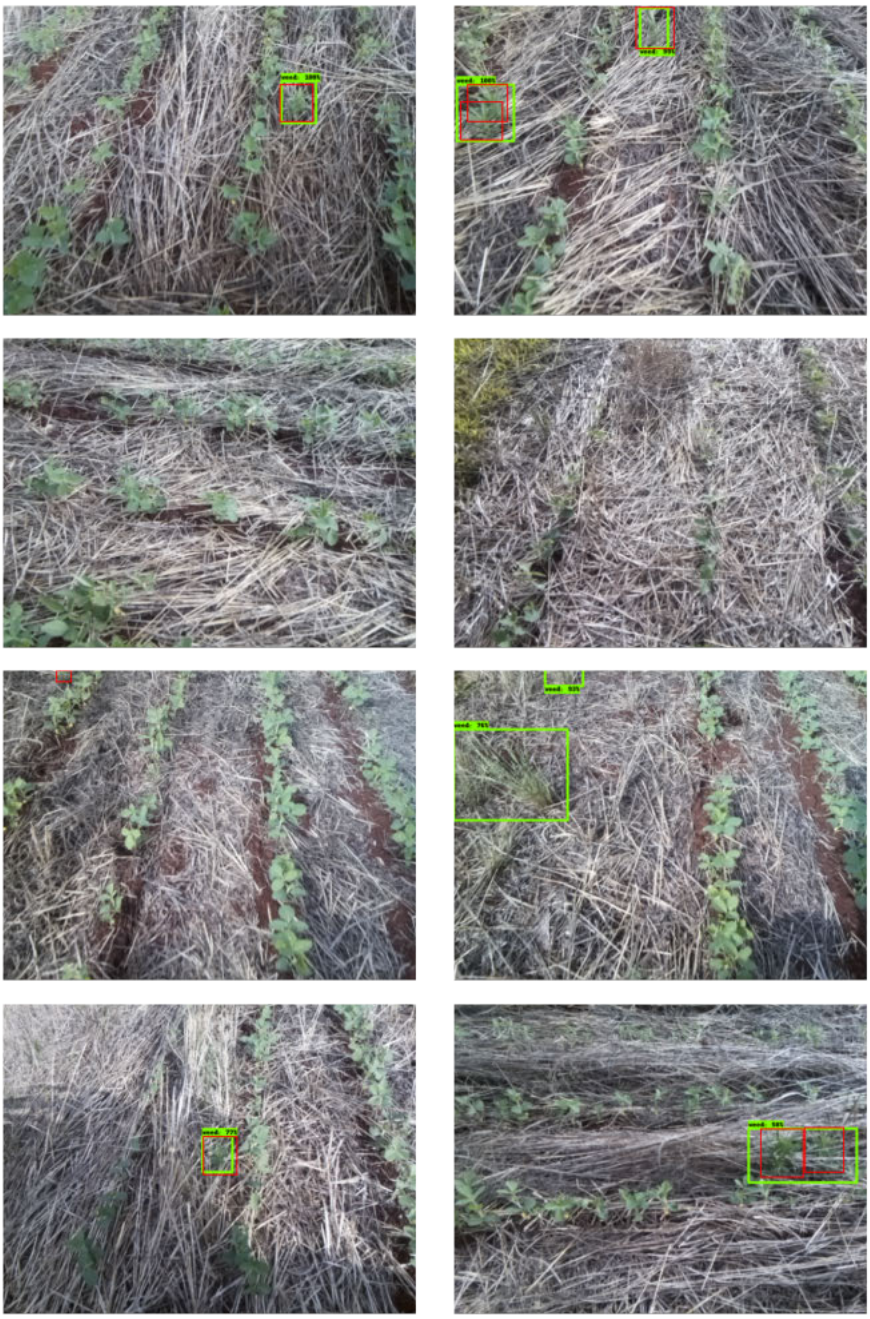
APÊNDICE A – Resultados de inferência em fotos

Figura 30 – Resultados Faster R-CNN/ResNet50 no banco de dados de teste (Os labels estão em azul)



Fonte: Autoria Própria.

Figura 31 – Resultados SSD-MobileNetV2 no banco de dados de teste (Os labels estão em vermelho)



Fonte: Autoria Própria.

Figura 32 – Resultados EfficientDet0 no banco de dados de teste (Os labels estão em vermelho)

Fonte: Autoria Própria.

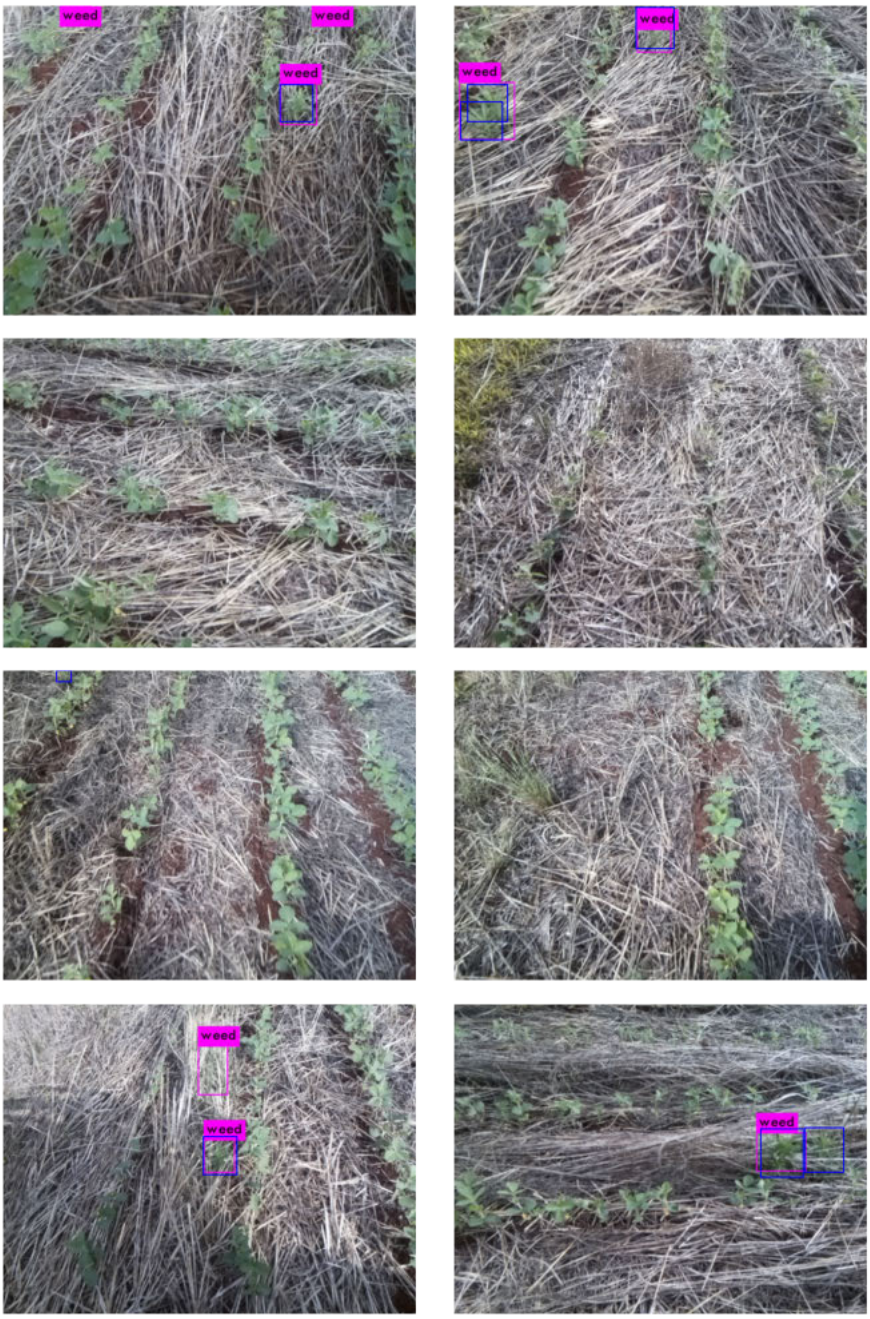
Figura 33 – Resultados RetinaNet50 no banco de dados de teste (Os labels estão em vermelho)

Fonte: Autoria Própria.

Figura 34 – Resultados YOLOv4 no banco de dados de teste (Os labels estão em azul)

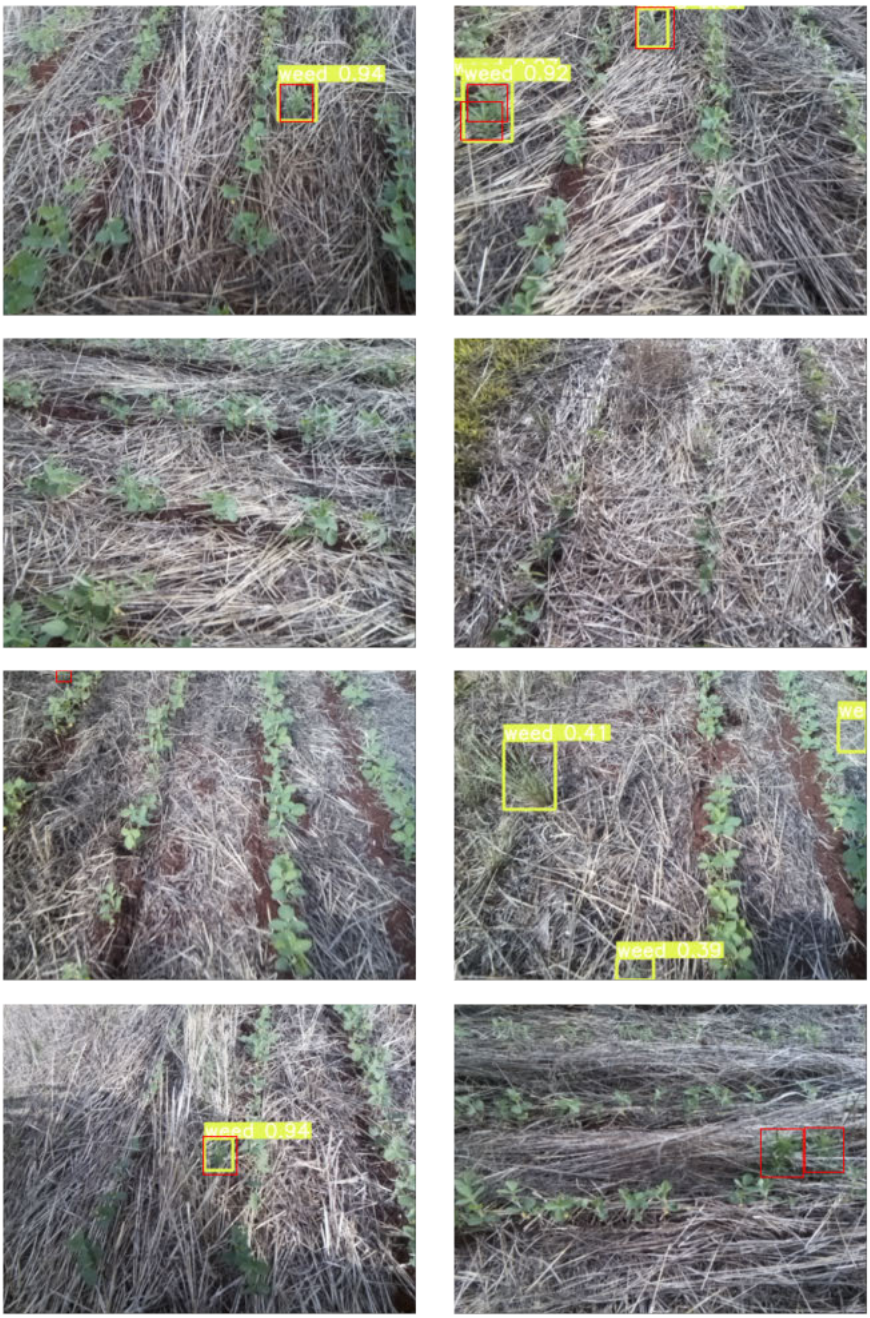
Fonte: Autoria Própria.

Figura 35 – Resultados YOLOv4-tiny no banco de dados de teste (Os labels estão em azul)



Fonte: Autoria Própria.

Figura 36 – Resultados YOLOv5s no banco de dados de teste (Os labels estão em vermelho)

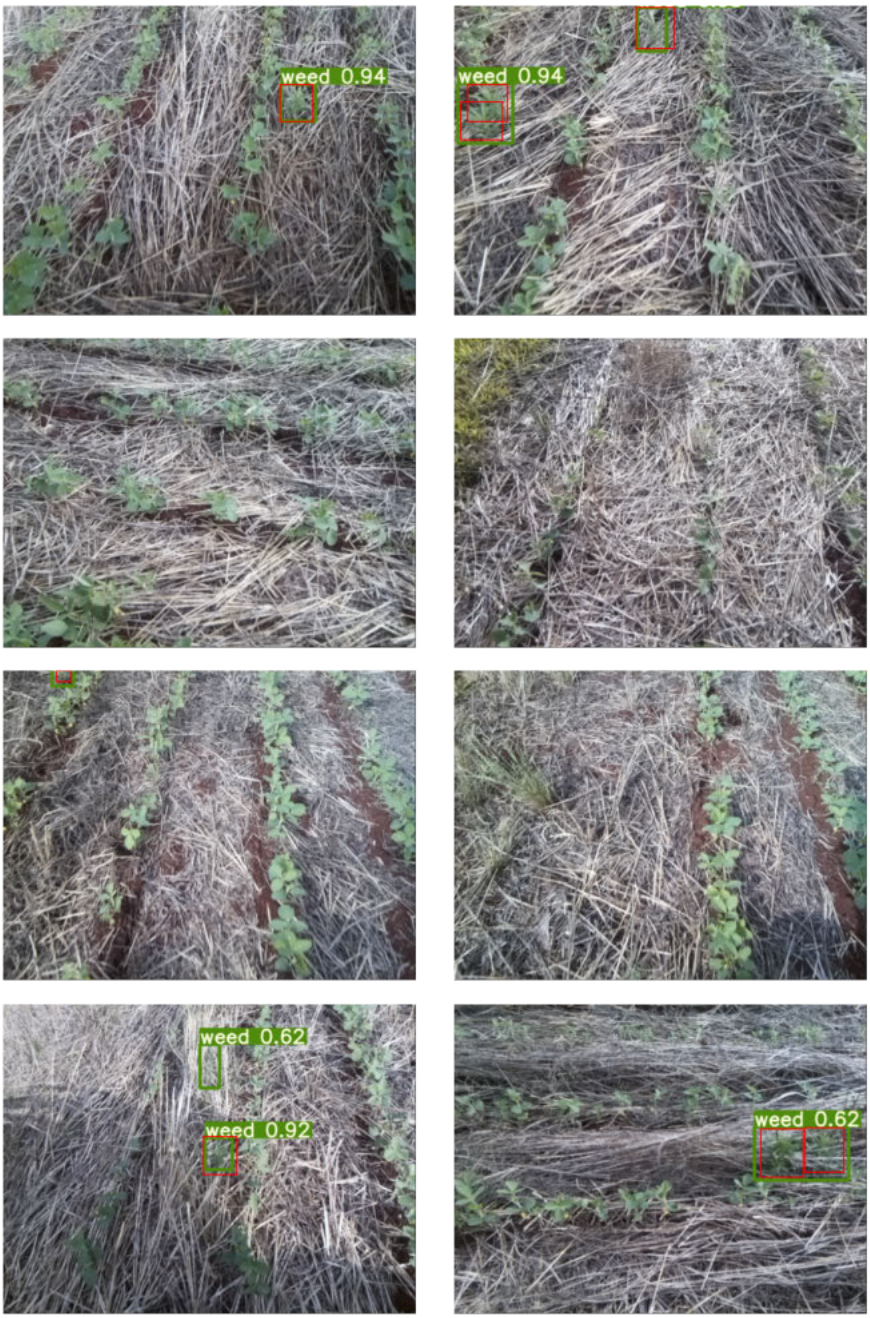


Fonte: Autoria Própria.

Figura 37 – Resultados YOLOv5m no banco de dados de teste (Os labels estão em azul)

Fonte: Autoria Própria.

Figura 38 – Resultados YOLOv5x no banco de dados de teste (Os labels estão em vermelho)



Fonte: Autoria Própria.