UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Programa de Pós-Graduação em Engenharia Elétrica

Redes Neurais Profundas com Redução de Pesos para Uso em Hardwares de Baixo Custo Aplicadas à Identificação de Espécies Invasoras em Cultivares

Luiz Carlos Marques Junior

Bauru
2023

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

Programa de Pós-Graduação em Engenharia Elétrica

Redes Neurais Profundas com Redução de Pesos para Uso em Hardwares de Baixo Custo Aplicadas à Identificação de Espécies Invasoras em Cultivares

LUIZ CARLOS MARQUES JUNIOR

Tese apresentada à Faculdade de Engenharia de Bauru da Universidade Estadual Paulista "Júlio de Mesquita Filho" para obtenção do título de Doutor em Engenharia Elétrica, pelo Programa de Pós-Graduaçãoem Engenharia Elétrica.

Orientador: Prof. Dr. José Alfredo Covolan Ulson

Bauru

2023

Marques Junior, Luiz Carlos.
Redes Neurais Profunds com Redução de Pesos para Uso em Hardwares de Baixo Custo Aplicados à Identificação de Espécies Invasoras em Cultivares /Luiz Carlos Marques Junior, 2023
96 f.:il.

Orientador: José Alfredo Covolan Ulson

Tese (Doutorado) – Universidade Estadual Paulista (Unesp). Faculdade de Engenharia, Bauru, 2023

1. Agroindústria. 2. Plantas Daninhas. 3. Aprendizado Profundo. 4. Redes Neurais Convolucionais. 5. Quantização Pós Treinamento. I. Universidade Estadual Paulista. Faculdade de Engenharia. II. Título.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Bauru

ATA DA DEFESA PÚBLICA DA TESE DE DOUTORADO DE LUIZ CARLOS MARQUES JUNIOR, DISCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA, DA FACULDADE DE ENGENHARIA - CÂMPUS DE BAURU.

Aos 24 dias do mês de julho do ano de 2023, às 08:30 horas, por meio de Videoconferência, realizou-se a defesa de TESE DE DOUTORADO de LUIZ CARLOS MARQUES JUNIOR, intitulada **REDES NEURAIS PROFUNDAS COM REDUÇÃO DE DIMENSÃO DOS PESOS PARA USO EM HARDWARES DE BAIXO CUSTO APLICADAS À IDENTIFICAÇÃO DE ESPÉCIES INVASORAS EM CULTIVARES**. A Comissão Examinadora foi constituida pelos seguintes membros: Prof. Dr. JOSÉ ALFREDO COVOLAN ULSON (Orientador(a) - Participação Virtual) do(a) Departamento de Engenharia Eletrica / Faculdade de Engenharia de Bauru UNESP, Prof. Dr. DANILO HERNANE SPATTI (Participação Virtual) do(a) Departamento de Sistemas de Computação / Instituto de Ciências Matemáticas e de Computação/USP, Prof. Dr. ADRIANO DE SOUZA MARQUES (Participação Virtual) do(a) Engenharia da Computação / Instituto Federal de São Paulo - IFSP - Câmpus Birigui, Prof. Dr. PEDRO DE OLIVEIRA CONCEIÇÃO JÚNIOR (Participação Virtual) do(a) Departamento de Engenharia Eletrica e de Computacao / Escola de Engenharia de Sao Carlos - Universidade de Sao Paulo, Prof. Dr. MARCELO NICOLETTI FRANCHIN (Participação Virtual) do(a) Departamento de Engenharia Eletrica / Faculdade de Engenharia de Bauru - UNESP. Após a exposição pelo doutorando e arguição pelos membros da Comissão Examinadora que participaram do ato, de forma presencial e/ou virtual, o discente recebeu o conceito final: aprovado _ _ _ _ _ _ _ . Nada mais havendo, foi lavrada a presente ata, que após lida e aprovada, foi assinada pelo(a) Presidente(a) da Comissão Examinadora.

Prof. Dr. JOSÉ ALFREDO COVOLAN ULSON

Epígrafe

“O conhecimento abre portas que não podem ser fechadas, rompe barreiras que nunca foram cruzadas e nos faz entender que na vida somos eternos aprendizes.”

Agradecimentos

Gostaria de agradecer primeiramente a Deus, por ter me capacitado e permitido a execução desta pesquisa que culminou nesta tese.

A minha esposa pelo apoio, compreensão e companheirismo em todos osmomentos.

A família, em especial aos meus pais, por terem me dado a possibilidadee por ter me apoiado nos estudos.

Ao meu orientador Prof. Dr. José Alfredo Covolan Ulson, pelo apoio, tempo dedicado a este trabalho e confiança depositada em mim na elaboração desta pesquisa.

A todos os professores e colegas que passaram em minha vida e gentilmente contribuiram na formação de meu conhecimento.

Resumo

O aumento na demanda por alimentos bem como a alta no preço dos insumos para sua produção são fatores que afetam a agroindústria e a população. Neste cenário, outro desafio é o controle de pragas e doenças, dentre essas pragas destacam-se as espécies invasoras (plantas daninhas), nos cultivares de milho e soja, que são os grãos mais produzidos no mundo. Atualmente, a identificação dessas plantas é um processo insalubre, laborioso e sujeito a falhas quando realizado pelo método manual. Quando o controle é feito por máquinas de maneira não seletiva, o agricultor estará beneficiando espécies resistentes tornando o manejo ineficiente, além de causar impactos ambientais pelo uso excessivo de herbicidas. Portanto, existe a necessidade de um sistema embarcado de baixo custo para a identificação de plantas daninhas que permita seu uso em tratores ou diretamente em plataformas robóticas. Com base nestes desafios, para este trabalho foi proposto e explorado o processo de identificar e localizar espécies invasoras nas culturas de milho e soja por meio de um sistema de inteligência artificial com quantização e conversão de pesos neurais em um *hardware* de baixo custo. Para isso fez-se uso da visão computacional associada ao aprendizado profundo (*Deep Learning*) no treinamento de arquiteturas de redes neurais convolucionais (*Convolutional Neural* Network) *Yolov5* em um conjunto de imagens público da plantas daninhas prejudiciais às culturas de milho e soja. As arquiteturas foram analisadas com base nas métricas matriz de confusão, precisão, revocação, média das precisões médias (*mAP)* com limiar de 0,5 e média das precisões médias com limiar variando 0,5 a 0,95 (*mAP 0,5:0,95*). Após o treinamento fez-se uso da quantização pós treinamento para comprimir os pesos neurais para os formatos inteiro de 8 bits (*int8)*, ponto flutuante de 16 bits (*fp16)*, e a conversão para *ONNX* (*Open Neural Network Exchange*). Estes pesos juntamente com as imagens de validação foram transferidos para plataforma *Raspberry Pi 4 model B*, onde, realizou-se a avaliação dos modelos com as métricas descritas. Em seguida foi feita a inferência nas imagens do conjunto de validação. Os melhores resultados de validação foram obtidos com a rede *Yolov5l* com quantização em *int8* com *mAP 0,5* de 0,815, precisão de 0,834, revocação de 0,772 e *mAP* 0,5:0,95 de 0,413, na inferência a rede obteve 1 imagem inferida a cada 5,685 segundos. O modelo com menor tempo de inferência foi a *Yolov5n* com pesos convertidos para *ONNX*, onde foi obtido uma imagem inferida a cada 0,376 segundo. Neste contexto, os resultados apontam que a abordagem proposta é viável para utilização em campo em plataformas tratorizados/robotizadas. Tem-se como vantagens a possibilidade da aplicação seletiva de defensivos, poupando o meio ambiente, reduzindo custos e aumentando a eficiência na erradicação de plantas daninhas prejudiciais às culturas de milho e soja.

Palavras-chave: Agroindústria, Plantas Daninhas, Aprendizado Profundo, Redes Neurais Convolucionais, Quantização Pós Treinamento.

Abstract

The increase in the demand for food as well as the rise in the price of the inputs for its production are factors that affect the agro-industry and the population. In this scenario, another challenge is the control of pests and diseases, including invasive species (weeds) in corn and soybean cultivars, which are the most produced grains in the world. Currently, identifying these plants is an unhealthy, laborious and error-prone process when carried out manually. When control is carried out by machines in a non-selective way, the farmer will be benefiting resistant species, making management inefficient and causing environmental impacts through the excessive use of herbicides. Therefore, there is a need for a low-cost embedded system for weed identification that can be used on tractors or directly on robotic platforms. Based on these challenges, this work proposed and explored the process of identifying and locating invasive plant species in corn and soybean crops using an artificial intelligence system with quantization and conversion of neural weights on low-cost hardware. For this purpose, computer vision associated with deep learning was used to train Yolov5 convolutional neural network architectures on a set of public images of weeds harmful to corn and soybean crops. The architectures were analyzed based on the following metrics: confusion matrix, precision, recall, mean average precision (mAP) with a threshold of 0.5 and mean average precision with a threshold ranging from 0.5 to 0.95 (mAP 0.5:0.95). After training, post-training quantization was used to compress the neural weights into 8-bit integer (int8) and 16-bit floating point (fp16) formats, and conversion to ONNX (Open Neural Network Exchange). These weights and the validation images were transferred to the Raspberry Pi 4 model B platform, where the models were evaluated using the metrics described. Inference was then made on the images in the validation set. The best validation results were obtained with the Yolov5l network with int8 quantization with mAP 0.5 of 0.815, precision of 0.834, recall of 0.772 and mAP 0.5:0.95 of 0.413. In inference, the network obtained 1 inferred image every 5.685 seconds. The model with the fastest inference time was Yolov5n with weights converted to ONNX, where an inferred image was obtained every 0.376 seconds. In this context, the results show that the proposed approach is viable for use in the field on tractor/robot platforms. The advantages include the possibility of selective application of defensives, which conserves the environment, reduces costs and increases efficiency in eradicating weeds that are harmful to corn and soybean crops.

Keywords: Agroindustry, Weeds, Deep Learning, Convolutional Neural Networks, Post-Training Quantization.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AP	Average Precision
CEPEA	Centro de Estudos Avançados em Economia Aplicada
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
CSPNET	Cross Stage Partial Network
EMBRAPA	Empresa Brasileira de Pesquisa Agropecuária
ESALQ	Escola Superior Luiz de Queiroz
FAO	Food and Agriculture Organization of the United Nations
FAOSTAT	Food and Agriculture Statistics
FLOPS	Floating Point Per Second
FP16	Float Point 16
FP32	Float Point 32
FPS	Frames por Segundo
GPU	Graphics Processing Units
INT8	Integer 8 bits
IOU	Intersection Over Union
mAP	Mean Average Precision
MB	Megabyte
PANET	Path Aggregation Network
PIB	Produto Interno Bruto
RNA	Redes Neurais Artificiais
RNC	Redes Neurais Convolucionais
SLIC	Simple Linear Iterative Clustering
USDA	United Stated Department of Agriculture
VANT	Veículo Aéreo Não Tripulado

Sumário

CAPÍTULO 01: INTRODUÇÃO

Neste capítulo é apresentado o contexto e relevância deste projeto de pesquisa, bem como a motivação, os objetivos gerais, específicos e a estrutura da tese.

1.1. Motivação

De acordo com o relatório da FAO é estimado que a população mundial atinja a marca de 9,7 bilhões de habitantes em 2050, consequentemente aumentando a demanda por alimentos. É estimado também que cerca de dois terços da população mundial viverá em áreas urbanas e um terço em áreas rurais (FAO, 2017).

Nesse contexto, o Brasil ocupa papel de destaque na agroindústria mundial, sendo o maior exportador de carne bovina do mundo e o segundo maior em exportação de grãos, de acordo com a EMBRAPA (EMBRAPA, 2021). Entretanto, um fator limitante para a produtividade agrícola é o controle de pragas e doenças como insetos e plantas daninhas, que podem causar perdas significativas. Segundo a EMBRAPA as plantas daninhas causam perdas diretas na produção via competição por recursos (sol, nutrientes do solo) e alelopatia (efeito que uma planta exerce sobre a outra via a liberação de compostos químicos) reduzindo a capacidade de produção das plantas, assim como perdas indiretas, como o aumento do custo de produção, dificuldade de colheita devido à infestação de plantas daninhas e depreciação na qualidade do alimento produzido. Caso não seja feito manejo das plantas daninhas as perdas podem chegar a 90% e com manejo atualmente as perdas são estimadas entre 13 e 15% para a produção de grãos (EMBRAPA, s.d).

Com base nesse cenário, a identificação de plantas daninhas é essencial para manter a produtividade alta, os custos de produção baixos e evitar o uso em excesso de herbicidas, prejudicando o meio ambiente. Porém, a identificação de pragas e doenças pelo método manual em especial as plantas daninhas é um trabalho sujeito a falhas humanas devido à grande extensão das plantações, também por ser um processo repetitivo e laborioso. Caso feito esse controle por máquinas de maneira não seletiva, o agricultor estará beneficiando espécies resistentes o que tornará esse manejo ineficiente.

Portanto, automatizar o processo de identificação de diferentes espécies de plantas daninhas para a aplicação localizada de defensivos agrícolas ou remoção via outros métodos é primordial. Nos últimos anos uma solução promissora para este desafio tem surgido que é a visão computacional associada com algoritmos de aprendizado de máquina (*machine learning*) e recentemente o aprendizado profundo (*deep learning)* bem como o uso de redes neurais convolucionais (*convolutional neurais networks)*.

Na literatura alguns trabalhos destacam-se relacionados a esse tema, como o uso de redes neurais convolucionais (*CNNs*) para a classificação de imagens de plantas daninhas (DOS SANTOS FERREIRA et al.,2017; SUH et al., 2018; TEIMOURI et al., 2018; TRONG et al., 2020) detecção de plantas daninhas em tempo real utilizando plataformas robotizadas (MILIOTO et al., 2018; PARTEL et al., 2019). Em um trabalho anterior (MARQUES JUNIOR; ULSON, 2018) foi proposto a comparação entre modelos de aprendizado profundo para a classificação de imagens, distinguindo solo, soja, plantas daninhas de folha larga e folha estreita, neste trabalho o melhor resultado de precisão foi obtido com a rede VGG16 com 96,5%. Entretanto, percebeu-se que além de classificar as plantas daninhas era necessário detectá-las, diferindo-as do plano de fundo, assim obtendo a localização das plantas e embarcando os algoritmos de inteligência artificial em plataformas robóticas ou tratorizadas, criando um sistema de aplicação automática de defensivos com aplicação localizada e seletiva.

Foi proposto então em outro trabalho (MARQUES JUNIOR; ULSON, 2021) a implementação e validação dos modelos de rede *Yolov5* (ULTRALYTICS, 2020) em um conjunto de imagens (*dataset*) para identificação e detecção de plantas daninhas resistentes ao herbicida Glifosato. Neste trabalho o melhor tempo de inferência (imagens analisadas pelo algoritmo de IA) foi de 62 FPS (frames por segundo) com a rede *Yolov5s*, já a melhor média das precisões médias com 0,5 de limiar (*mAP 0,5*) foi obtido com a rede *Yolov5m* com *mAP 0,5* de 0,77 e 26 *FPS*, inferindo localmente em um notebook *Samsung Odissey* com processador Core i7, 8 gigas de memória ram e placa gráfica Nvidia GTX 1050 com 4 gigas.

Apesar de promissor, o custo de implementação de um sistema como o descrito, tanto para a implantação em larga escala quanto para poucas máquinas, seria alto e inviável para os agricultores. Tendo em vista essa limitação e desafio proposto para

este trabalho foi explorar a implementação de redes neurais profundas dedicadas a detecção de objetos em um *hardware* de baixo custo e processamento.

Portanto, neste trabalho os distintos modelos da rede neural *Yolov5* foram treinados em um *dataset* público de plantas daninhas prejudiciais às culturas de milho e soja. Posteriormente, foi feita a quantização dos pesos neurais para as representações de ponto flutuante de 16 bits (*fp16)* inteiro de 8 bits (*int8)* e conversão desses pesos para o formato ONNX (*Open Neural Network Exchange*). Para a quantização os pesos neurais são convertidos do formato *pytorch* para *tensorflow* em seguida para as representações *tflite.fp16* e *tflite.int8*. Já a conversão para *ONNX* é feita por meio da função *torch.onnx.export* nativa do *pytorch.*

De pose dos pesos neurais, avaliações de desempenho foram realizadas a fim de constatar a viabilidade da abordagem proposta. Nos experimentos foi avaliada a inferência e identificação de plantas daninhas com os dados dos pesos neurais no módulo *Raspberry Pi 4 model B* de 4GB. De acordo com os nossos conhecimentos não existem trabalhos que exploraram a abordagem de quantização e conversão de pesos neurais para implantar um sistema de baixo custo aplicado na identificação de plantas daninhas prejudiciais nas culturas de milho e soja.

1.2. Objetivo Geral

Implementar uma abordagem baseada em redes neurais artificiais para identificação de plantas daninhas nas culturas de milho e soja a partir de imagens no espectro visível, possibilitando que esse sistema seja embarcado em veículos tratorizados.

1.3. Objetivos Específicos

Tem-se como objetivos específicos para este trabalho:

- Treinar os distintos modelos da *Yolov5* em um conjunto de imagens públicas de plantas daninhas prejudiciais as culturas de milho e soja, sendo essas plantas: Carrapicho de Carneiro, Rabo de Raposa, Caruru Gigante e Erva de Santiago Gigante;

- Gerar as matrizes de confusão e demais métricas para avaliação de assertividade desses modelos;

- Quantizar e converter os pesos neurais para os formatos *int8, fp16* e *ONNX* e tranferir os arquivos de pesos para o modulo *Raspberry Pi 4 model B* de 4GB.

- No modulo *Raspberry* gerar as métricas descritas no item anterior para avaliar assetividade, bem como avaliações de tempo de inferência por imagem para cada peso neural.

- De posse dos resultados de assertividade e inferência discutir os resultados e apontar quais modelos apresentaram melhor desempenho.

1.4. Estrutura da Tese

Os próximos capítulos desta tese estão organizados da seguinte maneira:

O capítulo 2 apresenta a revisão bibliográfica, contendo uma contextualização mais aprofundada a respeito da produção alimentícia levando em conta o Brasil e o mundo, posteriormente é dado enfoque no desafio principal deste trabalho que é o controle de plantas daninhas e quais são técnicas utilizadas. Em seguida os são explanados os conceitos, definições e abordagens no uso de *machine learning,* redes neurais artificiais e visão computacional nas tarefas de classificação, detecção de objetos e segmentação semântica. Em sequência as métricas para avaliação dos resultados dessas abordagens são descritos. Por fim o processo de quantização e conversão de pesos neurais é descrito seguido pelos principais trabalhos relacionados ao escopo da tese.

No capítulo 3 inicialmente é apresentado o fluxograma com os procedimentos metodológicos da tese, em seguida é descrita a metodologia utilizada desde a preparação do conjunto de imagens, arquitetura e características da rede neural *Yolov5*, o ambiente e dependencias utilizados para o treinamento dos modelos, por fim é descrito o processo de quantização e conversão dos pesos neurais, bem como apresentado *hardware* explorado nos experimentos e seus príncipais componentes.

No capítulo 4 são apresentados os resultados e discussões dos procedimentos experimentais, inicialmente são avaliados os resultados de matrizes de confusão, precisão, revocação média das precisões médias com limiar de 0,5 (mAP 0,5) e com limiar variando de 0,5 à 0,95 (mAP 0,5: 0,95) na representação de ponto fluturante e 32 *bits*. Em seguida as mesmas métricas apontadas são avaliadas na *Raspberry Pi 4 model B* de 4GB, com os pesos neurais de cada rede quantizado e convertido. Por fim é feito o levantamento de tempo de inferência de imagens para cada modelo na *Raspberry Pi 4 model B* de 4GB.

Ao final o capítulo 5 traz as conclusões, contribuições obtidas com este trabalho bem como perspectivas de melhoras para trabalhos futuros.

CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

Neste capítulo é apresentada a revisão bibliográfica da tese, partindo da contextualização do cenário de produção de grãos no brasil e no mundo, bem como destacar os principais desafios para esse setor, em especial o controle de plantas daninhas e as principais técnicas de controle adotadas.

Posteriormente, as principais definições sobre o aprendizado de máquina, visão computacional, redes neurais artificiais, métricas para avalição de modelos de redes neurais, quantização e de pesos neurais, bem como trabalhos de destaque no escopo desta tese são detalhados.

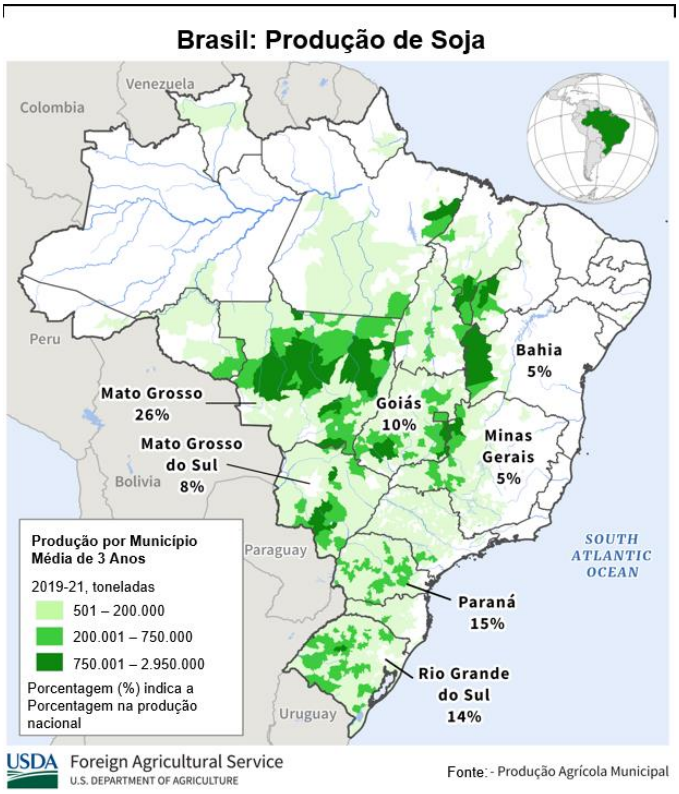
2.1. Produção Agrícola no Brasil e no Mundo

A agricultura exerce papel de destaque na economia Brasileira, de acordo com o CEPEA (Centro de Estudos Avançados em Economia Aplicada) da Esalq/USP (Escola Superior Luiz de Queiroz) no ano de 2021 o agronegócio teve participação de 27,4% no PIB (Produto Interno Bruto) brasileiro (CEPEA, 2021). Para o ano de 2023 de acordo com o relatório de Agosto de 2023 do Departamento de Agricultura dos Estados Unidos (USDA) o Brasil será o maior produtor de soja no mundo com 40% da produção mundial com 163 milhões de toneladas, para o milho o país é o terceiro maior produtor atrás de Estados Unidos com 32% e China com 23%, o Brasil tem 11% da produção mundial de milho com 129 mihões de toneladas.

Portanto a produção de grãos é fundamental para a segurança alimentar mundial, visto que além do consumo e geração de produtos diretos como óleo de soja, os grãos são utilizados como base alimentar animal estando também relacionados com a produção de carnes de aves, suínos e bovinos.

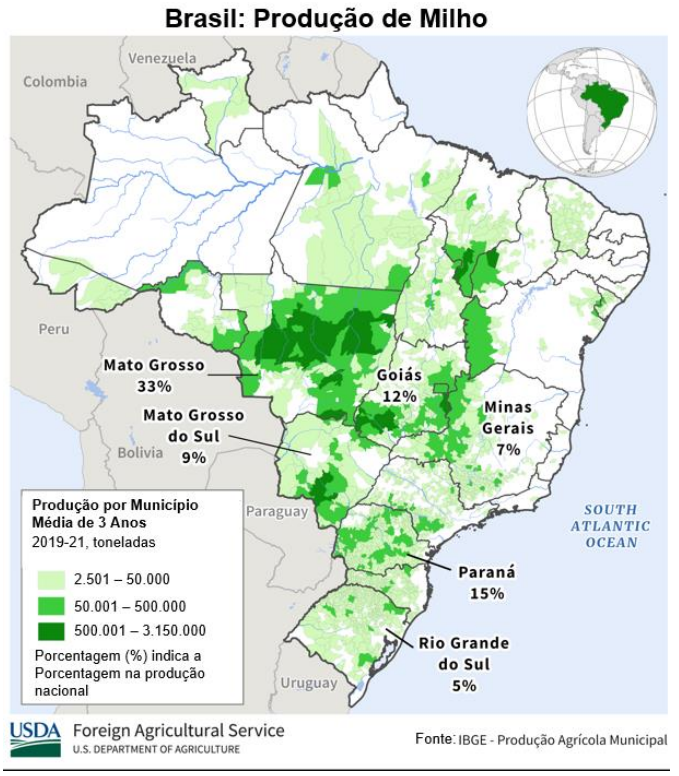
Em relação a produção de soja os estados como maior produção são Mato Grosso com 26%, Paraná com 15% e Rio Grande do Sul com 14%. Para a produção de Milho os estados líderes são Mato Grosso com 33%, Paraná com 15% e Goiás com 12%. Os dados apresentados são sintetizados nas figuras 1 referente a produção nacional de soja e figura 2 referente a produção nacional de milho.

Figura 1 - Produção de Soja no Brasil



Fonte: USDA (2023), adaptado pelo autor.

Figura 2 - Produção de Milho no Brasil



Fonte: USDA (2023), adaptado pelo autor.

Nesse cenário se por um lado o Brasil ocupa posição de liderança na produção de soja e é o terceiro maior produtor de milho no mundo, por outro enfrenta o desafio de manter os custos de produção competitivos. Além da necessidade de implementar técnicas de manejo eficientes para o controle de pragas e doenças, especialmente plantas daninhas que podem causar perdas de produção significativas.

2.2. Plantas Daninhas

De acordo com a EMBRAPA (Empresa Brasileira de Pesquisa Agropecuária) um dos fatores que mais afeta o rendimento e a produtividade agrícola é a ocorrênciade plantas daninhas, sendo causadoras de efeitos diretos na cultura plantada como a interferência (ação conjunta da competição e da alelopatia, que é a capacidade das plantas produzirem substâncias prejudiciais ou benéficas para outras plantas) e consequentemente a perda de rendimento, além de efeitos indiretos, como aumento do custo de produção, dificuldades na colheita, depreciação na qualidade do produto, e hospedagem de pragas e doenças. As perdas estimadas ocasionadas pelas plantas daninhas podem, em casos em que não é feito controle algum, alcançar 90%, normalmente estas perdas em média ficam entre 13 a 15% na produção de grãos (EMBRAPA, s.d).

Plantas daninhas englobam todas as plantas que interferem no crescimentoda cultura cultivada, em sua maioria são mais rusticas e adaptáveis a mudanças climáticas, sendo consideradas plantas indesejadas. Este tipo de planta costuma crescer em condições adversas, como ambientes secos ou úmidos, com temperaturas baixas ou elevadas e variados tipos de solos, também apresentarem capacidade de produzir sementes em abundância, com variadas formas de dispersão e a resistência ao ataque de pragas e doenças (EMBRAPA, s.d).

Das 350.000 espécies conhecidas de plantas, apenas 3.000 são cultivadas e aproximadamente 250 são universalmente consideradas plantas daninhas, dasquais cerca de 40% pertencem a apenas duas famílias: Poaceae (gramíneas) eAsteraceae (compostas). Por causa do seu caráter competitivo, as plantas daninhas garantem sua perpetuação por meio de dormência e germinação desuniforme das sementes. Estas habilidades conferem um difícil controle das espécies invasoras pelo fato de não germinarem todas ao mesmo tempo, mesmoem condições ideais de temperatura,

umidade e luz (EMBRAPA, s.d). O desenvolvimento das plantas invasoras é rápido, sendo que algumas espécies conseguem se reproduzir também através de bulbos, tubérculos, rizomas(caules) e enraizamento, atingindo maturidade em pouco tempo.

2.2.1. Técnicas Tradicionais para o Manejo de Plantas Daninhas

Por se tratar de um desafio que tem acompanhado a atividade agrícola desde seu início algumas técnicas tradicionais de manejo das plantas daninhas já se perpetuaram sendo estas técnicas:

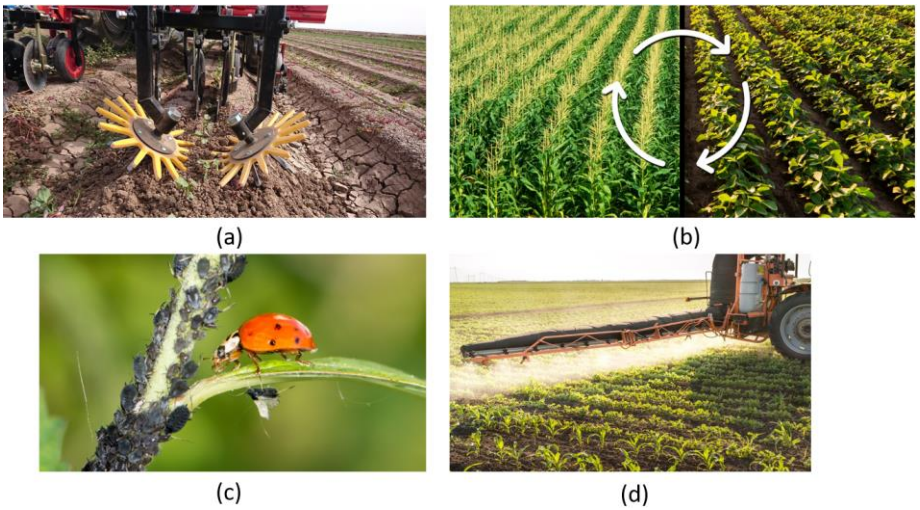
Controle Mecânico: Inicialmente era feita de maneira manual onde o agricultor realizava a capinação das plantas daninhas. Atualmente o controle mecânico usa implementos mecânicos (grades, discos ou dentes) em um trator para cortar ou arrancar as plantas daninhas (GROW, 2021).

Controle Cultural: O controle cultural de plantas daninhas refere-se a qualquer técnica que envolva a manutenção de condições de campo de modo que as plantas daninhas sejam menos propensas a se estabelecer e/ou aumentar em número. Como exemplo de controle cultural tem-se a rotação de culturas que evita o sobrepastoreio da mesma cultura, usando espécies forrageiras competitivas bem adaptadas e mantendo boa fertilidade do solo (FORAGES, 2021).

Controle Biológico: O controle biológico (biocontrole) envolve a introdução de inimigos naturais (insetos, ácaros e patógenos) de uma planta daninha alvo, assim reduzindo a densidade dessa planta. O biocontrole não erradica as plantas daninhas, mas pode reduzir as populações a níveis aceitáveis, ou suprimi-las, onde podem ser controladas em combinação com outros métodos (NSW DPI, 2022).

Controle Químico: O controle químico de plantas daninhas refere-se a qualquer técnica que envolva a aplicação de um produto químico (herbicida) nas plantas daninhas ou no solo para controlar a germinação e crescimento das espécies invasoras. Atualmente é a técnica de manejo mais utilizada, sendo que existem diversos herbicidas com distintos princípios ativos desenvolvidos para este tipo de manejo (FORAGES, 2021). Na figura 3 são ilustradas as quatro principais técnicas para o manejo e controle de plantas daninhas.

Figura 3 - Técnicas Tradicionais para o Controle de Plantas Daninhas (a) Controle Mecânico, (b) Controle Cultural, (c) Controle Biológico, (d) Controle Químico



Fonte: Elaborado pelo autor

Com o advento da mecanização agrícola foi possível automatizar o controle de plantas daninhas com destaque para o controle químico com herbicidas comerciais. Entretanto, atualmente na grande maioria dos casos o herbicida é aplicado em toda área plantada, sendo um dos maiores gastos na produção em larga escala de grãos como o milho e a soja, além de causar impactos ambientais negativos pelo uso excessivo de herbicida. Outra limitação é que essa aplicação não é seletiva, impossibilitando a troca de herbicida caso uma planta daninha resistente seja identificada. Como alternativas promissoras para este desafio técnicas de aprendizado de máquina (*machine learning*), aprendizado profundo (*deep learning*) associados com a visão computacional vem sendo estudadas e implementadas em campo para o controle seletivo de plantas daninhas.

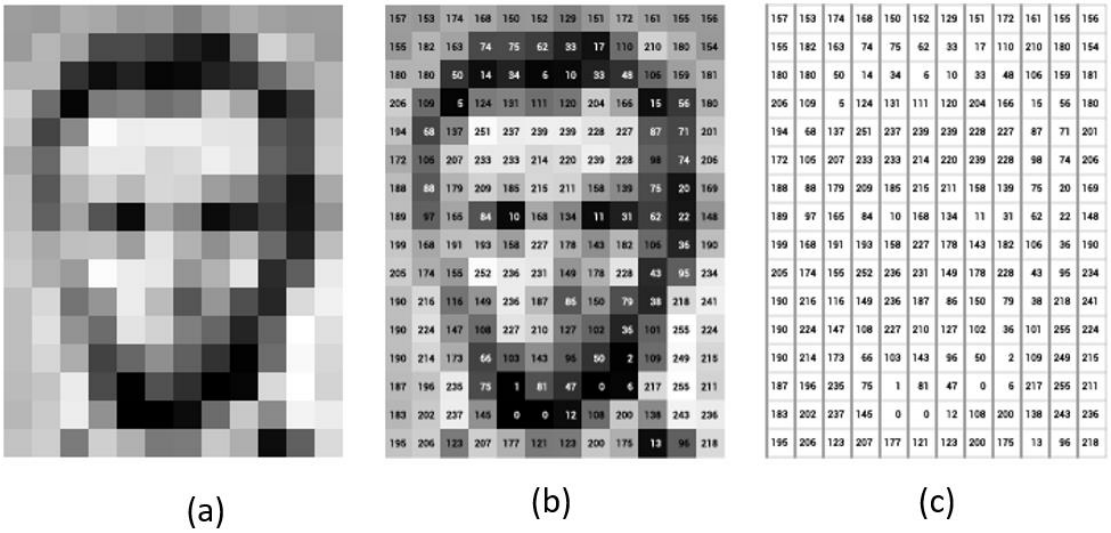
2.3. Visão Computacional

A visão é o sentido mais utilizado pelo ser humano, logo a identificação de diferentes cores, texturas e formas de objetos é uma tarefa comum executada constantemente. Por outro lado, representar uma imagem e seu conteúdo para um computador é um desafio, visto que cada informação da imagem precisa ter um valor numérico para que possa ser processado pelo computador. Analogamente ao que ocorre com a visão humana na visão computacional recebe-se dados de diferentes

sensores ópticos como câmeras, *scanners* e *lasers*, esses sinais recebidos são digitalmente processados.

A visão computacional surgiu com o objetivo principal de extrair e representar o conteúdo de imagens e vídeos como é feito pela visão humana. Na visão computacional a imagem digital poder ser descrita em tons de cinza, onde sua altura e largura é determinada pelo número de pixels, além de uma dimensão. Logo uma imagem pode ser representada como uma matriz de pixels variando entre 0 (preto) à 255 (branco), ou esses valores também podem ser normalizados entre 0 e 1 dividindo seu valor por 255. Na figura 4 é apresentado uma imagem em tons de cinza bem como a variação no valor de seus pixels de 0 à 255.

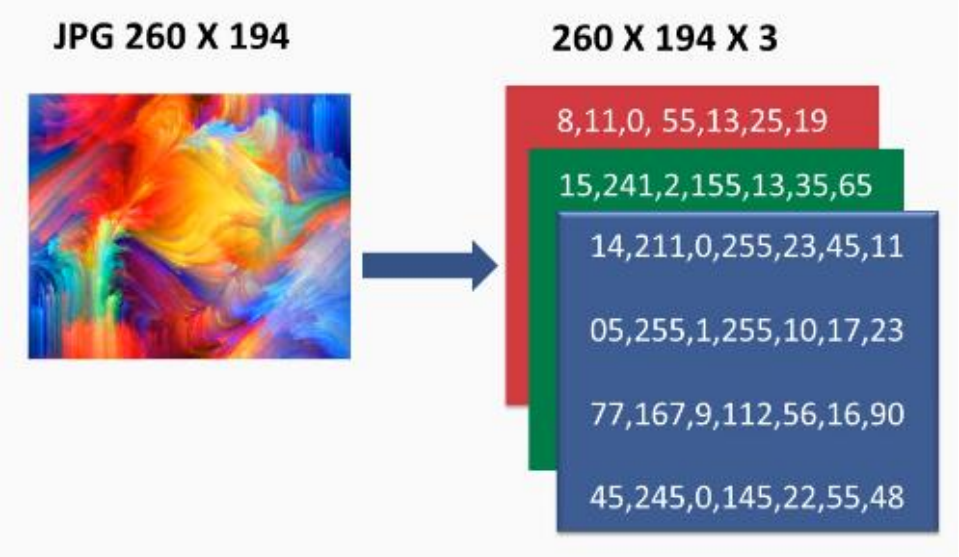
Figura 4 - Representação de Imagem em Tons de Cinza (a) Figura Pixelizada, (b) Figura Pixelizada com o Valores do Pixels, (c) Valores dos Pixels



Fonte: AI.Stanford (2022), adaptado pelo autor.

Outra representação muito utilizada em visão computacional são imagens coloridas conhecidas como RGB (*Red, Green, Blue)* ou vermelho, verde e azul, onde a imagem tem a altura e largura determinada pelo número de pixels, além de três dimensões, sendo cada cor uma dimensão. Na figura 5 é apresentado um exemplo de uma imagem colorida e as matrizes de valores para cada dimensão.

Figura 5 - Representação de Imagem em Colorida e suas Dimensões



Fonte: Ramo (2019)

Em uma imagem colorida cada canal do RGB possui valores com intensidades diferentes, uma vez que cada parte da imagem é mais ou menos representativa de acordo com a cor. A partir da obtenção dos valores de cada pixel em uma imagem a visão computacional possibilitou o desenvolvimento de técnicas mais sofisticadas para a extração de características como a identificação de texturas, formas e cores bem como a implementação de filtros para modificar ou melhorar propriedades das imagens.

As informações obtidas por meio do processamento digital de sinais e da visão computacional servem de entrada para algoritmos de aprendizado de máquina, que por sua vez identificam padrões e características tendo relevância para diversas aplicações como transporte, agricultura, medicina segurança etc.

2.4. Aprendizado de Máquina

O aprendizado de máquina é um subcampo da inteligência artificial, que é amplamente definido como a capacidade de uma máquina imitar o comportamento humano inteligente (MIT, 2021). Os sistemas de inteligência artificial são usados para executar tarefas complexas de maneira semelhante a abordagem humana para resolver problemas. Existem diversas implementações e tipos de algoritmos de aprendizado de máquina, porém, primordialmente pode-se dividir o aprendizado de máquina em três subcategorias:

Aprendizado de máquina supervisionado: Neste aprendizado os modelos (arquiteturas de aprendizado de máquina) são treinados em conjuntos de dados rotulados, assim, os modelos aprendem as características do conjunto de dados e tornam-se mais precisos de acordo com a quantidade e qualidade das informações. Os algoritmos de aprendizado supervisionado podem ser divididos em duas categorias:

- Classificação: Algoritmos de classificação tem por objetivo categorizar sua saída, por exemplo, um algoritmo treinado com fotos de plantas rotuladas em diferentes situações, todas rotuladas por pessoas. Nesse cenário a máquina aprenderia distintas maneiras de identificar plantas por conta própria.

- Regressão: Os algoritmos de regressão têm por objetivo comparar o valor estimado com o valor real, por exemplo, um algoritmo treinado com dados de uso de herbicida relacionando o uso ao tamanho da área e cultura cultivada. Quando o algoritmo for realizar a predição da quantidade de herbicida que deverá ser usado em uma nova área, o valor obtido deverá ser próximo ou igual ao valor que será utilizado na prática.

O aprendizado de máquina supervisionado é a abordagem mais utilizada atualmente, especialmente em tarefas diretamente relacionadas com imagens como a classificação, detecção de objetos e segmentação semântica.

Aprendizado de máquina não supervisionado: Neste aprendizado o modelo procura padrões em dados não rotulados, os algoritmos são baseados príncipalmente na clusterização de informações de acordo com suas características. No aprendizado de máquina não supervisionado pode-se encontrar padrões ou tendências que as pessoas não procuram explicitamente. Por exemplo, o algoritmo pode analisar imagens de plantações e identificar padrões não identificados anteriormente (MIT, 2021).

Aprendizado de Máquina por Reforço: Neste aprendizado é explorado a tentativa e erro para tomar a melhor ação, estabelecendo um sistema de recompensa, o aprendizado por reforço pode ser:

- Positivo: O aprendizado por reforço positivo tende a maximizar o estimulo quando um evento desejado ocorre, por exemplo, em robótica um evento positivo seria pegar um objeto e leva-lo do ponta A ao ponto B no menor tempo ou com menos passos para executar o processo.

- Negativo: O aprendizado por reforço negativo tende a maximizar o estimulo quando um evento indesejado é parado ou evitado. Por exemplo o braço robótico não deve pegar o objeto do ponto B e leva-lo ao ponto A, caso esse processo não ocorra ou seja parado em estágios iniciais, o algoritmo vai estimular esse comportamento.

O aprendizado por reforço é utilizado para treinar modelos para jogos ou na direção de veículos autônomos, sendo que alguns desses veículos são empregados na área agrícola, informando à máquina quando ela tomou as decisões corretas, o que a auxilia a aprender ao longo do tempo quais ações devem ser tomadas (MIT, 2021).

2.5. Redes Neurais Artificiais

As redes neurais são uma classe específica e comumente usada de algoritmos de aprendizado de máquina. As redes neurais artificiais (RNA) em sua concepção foram baseadas no neurônio biológico do cérebro humano, no qual milhões de nós de processamento são interconectados e organizados em camadas. O neurônio biológico pode ser dividido em quatro partes principais:

Dendritos: Os dendritos são terminais de recepção, responsáveis por receber impulsos nervosos (entradas).

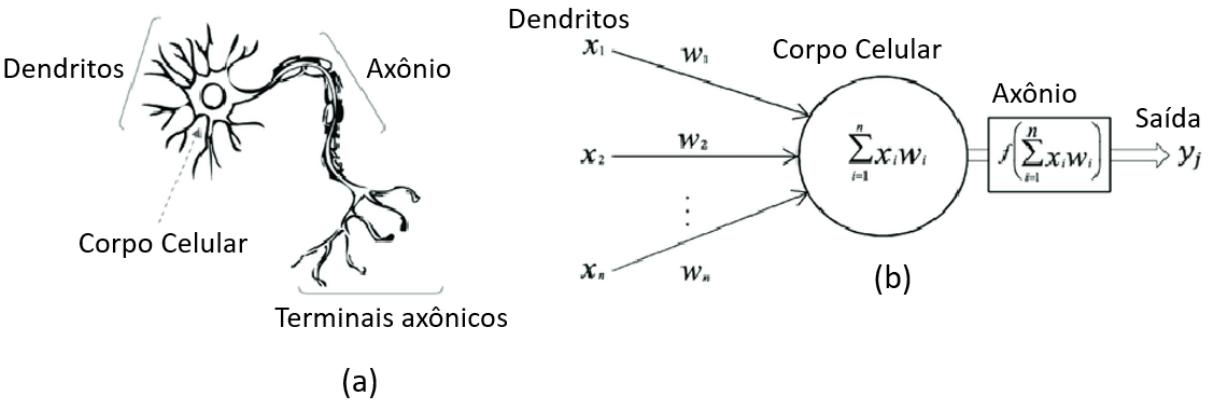
Corpo Celular: O corpo celular é responsável por processar os sinais de entrada e transmitir o impulso nervoso para o axônio.

Axônio: O axônio é responsável por receber e transportar o impulso nervoso até os terminais axônicos.

Terminal Axônico: Os terminais axônicos transmitem o impulso nervoso para os dendritos de outros neurônios (DEEP LEARNING BOOK, 2022).

Os primeiros neurônios matemáticos artificiais datam de 1943, quando o neurofisiologista *Warren McCulloch* e o matemático *Walter Pitts* escreveram um artigo sobre como os neurônios artificiais poderiam funcionar e para isso eles modelaram uma rede neural usando circuitos elétricos (W, S. MCCULLOCH, 1943). A rede neural de *McCulloch-Pitts* tinha dois tipos de entradas excitatória ou inibitória, sendo que os valores de entrada do modelo poderiam ser 0 ou 1. Na figura 6 são apresentados o neurônio biológico e seus elementos, bem como neurônio matemático proposto por *McCulloch-Pitts*, onde é feita uma analogia e comparação dos elementos de ambos os neurônios.

Figura 6 - Representação do Neurônio Biológico (a) Neurônio Matemático de McCulloch- Pitts (b)



Fonte: Meng et.al (2020), adaptado pelo autor.

Com base no item (b) da figura 6 o funcionamento do neurônio matemático proposto por *McCulloch-Pitts* pode ser dividido em:

Sinais de Entrada: Semelhante aos dendritos que recebem os sinais externos de entrada, neste caso os dados de {X1, X2, Xn}, sendo que cada dado terá o seu peso sináptico {W1, W2, Wn}.

Pesos Sináptico: Representados por {W1, W2, Wn}, são valores ponderados para cada entrada da rede, cada entrada terá um peso sináptico distinto podendo ser mais ou menos excitatório.

Combinador Linear: O combinador linear {Σ} soma os sinais de entrada e os multiplica pelos pesos sinápticos com o intuito de gerar uma potência de ativação. Seu funcionamento é análogo ao corpo celular do neurônio.

Função de Ativação: Representada por {*f*} a função de ativação define baseada na saída do combinador linear qual será o sinal de saída do neurônio, podendo ser inibitório 0 ou excitatório 1. Seu funcionamento é análogo ao axônio do neurônio.

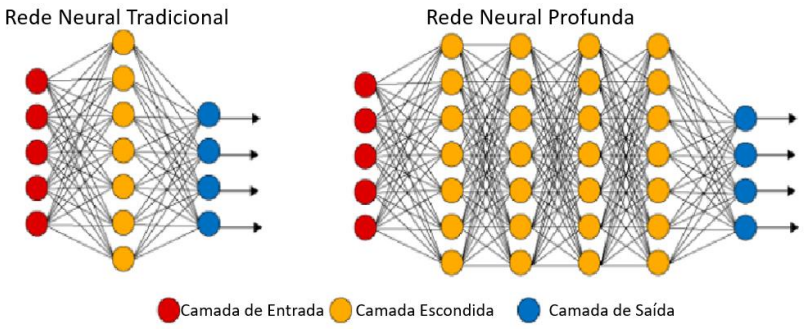
Sinal de saída: O sinal de saída {y} assemelha-se aos terminais axônicos transmitindo o impulso com o valor de saída como entrada para outros neurônios matemáticos interligados.

Em síntese uma rede neural é dividida entre a camada de entrada, uma ou mais camadas ocultas e camada de saída com o valor resultante. A partir do neurônio matemático as redes neurais tornaram-se cada vez mais profundas sendo implementadas em problemas mais complexos envolvendo várias informações de entrada, dentre essas informações destaca-se o uso de imagens.

2.6. Aprendizado Profundo

As redes de aprendizado profundo são redes neurais com muitas camadas, uma rede neural tradicional é composta por 3 camadas divididas em entrada (*input*) camada oculta (*hidden layer)* e saída (*output*), já uma rede neural profunda pode ter dezenas ou até centenas de camadas, aumentando sua capacidade de extração caracteristicas mais complexas. No aprendizado profundo camadas de neurônios matemáticos são usados para processar dados, compreender a fala humana e reconhecer objetos visualmente, como nas redes neurais tradicionais a primeira camada recebe os dados de entrada, enquanto a última retorna a saída (DEEP LEARNING BOOK, 2022). Na figura 7 é apresentada uma comparação entre uma rede neural tradicional e uma rede neural profunda.

Figura 7 - Estrutura de uma Rede Neural Tradicional e uma Rede Profunda

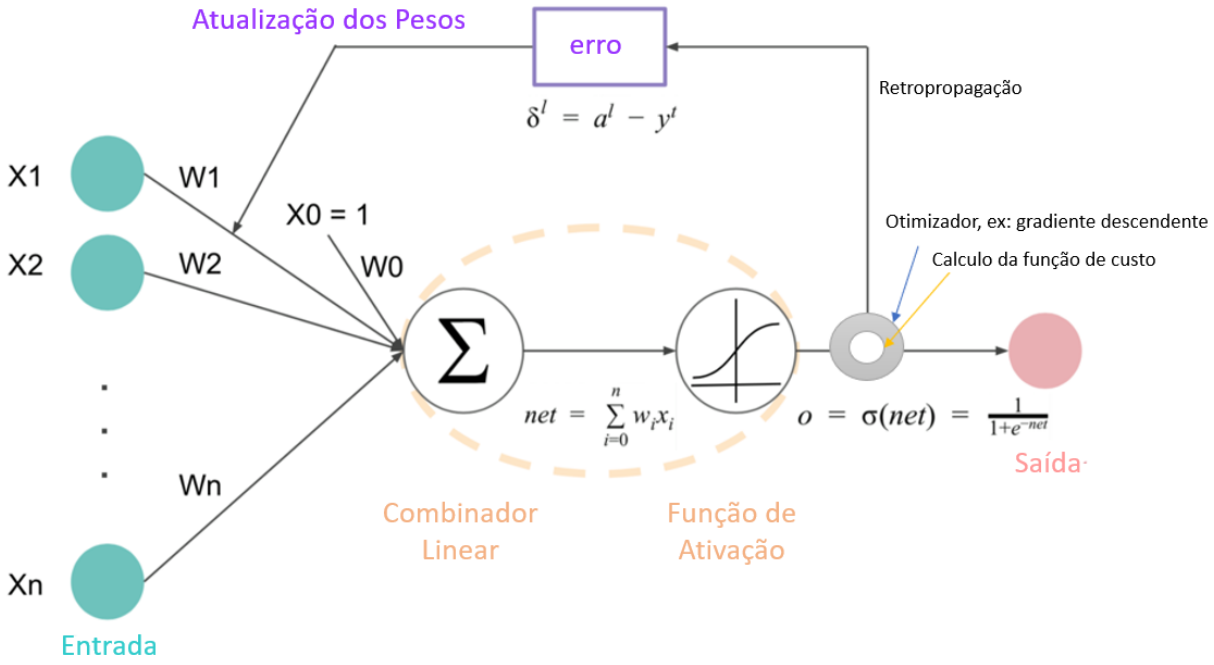


Fonte: Deep Learning Book (2022), adaptado pelo autor.

Outros dois avanços importantes relacionados com as redes neurais profundas foi a implementação do algoritmo de retropropagação (*Backpropagation*) (Rumelhart et.al, 1986) nesse algoritmo antes da camada de saída da rede neural foi implementada uma função de erro. Essa função de erro calcula a diferença entre o valor estimado pela rede e o valor real durante o treinamento, buscando minimizar o erro os sinais são retro propagados para as camadas iniciais da rede neural, onde os pesos sinápticos são atualizados com novos valores.

O segundo avanço foi a implementação do gradiente descendente (Werfel et.al, 2003) o que introduziu a implementação de otimizadores em arquiteturas de redes neurais. Na prática antes da camada de saída da rede é calculado o gradiente descente e com base no valor do gradiente é feita a retro propagação e atualização dos pesos sinápticos. Na figura 8 é apresentada a integração dos algoritmos de retropropagação e gradiente descendente em uma arquitetura de rede neural.

Figura 8 - Rede Neural com Retro propagação e Otimização via Gradiente Descendente



Fonte: Stackexchange (2018), adaptado pelo autor.

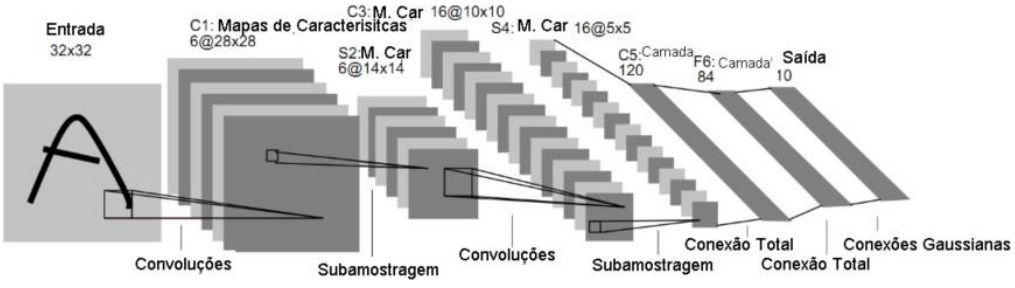
Assim como nas redes neurais tradicionais, as redes de aprendizado profundo foram modeladas com base no cérebro humano o que tem potencializado seu uso para tarefas de difícil abstração via algoritmos como classificação de imagens, detecção de objetos e o processamento de linguagem natural.

2.7. Redes Neurais Convolucionais

Uma rede neural convolucional, ou *CNN*, é uma rede neural de aprendizado profundo utilizada principalmente para extrair características (*features*) de imagens. Atualmente arquiteturas baseadas em *CNNs* tornaram-se estado da arte em diversas aplicações visuais sendo também implementadas no processamento de linguagem natural para classificação de texto (*DEEPAI*, s.d). O trabalho precursor demonstrando a aplicabilidade das Redes Neurais Convolucionais (RNC) foi realizado por *Yann Lecun*, nos laboratórios *AT&T BellLabs* em 1988. Em seu experimento o autor fez uso das redes neurais convolucionais para reconhecimento de caracteres escritos à mão, por meio deste experimento foi validada a hipótese que as redes neurais convolucionais poderiam ser utilizadas em aplicações práticas (LECUN, Y. et al., 1988).

Na figura 9 é apresentada a estrutura da rede convolucional proposta por *Yann Lecun.* Nessa arquitetura, tem-se na entrada uma imagem em preto e branco de 32x32 pixels, em seguida a imagem passa por um filtro convolucional, que é uma operação matemática feita por um sistema ou filtro linear invariante ao tempo. Após o filtro convolucional, é aplicada uma técnica denominada subamostragem (*subsampling*), que reduz o tamanho da imagem, esse processo se repete até a última camada que é uma rede neural totalmente conectada (*fully connected*), com o respectivo número de saídas de acordo com o número de classes.

Figura 9 - Rede Neural Convolucional proposta por Lecun, Y



Fonte: Lecun (1988), adaptado pelo autor.

Atualmente as redes neurais convolucionais alcançaram resultados notáveis em *datasets* renomados para tarefas de classificação de imagens (KRIZHEVSKY et al., 2012; SIMONYAN et al., 2015; SZEGEDY et al., 2015; HE et al., 2016) e detecção de objetos (SERMANET et al., 2014; REN et al., 2016; REDEMON et al., 2016).

2.8. Classificação, Detecção de Objetos e Segmentação Semântica em Imagens

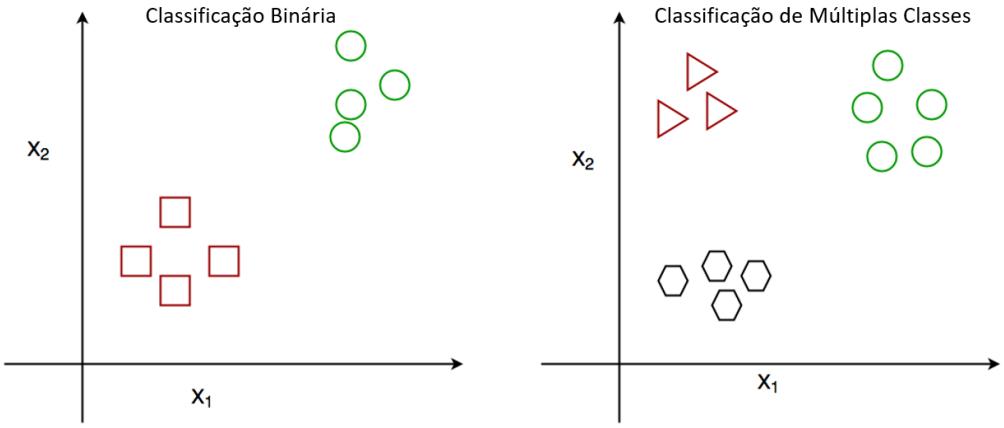
2.8.1. Classificação

A classificação de objetos foi o primeiro problema onde a visão computacional juntamente com algoritmos de *deep learning* foram aplicados. O trabalho precursor nesse campo foi o de (*LECUN, Y*. et al., 1988), onde o autor fez uso das redes neurais convolucionais para classificar dígitos escritos a mão. Outro marco importante relacionado a classificação de imagens foi o trabalho de (*KRIZHEVSKY* et al., 2012) usando redes neurais convolucionais no *ImageNet* (*DENG* et al., 2009) que foi um *dataset* lançado em 2009 e serviu de *benchmark* para diversos algoritmos de aprendizado de máquina. A classificação de objetos pode ser dividida em:

Classificação Binária: Na classificação binária o modelo categoriza os dados em duas classes, por exemplo se tem ou não uma planta na imagem, ou com base em outros atributos como plantas daninhas e cultura, separa os dados em duas classes, praga e plantação.

Classificação de Múltiplas Classes: Na classificação de múltiplas classes o modelo irá categorizar os dados em pelo menos duas classes distintas, por exemplo se tem uma planta dainha da espécie A, espécie B, ambas ou nenhuma das duas na imagem. Outro exemplo seria com base em alguns atributos classificar o estado que a cultura plantada está como germinação, vegetativo e reprodutivo. Na figura 10 são ilustradas as classificações binária e de múltiplas classes.

Figura 10 - Classificação Binária e de Múltiplas Classes



Fonte: Geeksforgeeks (2022), adaptado pelo autor.

2.8.2. Detecção de Objetos

A detecção de objetos é uma das principais áreas pesquisadas na visão computacional, pode ser definida como uma técnica de visão computacional que permite identificar e localizar objetos em uma imagem ou vídeo. Nos últimos anos diversas têm sido as abordagens e arquiteturas de redes neurais profundas aplicadas na detecção de objetos, sendo que o *COCO* (*Common Objects in Context*) *dataset* é o conjunto de imagens e métricas de referência na avaliação destes modelos com 80 classes diversas (*EVERINGHAM* et al., 2015). Na figura 11 são exemplificados a classificação de imagens e a detecção de objetos que envolve localizar e classificar os objetos de interesse.

Figura 11 - Classificação, Classificação com Localização e Detecção de Objetos

Fonte: MIT (2023), adaptado pelo autor.

2.8.3. Segmentação Semântica

Na segmentação semântica de imagens os pixels referentes a um objeto passam a representar a classe do objeto, por exemplo todos os pixels em uma imagem que compõem uma árvore ou um carro passam a pertencer a esta classe. A segmentação semântica é amplamente utilizada em aplicações médicas para diagnóstico de doenças e na direção autônoma de veículos, por permitir delimitar a área e perímetro dos objetos, entretanto seu processamento é maior se comparado a uma rede de detecção de objetos. Atualmente existem diversas arquiteturas de redes para segmentação semântica sendo a *U-Net* (*Ronneberger* et. al, 2015) a rede mais implementada. Na figura 12 é apresentado um exemplo entre uma imagem real, e uma máscara semântica anotada que é utilizada como entrada para a rede de segmentação semântica.

Figura 12 - Imagem de Entrada e Máscara de Entrada para Segmentação Semântica

Fonte: MIT (2023), adaptado pelo autor.

2.9. Quantização de Pesos Neurais e Conversão para ONNX

Durante o treinamento uma rede neural aprende uma série de padrões para que possa realizar a tarefa para a qual projetada, esses parâmetros são armazenados em um arquivo de pesos que contém as características principais adquiridas no processo de aprendizagem pela rede neural bem como a estrutura dessa rede. Por padrão os pesos são representados no formato de ponto flutuante de 32 bits *float32*, ou seja, são valores de ponto flutuante de 32 *bits* que permitem alta precisão e acurácia dos modelos.

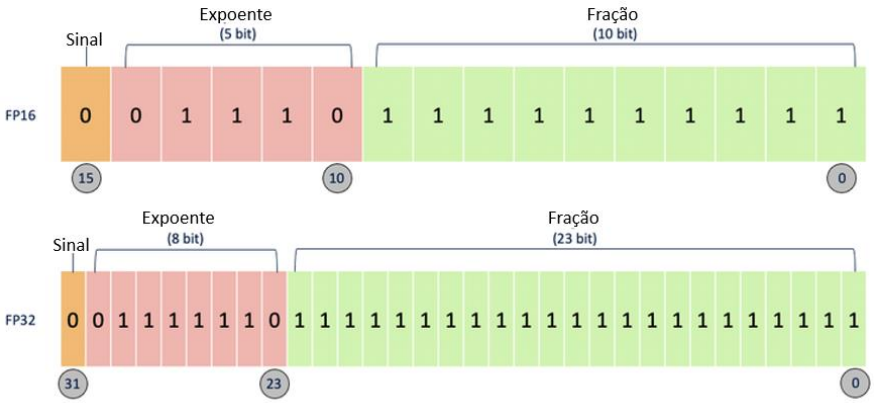
Entretanto algumas limitações que os pesos em formato *float32* causam na inferência são o alto consumo energético e a maior demanda por processamento, muitas vezes resultando na necessidade do uso de *GPUs* (*Graphics Processing Unit*) placas gráficas para processar as imagens localmente, assim, causando aumento no custo do *hardware* de inferência (ALLABOUTCIRCUITS, 2022). Uma alternativa para reduzir essa demanda é aplicar quantização nos pesos neurais, a quantização é o processo de reduzir tamanho dos pesos neurais representando estes pesos em formatos menores.

2.9.1. Quantização para Ponto Flutuante de 16 Bits

A quantização em o ponto flutuante de 16 bits fp16 (*floating point 16*) é explorada quando tem-se por objetivo comprimir os pesos neurais sem sacrificar a precisão do modelo consideravelmente. Enquanto que em ponto flutuante de 32 bits tem-se 1 bit para o sinal 8 bits para o expoente e 23 para a fração também conhecida como precisão ou mantissa. Em ponto flutuante de 16 *bits* tem-se 1 bit para o sinal, 5 *bits* para o expoente e 10 *bits* para a fração. A representação de dados em *fp16* é conhecida também como meia precisão visto que a fração

decresce de 23 bits em *fp32* para 10 em *fp16*. Na figura 13 são apresentadas as representações de valores em ponto flutuante de 16 e 32 bits.

Figura 13 - Representação de Dados em FP16 e FP32

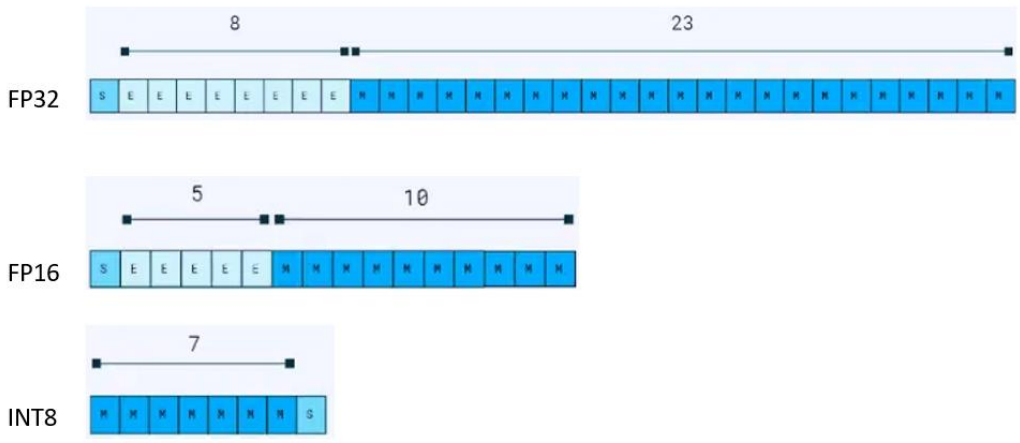


Fonte: Ahmed. (2021), adaptado pelo autor.

2.9.2. Quantização para Inteiro de 8 Bits

A quantização para inteiros de 8 bits int8 (*integer 8 bits*) é explorada quando se tem como objetivo embarcar modelos de redes neurais em equipamentos (*hardware*) de menor processamento, nessa representação os dados são divididos em 1 bit para o sinal e 7 bits para a fração ou mantissa, o que pode reduzir consideravelmente a precisão do modelo. Uma técnica utilizada para contornar a perda de precisão para o em *int8* é usar um *dataset* representativo, baseado nas imagens de treinamento para calibrar os pesos em *int8*. Na figura 14 são comparados os dados em fp32, f16 e int8 em relação a extensão desses valores, a precisão bem como a distribuição dos bits.

Figura 14 - Representação de Dados em FP32, FP16 e INT8



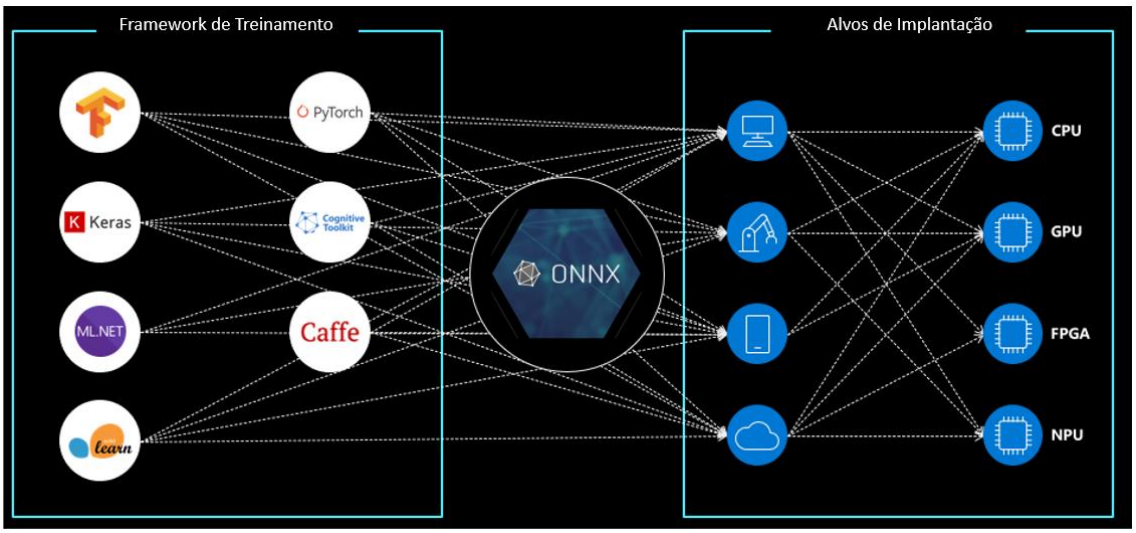
Fonte: DECI.AI (2023), adaptado pelo autor.

2.9.3. Conversão para ONNX

O *ONNX* (*Open Neural Network Exchange*) ou Trocador de Rede Neural Aberto é definido como um formato aberto para representar modelos tradicionais e de aprendizado profundo, o *ONNX* surgiu para permitir a interoperabilidade entre diferentes *frameworks* de treinamento de rede neural como *Tensorflow, Pytorch, Keras* em diferentes dispositivos como *CPUs, GPUs, FPGAs* etc (Microsoft, 2019).

Dessa maneira o *ONNX* possibilita implantar o mesmo modelo de Inteligência Artificial em diferentes dispositivos evitando a incompatibilidade de dependências e *hardware*, solucionando um dos principais desafios para implantar arquiteturas de *machine learning*. Na figura 15 é ilustrado a interoperabilidade do ONNX para diferentes *frameworks* e dispositivos.

Figura 15 - Conversão de Pesos Neurais para ONNX e Implantação em Diferentes Dispositivos



Fonte: Microsoft AI@Edge (2019), adaptado pelo autor.

Outra vantagem do *ONNX* é poder utilizar o *ONNX Runtime Inferece* (Inferência em tempo de execução), o *runtime inferece* é um mecanismo de inferência de alto desempenho para implantar modelos com pesos neurais em *ONNX* em ambiente de produção, sendo otimizado tanto para aplicações em nuvem quanto em dispositivos embarcados (Microsoft, 2022).

2.10. Métricas para Avaliação de Resultados

Para este trabalho o desafio proposto foi identificar plantas daninhas, para isso foram utilizadas as redes neurais Yolov5, sendo que a arquitetura Yolov5 está diretamente relacionada com a tarefa de detecção de objetos. Por se tratar de uma tarefa frequentemente explorada algumas métricas para a avaliação de performance e resultados dos modelos de *machine learning* para a detecção de objetos já estão consolidadas sendo as principais métricas:

Verdadeiro Positivo (*True Positive*): Refere-se à quantidade de objetos positivos aos quais o modelo classificou como positivo, ou seja, corretamente classificou.

Falso Negativo (*False Negative*): Refere-se à quantidade de objetos dos quais o modelo incorretamente classificou, ou seja, a classificação deveria ser positiva, mas o modelo classificou como negativa.

Falso Positivo (*False Positive*): Refere-se à quantidade de objetos dos quais o modelo incorretamente classificou, ou seja, a classificação deveria ser negativa,mas o modelo classificou como positiva.

Verdadeiro Negativo (*True Negative*): Refere-se à quantidade de objetos negativos dos quais o modelo classificou como negativo, ou seja, corretamente classificado (PAPERSPACE, 2020).

Precisão (*Precision*): A precisão é calculada como a relação entre o número deobjetos positivos corretamente classificados em relação ao total de objetos positivos classificados, seja corretamente ou incorretamente. Essa é uma métrica constantemente avaliada visto que está diretamente relacionada com a capacidade do modelo avaliado em classificar objetos corretamente. A equação que define a precisão é apresentada em (1).

$$Precisão = \frac{Verdadeiro_{positivo}}{Verdadeiro_{positivo} + Falso_{positivo}} \quad (1)$$

Revocação (*Recall*): A revocação é calculada como a relação entre o número de objetos positivos corretamente classificados em relação ao total de objetos positivos no *dataset*. Diferente da precisão e revocação é um importante indicador de quantos objetos estão sendo identificados como positivos, mesmo que incorretamente. A revocação juntamente com a precisão são métricas complementares, uma vez que um modelo pode apresentar uma alta revocação e baixa precisão indicando a existência de muitos falsos positivos, quando o modelo tem uma alta precisão e baixa revocação essa métrica indica que o modelo é preciso, porém consegue identificar poucos objetos. A equação que representa a revocação é apresentada em (2)

$$Revocação = \frac{Verdadeiro_{positivo}}{Verdadeiro_{positivo} + Falso_{negativo}} \quad (2)$$

Pontuação F1 (*F1 Score*): Com os dados de precisão e revocação pode-se gerar a curva de precisão/revocação e por meio desta curva visualmente e numericamente determinar em qual ponto há um melhor balanço entre as duas métricas, tendo valores de precisão e revocação balanceados. Outra maneira de calcular essa relação é por meio da pontuação f1 (*f1 score*), essa pontuação indica o valor ideal entre a precisão e a revocação baseando-se nestes valores obtidos previamente.

A pontuação f1 varia de 0 à 1, sendo que quando pontuação f1 é alta próximo a 1 indica que tanto a precisão quanto a revocação estão altas, já uma pontuação f1 baixa indica o desbalanceamento entre a precisão e a revocação. A equação da pontuação f1 é apresentado na equação 3, já a curva de precisão e revocação e apresentada na figura 16.

$$f1 = 2\frac{Precisão * Revocação}{Precisão + Revocação} \quad (3)$$

Figura 16 - Curva Precisão-Revocação

Fonte: Paperspace (2020), adaptado pelo autor.

No exemplo mostrado na figura 16 o melhor balanço encontrado foi com o valor de precisão de 0,875 e revocação 0,778, inserindo esses valores na equação 3 para determinar a pontuação f1, obtém-se uma pontuação f1 de 0,824, indicando um bom balanço entre precisão e revocação.

Precisão Média (*Average Precision, AP*): A precisão média sintetiza a relação entre precisão e revocação via a média de todas as precisões. A precisão média é calculada por meio de um loop que passa por todas as precisões/revocações, a diferença entre as revocação atuais e as próximas é calculada e depois multiplicada pela precisão atual. Assim, a precisão média é a soma ponderada das precisões em cada limiar (*threshold*) onde o peso é o aumento da revocação. A equação que define a precisão média é apresentada a seguir (4).

$$AP = \sum_{k=0}^{k=n-1} [Revocações(k) - Revocações(k+1)] * Precisões\ (k) \qquad (4)$$

$$(4)$$

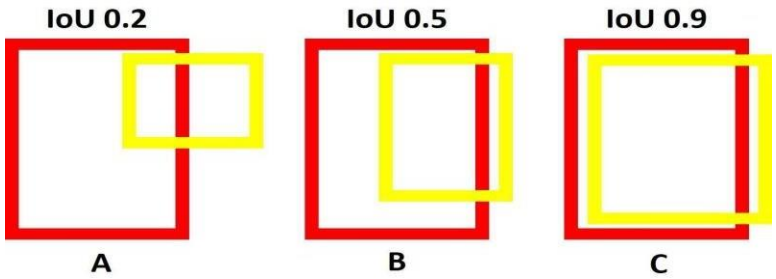
$$Revocações(n) = 0, Precisões(n) = 1$$

$$n = Número\ de\ limiares$$

Intersecção Sob União (*Intersection Over Union, IoU*): Outra métrica importante na avaliação de modelos de detecção de objetos é a intersecção sob união ou IoU. Esta métrica é calculada dividindo a área de intersecção entre as 2 caixas delimitadoras de objetos (*bounding boxes*), uma predita pelo modelo de IA e outra que é a anotação real, pela área de sua união. Quanto maior a IoU melhor a predição da caixa delimitadora, a equação que representa a IoU bem como um exemplo de diferentes valores de IoU são apresentados na equação (5) e figura 17.

$$IoU = \frac{Area\ de\ Intersecção}{Area\ de\ União} \tag{5}$$

Figura 17 - Representação de Intersecção sob União de (a) 0,2, (b) 0,5 e (c) 0,9



Fonte: Paperspace (2020)

Média das Precisões Médias (*Mean Average Precision, mAP*): Refere-se ao valor médio das precisões médias para cada classe. Em termos práticos inicialmente a precisão média (AP) é calculada para cada classe, levando em conta o limiar de IoU podendo ser 0,5 de intersecção ou uma intersecção que varia de 0,5 à 0,95. Uma vez que que a AP foi calculada a média desse valor para todas as classes é a média de precisões médias (MPM). A equação que representa o cálculo da MPM é apresentada na equação (6) (PAPERSPACE, 2020).

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \tag{6}$$

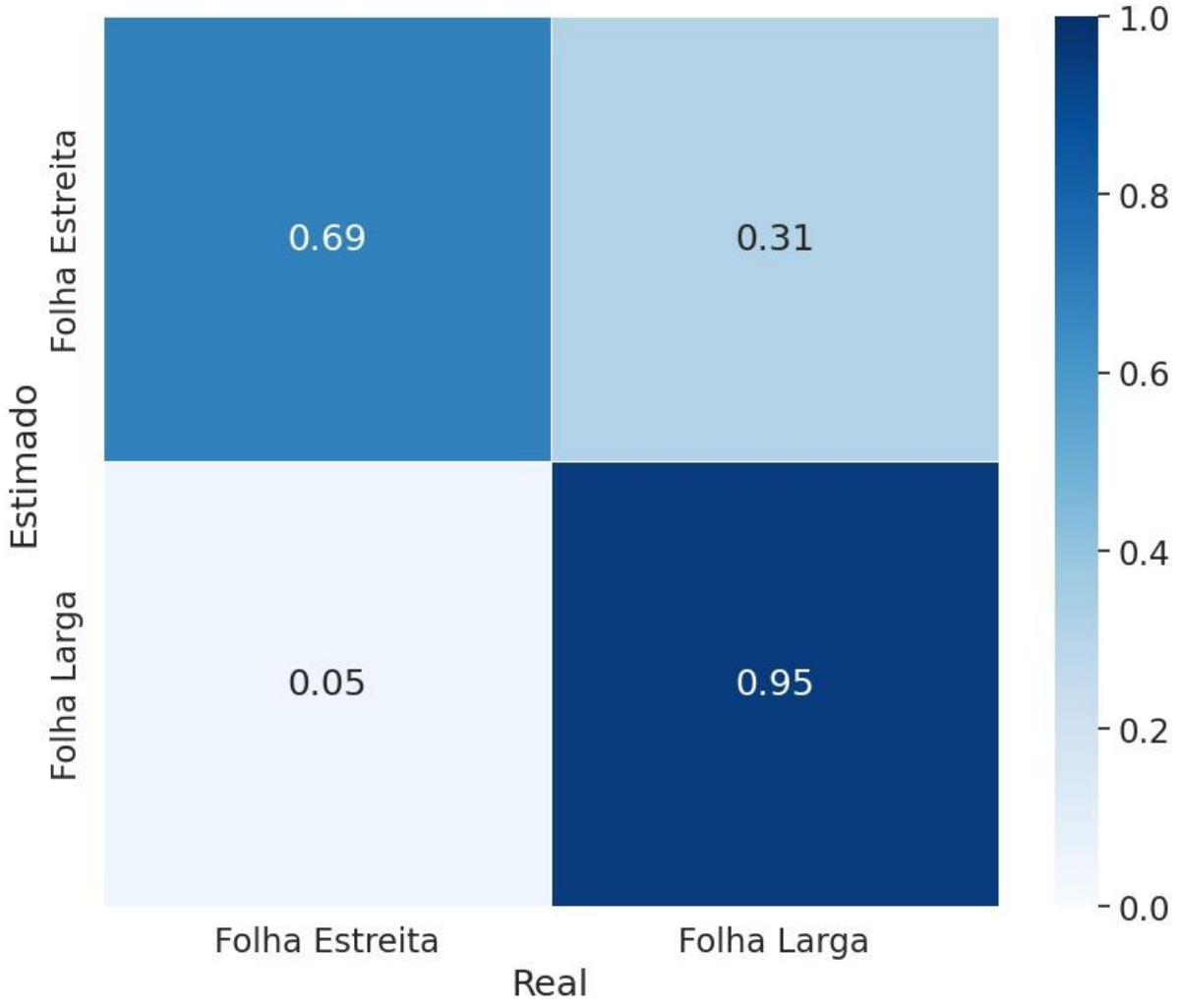
$$AP_k = A\ AP\ das\ classes\ k$$

$$n = O\ número\ de\ classes$$

Matriz de Confusão (*Confusion Matrix*): A matriz de confusão resume graficamente as primeiras definições apresentadas nas métricas sendo que um objeto

pode ser identificado como verdadeiro positivo, falso negativo, falso positivo ou verdadeiro negativo em uma escala de 0 à 1 quando os valores estão normalizados, entre n classes que o modelo foi treinado. Na figura 18 tem-se um exemplo da matriz de confusão de uma rede neural treinada para classificacar plantas daninhas entre folha larga e folha estreita. Essa matriz é acompanhada por uma legenda que varia a intensidade de coloração de acordo com o nível de assertividade das amostras.

Figura 18 - Matriz de Confusão Indicando a Classificação Entre Plantas de Folha Larga e Folha Estreita



Fonte: Elaborado pelo autor

No exemplo da figura 18 tem-se 0.95 ou 95% das plantas daninhas de folha larga folhas corretamente classificadas, enquando 0,05 ou 5% não foram. Para as plantas daninhas de folha estreita os resultados foram inferiores com 0,69 ou 69% das plantas foram corretamente classificadas e 31% não foram.

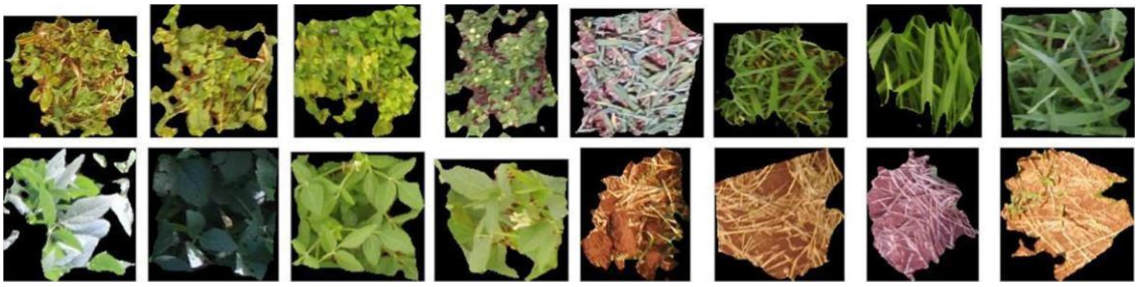
As métricas apresentadas serviram de base para avaliar osmodelos Yolov5 com e sem quantização de pesos neurais. No próximo item são abordados trabalhos de destaque relacionados a classificação e detecção de plantas daninhas usando CNNs.

2.11. Abordagens baseadas em Redes Neurais Profundas para a Identificação de Plantas Daninhas

Nos últimos anos diversas têm sido as abordagens que fazem uso de um ou mais modelos de aprendizado de máquina seja na classificação ou detecção de plantas daninhas, dentre estes trabalhos algumas abordagens que fizeram uso de redes neurais convolucionais e destacam-se são:

No cenário nacional, (*DOS SANTOS FERREIRA* et al., 2017) implementou redes neurais convolucionais na tarefa de classificação entre soja, solo, plantas daninhas de folha larga e folha estreita. Neste trabalho, o autor criou um banco de imagens com mais de 15.000 imagens, além de ter feito a coleta com um VANT (Veículo Aéreo Não tripulado) *DJI Phanton3* com uma altura média de 2 metros do solo. O trabalho destaca-se principalmente por ser uma das primeiras pesquisas usando VANTs e redes convolucionais para a classificação de plantas daninhas no Brasil. O autor também realizou a segmentação das imagens do plano de fundo utilizando o algoritmo SLIC (*Simple Linear Iterative Clustering*) *Superpixels* ou Simples Clusterização Linear Iterativa de *Superpixels*. A figura 19 apresenta algumas das imagens já segmentadas utilizadas no algoritmo.

Figura 19 - Imagens Segmentadas Utilizadas para Classificação de Plantas Daninhas



Fonte: Dos Santos Ferreira, A. (2017)

SUH et al., (2018) implementou e comparou a performance de 6 CNNs, *AlexNet* (KRIZHEVSKY et al., 2012), *VGG-19* (SIMONYAN et al., 2015), *GoogLeNet* (SZEGEDY et al., 2015), ResNet-50(HE et al., 2016), *ResNet-101* (HE et al., 2016) e *InceptionV3* (SZEGEDY et al., 2016) para a tarefa de classificação entre beterraba sacarina e plantas daninhas realizando transferência de aprendizado de pesos neurais pré treinados (*transfer learning*) (WEISS et al., 2016). Neste trabalho a maior acurácia de classificação foi obtida com a rede *VGG-19* (98,7%), entretanto foi o modelo que levou mais tempo para inferir devido o número de parâmetros. Por outro lado, a rede *AlexNet* obteve uma acurácia de classificação de 97%, mesmo em diferentes condições de iluminação e teve tempo de inferência menor.

No trabalho de (TEIMOURI et al., 2018) foi demonstrado que utilizar pesos pré treinados em grandes *datasets* de classificação de imagens como o *ImageNet* (DENG et al., 2009) são benéficos para o treinamento de modelos cujo intuito é classificar plantas daninhas, mesmo que esses modelos com pesos pré treinados não tenham inicialmente rótulos correlatos. Com a transferência de aprendizado o autor conseguiu reduzir o número de iterações de treinamento (épocas) e acelerou o processo de convergência do modelo. Para isso refinou-se (*fine- tuned*) a arquitetura *InceptionV3* e fez-se a classificação de dezoito espécies de plantas daninhas dispostas em distintos estágios de maturação com base no número de folhas. Como resultados obteve-se uma acurácia de classificação entre 46% a 78% e uma precisão média de 70% na contagem de folhas.

Outro trabalho de destaque é o de (MILIOTO et al., 2018) onde o autor fez uso das redes neurais convolucionais visando à automação na detecção e erradicação de plantas daninhas, em sua pesquisa o autor fez a classificação entre a beterraba-sacarina em estágio inicial, plantas daninhas e plano de fundo, aplicando segmentação semântica (*Semantic Segmentation)* que leva em conta uma máscara semântica atribuída as classes de interesse. Como resultado principal obtido conseguiu-se detectar e eliminar plantas daninhas em tempo real com um processamento de 20 frames por segundo (FPS). Para o treinamento da rede neural foram usadas 15.197 imagens de três locais distintos, um computador com processador Intel i7 e placa gráfica NVIDIA GTX1080Ti foi embarcado na plataforma robótica denominada *Bonirob* (Bosch) juntamente com outros sensores, para tanto navegar de maneira autônoma no campo quanto para inferir e eliminar as plantas

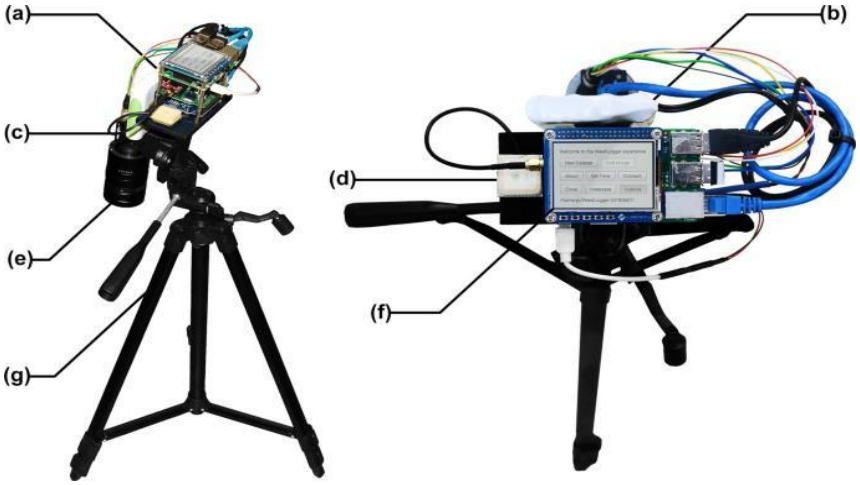
daninhas. Na figura 20 são apresentadas a plataforma *Bonirob* e as imagens que foram obtidas e processadas em tempo pela plataforma.

Figura 20 - Plataforma de Coleta de Imagens Bonirob e Plantas Analisadas

Fonte: Milioto et al. (2018)

No trabalho de (Olsen et al., 2019) foi criado um *dataset* próprio contendo 17.509 imagens de 8 tipos diferentes de plantas daninhas prejudiciais na Austrália. Para a coleta das imagens o autor desenvolveu um dispositivo de coleta (*WeedLogger in-field instrument*) para capturar as imagens com coordenadas GPS. Posteriormente foi feita a implementação e comparação entre as redes *ResNet-50* e *InceptionV3* na classificação das imagens do *dataset* ambas obtiveram resultados próximos 95,7% e 95,1%, sendo que a *ResNet-50* obteve o melhor resultado. A *dataset* bem como os scripts de implementação estão publicamente disponíveis em https://github.com/AlexOlsen/DeepWeeds, na figura 21 é apresentado o *WeedLogger* e seus componentes.

Figura 21 - Aparelho WeedLogger para coleta de imagens com coordenadas GPS



Fonte: Olsen et al. (2019)

Em um trabalho anterior (MARQUES JUNIOR; ULSON, 2018), foi criado um *dataset* de imagens de plantas daninhas prejudiciais para cultura de soja e que também apresentam resistência ao herbicida glifosato (Buva, *Conyza Bonariensis*; Capim-Azevém, *Lolium Multiflorum*; Capim-Amargoso, *Digitaria Insularis*; Capim-Pé-de-Galinha, *Eleusine Indica*; Caruru Plameri, *Amaranthus Palmeri*) em seguida foi feito o treinamento e comparação entre quatro arquiteturas de redes neurais convolucionais para a classificação das plantas (*VGG16*, *ResNet50*, *InceptionV3* e *InceptionResNetV2 (*SZEGEDY et al., 2016). Inicialmente os modelos foram treinados por 20 épocas sendo que a rede *InceptionV3* teve o melhor desempenho, com 84,73% de exatidão na classificação das cinco espécies. Seguida pela arquitetura *InceptionResNetV2* com 82,87%, *VGG16* com 80,60%, a arquitetura*ResNet50* obteve o pior resultado com 20,00% de exatidão. Determinado o melhor modelo a rede *InceptionV3* foi treinada com 40 épocas, onde obteve-se 88,50% de exatidão.

PARTEL et al., (2019) desenvolveu um sistema de aplicação inteligente (*smart spray*) para aplicação localizada de herbicidas em plantas daninhas por meio de um veículo tratorizado. Neste trabalho o autor comparou a performance das redes *Faster R-CNN* (REN et al., 2015), *YOLO-v3* (REDMON; FARHADI, 2018), *ResNet-50*, *ResNet-101*, e *Darknet-53* (REDMON, s.d.) para a inferência local foi utilizado um *notebook* com uma GPU NVIDIA 1070 Ti com 8 GB de RAM. Nos experimentos foi obtido 28 frames por segundo de inferência local e com base nos valores de precisão e revocação o melhor modelo foi a *ResNet-50*.

TRONG et al., (2020) propuseram um sistema de votação e fusão tardia multimodal de redes neurais profundas (*Fusion of Multimodal Deep Neural Networks*) para classificar plantas daninhas do *CNU Weed Dataset*, sendo composto por 208.477 imagens divididas em 21 espécies e 5 famílias de plantas daninhas. Nessa implementação os autores treinaram e compararam 5 redes pré-treinadas *NASNet* (ZOPH et al., 2017), *Resnet*, *Inception-Resnet*, *Mobilenet* (HOWARD et al., 2017) e *VGG*, posteriormente foi calculado um vetor de pontuação de duas maneiras via probabilidade condicional bayesiana ou determinando pesos de prioridade para que os melhores modelos de redes neurais profundas tenham uma maior contribuição para a pontuação. De acordo com os autores a fusão dos 5 modelos resultou em uma acurácia de classificação de 98,77%.

Em um trabalho mais recente (MARQUES JUNIOR; ULSON, 2021) foi feita a implementação e validação dos modelos de rede Yolov5 em um conjunto de imagens (*dataset*) customizado para a detecção de plantas daninhas resistentes ao herbicida glifosato. Com principais contribuições, o melhor tempo de inferência foi de 62 frames por segundo (FPS) usando a rede Yolov5s e a melhor média de precisões médias com limiar de 0,5 (mAP 0,5) foi obtida com a rede Yolov5m tendo mAP 0,5 de 0,77 e 26 FPS emum notebook *Samsung Odissey* com processador Core i7, 8GB de memória ram eplaca gráfica Nvidia GTX 1050 com 4 GB.

Todas as abordagens apresentadas trouxeram alguma contribuição significativa para a pesquisa e desenvolvimento de sistemas para a classificação e detecção plantas daninhas, seja pela implementação e comparação de múltiplos modelos, ou pela inferência local em plataformas tratorizadas. No entanto nenhum dos trabalhos abordou a compactação de pesos neurais via quantização para os formatos *fp16* e *int8,* além da conversão para *ONNX*. Também não esteve no escopo destes trabalhos avaliar a inferência de de imagens localmente, usando dispositivos de baixo custo e processamento limitado nos arquivos de pesos neurais quantizados e convertidos.

Alinhado com este desafio no capítulo a seguir são apresentados os procedimentos metológicos utilizados neste trabalho. Elencando a preparação de dataset, treinamentos dos modelos *Yolov5,* quantização e conversão dos pesos neurais bem como a descrição e transferência desses arquivos para a *Raspberry Pi Model B* 4GB.

CAPÍTULO 3: METODOLOGIA

Neste capítulo são abordados os procedimentos metodológicos executados nesta tese incluindo a preparação do conjunto de imagens para treinamento e validação, o treinamento dos modelos *Yolov5*, a quantização e conversão dos pesos neurais bem como a transferência dos pesos neurais para a *Raspberry Pi Model B* 4GB.

3.1. Procedimentos Metodológicos

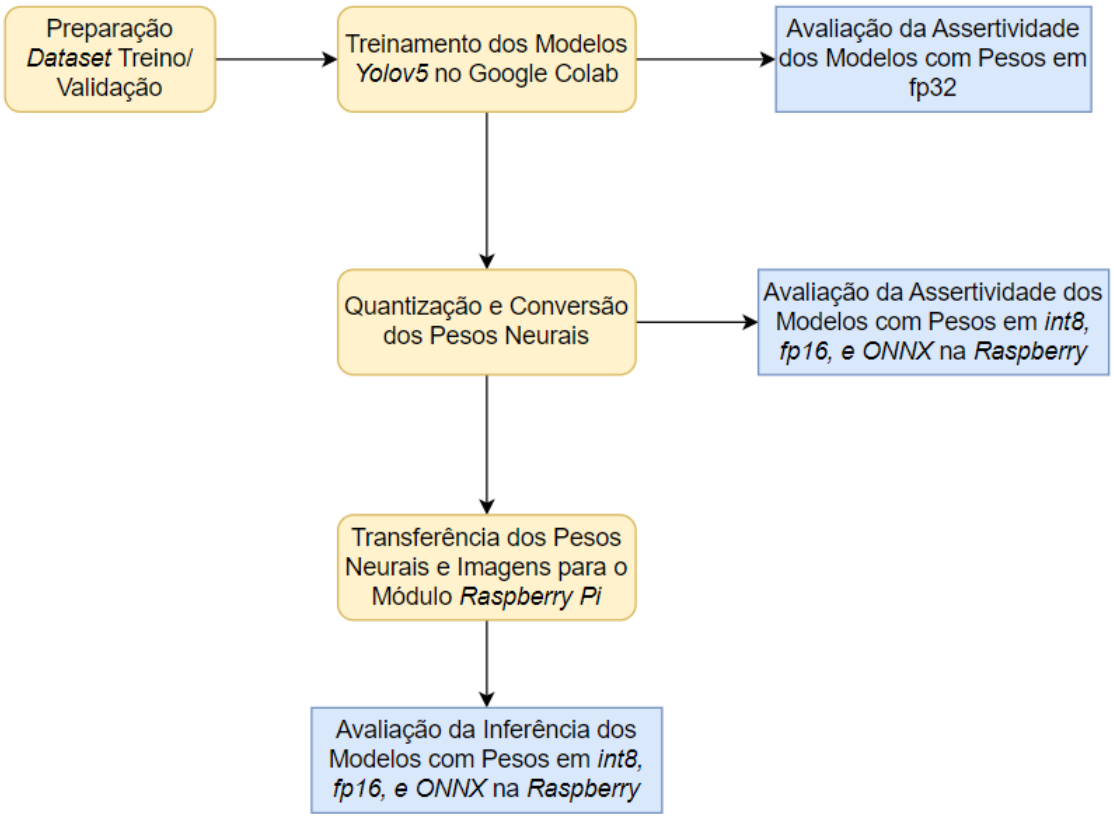
Os procedimentos metodológicos desta tese podem ser divididos em 7 partes principais:

- **Preparação do *Dataset*:** Envolve a preparação dos dados (imagens e rótulos) separando-os nos conjuntos de treinamento e validação.
- **Treinamento das arquiteturas *Yolov5***: Envolve o treinamento dos distintos modelos *Yolov5* no *Google Colab*, para a obteção dos pesos neurais é métricas de assetividade.
- **Avalição da Assertividade:** Envolve avaliar os resultados de validação obtidos nos treinamentos dos modelos *Yolov5* com representação de dados em *fp32*.
- **Quantização e Conversão:** Envolve quantizar os pesos neurais representados em *fp32* para *fp16* e *int8*. Bem como converter os pesos neurais para *ONNX.*
- **Tranferência dos Pesos Neurais:** Nessa etapa os pesos neurais quantizados e convertidos, assim como as imagens do conjunto de validação são transferidos para a *Raspberry Pi Model B 4GB.*
- **Avalição da Assertividade na *Raspberry Pi Model B 4GB*:** Envolve avaliar os resultados de validação obtidos com as representações de pesos neurais em *fp16, in8* e *ONNX*.

- **Avalição da Assertividade na *Raspberry Pi Model B 4GB*:** Envolve avaliar o tempo de inferência que cada representação de peso neural levou no conjunto de imagens de validação.

Na figura 22 é apresentado o fluxograma dos procedimentos metodológicos que resumo as etapas descritas.

Figura 22 - Fluxo de Procedimentos Metodológicos da Tese



Fonte: Elaborado pelo autor

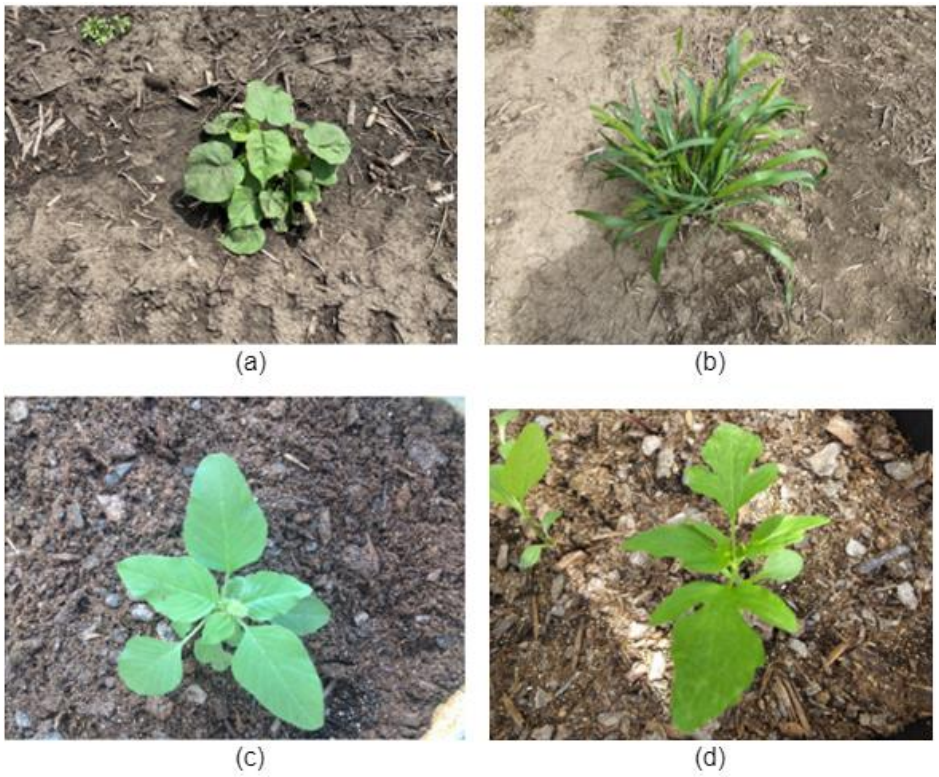
3.2. Conjunto de Imagens

Foi utilizado como conjunto de imagens (*dataset*) para treinamento e validação dos modelos Yolov5 o *4Weed Dataset* (AGGARWAL et al., 2022), produzido por membros da Universidade de *Purdue* o *dataset* engloba 4 espécies de plantas daninhas prejudiciais as culturas de milho e soja, divididas em:

- Carrapicho de Carneiro (*Xanthium Strumarium*);
- Rabo de Raposa (*Setaria Viridis*);
- Caruru Gigante (*Amaranthus Retroflexus*);
- Erva de Santiago Gigante (*Ambrosia Trifida*).

Todas as imagens foram capturadas nos campos experimentais de milho e soja da Universidade de *Purdue* com diferentes modelos de câmeras tendo resoluções distintas, o *dataset* encontra-se publicamente disponível. Na figura 23 são apresentadas as 4 espécies de plantas daninhas, enquanto que na tabela 1 tem-se o número total de imagens e de rótulos (plantas daninhas), divididos entre as quatro espécies de plantas daninhas que compõem o *dataset*.

Figura 23 - Espécies de plantas daninhas do 4WeedDtaset: (a) Carrapicho de Carneiro, (b) Rabo de Raposa, (c) Caruru Gigante e (d) Erva de Santiago Gigante



Fonte: Aggarwal (2022), adaptado pelo autor

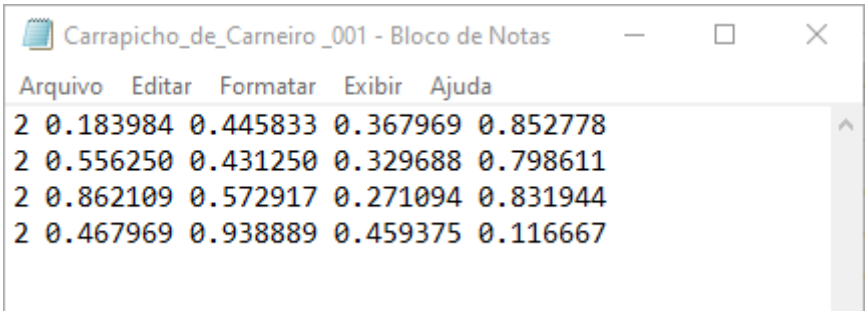
Tabela 1 - Distribuição de Imagens por Espécie e Rótulos de Plantas Daninhas

Espécie de Planta	Número de Imagens	Número de Plantas
Carrapicho de Carneiro	159	728
Rabo de Raposa	139	274
Caruru Gigante	170	202
Erva de Satiago	150	339
Total	618	1543

Fonte: Elaborado pelo autor

O formato de anotação das imagens do *dataset* está de acordo com o formato aceito pelas arquiteturas de rede neural do tipo *Yolo*, que é um arquivo em txt dividido em 5 colunas para cada imagem. Na primeira coluna um número inteiro iniciando em 0 é atribuído à classe ao qual o objeto pertence, as outras 4 colunas referem-se às coordenadas da localização da caixa delimitadora (*bounding box*) que contorna o objeto de interesse, o valor das coordenadas é normalizado variando de 0 à 1. Um exemplo de anotação no formato *Yolo* é apresentado na figura 24.

Figura 24 - Formato de Anotação Yolo



Fonte: Elaborado pelo autor

3.3. Algoritmo Yolov5

As redes neurais do tipo *Yolo* (*You Only Look Once*), (REDMON, 2016) obtiveram resultados expressivos nos últimos anos na tarefa de detecção de objetos, desde então essas redes têm sido amplamente estudadas, melhoradas e implementadas pela comunidade de visão computacional e *machine learning*. Tendo como base a *Yolov4* (BOCHKOVSKIY et al., 2020) o algoritmo *Yolov5* (ULTRALYTICS, 2020) traz como características principais da arquitetura os seguintes pontos:

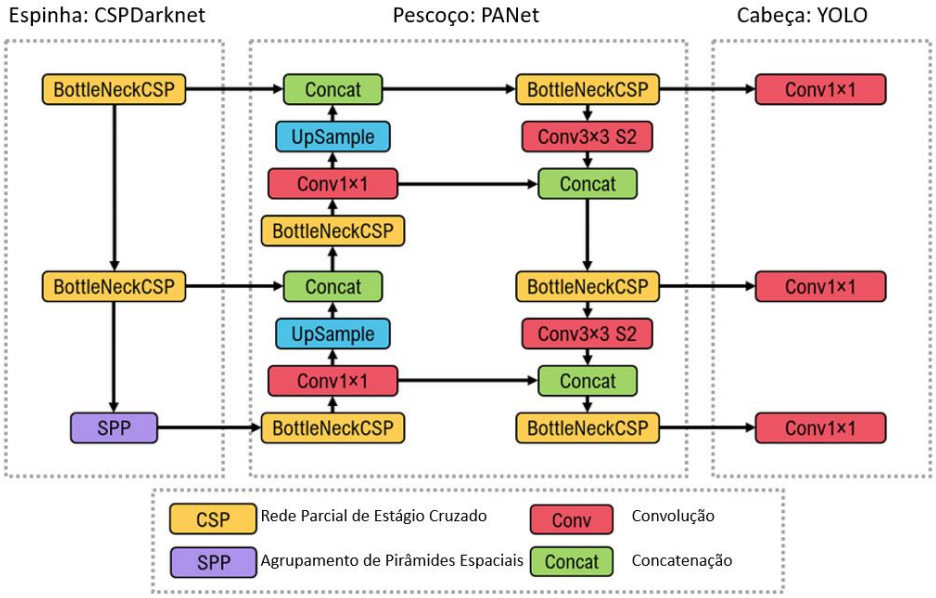
- **A Rede Parcial de Estágio Cruzado (*Cross Stage Partial Network, CSPDarknet*)**: A rede parcial de estágio cruzado (CSPNet) foi proposta por (WANG et al., 2020) nessa arquitetura a *CSPNet* aumenta a velocidade e precisão de inferência integrando alterações de gradiente em um mapa de características (*feature map*) e reduz os parâmetros e FLOPS (*Floating Points Operations Per Second*) ou operações de pontos flutuantes por segundo (XUE et al., 2021), tornando o modelo mais rápido e preciso. A *CSPNet* é integrada a arquitetua *Darknet-53* que é uma rede neural convolucional introduzida na arquitetura *Yolov3* onde serviu de espinha

dorsal *backbone* do modelo (Redmon et al., 2018). Assim a espinha dorsal da *Yolov5* é composta pela *CSPDarknet*, tendo ainda um último componente que é o Agrupamento de Pirâmides Espaciais (*Spatial Pyramid Pooling, SPP*) a *SPP* agrega a informação que recebe das entradas e retorna uma saída de tamanho fixo, assim, tem a vantagem de aumentar significativamente o campo receptivo e segregar os recursos de contexto mais relevantes sem diminuir a velocidade da rede.

- **A Rede de Agregação de Caminhos (*Path Aggregation Network, PANet*)** (*WANG* et al., 2019): Essa rede é denominada como pescoço ou *neck* da arquitetura, o propósito desta arquitetura é auxiliar na propagação recursos de baixo nível, bem como a melhora no uso de sinais de localização, o que aumenta a precisão e capacidade do modelo em localizar um objeto de interesse (XUE et al., 2021).

- **A Rede Yolov5**: Conhecida como cabeça ou *head* do modelo de algoritmo de inteligência artificial, está arquitetura gera 3 diferentes tamanhos de mapas de características (18 × 18, 36 × 36, 72 × 72), que possibilita à arquitetura ter distintas escalas de detecção (*REDMON* et al., 2016), podendo identificar objetos de interesse pequenos, médios e grandes (*XUE et al*., 2021). Sumarizando, inicialmente os dados são inseridos na *CSPDarknet* para extração de características, em seguida alimentam a *PANet* onde é feita a fusão de características, já na cabeça rede *YOLO* gera os resultados da detecção de objetos retornando um vetor com classe, confiança, localização e tamanho da imagem. Portanto o modelo *Yolov5* não é apenas uma rede neural, mas um conjunto de arquiteturas de redes neurais que foram associadas na tarefa de detecção de objetos. Na figura 25 são apresentados os elementos da arquitetura *YoloV5*.

Figura 25 - Arquitetura Yolov5 composta por: Espinha Dorsal (Backbone) CSPDarkenet, Pescoço (Neck) PANet e Cabeça (Head) Yolo



Fonte: Xue et al. (2021) adaptado pelo autor.

Outro aspecto importante da rede *Yolov5* é possuir diferentes versões, variando o número de parâmetros e *FLOPS*, o que impacta na precisão e tempo de inferência. Esses modelos vão desde o modelo com mais parâmetros *Yolov5n* ao modelo com mais *Yolov5x*, normalmente o modelo *Yolov5x* é o mais preciso, no entanto é o que consome maior processamento e leva mais tempo para inferir, já o modelo *Yolov5n* é o com menor tempo de inferencia, porém é menos preciso. Na tabela 2 são apresentados os diferentes modelos bem como o número de parâmetros em milhões M e operações de pontos flutuantes (*FLOPS*) em bilhões B para cada modelo e valores dos pesos neurais em *megabytes* (*MB*) representados em fp32.

Tabela 2 - Variações de modelos da Rede Neural Yolov5 e suas características

Modelo	Parâmetros(M)	FLOPS 640(B)	Pesos (MB)
YOLOv5n	1,9	4,5	3,9
YOLOv5s	7,2	16,5	14,5
YOLOv5m	21,2	49,0	42,3
YOLOv5l	46,5	109,1	92,9
YOLOv5x	86,7	205,7	173,2

Fonte: Elaborado pelo autor

A arquitetura *Yolo* conta ainda com uma série de hiperparâmetros para treinamento e augmentação de dados, sendo os principais:

- **Lr0:** Refere-se a taxa de aprendizagem (*learning rate*) aplicada na arquitetura;
- **Momentum:** Refere-se à aplicação de um valor de momento no processo de aprendizado do modelo uma abordagem tradicional ao treinar redes neurais.
- **Iou_T:** Refere-se ao limiar de interseção sob união (*intersection over union)* entre as caixas delimitadoras durante o treinamento.
- **Hsv:** Refere-se a técnica de augmentação de imagens no espaço H, S, V que seria matiz (*Hue*), saturação (*Saturation*) e valor (*Value*).
- **Flip:** Refere-se a técnica de virar uma ou mais imagens, podendo ser para esquerda, direita, para baixo ou para cima.
- **Mosaic:** Refere-se a augmentação onde as imagens com os objetos de interesse são construídas como um mosaico.

Na figura 26 são apresentados todos os hiperparâmetros de treinamento e augmentação utilizados nos experimentos as redes *Yolov5.*

Figura 26 - Hiperparâmetros de Treinamento e Augmentação para as Redes Yolov5

34 lines (33 sloc) | 1.65 KB

```yaml
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license
# Hyperparameters for low-augmentation COCO training from scratch
# python train.py --batch 64 --cfg yolov5n6.yaml --weights '' --data coco.yaml --img 640 --epochs 300 --linear
# See tutorials for hyperparameter evolution https://github.com/ultralytics/yolov5#tutorials

lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
copy_paste: 0.0  # segment copy-paste (probability)
```

Fonte: ULTRALYTICS. (2020)

3.4. Treinamento Modelos Yolov5

Para o treinamento das arquiteturas de fez-se uso da linguagem de programação *Python 3.9* e do framework de código aberto (*open source*) para *machine learning Pytorch* (*PASZKE et al*., 2019). Devido a diferença de parâmetros entre os modelos utilizados o treinamento foi realizado no ambiente do *Google Colab Pro* (*BISONG*, 2019), por disponibilizar tempo de treinamento maior e máquinas com maior processamento gráfico (*GPU*) e memória *RAM*. A resolução de 640 x 640 pixels foi definida como padrão para o treinamento uma vez que existe uma variabilidade de resoluções entre as imagens do *dataset*, todas as arquiteturas foram treinadas por 150 épocas. O *dataset* de treinamento foi composto por 518 imagens com 1230 rótulos no total, divididos entre as 4 espécies de plantas daninhas, a distribuição dos rótulos bem como o número de imagens por espécie de planta está sintetizada na tabela 3.

Tabela 3 - Número de Imagens por Espécie de Planta no Dataset de Treinamento

Espécie de Planta	Número de Imagens	Número de Plantas
Carrapicho de Carneiro	134	612
Rabo de Raposa	114	205
Caruru Gigante	145	168
Erva de Satiago	125	245
Total	518	1230

Fonte: Elaborado pelo autor

Para a validação foram usadas 25 imagens de cada espécie de planta daninha totalizando 100 imagens com 313 rótulos distribuídos para as 4 espécies. Na tabela 4 tem-se a distribuição dos rótulos do *dataset* de validação.

Tabela 4 - Número de imagens por Espécie de Planta no Dataset de Validação

Espécie de Planta	Número de Imagens	Número de Plantas
Carrapicho de Carneiro	25	116
Rabo de Raposa	25	69
Caruru Gigante	25	34
Erva de Satiago	25	94
Total	100	313

Fonte: Elaborado pelo autor

A espécie Carrapicho de Carneiro possui mais rótulos para treinamento (612) e validação (116), seguida pela espécie Erva de Santiago com 245 rótulos no treinamento e 94 na validação, a espécie Rabo de Raposa é a terceira com 205 rótulos para treinamento e 69 para validação, por último a espécie Caruru Gigante é a que possui a menor quantidade de rótulos no treinamento (168) e validação (34).

Ao final do treinamento foram obtidas as métricas para avaliação dos resultados, bem como um arquivo no formato .pt (extensão *pytorch*), que contém os pesos neurais dos modelos treinados representando os dados em ponto flutuante de 32 bits (*fp32*).

3.5. Quantização e Conversão dos Pesos Neurais

Finalizado o treinamento e validação das redes neurais pelo *Google Colab* e obtido os arquivos de pesos para cada modelo, foi realizada a quantização pós treinamento e a conversão dos pesos neurais para o formato *ONNX*. Originalmente os pesos neurais são obtidos no formato *pytorch* representado pela extensão .pt em ponto flutuante de 32 *bits*, tanto para a quantização em formato de ponto flutuante de 16 *bits* quanto para a quantização em int8 os pesos neurais foram convertidos de *Pytorch* para *Tensorflow* representados no formato *saved_model*. Com os pesos em *Tensorflow* foram realizados os seguintes procedimentos para a quantização:

- **Quantização para fp16:** Com os pesos no formato *saved_model* esse arquivo é utilizado como entrada para o procedimento de conversão por meio da classe `tf.lite.TFLiteConverter.from_keras_model` que recebe o peso neural para a conversão em seguida as operações de conversão suportadas são executadas pelo `[tf.lite.OpsSet.TFLITE_BUILTINS]` e por fim é indicada a representação para salvar o peso neural para o formato tflite.fp16 via `[tf.float16]`.
- **Quantização para int8**: Semelhante ao que ocorre com a quantização em fp16 na quantização para o formato int8 inicialmente tem-se o peso neural no formato *saved_model* esse peso é utilizado como entrada para a classe `tf.lite.TFLiteConverter.from_keras_model`, na quantização int8 antes de definir as operações de conversão um conjunto de imagens representativos é criado por meio da função `representative_dataset_gen,`

o conjunto de imagens representativo é criado a partir das imagens de treinamento com o intuito de calibrar os pesos neurais a serem salvos formato int8, o que permite que esse formato de representação mantenha a acuraria. Em seguida as operações de conversão suportadas são executadas por meio *[tf.lite.OpsSet.TFLITE_BUILTINS_INT8]* e por fim é indicado a representação que o peso neural no formato tflite.int8 deverá ser salvo via *tf.uint8*.

Na figura 27 o trecho de código responsável pela quantização fp16 e int8, bem como os itens descritos são apresentados.

Figura 27 - Trecho de código para Quantização nos Formatos fp16 e int8

```
@try_export
def export_tflite(keras_model, im, file, int8, data, nms, agnostic_nms, prefix=colorstr('TensorFlow Lite:')):
    # YOLOv5 TensorFlow Lite export
    import tensorflow as tf

    LOGGER.info(f'\n{prefix} starting export with tensorflow {tf.__version__}...')
    batch_size, ch, *imgsz = list(im.shape)  # BCHW
    f = str(file).replace('.pt', '-fp16.tflite')

    converter = tf.lite.TFLiteConverter.from_keras_model(keras_model)
    converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS]
    converter.target_spec.supported_types = [tf.float16]
    converter.optimizations = [tf.lite.Optimize.DEFAULT]
    if int8:
        from models.tf import representative_dataset_gen
        dataset = LoadImages(check_dataset(check_yaml(data))['train'], img_size=imgsz, auto=False)
        converter.representative_dataset = lambda: representative_dataset_gen(dataset, ncalib=100)
        converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
        converter.target_spec.supported_types = []
        converter.inference_input_type = tf.uint8  # or tf.int8
        converter.inference_output_type = tf.uint8  # or tf.int8
        converter.experimental_new_quantizer = True
        f = str(file).replace('.pt', '-int8.tflite')
    if nms or agnostic_nms:
        converter.target_spec.supported_ops.append(tf.lite.OpsSet.SELECT_TF_OPS)

    tflite_model = converter.convert()
    open(f, 'wb').write(tflite_model)
    return f, None
```

Fonte: ULTRALYTICS. (2020)

- **Conversão para *ONNX:***

Em seguida o mesmo arquivo de com pesos de treinamento no formato *pytorch* com a extensão .pt foi convertido para *ONNX* por meio da função *torch.onnx.export*, permitindo que o peso neural no formato *ONNX* possa ser carregado para inferência em *ONNX Runtime Inference*. Na figura 28 é apresentado o trecho de código

responsável pela conversão de *pytorch* para *ONNX.* Com todas os pesos neurais das redes quantizados e convertidos foi feita a transferência desses arquivos para a *Raspberry Pi 4 Model B* 4GB juntamente com as 100 imagens para validação de resultados e inferência local.

Figura 28 - Trecho de código para Conversão no Formato ONNX

```python
@try_export
def export_onnx(model, im, file, opset, dynamic, simplify, prefix=colorstr('ONNX:')):
    # YOLOv5 ONNX export
    check_requirements('onnx>=1.12.0')
    import onnx

    LOGGER.info(f'\n{prefix} starting export with onnx {onnx.__version__}...')
    f = file.with_suffix('.onnx')

    output_names = ['output0', 'output1'] if isinstance(model, SegmentationModel) else ['output0']
    if dynamic:
        dynamic = {'images': {0: 'batch', 2: 'height', 3: 'width'}}  # shape(1,3,640,640)
        if isinstance(model, SegmentationModel):
            dynamic['output0'] = {0: 'batch', 1: 'anchors'}  # shape(1,25200,85)
            dynamic['output1'] = {0: 'batch', 2: 'mask_height', 3: 'mask_width'}  # shape(1,32,160,160)
        elif isinstance(model, DetectionModel):
            dynamic['output0'] = {0: 'batch', 1: 'anchors'}  # shape(1,25200,85)

    torch.onnx.export(
        model.cpu() if dynamic else model,  # --dynamic only compatible with cpu
        im.cpu() if dynamic else im,
        f,
        verbose=False,
        opset_version=opset,
        do_constant_folding=True,  # WARNING: DNN inference with torch>=1.12 may require do_constant_folding=False
        input_names=['images'],
        output_names=output_names,
        dynamic_axes=dynamic or None)

    # Checks
    model_onnx = onnx.load(f)  # load onnx model
    onnx.checker.check_model(model_onnx)  # check onnx model

    # Metadata
    d = {'stride': int(max(model.stride)), 'names': model.names}
    for k, v in d.items():
        meta = model_onnx.metadata_props.add()
        meta.key, meta.value = k, str(v)
    onnx.save(model_onnx, f)
```

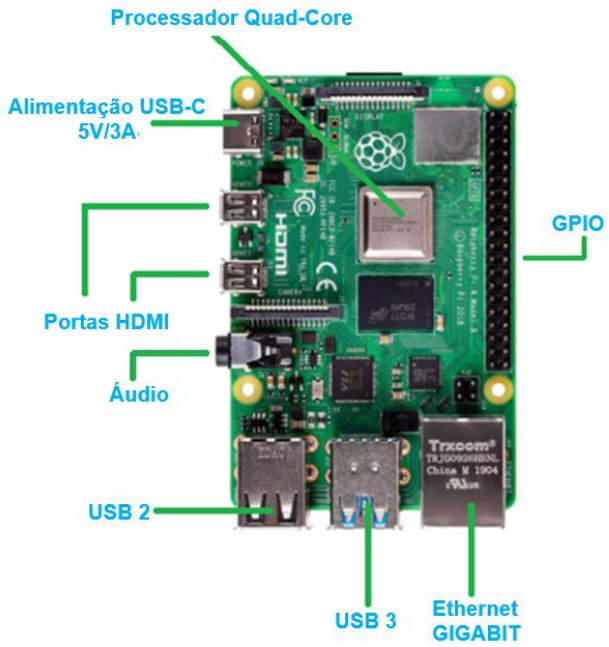
Fonte: ULTRALYTICS. (2020)

3.6. Hardware de Inferência

Para a validação e inferência dos pesos neurais nos formatos *tflite.fp16, tflite.int8* e *ONNX* foi utilizada a placa *Raspberry Pi 4 model B* de 4GB. Denominado como mini computador, a placa conta com um processador *Quad core Cortex-A72 (ARMv8) de 64 bits 1.5GHz SoC* e 4 GB de RAM DDR4, para evitar sobreaquecimento no processador 2 *coolers* foram instalados junto com dissipadores térmicos de alumínio, além de uma base superior e inferior em acrílico para facilitar o manuseio da placa. O sistema operacional utilizado foi o padrão de fábrica que é o *raspbian* um sistema operacional baseado em *Linux*.

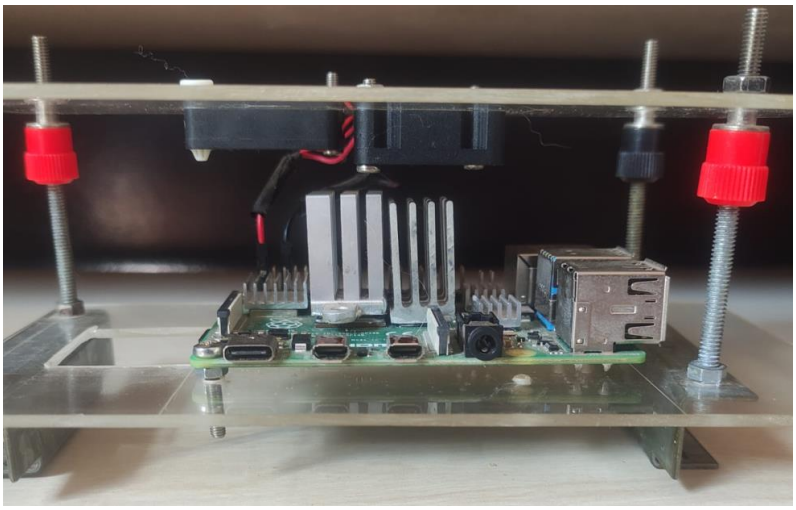
No momento em que este trabalho foi desenvolvido o modelo de placa utilizada foi encontrado em um site de revenda oficial por R$ 734,90 (FILIPEFLOP, 2022). Foi utilizado como parâmetro de custo acessivel o valor de até R$ 1000,00, uma vez que o mini computador pode ser adquirido por preço inferior ao parâmetro estabelecido, nesse contexto considera-se um sistema de baixo custo para processamento e inferência local. Na figura 29 são apresentados os componentes do módulo *Raspberry Pi 4 model B* de 4GB enquanto que na figura 30 é apresentada a placa utilizada para nos experimentos.

Figura 29 - Principais Componentes do Módulo Raspberry Pi 4 Model B 4GB



Fonte: Theengineeringproject (2021) adaptado pelo autor

Figura 30 - Placa Raspberry Pi Model B 4GB Utilizada nos Experimentos



Fonte: Elaborado pelo autor

Os pesos dos modelos *Yolov5n, Yolov5s, Yolov5m, Yolov5l e Yolov5x* quantizados para *fp16, int8* e convertidos para o formato *ONNX* foram tranferidos para o módulo *Raspberry Pi Model B 4GB*. Inicialmente foi feita a validação desses pesos com base nas 100 imagens dos *dataset* de validação, que também foram transferidas para o módulo. Onde obteve-se as matrizes de confusão e demais métricas para avaliação dos resultados.

Posteriormente foi feita a inferência para avaliar tempo de detecção de imagens das distintas arquiteturas e seus respectivos pesos neurais. Os códigos para treinamento, validação, conversão e quantização de pesos neurais estão disponíveis no repositório https://github.com/Luiz-Carlos88/Phd_Thesis-Embedded-Yolov5.git.

Os resultados de validação com pesos em fp32, bem como os resultados de validação com os pesos quantizados, convertidos para *ONNX* e o tempo de inferência dos modelos *Yolov5* são apresentados no próximo capítulo.

CAPÍTULO 4: RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os resultados do treinamento dos modelos, bem como os resultados de validação e inferência com os pesos neurais quantizados e convertidos.

4.1. Resultado Treinamento Modelos *Yolov5*

Finalizado o treinamento das arquiteturas foi feita a validação nas imagens no *dataset* de validação. A partir desta validação obteve-se as matrizes de confusão e demais métricas para avaliar os resultados dos modelos *Yolov5n, Yolov5s, Yolov5m, Yolov5l* e *Yolov5x*, todos como pesos neurais em ponto flutuante de 32 *bit*s fp32.

Na figura 31 é apresentada a matriz de confusão para o modelo *Yolov5n*, nesse modelo o melhor resultado foi obtido para a espécie Carrapicho de Carneiro com 0,86, seguido por Caruru Gigante com 0,79 e Rabo de Raposa com 0,71. A rede teve resultados piores para classificar a espécie Erva de Santiago obtendo apenas 0,48, fazendo uma média desses valores a rede *Yolov5n* obteve 0,71 ou 71% de acertos.

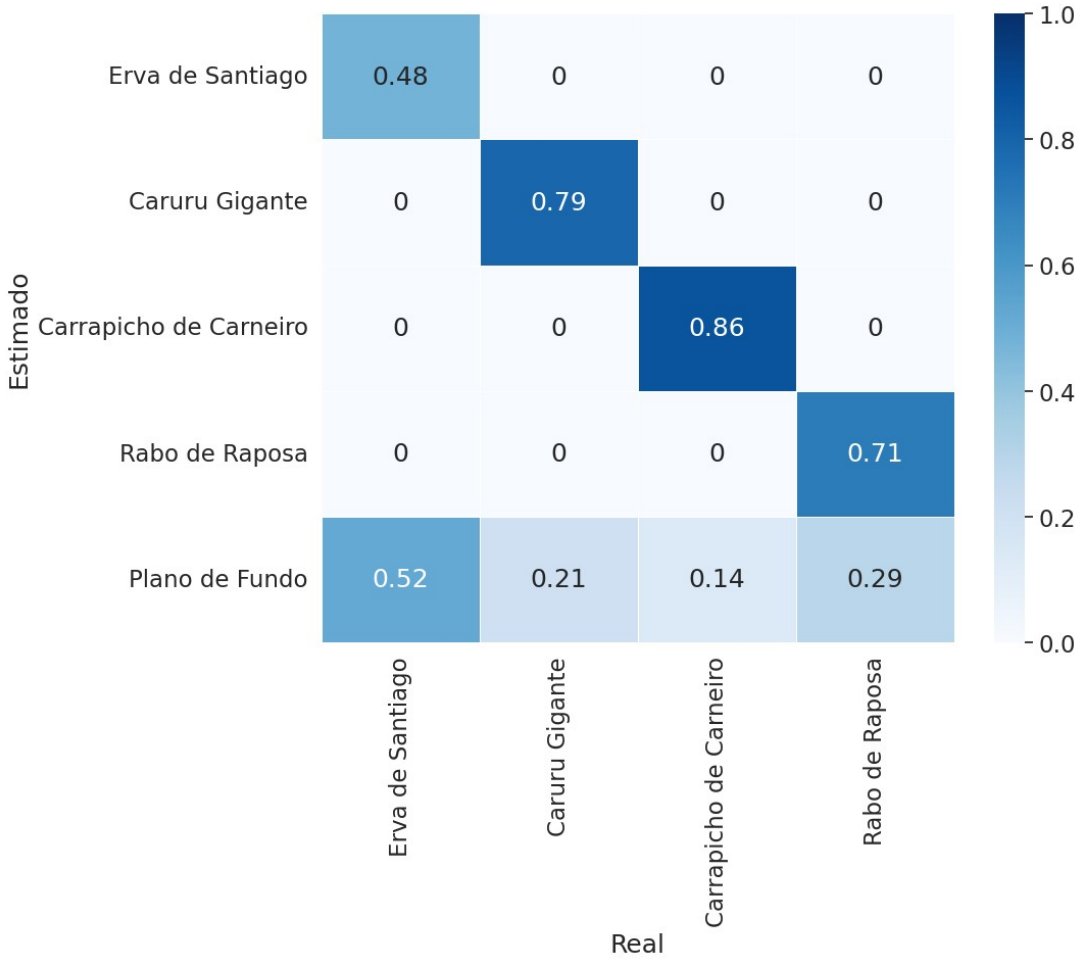
A matriz de confusão do modelo *Yolov5s* é apresentada na figura 32. Esse modelo obteve resultados melhores comparando com a *Yolov5n* com 0,91 para Carrapicho de Carneiro, 0,85 para Caruru Gigante e 0,74 para Rabo de Raposa. A espécie Erva de Santiago continuou sendo a de piores resultados com 0,56, fazendo uma média desses valores a rede *Yolov5s* obteve 0,76 ou 76% de acertos.

A matriz de confusão da rede *Yolov5m* é apresentada na figura 33. Essa rede obteve o melhor resultado de todas para a espécie Rabo de Raposa com 0,81, Carrapicho de Carneiro continuou como a espécie melhor classificada dentre os modelos com 0,90 seguida por Caruru Gigante com 0,85. Por fim, obteve-se 0,53 para Erva de Santiago, a média obtida pelo modelo Yolov5m foi de 0,77 ou 77%.

A matriz de confusão da rede *Yolov5l* é apresentada na figura 34. Essa rede obteve o melhor resultado de todas para a espécie Carrapicho de Carneiro com 0,92, seguida por Caruru Gigante com 0,85 e Rabo de Raposa com 0,80. Foi obtido também 0,55 para Erva de Santiago, a média obtida pelo modelo *Yolov5l* foi de 0,78 ou 78%.

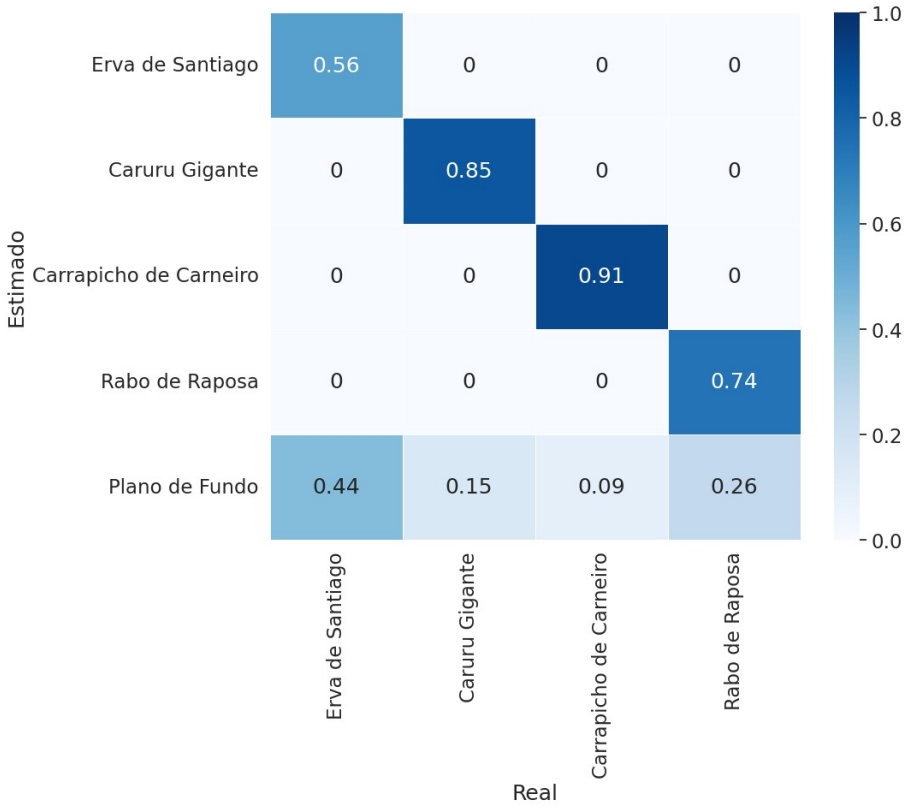
Na figura 35 é apresentada a matriz de confusão da rede *Yolov5x*. Essa rede obteve os melhores resultados para as espécies Erva de Santiago com 0,59 e Caruru Gigante 0,88. Obteve também bons resultados para Carrapicho de Carneiro com 0,89 e Rabo de Raposa com 0,77. Fazendo a média desses valores a rede *Yolov5x* obteve 0,782 ou 78,2% de acertos, tendo uma performance pouco superior se comparada com a *Yolov5l*. As matrizes de confusão para os modelos *Yolov5* são apresentados nas figuras a seguir.

Figura 31 - Matriz de Confusão Yolov5n com pesos em fp32



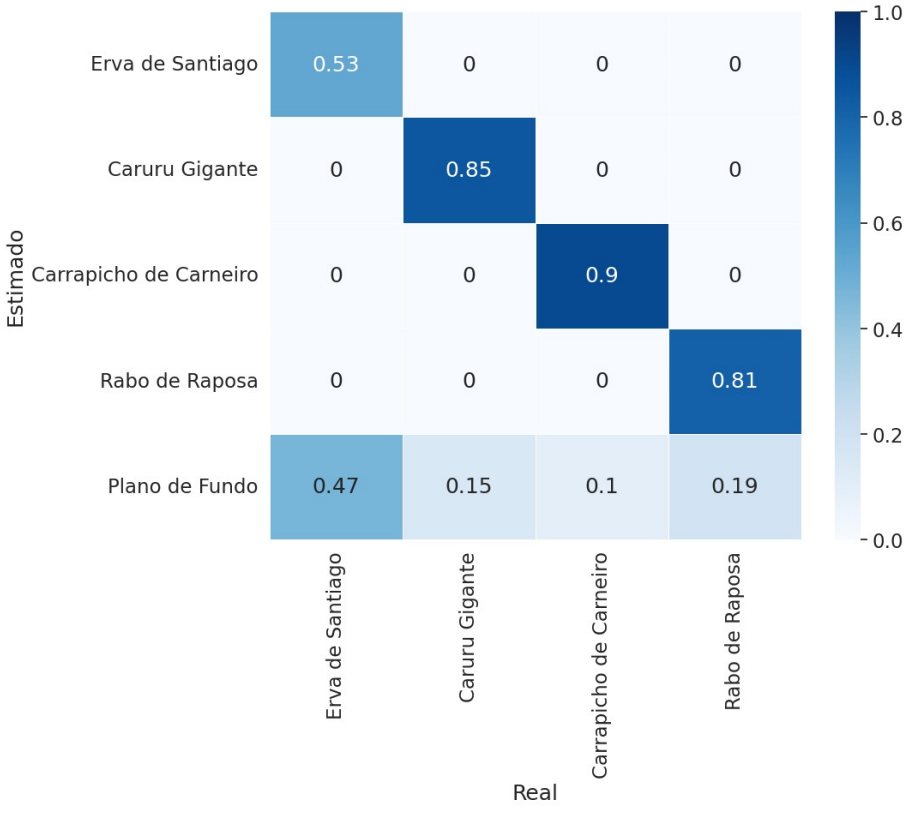
Fonte: Elaborado pelo autor

Figura 32 - Matriz de Confusão Yolov5s com pesos em fp32



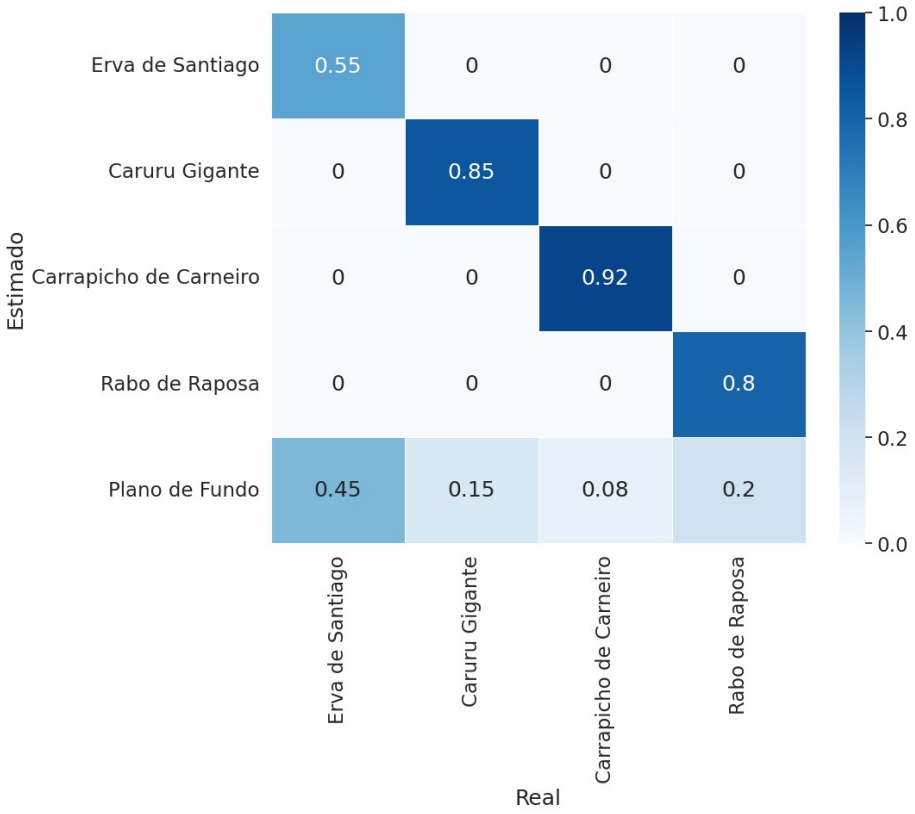
Fonte: Elaborado pelo autor

Figura 33 - Matriz de Confusão Yolov5m com pesos em fp32



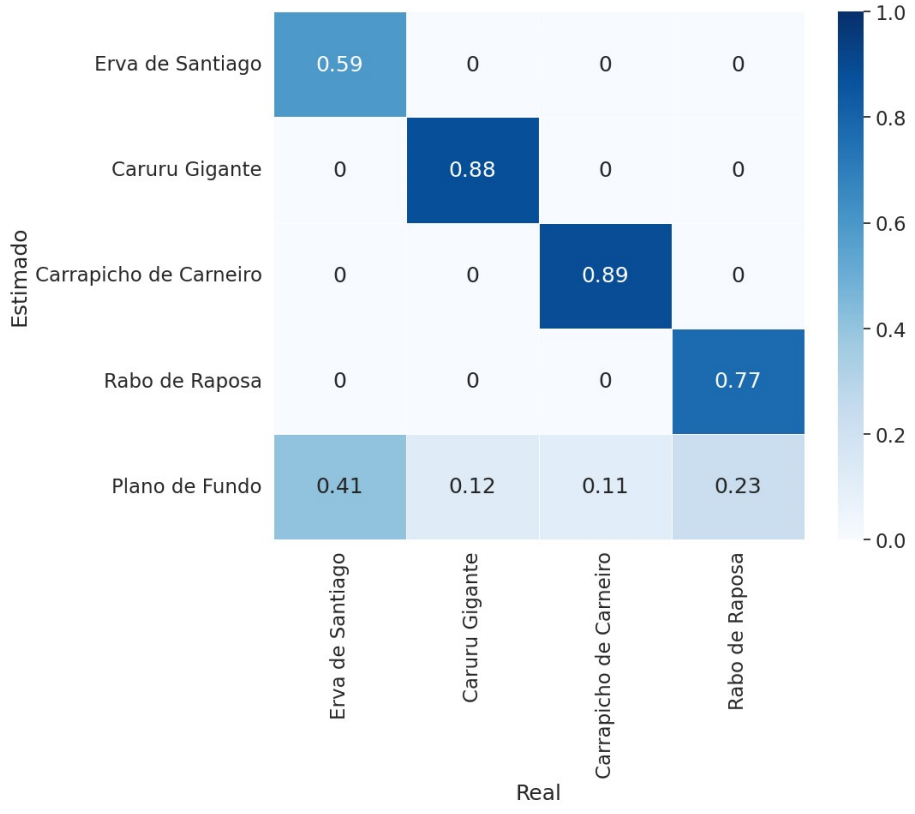
Fonte: Elaborado pelo autor

Figura 34 - Matriz de Confusão Yolov5l com pesos em fp32



Fonte: Elaborado pelo autor

Figura 35 - Matriz de Confusão Yolov5x com pesos em fp32



Fonte: Elaborado pelo autor

Levando em conta o balanço entre performance e precisão, e com base nas matrizes de confusão a rede *Yolov5s* é a mais equilibrada, visto que é o segundo modelo com menor representação de pesos neurais 14,5 *megabytes* e obteve resultados satisfatórios com 0,76 ou 76% de acertos com base na matriz de confusão. Na tabela 5 são apresentadas as demais métricas para avaliação dos modelos como a Precisão (*precision*) a Revocação (*recall*) a média das precisões médias com 0,5 de limiar e a média das precisões médias com liminar variando de 0,5 a 0,95.

Diferente dos resultados obtidos com as matrizes de confusão no *mAP* os modelos são penalizados caso as caixas delimitadores (*boundinx boxes*) não se sobreponham em pelo menos 50% (*mAP 0,5)* ou entre 50% à 95% (*mAP 0,5:0,95)*. Também é levado em conta o balanço entre a precisão e a revocação, visto que um modelo pode ser preciso, mas a quantidade de dados acertados é pequena se comparado com o *dataset*, ou um modelo pode indicar que uma série de imagens representam uma espécie de planta daninha, entretanto são falsos positivos. A média destas métricas para cada modelo também é apresentada na tabela 5, sendo que a rede *Yolov5l* obteve melhores resultados com 0,792 de *mAP 0,5* seguida pela Yolov5x com *mAP 0,5* de 0,784.

Após a validação com os pesos no formato *fp32*, foi realizada a validação e inferência no módulo *Raspberry Pi 4 model B* de 4GB, com os pesos neurais quantizados e convertidos. Os resultados obtidos nesses experimentos são apresentados nos itens a seguir.

Tabela 5 - Resultados de Validação Redes Yolov5 pesos em fp32

Rede Neural	Espécie de Planta	Precisão	Revocação	mAP 0,5	mAP 0,5:0,95
Yolov5n	Erva de Santiago	0,744	0,495	0,563	0,278
	Caruru Gigante	0,961	0,824	0,950	0,571
	Carrapicho de Carneiro	0,855	0,866	0,930	0,516
	Rabo de Raposa	0,646	0,638	0,581	0,152
	Média	**0,802**	**0,705**	**0,756**	**0,379**
Yolov5s	Erva de Santiago	0,723	0,511	0,556	0,274
	Caruru Gigante	1	0,853	0,915	0,625
	Carrapicho de Carneiro	0,776	0,897	0,916	0,496
	Rabo de Raposa	0,608	0,63	0,583	0,177
	Média	**0,777**	**0,722**	**0,742**	**0,393**
Yolov5m	Erva de Santiago	0,742	0,511	0,566	0,282
	Caruru Gigante	0,961	0,853	0,861	0,625
	Carrapicho de Carneiro	0,845	0,891	0,921	0,517
	Rabo de Raposa	0,635	0,730	0,665	0,187
	Média	**0,796**	**0,746**	**0,753**	**0,403**
Yolov5l	Erva de Santiago	0,832	0,543	0,641	0,317
	Caruru Gigante	0,991	0,912	0,922	0,653
	Carrapicho de Carneiro	0,845	0,931	0,923	0,508
	Rabo de Raposa	0,592	0,768	0,680	0,186
	Média	**0,815**	**0,788**	**0,792**	**0,416**
Yolov5x	Erva de Santiago	0,754	0,587	0,609	0,298
	Caruru Gigante	0,996	0,941	0,967	0,660
	Carrapicho de Carneiro	0,874	0,897	0,923	0,527
	Rabo de Raposa	0,646	0,696	0,635	0,182
	Média	**0,817**	**0,780**	**0,784**	**0,417**

Fonte: Elaborado pelo autor

4.2. Resultados Quantização e Conversão Yolov5n

Os arquivo em formato *tflite.fp16, tflite.int8* e *ONNX* do modelo *Yolov5n* foram tranferidos para a placa *Raspberry Pi*, para a validação e inferência local. Na tabela 6 são apresentados os valores dos pesos neurais quantizados e convertidos em *megabytes* (MB) para esse modelo.

Tabela 6 - Pesos Neurais Rede Yolov5n Quantizados e Convertidos

Rede Neural	Arquivo de Pesos em (MB)
Yolov5n fp16	3,7
Yolov5n int8	2,0
Yolov5n onnx	7,5

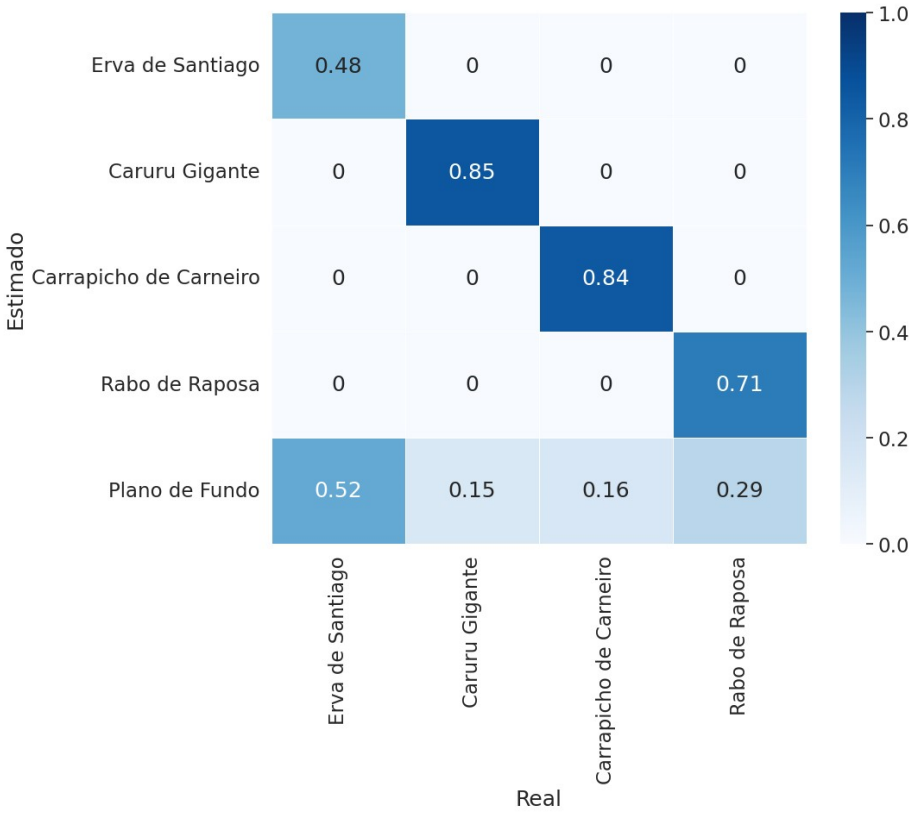
Fonte: Elaborado pelo autor

Após a validação local dos pesos neurais do modelo *Yolov5n* foram obtidas as matrizes de confusão bem como as demais métricas para avaliar os resultados. Na figura 36 é apresentada a matriz de confusão da rede *Yolov5n* com pesos quantizados para fp16, os melhores resultados foram para as espécies Caruru Gigante 0,85, Carrapicho de Carneiro com 0,84 e Rabo de Raposa com 0,71. O pior resultado foi para Erva de Santiago com 0,48, fazendo uma média desses valores obteve-se 0,71 ou 71% de acertos.

A matriz de confusão da *Yolov5n* com pesos quantizados para *int8* é apresentada na figura 37. Com a quantização para int8 os resultados foram inferiores quando comparados com a quantização fp16, com 0,82 para Caruru Gigante e Carrapicho de Carneiro, seguida por Rabo de Raposa com 0,71 e Erva de Santiago com 0,47. Fazendo uma média desses valores obteve-se 0,70 ou 70% de acertos.

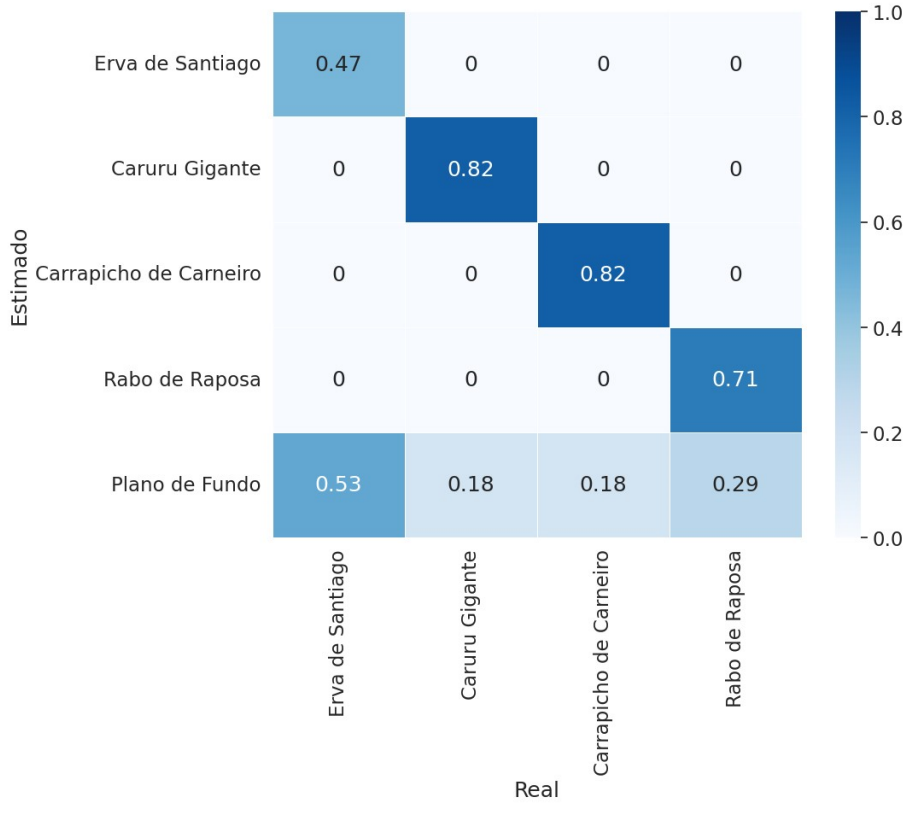
Na figura 38 é apresentada a matriz de confusão com os pesos convertidos para *ONNX*. Os resultados obtidos foram idênticos aos apresentados na matriz de confusão da figura 36, onde os melhores resultados foram para as espécies Caruru Gigante 0,85, Carrapicho de Carneiro com 0,84 e Rabo de Raposa com 0,71, já o pior resultado foi para Erva de Santiago com 0,48, a média dos valores foi de 0,71 ou 71%. As matrizes de confusão são apresentadas nas figuras a seguir.

Figura 36 - Matriz de Confusão Yolov5n com pesos quantizados fp16



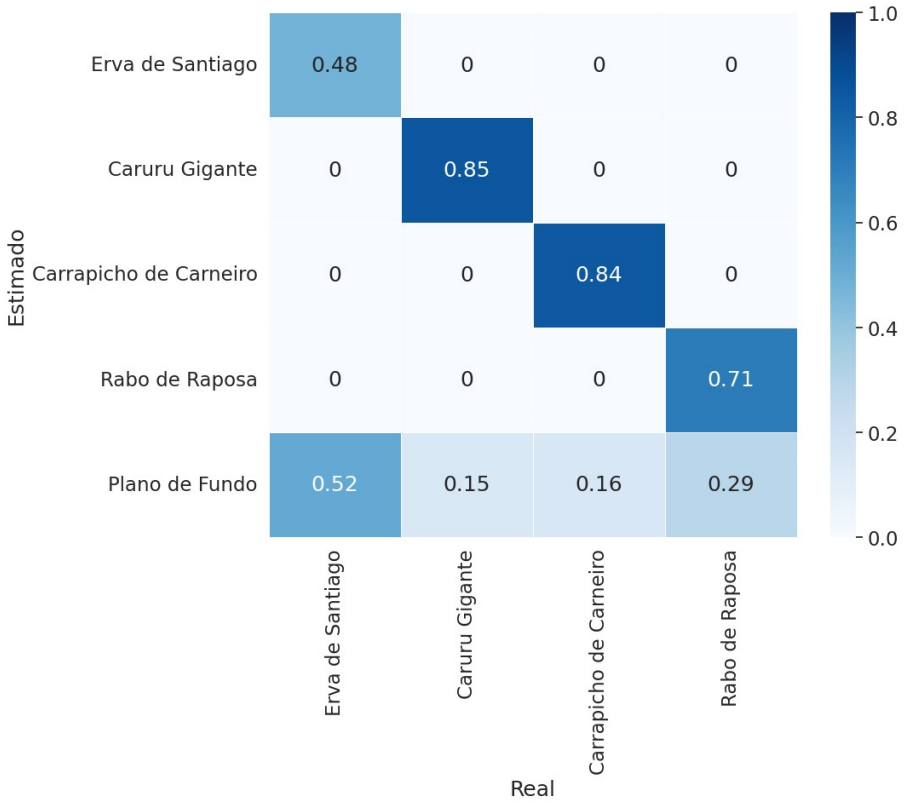
Fonte: Elaborado pelo autor

Figura 37 - Matriz de Confusão Yolov5n com pesos quantizados int8



Fonte: Elaborado pelo autor

Figura 38 - Matriz de Confusão Yolov5n com pesos convertidos para ONNX



Fonte: Elaborado pelo autor

Para a *Yolov5n* com base nas matrizes de confusão, tanto a quantização fp16 quanto a conversão para *ONNX* foram as abordagens que geraram melhores resultados na validação local. Por se tratar do modelo com menos parâmetros a quantização para int8 prejudicou a acurácia do modelo.

Na tabela 7 são apresentados os resultados com base nas métricas de avaliação para a *Yolov5n* com quantização e conversão dos pesos neurais. Essas métricas são a Precisão (*precision*) a Revocação (*recall*) a média das precisões médias com 0,5 de limiar e a média das precisões médias com liminar variando de 0,5 a 0,95.

O modelo com pesos convertidos para *ONNX* obteve resultados semelhantes aos das matrizes de confusão da rede *Yolov5n* com pesos quantizados para fp16. Ambos obtiveram os mesmos resultados para Precisão (0,805), Revocação (0,691) e *mAP 0,5:0,95* (0,374), tendo como média das precisões médias com 0,5 *mAP 0,5* de limiar 0,727.

Com a quantização para int8 os resultados foram inferiores com 0,712 de média das precisões médias com 0,5 de limiar, portanto, a quantização para fp16 e a conversão para *ONNX* apresentaram melhores resultados para a *Yolov5n*.

Tabela 7 - Resultados de Validação Rede Yolov5n pesos Quantizados e Convertidos

Rede Neural	Espécie de Planta	Precisão	Revocação	mAP 0,5	mAP 0,5:0,95
Yolov5n fp16	Erva de Santiago	0,803	0,478	0,573	0,287
	Caruru Gigante	0,907	0,861	0,879	0,551
	Carrapicho de Carneiro	0,858	0,831	0,915	0,507
	Rabo de Raposa	0,65	0,594	0,539	0,153
	Média	**0,805**	**0,691**	**0,727**	**0,374**
Yolov5n int8	Erva de Santiago	0,665	0,532	0,563	0,28
	Caruru Gigante	0,867	0,882	0,882	0,544
	Carrapicho de Carneiro	0,774	0,862	0,896	0,475
	Rabo de Raposa	0,56	0,594	0,509	0,139
	Média	**0,717**	**0,718**	**0,712**	**0,359**
Yolov5n onnx	Erva de Santiago	0,803	0,478	0,574	0,285
	Caruru Gigante	0,907	0,861	0,879	0,552
	Carrapicho de Carneiro	0,858	0,831	0,915	0,506
	Rabo de Raposa	0,651	0,594	0,539	0,153
	Média	**0,805**	**0,691**	**0,727**	**0,374**

Fonte: Elaborado pelo autor

4.3. Resultados Quantização e Conversão Yolov5s

Após os experimentos feitos com a rede *Yolov5n*, utilizou-se a mesma abordagem com o modelo *Yolov5s*. Os valores dos pesos neurais em *megabytes* (MB) nos formatos *tflite.fp16, tflite.int8* e *ONNX* para o modelo *Yolov5s* são apresentados na tabela 8. Os resultados obtidos nas matrizes de confusão e métricas de validação são apresentados a seguir.

Tabela 8 - Pesos Neurais Rede Yolov5s Quantizados e Convertidos

Rede Neural	Arquivo de Pesos em (MB)
Yolov5s fp16	14,2
Yolov5s int8	7,4
Yolov5s onnx	28,5

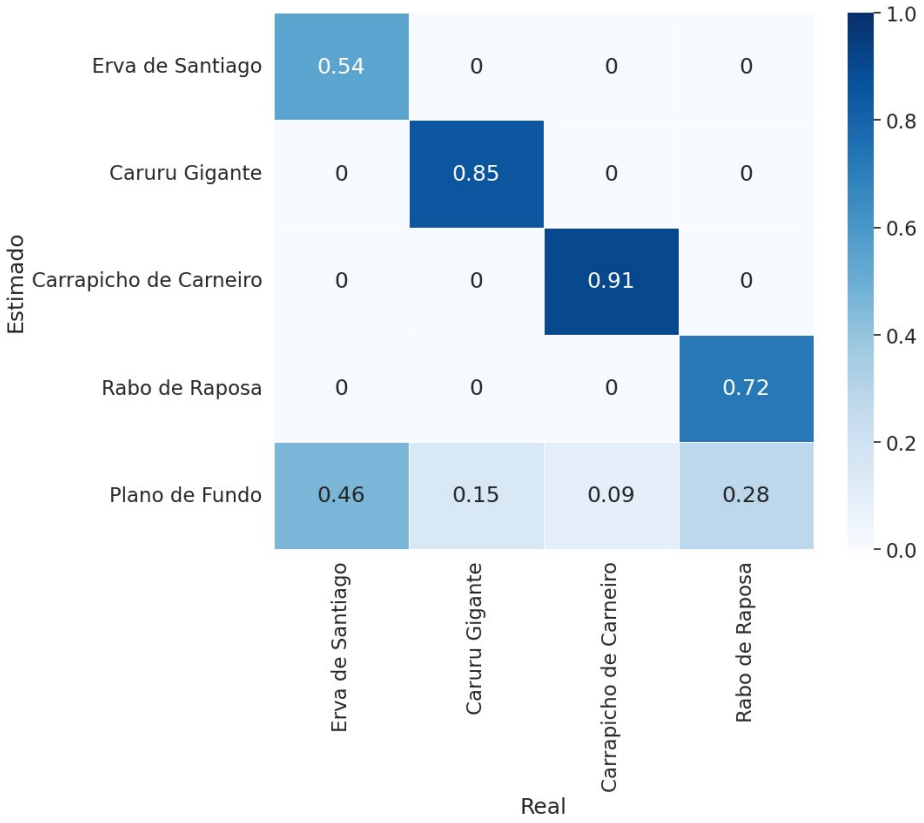
Fonte: Elaborado pelo autor

Na figura 39 é apresentada a matriz de confusão da rede *Yolov5s* com pesos quantizados para fp16, os melhores resultados foram obtidos para as plantas daninhas Carrapicho de Carneiro com 0,91, seguida por Caruru Gigante com 0,85 e Rabo de Raposa com 0,72, o pior resultado foi para Erva de Santiago com 0,54. Fazendo uma média dos valores da matriz obteve-se 0,75 ou 75% de acertos.

A matriz de confusão da rede *Yolov5s* com pesos quantizados para *int8* é apresentada na figura 40. Com base na matriz de confusão essa arquitetura obteve os melhores resultados dentre as 3 representações de pesos neurais, com 0,94 para Carrapicho de Carneiro, 0,85 para Caruru Gigante, 0,74 para Rabo de Raposa e 0,55 para Erva de Santiago. A média desses valores foi de 0,77 ou 77% de acertos.

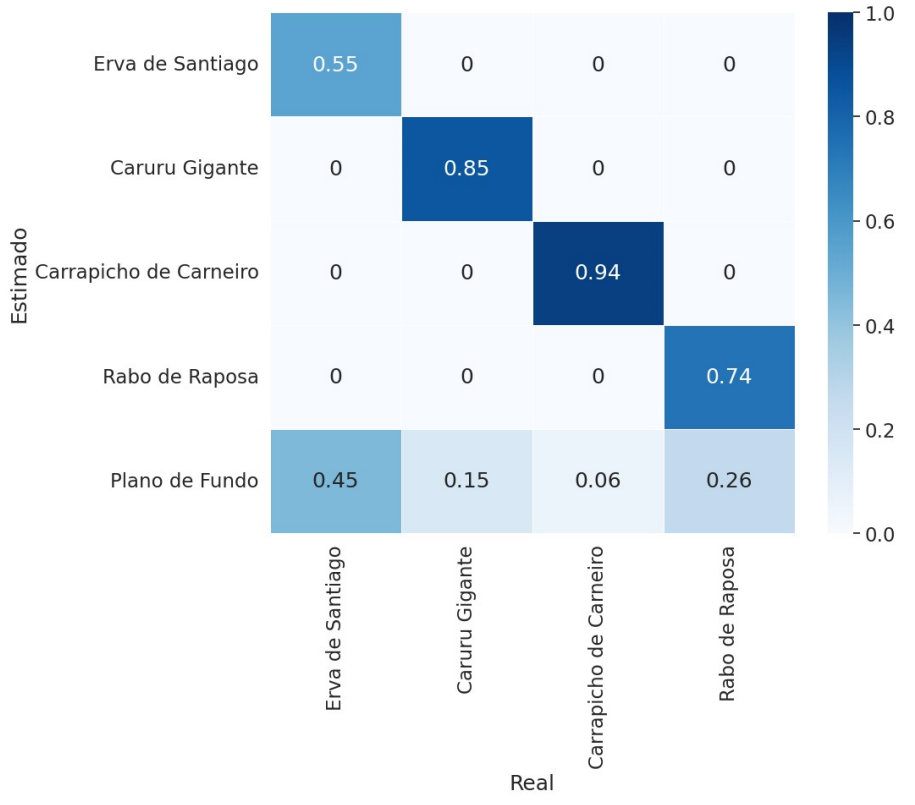
Na figura 41 é apresentada a matriz de confusão da rede *Yolov5s* com os pesos convertidos para *ONNX*. Assim como os resultados apresentados na rede *Yolov5n* com quantização fp16 e conversão para *ONNX*, os resultados obtidos nas matrizes de confusão com a conversão para *ONNX* para a rede *Yolov5s* foram idênticos aos da matriz de confusão da figura 39 com quantização em fp16. Com 0,91 para Carrapicho de Carneiro, seguida por Caruru Gigante com 0,85 e Rabo de Raposa com 0,72, o pior resultado foi para Erva de Santiago com 0,54. Fazendo uma média dos valores da matriz obteve-se 0,75 ou 75% de acertos.

Figura 39 - Matriz de Confusão Yolov5s com pesos quantizados fp16



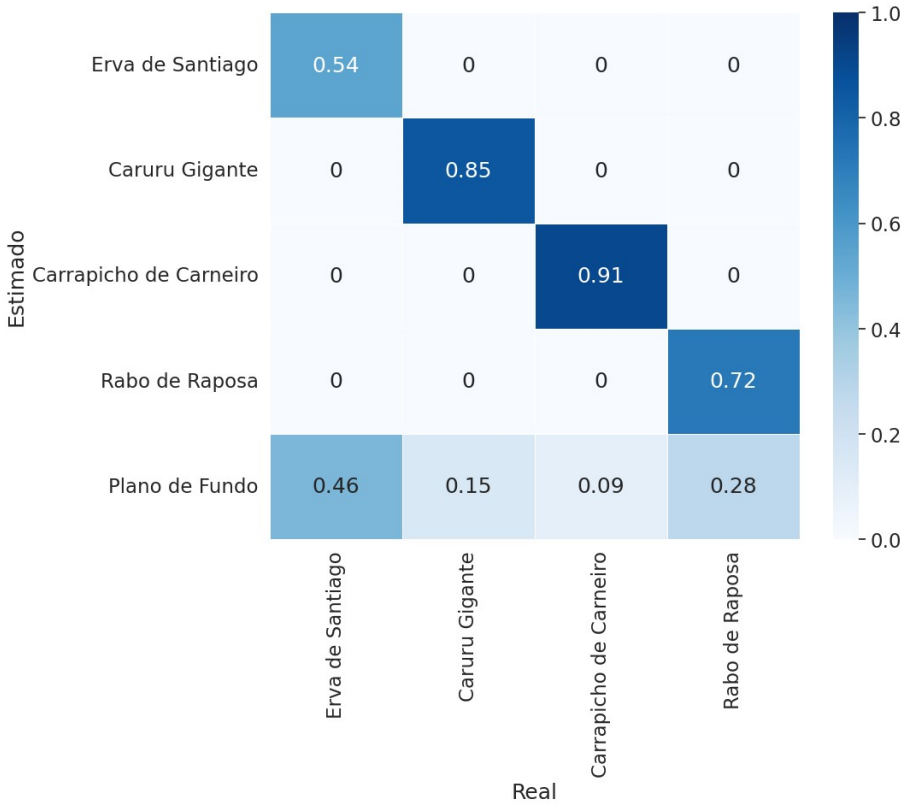
Fonte: Elaborado pelo autor

Figura 40 - Matriz de Confusão Yolov5s com pesos quantizados int8



Fonte: Elaborado pelo autor

Figura 41 - Matriz de Confusão Yolov5s com pesos convertidos para ONNX



Fonte: Elaborado pelo autor

Diferente dos resultados obtidos com a rede *Yolov5n* na rede *Yolov5s* a quantização para int8 gerou os melhores resultados com base nas matrizes de confusão. Isso ocorre principalmente pelo aumento do número de parâmetros, que permitiu a rede manter acurácia mesmo como pesos neurais representados em *int8*.

Na tabela 9 são apresentados os resultados das métricas de avaliação do modelo *Yolov5s* com quantização e conversão dos pesos neurais. Assim como demonstrado pelas matrizes de confusão a rede *Yolov5s* com quantização para int8 apresentou os melhores resultados com *mAP 0,5* de 0,77, Precisão de 0,822 e Revocação de 0,715, somente para a métrica *mAP 0,5:0,95* as representações em fp16 e *ONNX* foram superiores. A rede *Yolov5s* com quantização para fp16 e a conversão em *ONNX* apresentaram resultados idênticos com mAP 0,5 de 0,752, Precisão de 0,804 e Revocação de 0,711.

Tabela 9 - Resultados de Validação Rede Yolov5s pesos Quantizados e Convertidos

Rede Neural	Espécie de Planta	Precisão	Revocação	mAP 0,5	mAP 0,5:0,95
Yolov5s fp16	Erva de Santiago	0,774	0,475	0,553	0,275
	Caruru Gigante	1	0,842	0,922	0,586
	Carrapicho de Carneiro	0,809	0,888	0,918	0,485
	Rabo de Raposa	0,632	0,638	0,615	0,177
	Média	**0,804**	**0,711**	**0,752**	**0,381**
Yolov5s int8	Erva de Santiago	0,805	0,483	0,576	0,275
	Caruru Gigante	0,981	0,824	0,937	0,598
	Carrapicho de Carneiro	0,833	0,888	0,919	0,469
	Rabo de Raposa	0,668	0,667	0,648	0,175
	Média	**0,822**	**0,715**	**0,77**	**0,379**
Yolov5s onnx	Erva de Santiago	0,774	0,475	0,554	0,278
	Caruru Gigante	1	0,842	0,922	0,586
	Carrapicho de Carneiro	0,804	0,888	0,918	0,487
	Rabo de Raposa	0,633	0,638	0,614	0,178
	Média	**0,803**	**0,711**	**0,752**	**0,382**

Fonte: Elaborado pelo autor

4.4. Resultados Quantização e Conversão Yolov5m

Os valores dos pesos neurais em *megabytes* (MB) para os formatos *tflite.fp16, tflite.int8* e *ONNX* para o modelo *Yolov5m* são apresentados na tabela 10.

Tabela 10 - Pesos Neurais Rede Yolov5m Quantizados e Convertidos

Rede Neural	Arquivo de Pesos em (MB)
Yolov5m fp16	41,9
Yolov5m int8	21,5
Yolov5m onnx	83,9

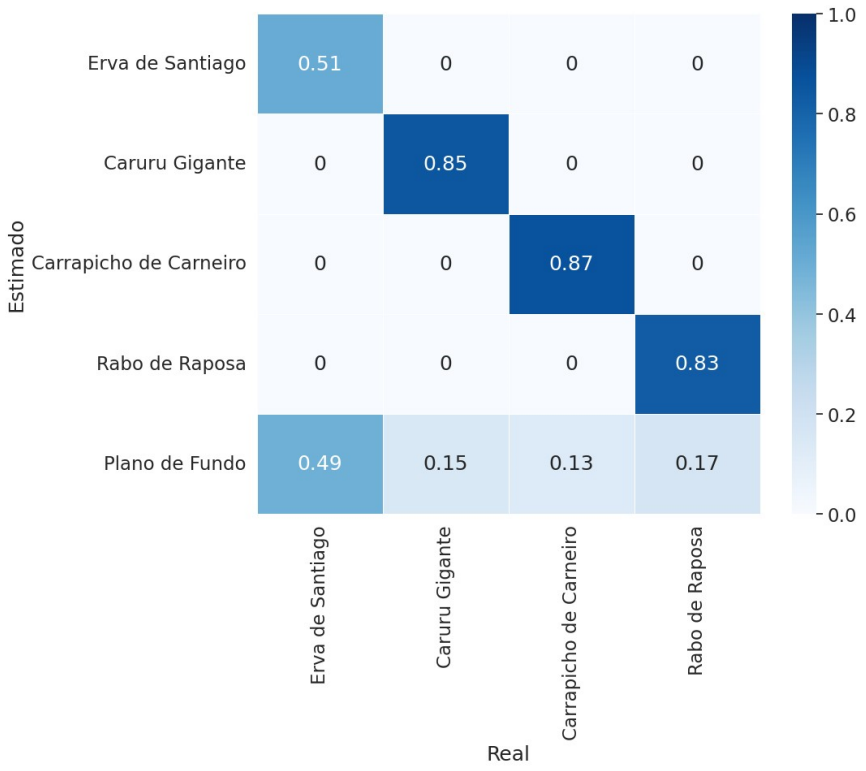
Fonte: Elaborado pelo autor

Na figura 42 é apresentada a matriz de confusão da rede *Yolov5m* com pesos quantizados para fp16. Para essa abordagem os melhores resultados obtidos foram para Carrapicho de Carneiro com 0,87, Caruru Gigante com 0,85 e Rabo de Raposa com 0,83, todos acima de 0,8. O pior resultado foi para Erva de Santiago com 0,51, a média dos valores da matriz foi de 0,765 ou 76,5% de acertos.

Na figura 43 é apresentada a matriz de confusão da rede *Yolov5m* com pesos quantizados para *int8*. Esse modelo foi pouco inferior ao modelo em fp16. Sendo que os melhores resultados foram para Carrapicho de Carneiro com 0,88, Caruru Gigante com 0,85 e Rabo de Raposa com 0,80. O pior resultado foi para Erva de Santiago com 0,51, a média dos valores da matriz de confusão Yolov5m com quantização para int8 foi de 0,762 ou 76,2% de acertos.

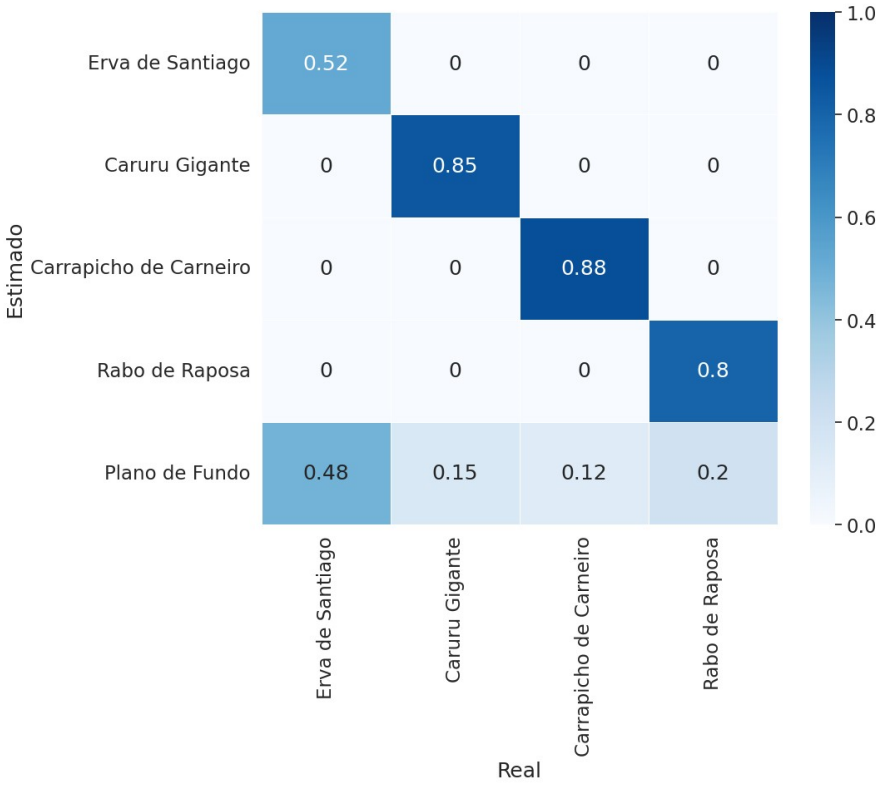
Na figura 44 é apresentada a matriz de confusão da rede *Yolov5m* com os pesos convertidos para *ONNX*. Os resultados obtidos para essa representação foram idênticos aos da matriz de confusão da figura 42 com quantização em fp16.

Figura 42 - Matriz de Confusão Yolov5m com pesos quantizados fp16



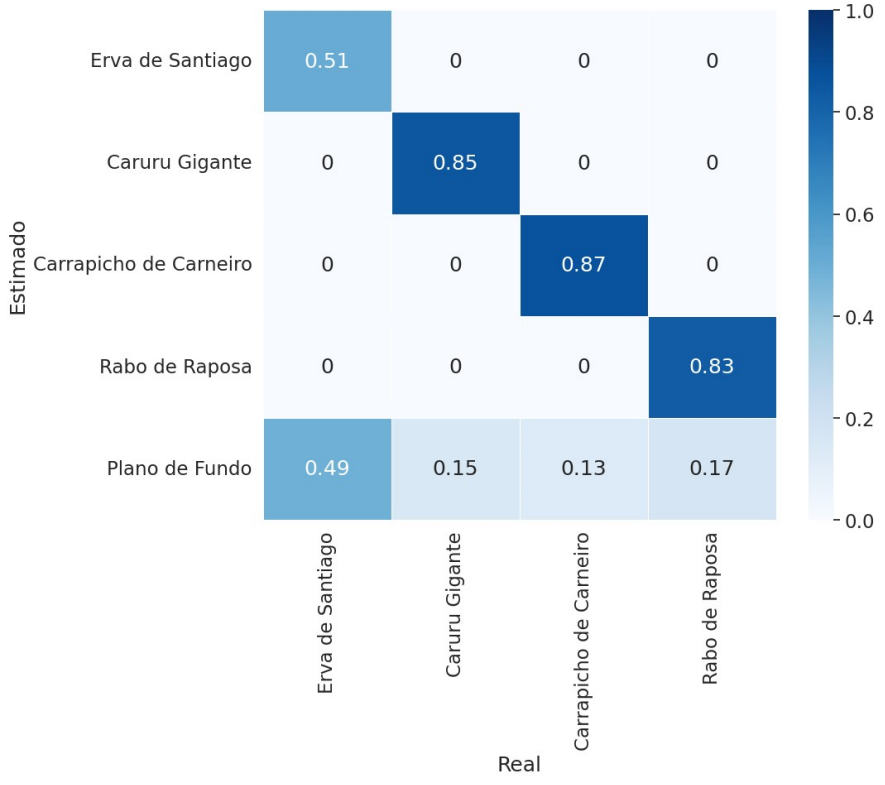
Fonte: Elaborado pelo autor

Figura 43 - Matriz de Confusão Yolov5m com pesos quantizados int8



Fonte: Elaborado pelo autor

Figura 44 - Matriz de Confusão Yolov5m com pesos convertidos para ONNX



Fonte: Elaborado pelo autor

Na tabela 11 são apresentados os resultados para as métricas de avalição para o modelo *Yolov5m* com quantização e conversão dos pesos neurais.

A rede *Yolov5m* com quantização para int8 teve os melhores resultados com mAP 0,5 de 0,762, Precisão com 0,834 e Revocação de 0,712, o mAP 0,5:0,95 foi pouco inferior comparado com as representações fp16 e *ONNX*.

As redes *Yolov5m* com quantização fp16 e a conversão em *ONNX* apresentaram resultados idênticos com mAP 0,5 de 0,738, Precisão de 0,822, Revocação de 0,706 e *mAP 0,5:0,95* de 0,39. Levando em conta os resultados de precisão, revocação e *mAP 0,5* o modelo com quantização para *int8* foi superior aos modelos com quantização *fp16* e conversão em *ONNX*, demonstrando uma estabilidade do modelo mesmo com pesos neurais em um formato de menor precisão.

Tabela 11 - Resultados de Validação Rede Yolov5m pesos Quantizados e Convertidos

Rede Neural	Espécie de Planta	Precisão	Revocação	mAP 0,5	mAP 0,5:0,95
Yolov5m fp16	Erva de Santiago	0,8	0,469	0,546	0,274
	Caruru Gigante	0,997	0,853	0,88	0,614
	Carrapicho de Carneiro	0,842	0,836	0,893	0,492
	Rabo de Raposa	0,65	0,667	0,634	0,179
	Média	**0,822**	**0,706**	**0,738**	**0,39**
Yolov5m int8	Erva de Santiago	0,835	0,479	0,587	0,267
	Caruru Gigante	1	0,852	0,888	0,601
	Carrapicho de Carneiro	0,839	0,853	0,898	0,476
	Rabo de Raposa	0,662	0,667	0,675	0,181
	Média	**0,834**	**0,712**	**0,762**	**0,381**
Yolov5m onnx	Erva de Santiago	0,8	0,468	0,546	0,273
	Caruru Gigante	0,997	0,853	0,88	0,614
	Carrapicho de Carneiro	0,842	0,836	0,893	0,493
	Rabo de Raposa	0,65	0,667	0,635	0,18
	Média	**0,822**	**0,706**	**0,738**	**0,39**

Fonte: Elaborado pelo autor

4.5. Resultados Quantização e Conversão Yolov5l

Após os experimentos com as redes *Yolov5n, Yolov5s e Yolov5m* foi feita a validação da rede *Yolov5l.* Os valores dos pesos neurais em *megabytes* (MB) para os formatos *tflite.fp16, tflite.int8* e *ONNX* para o modelo *Yolov5l* são apresentados na tabela 12.

Tabela 12 - Pesos Neurais Rede Yolov5m Quantizados e Convertidos

Rede Neural	Arquivo de Pesos em (MB)
Yolov5l fp16	92,5
Yolov5l int8	47,1
Yolov5l onnx	184,9

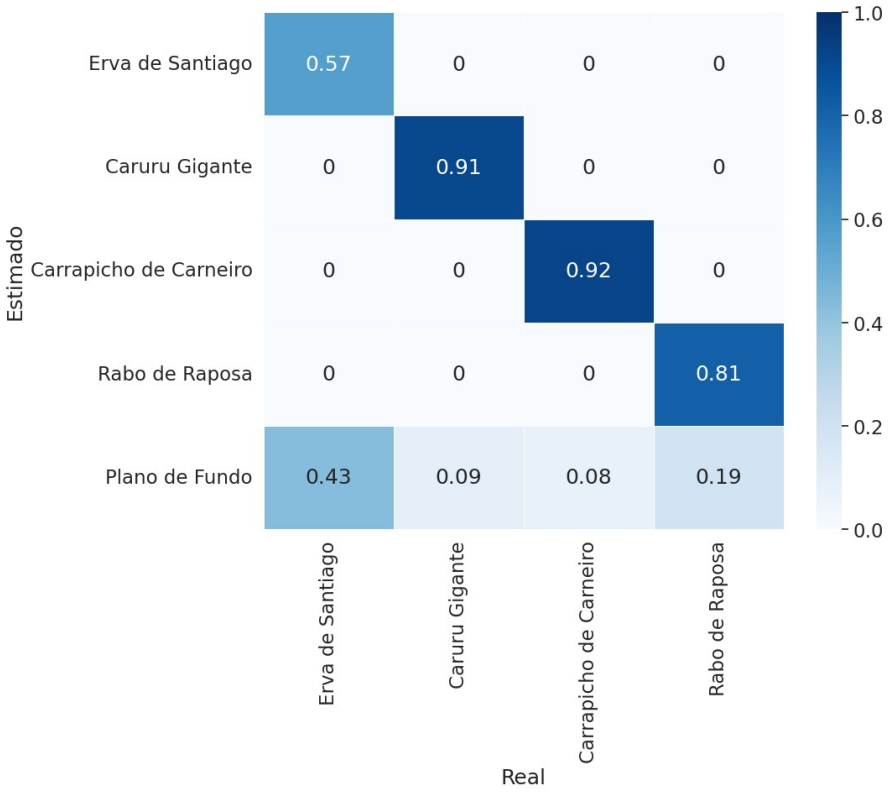
Fonte: Elaborado pelo autor

Na figura 45 é apresentada a matriz de confusão da rede *Yolov5l* com pesos quantizados para *fp16*. Os melhores resultados foram obtidos para a planta daninha Carrapicho de Carneiro com 0,92, seguida por Caruru Gigante com 0,91 e Rabo de Raposa com 0,81, com destaque para as duas primeiras espécies com valores acima de 0,9. Novamente o pior resultado foi para Erva de Santiago com 0,57, a média dos valores da matriz foi de 0,802 ou 80,2% de acertos.

A matriz de confusão da rede *Yolov5l* com pesos quantizados para *int8* é apresentada na figura 46. Esse modelo foi pouco inferior ao modelo em *fp16*, os melhores resultados foram Carrapicho de Carneiro com 0,92, Caruru Gigante com 0,91 e Rabo de Raposa com 0,81, semelhante a abordagem em *fp16* a rede *Yolov5l* com quantização *int8* obteve resultados superiores a 0,9 para as duas primeiras espécies de plantas. O pior resultado foi para Erva de Santiago com 0,54, a média dos valores da matriz foi de 0,762 ou 76,2% de acertos.

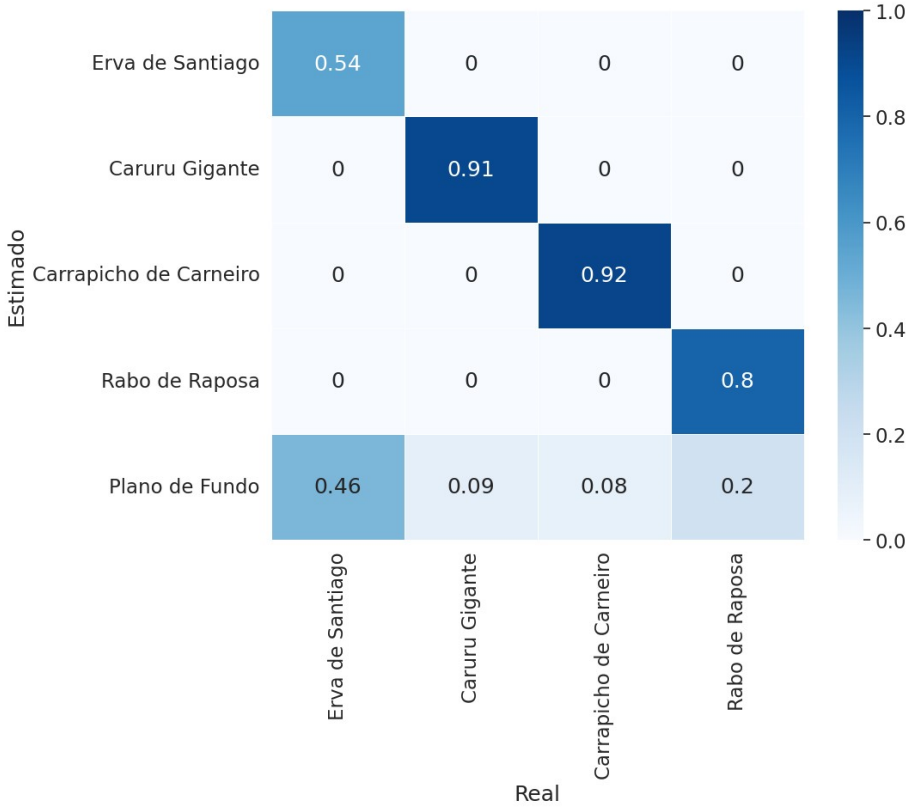
Na figura 47 é apresentada a matriz de confusão da rede *Yolov5l* com os pesos convertidos para ONNX. Os resultados obtidos foram idênticos aos da matriz de confusão da figura 45 com quantização em fp16, sendo que a média dos valores da matriz foi de 0,802 ou 80,2% de acertos.

Figura 45 - Matriz de Confusão Yolov5l com pesos quantizados fp16



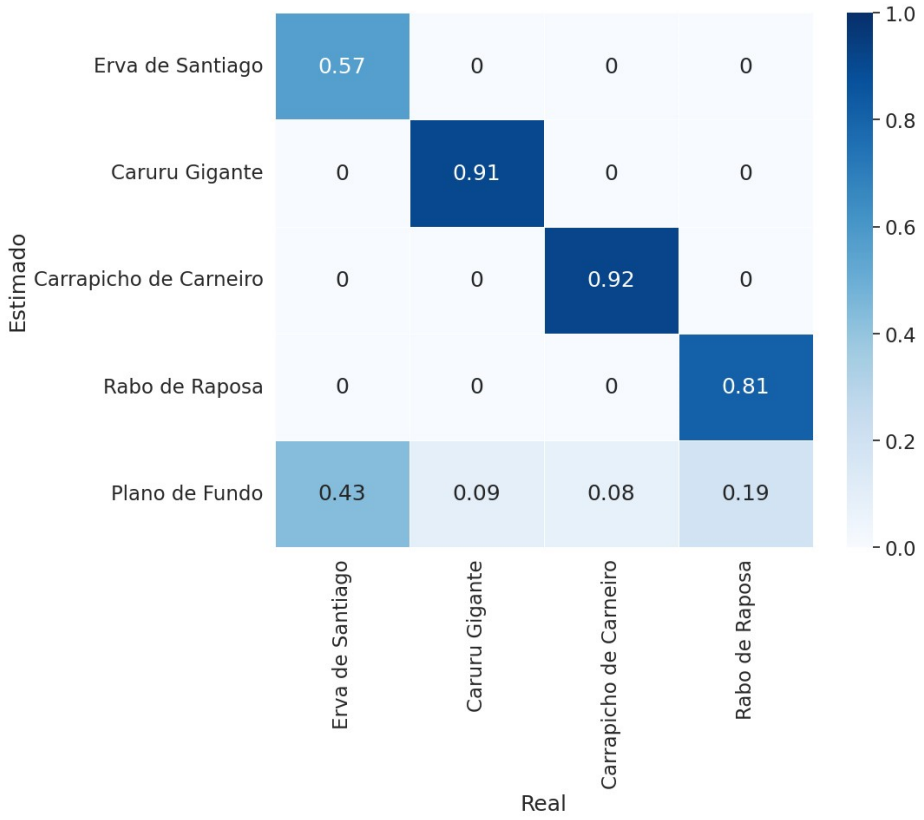
Fonte: Elaborado pelo autor

Figura 46 - Matriz de Confusão Yolov5l com pesos quantizados int8



Fonte: Elaborado pelo autor

Figura 47 - Matriz de Confusão Yolov5l com pesos convertidos para ONNX



Fonte: Elaborado pelo autor

Na tabela 13 são apresentados os resultados do modelo *Yolov5l* com quantização e conversão dos pesos neurais com base nas métricas para avaliação. A rede *Yolov5l* com quantização para *int8* teve os melhores resultados *com mAP 0,5* com 0,815, Precisão com 0,834 e Revocação de 0,772. O *mAP 0,5:0,95* do modelo quantizado para *int8* foi pouco inferior quando comparado com o modelo quantizado para *fp16* e convertido para *ONXX*

As redes *Yolov5l* com quantização fp16 e a conversão em *ONNX* apresentaram resultados idênticos com *mAP 0,5* de 0,801, Precisão de 0,823, Revocação de 0,762 *e mAP 0,5:0,95* de 0,39. Levando em conta os resultados de precisão, revocação e *mAP 0,5* o modelo com quantização para int8 foi superior aos modelos com quantização *fp16* e a conversão em *ONNX*.

Tabela 13 - Resultados de Validação Rede Yolov5l pesos Quantizados e Convertidos

Rede Neural	Espécie de Planta	Precision	Recall	mAP 0,5	mAP 0,5:0,95
Yolov5l fp16	Erva de Santiago	0,875	0,511	0,632	0,315
	Caruru Gigante	0,981	0,912	0,969	0,666
	Carrapicho de Carneiro	0,835	0,914	0,917	0,501
	Rabo de Raposa	0,602	0,71	0,686	0,188
	Média	**0,823**	**0,762**	**0,801**	**0,418**
Yolov5l int8	Erva de Santiago	0,926	0,529	0,659	0,316
	Caruru Gigante	0,963	0,912	0,969	0,662
	Carrapicho de Carneiro	0,816	0,922	0,92	0,487
	Rabo de Raposa	0,632	0,725	0,713	0,189
	Média	**0,834**	**0,772**	**0,815**	**0,413**
Yolov5l onnx	Erva de Santiago	0,893	0,521	0,658	0,317
	Caruru Gigante	0,979	0,912	0,969	0,666
	Carrapicho de Carneiro	0,835	0,914	0,917	0,501
	Rabo de Raposa	0,598	0,711	0,686	0,188
	Média	**0,826**	**0,764**	**0,807**	**0,418**

Fonte: Elaborado pelo autor

4.6. Resultados Quantização e Conversão Yolov5x

Os valores dos pesos neurais em *megabytes* (MB) nos formatos *tflite.fp16, tflite.int8* e *ONNX* para o modelo *Yolov5x* são apresentados na tabela 14.

Tabela 14 - Pesos Neurais Rede Yolov5x Quantizados e Convertidos

Rede Neural	Arquivo de Pesos em (MB)
Yolov5sx fp16	172,6
Yolov5x int8	87,6
Yolov5x onnx	345,2

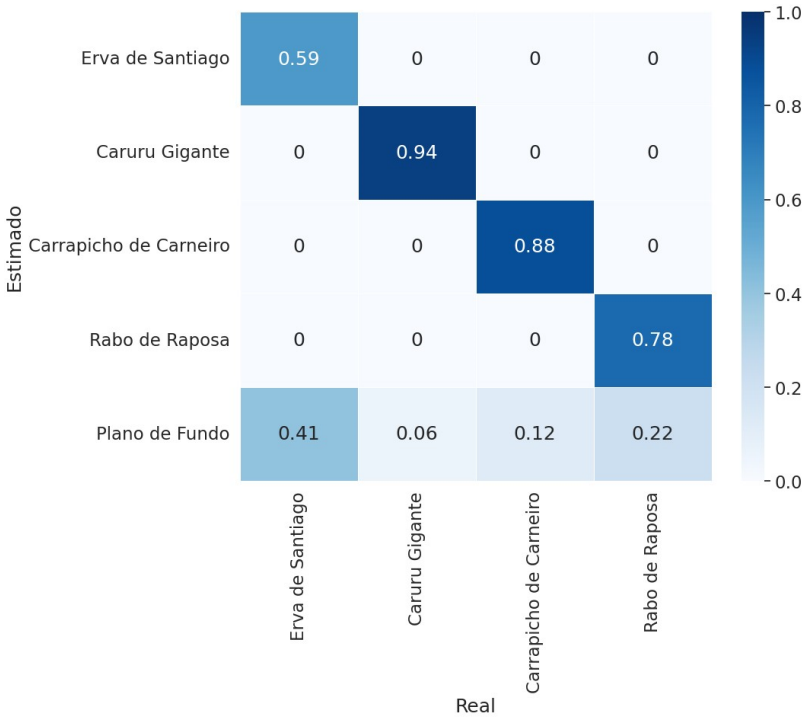
Fonte: Elaborado pelo autor

Na figura 48 é apresentada a matriz de confusão da rede *Yolov5x* com pesos quantizados para *fp16*. Os melhores resultados foram obtidos para Caruru Gigante com 0,94 seguido por Carrapicho de Carneiro com 0,88 e Rabo de Raposa com 0,78. Erva de Santiago apresentou resultado inferior as demais espécies com 0,59, a média dos valores da matriz foi de 0,7972 ou 79,7% de acertos.

A matriz de confusão da rede *Yolov5x* com pesos quantizados para *int8* é apresentada na figura 49. Esse modelo foi pouco inferior ao modelo em *fp16*, os melhores resultados foram para Caruru Gigante com 0,94 seguido por Carrapicho de Carneiro com 0,87 e Rabo de Raposa com 0,77. O pior resultado foi para Erva de Santiago com 0,55, a média dos valores da matriz foi de 0,782 ou 78,2% de acertos.

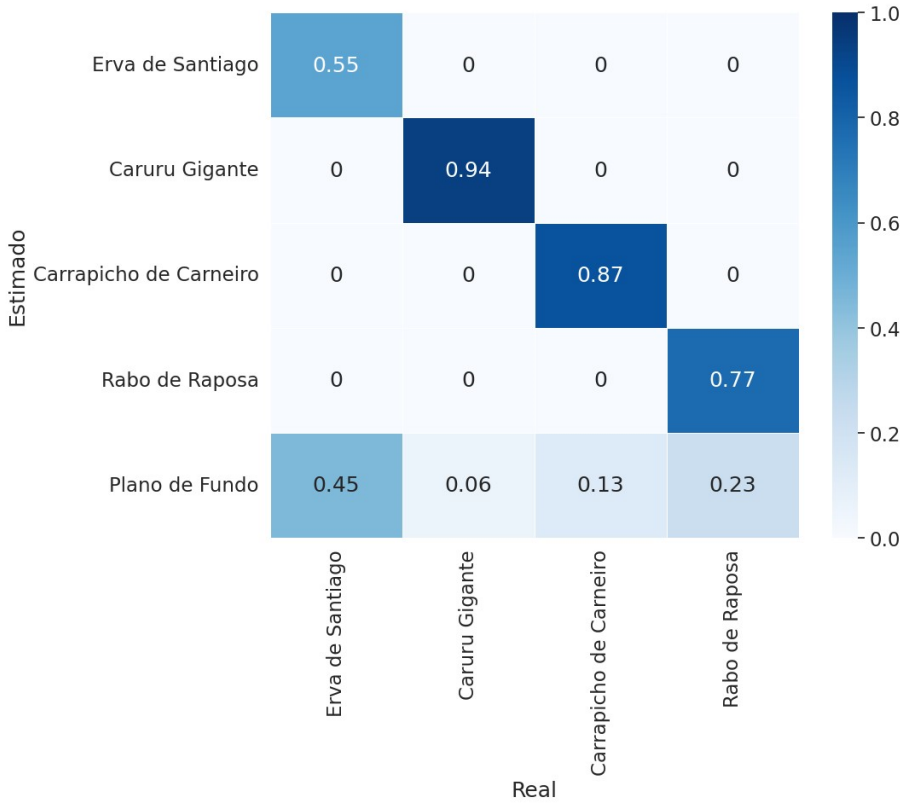
Na figura 50 é apresentada a matriz de confusão da rede *Yolov5x* com os pesos convertidos para *ONNX*. Os resultados obtidos foram idênticos aos da matriz de confusão da figura 48 com quantização em *fp16*. Onde os melhores resultados foram obtidos para Caruru Gigante com 0,94 seguido por Carrapicho de Carneiro com 0,88 e Rabo de Raposa com 0,78. Erva de Santiago apresentou resultado inferior as demais espécies com 0,59, a média dos valores da matriz foi de 0,7972 ou 79,7% de acertos.

Figura 48 - Matriz de Confusão Yolov5x com pesos quantizados fp16



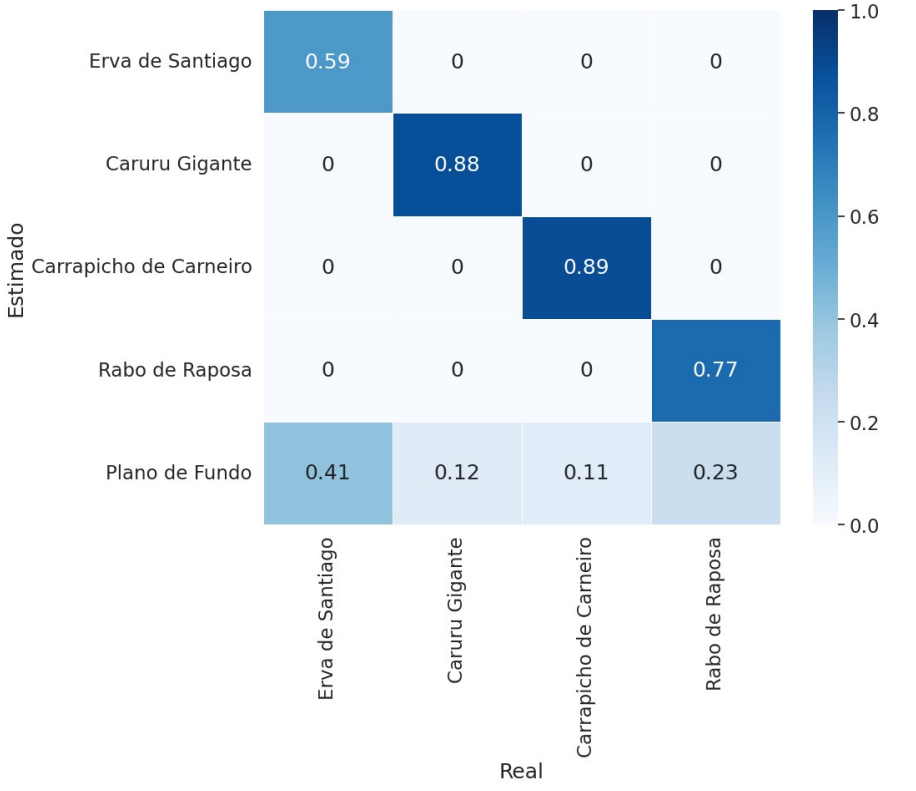
Fonte: Elaborado pelo autor

Figura 49 - Matriz de Confusão Yolov5x com pesos quantizados int8



Fonte: Elaborado pelo autor

Figura 50 - Matriz de Confusão Yolov5x com pesos convertidos para ONNX



Fonte: Elaborado pelo autor

Na tabela 15 são apresentados os resultados do modelo *Yolov5x* com quantização e conversão dos pesos neurais. A rede *Yolov5l* com quantização para int8 teve os piores resultados com mAP 0,5 de 0,759, Precisão de 0,777, Revocação de 0,739 e mAP 0,5:0,95 com 0,384.

As redes *Yolov5x* com quantização fp16 e com conversão em *ONNX* apresentaram os melhores resultados, sendo que estes resultados foram idênticos para ambas abordagens. Com *mAP 0,5* de 0,785, Precisão de 0,804, Revocação de 0,774 e *mAP 0,5:0,95* de 0,41.

A rede *Yolov5x* obteve os melhoes resultados para as abordagens de quantização em *fp16* e conversão para *ONNX*. Semelhante ao que ocorreu com a rede *Yolov5n* que é o modelo com o menor número de parâmetros.

Portantanto, os melhores resultados foram obtidos com as arquiteturas intermediarias *Yolov5s, Yolov5m e Yolov5l* com os pesos quantizados em inteiros de 8 *bits* (*int8*), enquanto que as redes extremas com mais e menos parâmetros respectivamente *Yolov5n* e *Yolov5x* apresentaram melhores resultados com a quatização em ponto flutuante de 16 *bits* (fp16) e conversão para o *ONNX* (*Open Neural Network Exchange*).

Após a validação dos pesos neurais quantizados para *fp16, int8* e convertidos para *ONNX* foi realizada a inferência local com esses pesos. Para a inferência foi feita a detecção de plantas daninhas nas imagens do *dataset* de validação. Os resultados das inferêcias dos modelos *Yolov5n, Yolov5s, Yolov5m, Yolov5l e Yolov5x* representados nos formatos *int8, fp16* e *ONNX* são apresentados a seguir.

Tabela 15 - Resultados de Validação Rede Yolov5l pesos Quantizados e Convertidos

Rede Neural	Espécie de Planta	Precision	Recall	mAP 0,5	mAP 0,5:0,95
Yolov5x fp16	Erva de Santiago	0,771	0,574	0,619	0,291
	Caruru Gigante	0,899	0,941	0,962	0,652
	Carrapicho de Carneiro	0,891	0,888	0,916	0,517
	Rabo de Raposa	0,656	0,692	0,644	0,182
	Média	**0,804**	**0,774**	**0,785**	**0,41**
Yolov5x int8	Erva de Santiago	0,76	0,532	0,592	0,266
	Caruru Gigante	0,887	0,925	0,961	0,639
	Carrapicho de Carneiro	0,86	0,848	0,913	0,476
	Rabo de Raposa	0,602	0,652	0,57	0,153
	Média	**0,777**	**0,739**	**0,759**	**0,384**
Yolov5x onnx	Erva de Santiago	0,774	0,574	0,619	0,291
	Caruru Gigante	0,899	0,941	0,962	0,652
	Carrapicho de Carneiro	0,894	0,888	0,916	0,519
	Rabo de Raposa	0,656	0,691	0,644	0,182
	Média	**0,806**	**0,774**	**0,785**	**0,411**

Fonte: Elaborado pelo autor

4.7. Resultados de Inferência Redes Yolov5 com Pesos Quantizados e Convertidos

Para avaliar o tempo de inferência dos modelos *Yolov5* usou-se as 100 imagens do conjunto de validação, onde foi feita a detecção das plantas invasoras em cada imagens e obteve-se o tempo de inferência para cada modelo.

Na tabela 16 é apresentado o tempo médio de inferência da rede *Yolov5n*. Para esse modelo apesar de ter pesos neurais maiores, a representação em *ONNX* feita via *ONNX runtime inference* teve o menor e melhor tempo de inferência. Com uma imagen inferida a casa 0,376 segundo. Ou seja, praticamente 3 imagens por segundo.

Com a quantização int8 foi obtido 0,398 imagens inferidas por segundo e com a quantização fp16 0,512, praticamente 2 imagens inferidas por segundo.

Tabela 16 - Resultados de Inferência Rede Yolov5n pesos Quantizados e Convertidos

Rede Neural	Tempo de inferência imagens (s)
Yolov5n fp16	0,512
Yolov5n int8	0,398
Yolov5n onnx	0,376

Fonte: Elaborado pelo autor

Na tabela 17 é apresentado o tempo médio de inferência da rede *Yolov5s*, nessa configuração a inferência com pesos quantizados para *int8* teve os melhores resultados com 1,048 imagem inferida por segundo. Ou seja, aproximadamente uma imagem inferida por segundos.

A conversão para *ONNX* apresentou o segundo melhor tempo de inferência com 1,084 imagem inferida por segundo.

Tabela 17 - Resultados de Inferência Rede Yolov5s pesos Quantizados e Convertidos

Rede Neural	Tempo de inferência imagens (s)
Yolov5s fp16	1,740
Yolov5s int	1,048
Yolov5s onnx	1,084

Fonte: Elaborado pelo autor

Na tabela 18 é apresentado o tempo médio de inferência da rede *Yolov5m*. Novamente a inferência com pesos quantizados para *int8* teve os melhores com 2,738 imagem inferida por segundos.

A conversão para *ONNX* teve o segundo melhor tempo de inferência com 3,023 imagem inferida por segundo. Por fim com a quantização em fp16 foi obtido 5,166 imagem por segundos. Ou seja enquanto que com a conversão *int8* a inferência é de

aproximadamente 1 frame a cada 2,7 segundos com a quantização para fp16 tem-se 1 frame inferido a cada 5,1 segundos.

Tabela 18 - Resultados de Inferência Rede Yolov5m pesos Quantizados e Convertidos

Rede Neural	Tempo de inferência imagens (s)
Yolov5m fp16	5,166
Yolov5m int8	2,738
Yolov5m onnx	3,023

Fonte: Elaborado pelo autor

Na tabela 19 é apresentado o tempo médio de inferência da rede *Yolov5l*, nessa configuração a inferência com pesos quantizados para *int8* teve os melhores resultados com 5,685 imagem inferida por segundos.

A conversão para *ONNX* teve o segundo melhor tempo de inferência com 6,252 imagem inferida por segundo. Ja a representação da *Yolov5l* com a quantização em fp16 teve o maior tempo de inferência foi obtido 11,817 imagem inferida por segundos. Praticamente o dobro de tempo de inferência comparando com os resultados de *int8*.

Tabela 19 - Resultados de Inferência Rede Yolov5l pesos Quantizados e Convertidos

Rede Neural	Tempo de inferência imagens (s)
Yolov5l fp16	11,817
Yolov5l int8	5,685
Yolov5l onnx	6,252

Fonte: Elaborado pelo autor

Na tabela 20 é apresentado o tempo médio de inferência da rede *Yolov5x*. Nessa configuração a inferência com pesos quantizados para int8 teve os melhores resultados com 10,302 imagem inferida por segundo.

A conversão para *ONNX* teve o segundo menor tempo de inferência com 11,402 imagem inferida por segundo. Para a quantização em *fp16* foi obtido 23,111 imagem por segundo. Ou seja enquanto que com a quantização em int8 a inferência é de aproximadamente 1 frame a cada 10,3 segundos para o modelo *Yolov5x,* com a quantização em fp16 tem-se 1 imagem inferida a cada 23 segundos. Na tabela 21 os resultados de tempo de inferência para todos os modelos com quantização e conversão dos pesos neurais estão sintetizados.

Tabela 20 - Resultados de Inferência Rede Yolov5x pesos Quantizados e Convertidos

Rede Neural	Tempo de inferência imagens (s)
Yolov5x fp16	23,111
Yolov5x int8	10,302
Yolov5x onnx	11,402

Fonte: Elaborado pelo autor

Tabela 21 - Resultados de Inferência Redes Yolov5 com pesos Quantizados e Convertidos

Rede Neural	Tempo de inferência imagens (s)
Yolov5n- fp16	0,512
Yolov5n- int8	0,398
Yolov5n-onnx	0,376
Yolov5s- fp16	1,740
Yolov5s- int8	1,048
Yolov5s- onnx	1,084
Yolov5m- fp16	5,166
Yolov5m-int8	2,738
Yolov5m- onnx	3,023
Yolov5l- fp16	11,817
Yolov5l-int8	5,685
Yolov5l- onnx	6,252
Yolov5x- fp16	23,111
Yolov5x-int8	10,302
Yolov5x- onnx	11,402

Fonte: Elaborado pelo autor

Um sistema de identificação de plantas daninhas é necessário, visto o aumento na demanda alimentícia, dos custos de produção causados pelas espécies invasoras e por possibilitar a aplicação localizada diferindo as espécies de plantas causando menor impacto ambiental.

Com base nessa demanda, neste trabalho foi utilizado um conjunto de imagens públicas de plantas daninhas prejudiciais para as culturas de milho e soja, onde foram treinados os modelos *Yolov5.* Após o treinamento obteve-se as métricas Precisão (*precision*), Revocação (*recall*), Média das Precisões Médias com 0,5 de Limiar (*mAP 0,5*) e a Média das Precisões Médias com Liminar variando de 0,5 a 0,95 (*mAP*) no *dataset* de validação.

Na validação com os pesos em *float32* a rede *Yolov5l* obteve os melhores resultados com *mAP 0,5* de 0,792, precisão de 0,815, revocação de 0,788 e *e mAP 0,5:0,95* de 0,416. Levanto em conta os valores da matriz de confusão desse modelo a *Yolov5l* teve 0,78 ou 78% de acertos entre as 4 espécies de plantas daninhas.

Posteriormente fez-se uso da quantização pós treinamento para comprimir os pesos neurais para os formatos inteiro de 8 *bits int8* e ponto flutuante de 16 *bits fp16*, bem como converter os pesos para *ONNX* ou trocador de rede neural aberto. Esses pesos juntamente com o *dataset* de validação foram transferidos para *Raspberry Pi 4 model B 4GB*, onde foi feita a validação e inferência dos modelos.

Os melhores resultados de validação foram obtidos com a rede *Yolov5l* com quantização em int8 com *mAP 0,5* de 0,815, precisão de 0,834, revocação de 0,772 e e *mAP 0,5:0,95* de 0,413. Em relação a inferência, a Yolov5l teve os melhores resultados com a quantização em *int8,* onde foi obtido uma imagem inferida a cada 5,685. Levanto em conta os valores da matriz de confusão desse modelo a *Yolov5l* teve melhores resultados com a quantização para *fp16* e conversão para *ONNX* com 0,802 ou 80,2% de acertos entre as 4 espécies de plantas daninhas.

Cabe salientar também que tanto os modelos quantizados em *fp16* quanto os convertidos para *ONNX* apresentaram os mesmos resultados para as matrizes de confusão e demais métricas computadas diferindo apenas em tempo de inferência.

O modelo mais balanceado foi a *Yolov5s* com pesos quantizados em *int8* com *mAP 0,5* de 0,77, precisão de 0,822, revocação de 0,715 e e *mAP 0,5:0,95* de 0,379. Levando em conta a média dos resultados das matrizes de confusão, novamente os melhores resultados foram obtidos com a representação em *int8*, com 0,77 ou 77% Em relação a inferência com a quantização em *int8* foi obtido uma imagem inferida a cada 1,048 segundos.

O modelo com menor tempo de inferência foi a *Yolov5n* com pesos convertidos para *ONNX* com *mAP 0,5* de 0,727, precisão de 0,805, revocação de 0,691 e e *mAP 0,5:0,95* de 0,374. Em relação a inferência com a conversão para *ONNX* foi obtido uma imagem inferida a cada 0,376 segundo para as imagens e a cada 0,376 segundo, cerca de 3 imagens inferidas por segundo. Tal resultado permite que testes mais aprofundados sejam feitos para implementar esse sistema em plataformas tratorizadas.

CAPÍTULO 5: CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo são apresentas as conclusões com base nos resultados obtidos, também são elencadas as principais contribuições dessa tese e propostas abordagens para trabalhos futuros.

5.1. Conclusões

Para este trabalho foi proposto um sistema para identificar as espécies invasoras para as culturas de milho e soja utilizando as arquiteturas *Yolov5* quantizadas para os formatos fp16, int8 e convertidas para *ONNX* na placa *Raspberry Pi model B 4GB.*

Como principal contribuição deste trabalho, foi proposto e demonstrado uma abordagem para implantar um sistema de inteligência artificial com quantização e conversão de pesos aplicados na identificação e detecção seletiva de plantas daninhas prejudiciais as culturas de milho e soja em um *hardware* de baixo custo. Tal abordagem traz como vantagens: i) Permite a identificação *on-the-go* de plantas daninhas em plataformas tratorizadas; ii) Permite a erradicação de plantas daninhas de forma seletiva; iii) Promove a economia de insumos agrícolas; iv) Contribui para a redução de impacto ambiental; v) O sistema apresenta baixo custo para implementação. Como principal desvantagem pode-se citar: i) A velocidade de deslocamento em campo é sensível ao desempenho do *hardware* de baixo custo utilizado.

5.2. Propostas para Trabalhos Futuros

Como sugestões para trabalhos futuros sugere-se: i) Criação de vídeos obtidos com câmeras acopladas em sistemas tratorizados para avaliar a inferência das diferentes representações de pesos neurais; ii) Implantação do sistema de identificação de plantas daninhas em uma plataforma robotizada ou acoplado em um trator agrícola para visando apurar o aprendizado para outras espécies de plantas daninhas encontradas no Brasil, iii) Avaliar o desempenho e a velocidade de deslocamento em campo para outras plataformas de hardware de baixo custo.

REFERÊNCIAS

A. PASZKE et al. **PyTorch: An Imperative Style, High-Performance Deep Learning Library**. *In NeurIPS*, Vancouver, v. 32, p. 8024—8035, 2019.

AGGARWAL, V. et al. **4Weed Dataset: Annotated Imagery Weeds Dataset**. Cite arXiv:2204.00080, 2022.

AI. STANFORD. **Totorial 1: Image Filtering**. Estados Unidos 2022. Disponível em: https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html . Acesso em 25/05/2022.

AHMED. S et al. OpTorch: **Optimized deep learning architectures for resource limited environments.** Cite arXiv:2105.00619, 2021.

ALLABOUTCIRCUITS. Neural Network Quantization: **What Is It and How Does It Relate to TinyML.** Estados Unidos, 2022. Disponível em: https://www.allaboutcircuits.com/technical-articles/neural-network-quantization-what-is-it-and-how-does-it-relate-to-tiny-machine-learning/ . Acesso em 30/04/2022.

BISONG. E. Google Colaboratory. In: **Building Machine Learning and Deep Learning Models on Google Cloud Platform.** Apress, Berkeley, CA, 2019.

BOCHKOVSKIY. A et al. **YOLOv4: Optimal Speed and Accuracy of Object Detection**. Cite arXiv:2004.10934, 2020.

CEPEA. **PIB-AGRO/CEPEA**. Brasil, 2022. Disponível em: https://www.cepea.esalq.usp.br/br/releases/pib-agro-cepea-pib-do-agro-cresce-8-36-em-2021-participacao-no-pib-brasileiro-chega-a-27-4.aspx. Acesso em 07/05/2022.

DECI.AI. **The Ultimate Guide to Dee Learning Model Quantization and Quantization-Aware Training.** Estados Unidos, 2023. Disponível em: https://deci.ai/quantization-and-quantization-aware-training/. Acesso em 22/03/2023.

DEEPAI. **Convolutional Neural Network**. Estados Unidos, 2022. Disponível em: https://deepai.org/machine-learning-glossary-and-terms/convolutional-neural-network>/. Acesso em 22/04/2022.

DEEP LEARNING BOOK. **Capítulo 3 – O Que São Redes Neurais Artificiais Profundas ou Deep Learning, Biológico e Matemático.** Estados Unidos, 2022. Disponível em: https://www.deeplearningbook.com.br/o-que-sao-redes-neurais-artificiais-profundas/ Acesso em 21/04/2022.

DEEP LEARNING BOOK. **Capítulo 4 – O Neurônio, Biológico e Matemático.** Estados Unidos, 2022. Disponível em: http://deeplearningbook.com.br/o-neuronio-biologico-e-matematico/. Acesso em 21/04/2022.

DENG, J. et al. **ImageNet: A Large-Scale Hierarchical Image Database**. *In CVPR.IEEE , Miami, v. 1, p. 248 - 255, 2009.*

DOS SANTOS FERREIRA, A. et al. **Weed detection in soybean crops using convnets**. *Computers and Electronics in Agriculture*, Elsevier, v. 143, p. 314-324, 2016.

EMBRAPA. **Plantas Daninhas**. Brasil, s.d. Disponível em: https://www.embrapa.br/tema-plantas-daninhas/sobre-o-tema. Acesso em 12/10/2021.

EMBRAPA. **Estudos Socioeconômicos e Ambientais.** Brasil, 2021. Disponível em https://www.embrapa.br/busca-de-noticias/-/noticia/62619259/brasil-e-o-quarto-maior-produtor-de-graos-e-o-maior-exportador-de-carne-bovina-do-mundo-diz-estudo. Acesso em 08/10/2021.

EVERINGHAM, M. et al. **The pascal visual object classes challenge: A retrospective.** *International Journal of Computer. Vision*, v. 111, no. 1, p. 98–136, 2015.

FELIPEFLOP. **Raspberry Pi 4 Model B Anatel.** Brasil, 2022. Disponível em: https://www.filipeflop.com/produto/raspberry-pi-4-model-b/. Acesso em 22/04/2022.

FAO. **The future of food and agriculture – Trends and challenges. Roma, 2017**. Roma, 2017. Disponível em: https://www.fao.org/3/i6583e/i6583e.pdf. Acesso em 15/07/2021.

FAO. **Agricultural production statistics. 2000–2020. FAOSTAT Analytical Brief Series No. 41**. Roma, 2022. Disponível em: https://www.fao.org/3/cb9180en/cb9180en.pdf . Acesso em 10/05/2022.

FORAGES. **Describe the five general categories of weed control methods**. Estados Unidos, 2021. Disponível em: https://forages.oregonstate.edu/nfgc/eo/onlineforagecurriculum/instructormaterials/availabletopics/weeds/control. Acesso em 15/05/2022.

GROW. **Mechanical Weed Control**. Estados Unidos, 2021. Disponível em: https://growiwm.org/mechanical-weed-control/. Acesso em 15/05/2022.

HE, K. et al. **Deep Residual Learning for Image Recognition**. *In CVPR.IEEE*, Las Vegas, v. 1, p. 770-778, 2016.

HOWARD, A. G. et al. **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. Cite arxiv:1704.04861, 2017.

KRIZHEVSKY, A. et al. **ImageNet Classification with Deep Convolutional Neural Networks.** *Advances in Neural Information Processing Systems*, San Diego, v. 25, p. 1097-1105, 2012.

LECUN, Y. et al. **Gradient-based learning applied to document recognition.** *Proceedings of the IEEE*, v. 88, p. 2278–2324, 1998.

MARQUES JUNIOR, L. C.; ULSON, J. A. C. **Utilização De Redes Neurais Convolucionais Em Processos De Classificação De Espécies Vegetais Na Agroindústria Da Soja.** *In: INDUSCON. IEEE*, São Paulo, v. 1, p. 2226-2231, 2018.

MARQUES JUNIOR, L. C.; ULSON, J. A. C. **Real Time Weed detection using Computer vision and Deep learning.** *In: INDUSCON. IEEE*, São Paulo, v. 1, p. 1131-1137, 2021.

MEDIUM. **Hardware for Deep Learning**. Part 3: GPU. Estados Unidos, 2018. Disponível em: <https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664#ea08> Acesso em 22/04/2022.

MILIOTO, A. et al. **Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs**. *In: ICRA. IEEE*, Áustria, v. 1, p. 2229–2235, 2018.

MIT. **Machine learning, explained**. Estados Unidos, 2021. Disponível em: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>. Acesso em 21/04/2022.

MIT. **Robotic Manipulation**. Estados Unidos, 2021. Disponível em:< https://manipulation.csail.mit.edu/segmentation.html>. Acesso em 23/03/2023.

NSW DPI. **About Weed Biological Control.** Australia, 2022. Disponível em: https://www.dpi.nsw.gov.au/biosecurity/weeds/weed-control/biological-control/about-weed-biological control. Accesso em: 15/05/2022.

OLSEN, A. et al. **DeepWeeds: A Multiclass Weed Species Image Dataset for Deep Learning.** *Sci Rep* 9, 2058 (2019).

PARTEL, V. et al. **Smart sprayer for precision weed control using artificial intelligence: Comparison of deep learning frameworks**. *Association for the Advancement of Articial Intelligence*. 2019.

PAPERSPACE. **Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall**. Estados Unidos, 2020. Disponível em: https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/. Acesso em 30/04/2022.

PAPERSPACE. **Evaluating Object Detection Models Using Mean Average Precision (mAP)**. Estados Unidos, 2020. Disponível em: https://blog.paperspace.com/mean-average-precision/. Acesso em 30/04/2022.

RAMO, K. **Hands-On Java Deep Learning for Computer Vision.** Birmingham, UK: Packt Publishing, 2019.

REDMON, J. et al. **You Only Look Once: Unified, Real-Time Object Detection.** *In CVPR.IEEE*, Las Vegas, v. 1, p. 779 - 788, 2016.

REDMON, J., FARHADI, A. **YOLO9000: Better, Faster, Stronger**. *In CVPR. IEEE*, p. 6517-6525, 2017.

REDMON, J., FARHADI, A. **Yolov3: An incremental improvement**. Cite arXiv:1804.02767, 2018.

REDMON, J. **Darknet13**. Estados Unidos, 2013. Disponível em: https://pjreddie.com/darknet/. Accesso em: 18/03/2022.

REN, S.et al. **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks**. *In NIPS*, pp. 91–99, 2015.

REN, S. et al. **Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.** *In Transactions on Pattern Analysis and Machine Intelligence*. *IEEE*, v.39, p. 1137 - 1149, 2016.

SERMANET, P et al. **Overfeat: Integrated recognition, localization and detection using convolutional networks**. *International Conference on Learning Representations*, Banff, 2014.

SIMONYAN, K; ZISSERMAN, A. **Very deep convolutional networks for Large-Scale image Recognition.** *International Conference on Learning Representations*, San Diego, 2015.

SZEGEDY, C. et al. **Going Deeper with Convolutions**. *In CVPR.IEEE*, Boston, v. 1, p. 1 - 9, 2015.

SZEGEDY, C. et al. **Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning**. *In ICLR*, San Juan, 2016.

SZEGEDY, C. et al. **Rethinking the Inception Architecture for Computer Vision**. *In CVPR. IEEE***,** Las Vegas, v. 1, p.2818 - 2826, 2016.

SUH, H. K. et al. **Transfer learning for the classication of sugar beet and volunteer potato under eld conditions**. *Biosystems engineering*, Elsevier, v. 174, p. 50-65, 2018.

TEIMOURI, N. et al**. Weed Growth Stage Estimator Using Deep Convolutional Neural Networks**. *Sensors*, vol. 18, no. 5, p. 1580, 2018.

THEENGINEERINGPROJECTS. **What is a Raspberry Pi 4? Pinout, Spec, Projects & Datasheet**. Estados Unidos, 2021. Disponível em: https://www.theengineeringprojects.com/2021/03/what-is-raspberry-pi-4-pinout-specs-projects-datasheet.html . Acesso em 13/03/2023

TRONG, V. H**. Late fusion of multimodal deep neural networks for weeds classication**. *Computers and Electronics in Agriculture*, vol. 175, p. 105506, 2020.

ULTRALYTICS. **Yolov5 by Ultralytics**. Estados Unidos, 2020. Disponível em: https://github.com/ultralytics/yolov5 . Acesso em 15/10/2021.

USDA. **Soybeam Explorer**. Disponível em: https://ipad.fas.usda.gov/cropexplorer/cropview/commodityView.aspx?cropid=2222000. Acesso em 15/04/2023.

USDA. **Corn Explorer**. Disponível em: https://ipad.fas.usda.gov/cropexplorer/cropview/commodityView.aspx?cropid=0440000 . Acesso em 15/04/2023.

W, S. MCCULLOCH.; Pitts, W. **A logical calculus of the ideas immanent in nervous activity**. *The bulletin of mathematical biophysics*, v. 5, p.115–133, 1943.

WANG, K. et al. Panet: **Few shot image semantic segmentation with prototype alignment**. *In ICCV.IEEE*, Seoul, Coreia, p. 9197– 9206, 2019.

WANG, C.Y. et al. CSPNet: **A New Backbone that can Enhance Learning Capability of CNN**. *In CVPR. IEEE*, Washington, DC, p. 1571–1580, 2020.

WEISS, K. et al. A survey of transfer learning. *Journal of Big Data*, vol. 3, p. 9, 2016.

XU, R. et al. **A Forest Fire Detection System Based on Ensemble Learning**. *Forests*, v. 12, no. 2, p. 217, 2021.

ZOPH, B. et al. **Learning Transferable Architectures for Scalable Image Recognition.** Cite arxiv:1707.07012, 2017.