UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATHEUS FERNANDES KOVALESKI

# Segmentação de plantações de soja para detecção de ervas daninha em dispositivos embarcados

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Profª. Dr. Cláudio Fernando Resin Geyer
Co-orientador: Prof. Dr. Julio Cesar Santos dos Anjos

Porto Alegre
2023

*“Há mais mistérios entre o céu e a terra*
*do que a vã filosofia dos homens possa imaginar.”*

— SHAKESPEARE, WILLIAM

## AGRADECIMENTOS

Em primeiro lugar, gostaria de expressar minha profunda gratidão aos meus familiares, que foram fonte de inspiração e apoio inabalável ao longo de todo o processo. Eles desempenharam um papel fundamental no meu caminho até aqui, e sou imensamente grato por sua presença constante e incentivo.

Em segundo lugar, desejo estender meus agradecimentos ao meu orientador e coorientador, o Professor Dr. Claudio Fernando Resin Geyer e ao Professor Dr.Julio Cesar Santos dos Anjos. Suas orientações, contínuas e valiosas contribuições, foram essenciais para o desenvolvimento deste trabalho e para o meu crescimento acadêmico. Estou imensamente grato por todo o tempo e dedicação que eles dedicaram a me orientar, além das oportunidades valiosas que ambos me proporcionaram ao longo desse período.

Também é importante mencionar meu amigo e antigo chefe, Ismael Scheeren. Sua generosidade ao permitir que eu desenvolvesse meu TCC em sua empresa foi um privilégio. Além disso, seu conhecimento e orientação foram inestimáveis durante todo o processo. Aprendi muito com sua experiência e orientação.

Por último, mas não menos importante, quero expressar um agradecimento especial a todos os amigos que fiz ao longo desta jornada da minha vida acadêmica. Suas amizades e apoio tornaram essa trajetória mais enriquecedora e significativa. Sou grato por compartilharmos essa experiência e por todos os momentos memoráveis que passamos juntos.

## RESUMO

Uma das etapas mais importantes para o plantio de algum alimento é a sua fase de desenvolvimento. Nessa etapa é preciso garantir que as plantas estejam saudáveis para que se possa extrair o máximo proveito do grão usado no plantio. Durante essa etapa é feito o uso de agrotóxicos/inseticidas para evitar que alguma praga atinja a plantação. Essa aplicação é feita em toda a área de plantio, já que a aplicação localizada do produto poderia acabar levando dias ou semanas, dado o fato que precisaria ser feito manualmente e mesmo assim poderia não ser bem aplicado. Um grande problema que surge com esse método de aplicação de agrotóxico, é que o uso desse tipo de produto em plantas pode causar intoxicação ou gerar certas alergias, além de ser muito custoso para os fazendeiros. Dado isso, o objetivo do trabalho foi o desenvolvimento de um modelo de Machine Learning, capaz de reconhecer e segmentar em N classes distintas uma imagem capturada em tempo real; o modelo em questão escolhido foi o pix2pix, um modelo baseado em GAN capaz de produzir segmentações com alta acurácia. Foram criados dois modelos para os experimentos, o primeiro seria a pix2pix original e o segundo seria uma versão modificada focada em aumentar a performance em termos de tempo de processamento. Os experimentos foram executados usando uma arquitetura ARM da família Jetson da Nvidia conhecida como nano, fornecida pela empresa Accore. Ao fim, apresentamos os resultados coletados dos experimentos em termos de acurácia e FPS para estimar a performance do nosso modelo. O presente trabalho obteve resultados de mais de 60% de acurácia, tendo em vista as limitações de *dataset*, ao passo que conseguimos ter uma taxa de FPS similar entre abordagens mais simples de detecção de instâncias.



**Palavras-chave:** Visão Computacional. Segmentação de imagens. Sistemas Embarcados. Rede Generativa Adversaria. Aprendizado de Máquina.

**Soya plantation segmentation to weed detection on embedded devices**


## ABSTRACT

One of the most important stages in planting any crop is its development phase. During this stage, it is crucial to ensure that the plants are healthy in order to extract the maximum benefit from the seeds used for planting. Agrochemicals/insecticides are used during this stage to prevent any pests from affecting the crop. This application is carried out across the entire planting area, as localized application of the product could take days or weeks, given that it would need to be done manually and might still not be applied effectively. One major problem that arises with this method of pesticide application in plants is that it can lead to poisoning or trigger certain allergies, in addition to being very costly for farmers. Given this, the objective of the study was to develop a Machine Learning model capable of recognizing and segmenting an image captured in real-time into N distinct classes. The chosen model for this purpose was pix2pix, a GAN-based model capable of producing highly accurate segmentations. Two models were created for the experiments, the first being the original pix2pix, and the second being a modified version focused on improving processing time performance. The experiments were conducted using an ARM architecture from Nvidia's Jetson family known as the Nano, provided by the Accore company. In the end, we presented the results collected from the experiments in terms of accuracy and FPS to estimate the performance of our model. The present work achieved results of over 60% accuracy, considering the limitations of the dataset, while also maintaining a similar FPS rate compared to simpler instance detection approaches.



**Keywords:** Computer Vision, Image Segmentation, Embedded systems, Generative Adversarial Networks, Machine Learning.

## LISTA DE FIGURAS

## LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| GAN | Generative Adversarial network |
| ML | Machine Learning |
| AI | Inteligencia Artificial ou *Artificial inteligence* |
| NN | *Artificial Neural Network* |
| MLP | *Multilayer Perceptron* |
| DNN | *Deep Neural Network* |
| CNN | *Convolutional Neural Network* |
| CL | *Convolutional Layer* |
| FCNN | *Fully Convolutional Neural Network* |

**SUMÁRIO**

# 1 INTRODUÇÃO

Uma das coisas mais importantes para a agricultura é o cuidado da plantação ao decorrer do tempo, para evitar perdas de grande proporção. Tais perdas, em sua maioria, vem da ocorrência de pragas nas plantações, tais como, algumas espécies de insetos, por exemplo: lagarta da soja, lagarta-do-cartucho e a lagarta-elasmo. Porém, quando falamos de pragas também precisamos considerar as daninhas, pragas que sugam os nutrientes das plantas e acabam ou reduzindo a produção eficiente da planta, ou simplesmente a matando.

Para tentar combater esse tipo de problema, se faz uso de produtos químicos como agrotóxicos e inseticidas. A aplicação desse tipo de produto pode ser feita de duas maneiras. A primeira maneira de se fazer isso é de forma manual, onde um aplicador coloca um equipamento, conhecido como Pulverizador, onde o mesmo tem a função de pulverizar o produto na plantação.

O problema com esse tipo de aplicação é que ela é demorada, podendo levar algumas horas para que a aplicação seja completada em toda a plantação. Outra maneira que se tem para fazer isso é a utilização de máquinas: como tratores, aviões e drones, para a aplicação de inseticidas. Diferente da primeira modalidade, onde a aplicação é feita por uma pessoa e de forma mais lenta, essa maneira possibilita a aplicação de forma mais rápida e muito mais paralelizada. Dessa forma, ocorre a eliminação da necessidade de se ter uma pessoa a todo momento próxima dos produtos, que em muitas vezes, podem causar problemas de saúde durante a aplicação. O maior problema que existe nesse caso é o gasto excessivo do produto na aplicação, tornando o processo mais caro.

Outro grande problema associado ao uso em larga escala de produtos químicos são os efeitos no organismo humano, como problemas de saúde, esses problemas de saúde podem ser separados em duas categorias, que seriam o tipo de efeito no organismo. Temos os efeitos agudos, que são aqueles que têm rápida reação, podendo causar problemas através do contato pela boca, respiração e pele e de forma crônica, podendo provocar abortos, impotência, além de poder, ter uma relação com o aparecimento de câncer.

Para resolver esses problemas, propomos uma solução envolvendo o treinamento de uma rede neural utilizando técnicas de visão computacional para fazer o reconhecimento de regiões onde existe infestação de ervas daninha. Isso possibilitará fazer a aplicação dos produtos de forma mais localizada, evitando o uso em demasia desses produtos prejudiciais à saúde humana e reduzindo os custos de produção alimentícia, tornando a

produção desses alimentos mais eficiente.

O modelo proposto para esse trabalho consiste na utilização de técnicas de *machine learning* e *deep learning* para segmentação de áreas de plantações em 5 classes, sendo elas caruru, folha estreita, folha larga, soja e nada; onde nada seria o solo onde a soja não foi plantada.

A abordagem escolhida para a criação desse modelo foi a pix2pix, proposta por (ISOLA et al., 2017). Essa abordagem faz a utilização de uma família de técnicas de treinamento conhecida como GAN, que seriam técnicas de treinamento onde se visa a utilização de uma rede adversária para melhorar a acurácia do modelo. Para ser mais preciso, a versão da GAN usada para esse trabalho consiste no uso de um modelo baseado em cGan (Conditional Generative Adversarial Network).

A cGan é uma técnica de GAN que consiste em se criar uma rede que tenta mapear as condições a serem identificadas. Um bom exemplo desse funcionamento são as redes cyclegan e pix2pix feita por (ISOLA et al., 2017), onde a rede identifica um estilo visual, como, por exemplo, um clima, como inverno ou verão e tenta fazer a mudança de estilo só mudando as características da imagem que condicionam a imagem a ter essa percepção clima. O funcionamento geral da estratégia GAN consiste em um gerador que seria a parte que produz os resultados esperados, ou seja, o resultado predito pela rede, e um discriminador que irá dizer o quão bom os resultados preditos estão quando comparados com os resultados esperados.

O modelo recebe uma imagem original, imagem essa que seria a capturada por uma câmera normal, e uma imagem segmentada utilizando uma técnica de superpixel conhecida como SLIC(JAIN, 2019), que divide a imagem em MxN segmentos, onde M seria a quantidade de segmentos por coluna e N a quantidade de segmentos por linha, e tentar convergir as bordas dos segmentos para melhor ajustar ao objeto dentro do segmento em questão. A imagem segmentada então é enviada para o modelo sendo usada como um *Ground Truth*.

Como mencionado antes, a rede GAN funciona com a utilização de duas arquiteturas para tentar aumentar a acurácia do modelo. Os modelos escolhidos para a implementação foram a rede unet-256 (GARDNER; DORLING, 1998), uma versão modificada da rede unet original, e a patchgan um discriminador para redes GAN que penaliza apenas estruturas em escala da imagem local. Daremos uma melhor explicação sobre o protótipo nos Capítulos seguintes.

## 1.1 Objetivos

Os objetivos deste trabalho consistem em desenvolver uma técnica de segmentação para ser utilizada em plantações de soja, com o intuito de permitir a execução em tempo real. Além disso, pretende-se explorar um espaço de pesquisa ainda não abordado no estudo de (ISOLA et al., 2017), a fim de avaliar o desempenho da sua arquitetura em cenários relacionados à agricultura. Essa análise será realizada em comparação com outros modelos atualmente empregados para a detecção e dispersão de inseticidas.

## 1.2 Organização do trabalho

A seguir descreveremos a estrutura adotada dos conteúdos deste trabalho de conclusão de curso.

- Capítulo 2: introdução dos algoritmos usados e detalhes técnicos necessários para justificar as escolhas do projeto desenvolvido;
- Capítulo 3: revisão do estado da arte, analisando de forma comparativa os trabalhos escolhidos como referência e o trabalho desenvolvido;
- Capítulo 4: detalhamento da metodologia usada no trabalho desenvolvido, tais como: funcionamento dos algoritmos implementados, uma melhor explicação das redes escolhidas;
- Capítulo 5: neste capítulo serão detalhados os itens necessários para implementar a metodologia empregada, apresentarei a validação das análises e discussão dos resultados obtidos, bem como apresenta hipóteses para explicar tais resultados;
- Capítulo 6: este capítulo é composto pelo resumo do trabalho e dos resultados obtidos, conclusões e elenca possíveis trabalhos futuros.

# 2 BACKGROUND

Neste capítulo, descreveremos as tecnologias usadas para a realização do trabalho, assim como dar um bom entendimento do porquê as mesmas foram adotadas para a resolução do trabalho proposto.

## 2.1 Ervas daninha

Ervas daninhas, também conhecidas como infestantes, são plantas que crescem indesejadamente em áreas controladas pelo homem, como campos de cultivo, hortas e jardins. Elas têm a capacidade de se reproduzir abundantemente e competir por recursos, prejudicando o rendimento das culturas. Existem dois principais tipos de ervas daninhas: monocotiledôneas (com folhas estreitas) e dicotiledôneas (com folhas largas). Essas plantas podem ter efeitos negativos, como dificultar a irrigação e a colheita, criar um ambiente propício para pragas e doenças, ou mesmo serem venenosas para o gado. No entanto, algumas ervas daninhas têm benefícios, como proteger o solo da erosão, melhorar a biodiversidade e regenerar ambientes urbanos. Algumas delas também são comestíveis ou têm propriedades medicinais. Para controlar as ervas daninhas, existem métodos preventivos, agronômicos, mecânicos, biológicos e químicos, embora o uso inadequado de herbicidas possa ser prejudicial ao meio ambiente e à saúde humana(IBERDROLA, 2023).

## 2.2 Processamento de imagem

O processamento de imagem refere-se à aplicação de técnicas e algoritmos para a análise e manipulação de dados multidimensionais adquiridos por diversos sensores, onde tanto a entrada quanto a saída do processo consistem em imagens. Essas técnicas têm várias aplicações em diferentes áreas, incluindo análise de recursos naturais, meteorologia, transmissão digital de sinais de televisão, análise biomédica, metalografia, obtenção de imagens médicas por ultrassom, radiação nuclear ou tomografia computadorizada, bem como automação industrial usando sensores visuais em robótica. O processamento de imagem é utilizado para melhorar a qualidade visual de características estruturais em imagens, facilitando a interpretação humana e gerando produtos que podem ser submetidos a análises adicionais. Isso inclui a análise de imagens de satélites para pesquisa de

recursos naturais, onde a qualidade da representação dos dados nas imagens desempenha um papel fundamental na extração de informações espectrais e na identificação de alvos de interesse. Além disso, o processamento de imagem permite a integração de diferentes tipos de dados georreferenciados, possibilitando uma análise abrangente de cenas em várias partes do espectro eletromagnético (GONZALEZ; WOODS, 2000).

### 2.2.1 Recorte

Em processamento de imagens, recorte ou *crop* é uma operação na qual uma parte específica de uma imagem maior é selecionada e extraída, enquanto o restante da imagem é descartado. Essa operação é frequentemente usada para remover áreas indesejadas de uma imagem ou para focar em uma região de interesse (ROI - *Region of Interest*) dentro da imagem original (GONZALEZ; WOODS, 2000).

Ao realizar o recorte de uma imagem, você está essencialmente definindo novos limites ou dimensões para a imagem resultante, que será composta apenas pela parte selecionada. O objetivo é isolar e preservar a área de interesse, descartando o restante.

O processo de recorte envolve a especificação das coordenadas ou dimensões da região a ser mantida na imagem, seja por meio de coordenadas de píxel específicas ou definindo uma região retangular. O resultado é uma imagem menor que contém apenas a parte recortada da imagem original.

O *crop* é uma operação comum em várias aplicações, incluindo edição de fotos, processamento de imagens médicas, visão computacional e análise de imagens em geral, quando é necessário focar em uma parte específica da imagem para análise ou apresentação.

### 2.2.2 Redimensionamento

Redimensionamento ou *resize* é uma operação que envolve a modificação do tamanho de uma imagem. Essa operação pode ser usada para aumentar ou diminuir as dimensões da imagem, ajustando sua largura e altura conforme as necessidades específicas. O redimensionamento de uma imagem é uma técnica comum em processamento de imagens e é realizada por meio de algoritmos que recalculam os valores de cor dos pixels da imagem original para criar uma nova imagem com as dimensões desejadas (GONZA-

LEZ; WOODS, 2000).

Quando uma imagem é redimensionada para um tamanho menor, isso é geralmente chamado de "redução"ou "diminuição", e quando é aumentada para um tamanho maior, é chamado de "ampliação". As técnicas de redimensionamento podem variar, e diferentes algoritmos podem ser usados para preservar a qualidade da imagem, minimizar perdas ou garantir um redimensionamento rápido.

A operação de *resize* é frequentemente utilizada em várias aplicações, como processamento de imagens digitais, edição de fotos, criação de miniaturas, ajuste de resolução para exibição em diferentes dispositivos, entre outros. É uma ferramenta importante para adaptar imagens a diferentes contextos e requisitos de exibição.

## 2.3 Visão computacional

A visão computacional é um campo multidisciplinar que se dedica à interpretação e análise de imagens e vídeos por meio de algoritmos e técnicas computacionais. Nesse contexto, a visão computacional busca replicar a habilidade humana de compreender e extrair informações a partir de dados visuais(figura 2.1). Ela envolve a aplicação de modelos físicos, probabilísticos e algoritmos eficientes para resolver problemas como reconhecimento de objetos, detecção de padrões, segmentação de imagens e reconstrução tridimensional a partir de imagens bidimensionais. É um campo com diversas aplicações práticas, abrangendo desde reconhecimento de caracteres até diagnóstico médico, automação industrial e veículos autônomos(SZELISKI, 2022).

## 2.4 Inteligencia artificial

Segundo (RUSSELL, 2010) Inteligencia Artificial ou *Artificial inteligence* (AI) pode ser definida por, mas não só, como a capacidade de construir entidades inteligentes. O campo de AI pode ser dividido em 4 categorias básicas: sistemas que pensam como humanos, sistemas que pensam de forma racional, sistemas que agem como humanos e sistemas que agem de forma racional. Tendo essas categorias e levando em consideração como (RUSSELL, 2010) descreve o que seria AI, podemos dizer que o campo de AI nada mais é que o campo onde focamos no desenvolvimento de sistemas inteligentes capazes de reproduzir certos comportamentos humano.

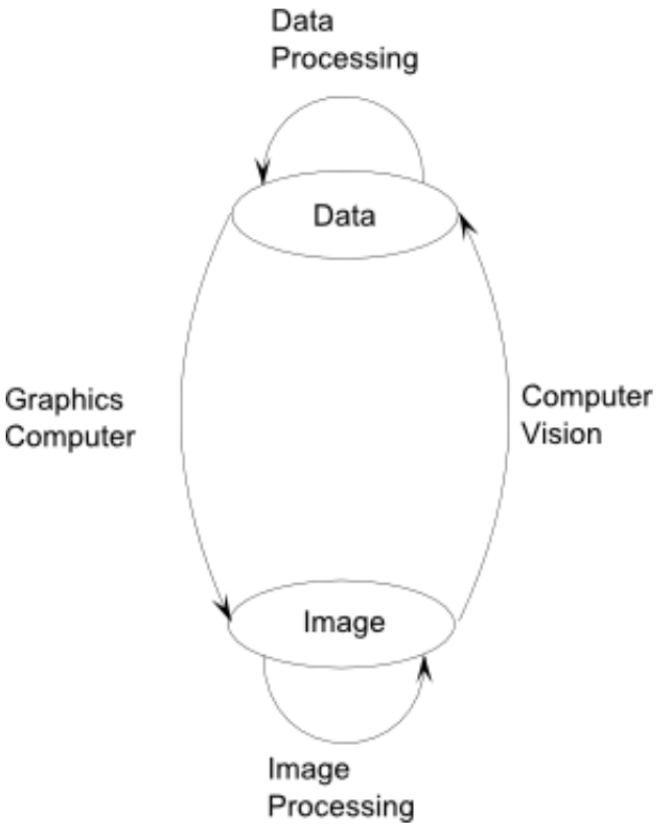



Figura 2.1 – Relação entre os diferentes tipos de tratamento de informação
Fonte: (TECNOLOGIA, 2012/03/22)

## 2.5 Aprendizado de Máquina

Quando damos uma olhada dentro da área de AI, chegamos a conclusão que podemos classificar-lá em algumas categorias, uma delas sendo a de Aprendizado de Máquina ou *Machine Learning* (ML). O objetivo da área de *Machine learning* seria tentar recriar o jeito que o ser humano aprende o reconhecimento de padrões para assim criar um modelo capaz de generalizar suas tarefas(BISHOP; NASRABADI, 2006). Desta forma, de forma bem geral, podemos dizer que ML é que a área que tente aplicar algoritmos de AI a um conjunto de dados na forma de encontrar padrões.

Podemos dividir a área de ML em algumas categorias: aprendizado supervisionado, aprendizado não-supervisionado, aprendizado semi-supervisionado e aprendizado por reforço(BISHOP; NASRABADI, 2006; RUSSELL, 2010; ENGELEN; HOOS, 2020). Cada uma destas categorias é melhor descrita nas Subseções a seguir.

### 2.5.1 Aprendizado Supervisionado

Aprendizado supervisionado é feito a partir de um conjunto de dados contendo características, que mapeia para um dado conjunto de rótulos (GOODFELLOW; BENGIO; COURVILLE, 2016).

### 2.5.2 Aprendizado não-supervisionado

Aprendizado não-supervisionado é onde o aprendizado é deixado todo a cargo dos próprios algoritmos, onde os mesmos vão tentar achar padrões entre os dados e separá-los entre si totalmente automática (RUSSELL, 2010).

### 2.5.3 Aprendizado semi-supervisionado

Nessa categoria, o algoritmo possui uma "mistura"dos dois paradigmas, tanto supervisionado quanto não-supervisionado. Com base na literatura. reconhece que o custo de rotular dados é elevado. Assim, o algoritmo roda primeiramente aprendendo o aquilo que possui rotulo, ao mesmo tempo que tenta identificar clusters nos dados que não possuem um rótulo em si (ENGELEN; HOOS, 2020).

### 2.5.4 Aprendizado por reforço

No aprendizado por reforço o algoritmo se comporta de forma exploratória, no caso o algoritmo tente a executar ações inicialmente de forma aleatória a fim de ganhar "recompensas"ou "punições"pelas ações feitas. Ao fim do algoritmo o aprendizado acorre pelo próprio algoritmo através das próprias experiências (RUSSELL, 2010).

## 2.6 Overfitting e Underfitting

Para termos noção de como o processo de treinamento está evoluindo, precisamos ter em mente que existe a necessidade de uma maneira de se medir essa evolução da rede, para isso utilizamos métricas que medem que consigam avaliar desempenho do

modelo, que nos fornecem valores objetivos para podermos saber como esse treinamento está sendo feito. Esse processo pode nos resultar em uma rede que tenha 2 comportamentos: o primeiro comportamento é quando a rede acaba durante o treinamento se saindo muito bem, mas quando avaliamos o modelo com o conjunto de teste o mesmo tem péssimo resultado, esse comportamento é chamado de *Overfitting*. Porem, é possível que o modelo, já durante o treinamento, apresente péssimos resultados, não conseguindo aprender a reconhecer os padrões dos dados. Nesse caso o nome dado é *underfitting* (BISHOP; NASRABADI, 2006).

Normalmente, quando estamos fazendo o treinamento da rede, estamos buscando um modelo que consiga generalizar bem para o domínio dos dados, nesse caso não teremos uma situação de *Overfitting* uma vez que não teremos *underfitting* já que o modelo já apresenta bons resultados.

## 2.7 Redes Neurais Artificiais

Umas das estratégias mais usadas atualmente para o treinamento de modelos é a de redes neurais artificiais. Uma rede neural artificial, também conhecido como *artificial neural network* (ANNs) ou *neural network* (NNs), são uma ramificação dos modelos de aprendizado de máquina que utilizam princípios da organização neural encontrados em redes neurais biológicas. No caso, uma NN é composta por um número N de neurônios, onde cada neurônio tenta simular o comportamento de um neurônio humano, com isso a rede tenta aprender a partir do reconhecimento de padrões existentes no *dataset*.

Como descrito acima, uma NN é composta por neurônios, onde esses neurônios podem ser representados pela Figura 2.4.

Onde temos que:

X0: seria o bias da rede, no caso o quanto a rede vai dar preferencia por mudança nos pesos dos neurônios;

$X_n$: seria os valores de input da rede, no caso os sensores usados para entender o que a rede esta recebendo;

$W_n$ : seria o valores associados a cada neurônio, no caso seria o peso dos neurônios para que assim a rede consiga entender o como gerar a saída esperada;

+: seria onde a somas dos pesos dado por $W_n$ é feita para que assim possa ser criado um valor para definir se aquele neurônio será ativado ou não.

*sign*: onde é feita a decisão de ativação do neurônio, no caso é nessa etapa onde é

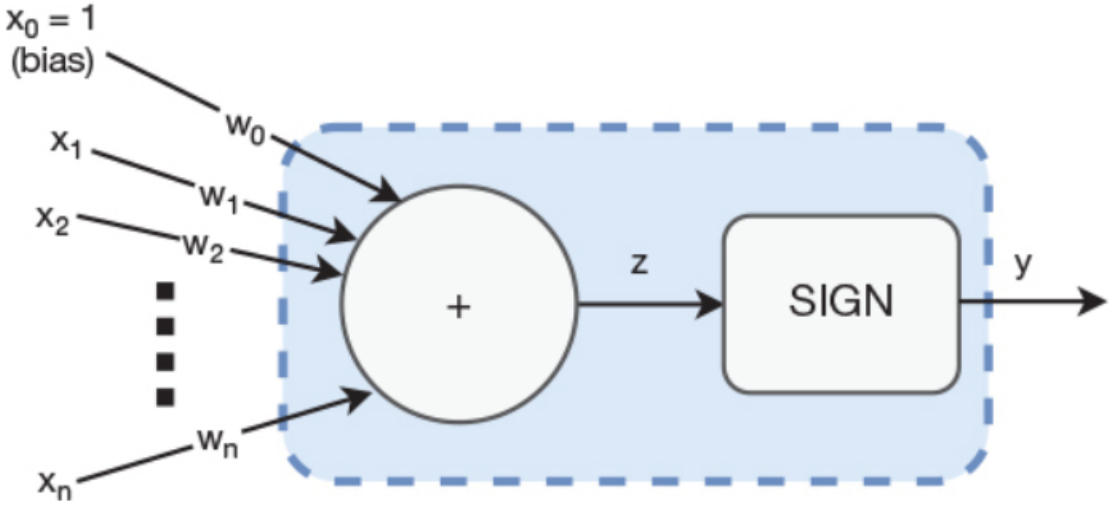



Figura 2.2 – Neurônio Artificial
Fonte: (EKMAN, 2021)

implementada a função de ativação, que servira para a criação de um limiar para a ativação do neurônio.

y: seria a saída da rede, que seria o valor associado ao neurônio ao fim do cálculo.

Um neurônio artificial pode ser formulado matematicamente da seguinte forma:

$$y = f(z)$$

$$z = \sum_{i=0}^{n} W_i X_i$$

$$f(z) = \begin{cases} \text{-1 : z < 0} \\ \text{1 : z} \geq 0 \end{cases}$$

$$X_0 = 1$$

### 2.7.1 Função de Ativação

Como descrito anteriormente, em um neurônio temos o que chamamos de Função de ativação. As funções de ativação não lineares ajudam a resolver problemas relacionados ao desaparecimento do gradiente. Elas também introduzem não-linearidade, potencializando a capacidade da rede neural de representar funções complexas. Essa não-linearidade evita que a rede se reduza a um mero modelo linear e permite que ela represente, de forma mais fiel, as características complexas e não lineares das interações

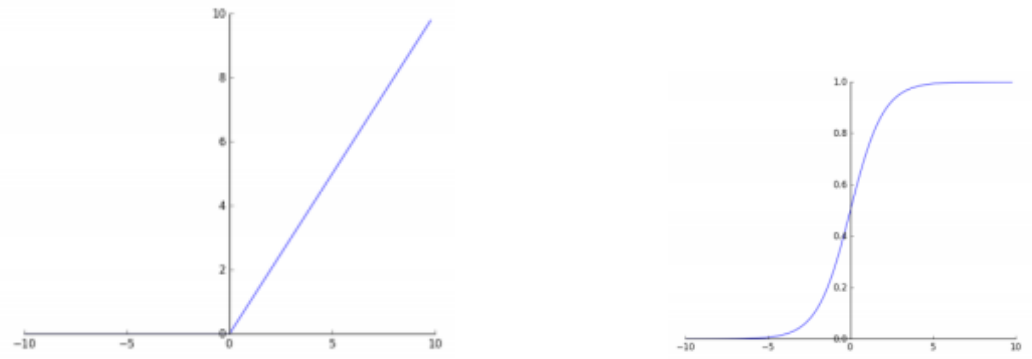
do mundo real.

Existem duas funções comumente usadas quando tratamos de funções de ativação: sigmoide e ReLU

No caso, a função *sigmoid* (Figura 2.3b) é muito utilizada quando queremos resolver um problema de classificação onde a saída de cada neurônio fica retido entre um intervalo de 0 a 1, onde o resultado seria a probabilidade da instância fornecida estar contido entre as classes analisadas.

A função ReLU (Figura 2.3a) tem como característica a utilização de valores não lineares. Por causa disso, normalmente essa função é utilizada como uma forma otimizar a rede, visto que seria uma função muito mais eficiente. Seu uso normalmente se dá em camadas ocultas da rede pois normalmente possuem uma quantidade maior de neurônios.

Figura 2.3 – Métodos de operação do Doc2Vec




(a) ReLU

(b) Sigmoide

### 2.7.2 Rede Neural Multicamadas

Quando estamos rodando um processo de treinamento, dificilmente conseguiremos montar uma rede que produza bons resultados com a penas uma camada de neurônios, no caso chamados de Perceptron. Para conseguirmos resolver esse problema, fazemos o uso de camadas ocultas, no caso chamada de Rede neural multicamadas, ou *Multilayer Perceptron*(MLP) (GOODFELLOW; BENGIO; COURVILLE, 2016).

Uma *MultiLayer Perceptron* consiste em uma rede de neurônios, onde a saída de um neurônio da camada N é a entrada de um neurônio na camada N+1. Uma rede neural pode ter mais de uma camada oculta, nesse caso essas camadas são usadas para dar um maior poder preditivo ao modelo, esse tipo de modelo é chamado de rede neural profunda,

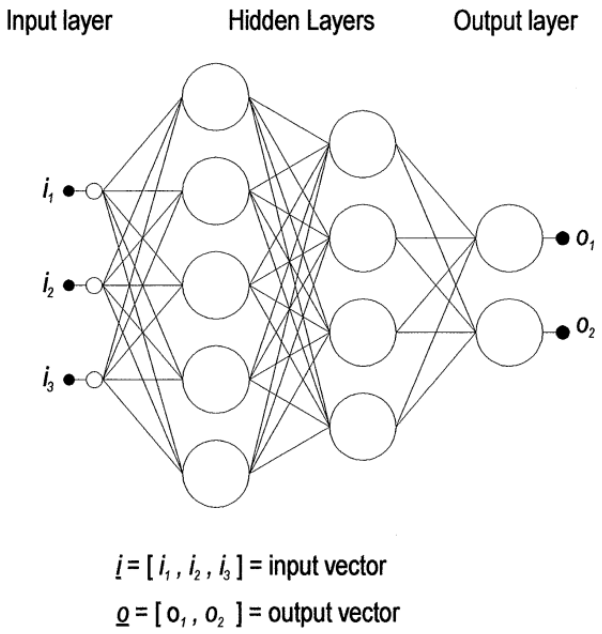



Figura 2.4 – Neurôonio Artificial
Fonte: (GARDNER; DORLING, 1998)

ou *deep neural network* (GOODFELLOW; BENGIO; COURVILLE, 2016).

## 2.8 Aprendizado Profundo

Redes neurais profundas ou aprendizado neural profundo, ou *Deep learning* , é uma subárea de aprendizado de máquina e AI que se concentra no uso de redes neurais com muitas camadas, chamadas de redes neurais profundas, para modelar e resolver tarefas complexas. Algoritmos de aprendizado profundo visam aprender automaticamente representações hierárquicas de características a partir de grandes quantidades de dados. Esses algoritmos têm mostrado um sucesso notável em várias aplicações de AI, incluindo visão computacional, processamento de linguagem natural e reconhecimento de fala.(GOODFELLOW; BENGIO; COURVILLE, 2016)

## 2.9 Redes Neurais Convolucionais

A Rede Neural Convolucional, ou *Convolutional Neural Network* (CNN), é uma arquitetura especializada de redes neurais profundas projetada para processamento de dados com uma estrutura em forma de grade, por exemplo, matrizes. As camadas totalmente conectadas conectam cada neurônio de uma camada a cada neurônio de outra camada. A

camada de convolução apresenta um filtro (ou kernel) que desliza sobre os dados e realiza operações é uma operação matricial. Operações de polling, envolvem seleção de valores de uma região, criando um mapa de características (GOODFELLOW; BENGIO; COURVILLE, 2016; EKMAN, 2021).

Uma rede neural profunda, por si só, não é inerentemente tolerante a variações de rotação e escala. No Capítulo 9, os autores (GOODFELLOW; BENGIO; COURVILLE, 2016; EKMAN, 2021) afirmam: "A convolução não é naturalmente equivariante a algumas outras transformações, como mudanças na escala ou rotação de uma imagem. São necessários outros mecanismos para lidar com esses tipos de transformações. É a camada de max polling que introduz essas invariancias."

Uma CNN, é composta por várias camadas que transformam uma entrada utilizando filtros de convolução de pequena extensão. Sua arquitetura consiste em três principais tipos de camadas: camadas de convolução, camadas de *pooling* e camadas totalmente conectadas. Cada uma dessas camadas desempenha um papel específico no processo de extração e aprendizado de características das imagens.(GOODFELLOW; BENGIO; COURVILLE, 2016)

Essa abordagem de redes neurais convolucionais tem sido bastante utilizadas em tarefas de visão computacional, reconhecimento de padrões e processamento de imagens, devido à sua capacidade de capturar informações contextuais e hierárquicas em dados visuais complexos.

### 2.9.1 Camada de Convolução

A Camada de Convolução, consiste em um conjunto de filtros não lineares, também chamados de *kernels*, de baixa dimensionalidade. Esses *kernels* percorrem sequencialmente toda a profundidade dos dados de entrada, aplicando operações de convolução e produzindo matrizes conhecidas como mapas de características ou *feature maps*. A combinação desses *feature maps*, gerados pelos diferentes filtros, representa a saída da CL (O'SHEA; NASH, 2015).

Durante o processo de treinamento da rede neural, nas camadas mais superficiais, esses filtros são ajustados para serem ativados na presença de características e padrões visuais mais simples, como bordas ou aglomerados de *píixeils* de valores semelhantes. À medida que avançamos nas camadas mais profundas, esses filtros contêm informações sobre padrões mais complexos, como a capacidade de distinguir partes específicas de

objetos.

A representação matemática da camada de convolução envolve a aplicação de uma operação de convolução discreta em um *array* multidimensional. Isso é especialmente relevante quando lidamos com imagens bidimensionais, conforme descrito na Equação 2.1 (GOODFELLOW; BENGIO; COURVILLE, 2016).

$$S[i,j] = (I * K)[i,j] = \sum_{m=1}^{M} \sum_{n=1}^{N} I[i-m, j-n]K[m,n] \tag{2.1}$$

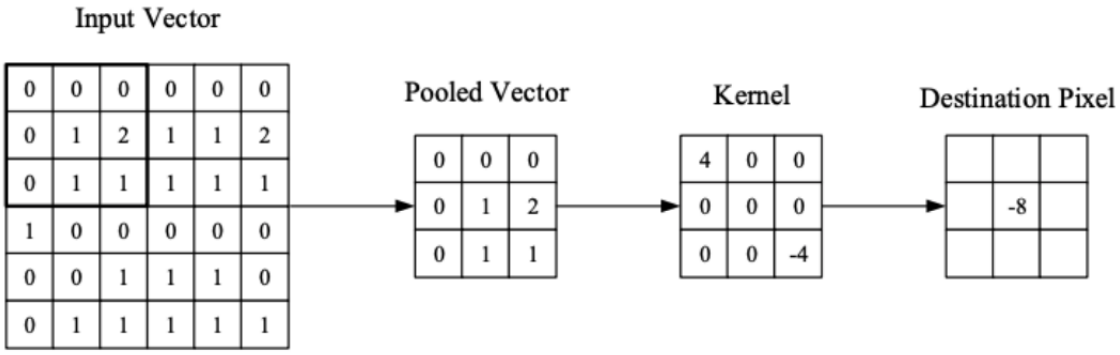



Figura 2.5 – Camada de convolução
Fonte: (GARDNER; DORLING, 1998)

A Figura 2.5 fornece uma representação visual da camada de convolução, ajudando a ilustrar como os filtros percorrem os dados de entrada e geram os *feature maps* que capturam informações relevantes para o aprendizado da rede neural.

#### *2.9.1.1 Stride*

Conforme explicado, os filtros de convolução percorrem sequencialmente toda a matriz de entrada, avançando píxel a píxel na imagem. O movimento ocorre em etapas ou passos, determinando o quanto o filtro se desloca a cada iteração. Quando o passo é igual a 1, a altura e a largura da camada de saída são mantidas iguais às da camada de entrada. Por outro lado, quando o passo é maior que 1, reduz-se a quantidade de sobreposições entre as regiões cobertas pelo filtro, o que, por sua vez, resulta em uma camada de saída com dimensões espaciais menores (O'SHEA; NASH, 2015).

Nesse contexto, o tamanho do passo ou *stride* desempenha um papel fundamental na determinação das dimensões das camadas de saída nas operações de convolução, afetando a resolução espacial das características extraídas.

#### *2.9.1.2 Preenchimento*

Durante o deslizamento dos filtros, é crucial abordar o tratamento das bordas da imagem. Para esse fim, emprega-se a técnica conhecida como *same padding* ou *zero padding*, na qual as margens da imagem são preenchidas com zeros. Isso é feito com o propósito de controlar as dimensões da camada de saída, garantindo que ela mantenha a mesma altura e largura que a camada de entrada, como ilustrado na Figura 2.5.

O tamanho do preenchimento é determinado pela Equação 2.2, na qual K representa o tamanho do filtro:

$$P = \frac{K - 1}{2} \tag{2.2}$$

É fundamental observar que ao empregar essa técnica de preenchimento, a dimensão espacial da camada de saída é ajustada conformede acordo com a fórmula da Equação 2.3. Nessa fórmula, V representa o volume de entrada, K é o tamanho do filtro, P é o valor de preenchimento, e S é o tamanho do passo do filtro (O'SHEA; NASH, 2015).

$$O = \frac{(V - K) + 2P}{S + 1} \tag{2.3}$$

Essas técnicas desempenham um papel crucial no controle das dimensões das camadas intermediárias em redes neurais convolucionais, garantindo que as informações sejam preservadas durante a convolução.

### 2.9.2 Camada de *pooling*

A camada de *pooling* tem como principal objetivo a gradual redução da dimensionalidade da representação, contribuindo para a diminuição do número de parâmetros e, consequentemente, reduzindo a complexidade computacional do modelo. Essa redução de dimensionalidade desempenha um papel crucial na prevenção do *overfitting*, como destacado por (O'SHEA; NASH, 2015).

O método mais comum de *pooling* é o *Max Pooling*, que consiste em diminuir as dimensões das camadas ao selecionar o valor máximo em cada região com base em um filtro de tamanho N×N. Essa abordagem permite eliminar informações insignificantes, ao mesmo tempo, em que cria uma invariância a pequenas mudanças e distorções locais, como mencionado por (GOODFELLOW; BENGIO; COURVILLE, 2016).

Assim, a camada de *pooling* desempenha um papel fundamental na simplificação das representações, tornando-as mais compactas e robustas, ao mesmo tempo, em que contribui para a eficácia geral do modelo.

### 2.9.3 Camada Totalmente Conectada

As Camadas Totalmente Conectadas, também conhecidas como *Fully Connected Layers* (FC), são camadas que seguem uma arquitetura de rede neural tradicional do tipo MLP *Multi-Layer Perceptron*, como exemplificado na Figura 2.4. Nessas camadas, todos os neurônios estão interconectados diretamente com as camadas adjacentes.

Essa camada desempenha um papel crucial no processo de classificação, uma vez que as características previamente extraídas nas camadas convolucionais da rede são utilizadas aqui para tomar decisões finais de classificação. É nessa etapa que as características aprendidas são combinadas e ponderadas para produzir as saídas da rede, resultando na classificação desejada.

Em resumo, as camadas totalmente conectadas representam o estágio final da rede neural, onde as informações processadas nas camadas convolucionais são empregadas para a tarefa final de classificação.

## 2.10 Redes Adversarias Generativas

Redes Generativas Adversarias (GANs), são usadas para modelar distribuições de dados de alta dimensão e funcionam através da competição entre dois componentes principais: o gerador e o discriminador. O gerador cria imagens falsas visando torná-las o mais realistas possível, enquanto o discriminador tenta distinguir essas imagens falsas das imagens reais. Ambos os componentes são treinados simultaneamente, com o gerador aprendendo apenas por meio de sua interação com o discriminador(CRESWELL et al., 2018).

O treinamento ocorre com base no erro do discriminador, usado para aprimorar o gerador, tornando-o capaz de produzir imagens falsas de qualidade superior. Essas redes são implementadas usando camadas convolucionais e/ou totalmente conectadas, e seus benefícios vão além da geração de imagens, já que as representações aprendidas podem ser aplicadas em diversas tarefas subsequentes.

Em resumo, GANs representam uma área promissora de pesquisa, com amplas aplicações em campos que vão além da geração de imagens, abrindo possibilidades interessantes para o desenvolvimento de representações úteis em outras áreas.

## 2.11 Segmentação Semântica

As redes convolucionais são comumente empregadas em tarefas de classificação, nas quais o resultado é um único rótulo de classe associado a uma imagem. No entanto, em muitos contextos visuais, especialmente no processamento de imagens biomédicas, a saída desejada precisa incluir informações sobre a localização de objetos. Isso significa que é necessário atribuir um rótulo de classe a cada pixel da imagem.

A tarefa que visa rotular cada pixel de uma imagem com uma classe correspondente ao que está sendo representado é conhecida como segmentação semântica. Como essa tarefa envolve a previsão para cada pixel da imagem, é comumente chamada de previsão densa. É importante destacar que, na segmentação semântica, o sistema não diferencia automaticamente objetos da mesma classe. Por exemplo, se houver dois objetos diferentes, mas da mesma classe, na imagem de entrada, o mapa de segmentação não os distinguirá como entidades separadas.

Para alcançar a segmentação semântica, é possível substituir a camada totalmente conectada por camadas de convolução, transformando a rede neural em uma *Fully Convolutional Neural Network* (FCNN). Isso permite que o modelo processe imagens de entrada de tamanhos variados em vez de se limitar a uma resolução fixa. Além disso, a saída da FCNN é um mapa de probabilidades para cada píxel, em vez de uma única saída para um único pixel. No entanto, devido às camadas de *pooling*, a resolução da saída é tipicamente muito menor do que a da imagem de entrada (LONG; SHELHAMER; DARRELL, 2014).

Em resumo, a segmentação semântica é uma tarefa importante que envolve a rotulagem de cada pixel de uma imagem com sua classe correspondente, e isso pode ser alcançado eficazmente por meio de redes neurais totalmente convolucionais, que permitem processar imagens de diferentes tamanhos e gerar mapas de probabilidade detalhados.

## 2.12 considerações

Com base nas técnicas e algoritmos delineados neste capítulo, torna-se viável a realização da implementação e da subsequente análise experimental dos estudos que foram examinados. Além disso, essas informações proporcionam uma compreensão detalhada dos procedimentos empregados na execução dos experimentos que serão apresentados nos próximos capítulos.

# 3 TRABALHOS RELACIONADOS

O atual capítulo pretende apresentar e comentar sobre trabalhos similares ao proposto pelo atual trabalho, além de apresentar uma análise comparativa entre os mesmo.

O trabalho conduzido por Anthoniraj, Karthikeyan e Vivek (2022), intitulado *Weed Detection Model Using Generative Adversarial Network and Deep Convolutional Neural Network*, fornece uma visão geral do emprego de técnicas de Aprendizado Profundo *Deep Learning* na agricultura, focando na detecção de ervas daninhas em plantações. Este estudo aborda três tópicos essenciais.

Primeiramente, o trabalho introduz uma abordagem que combina uma Rede Generativa Adversaria (GAN) com uma Rede Neural Convolucional Profunda (DCNN), denominada GAN-DCNN, para realizar a detecção das ervas na plantação. Em seguida, ocorre uma otimização dos parâmetros de treinamento, visando aprimorar a taxa de detecção. Por fim, o estudo realiza uma comparação entre o modelo proposto e outros modelos de segmentação disponíveis.

O modelo proposto pelo trabalho utiliza um extenso conjunto de dados composto por 17.509 imagens para treinamento, conhecido como *DeepWeeds* (OLSEN et al., 2019). A abordagem desenvolvida para abordar o problema de segmentação na agricultura se destaca pela utilização de um conjunto de dados amplo e pelos resultados alcançados, que, embora semelhantes a outros modelos, demonstram uma leve vantagem. Em resumo, o trabalho se destaca por sua contribuição para a detecção de ervas daninhas em ambientes agrícolas, através da aplicação de técnicas avançadas de *Deep Learning* e pela utilização de um conjunto de dados robusto, apresentando resultados competitivos em comparação com outros modelos similares.

O trabalho conduzido por (ISOLA et al., 2017), intitulado *Image-to-Image Translation with Conditional Adversarial Networks*, aprofunda o estudo do uso de técnicas de GAN condicionais com aplicações de uso geral. Os autores destacam que essa abordagem não apenas permite a aprendizagem de mapeamentos de imagens entre si, mas também capacita a rede a aprender a função de perda para o treinamento do modelo. O trabalho resulta no desenvolvimento de um software chamado "pix2pix".

Os autores do pix2pix exploram várias bases de dados para avaliar o desempenho da arquitetura estudada. Utilizaram conjuntos de dados que incluem mapas de cidades, contendo 1096 imagens do Google Maps da região metropolitana de Nova York (CORDTS et al., 2016), arquitetura urbana com 400 imagens (TYLECEK; SARA, 2013),

além de desenhos à mão, como imagens de sapatos do conjunto de dados UT Zappos50K, que contém 50 mil imagens (YU; GRAUMAN, 2014), e 137 mil imagens de bolsas de mão coletadas da Amazon (ZHU et al., 2018). No desfecho do estudo, os autores concluem que o uso de redes generativas condicionais demonstra ser uma abordagem promissora para tarefas de transferência de estilo, especialmente em cenários com estruturas bem definidas.

Ao avaliar os trabalhos mencionados anteriormente, é possível destacar diferenças significativas entre o trabalho atual e essas referências. No que se refere ao trabalho de (ANTHONIRAJ; KARTHIKEYAN; VIVEK, 2022), a principal distinção reside na ausência de uma análise mais detalhada direcionada especificamente para plantações de soja. No entanto, é importante notar que o autor reconhece essa limitação e aponta que é uma área que poderia ser explorada em seus futuros projetos de pesquisa. Além disso, outra discrepância evidente diz respeito ao tamanho do conjunto de dados utilizado. Na abordagem escolhida para o presente trabalho, empregamos um conjunto de dados contendo mais de 17 mil imagens de ervas daninhas, em contraste com o nosso modelo, que utiliza uma quantidade significativamente menor de dados na busca por resultados satisfatórios.

Quando se considera o trabalho proposto por (ISOLA et al., 2017), ele abre uma lacuna interessante que pode ser explorada por este trabalho em andamento. Essa lacuna refere-se à análise do desempenho das Redes Generativas Condicionais na segmentação de áreas relacionadas à agricultura, em particular, na segmentação de plantações de soja. Essa perspectiva representa um ponto de diferenciação fundamental entre o nosso trabalho atual e a pesquisa anterior, oferecendo uma abordagem inovadora e valiosa no contexto da agricultura de precisão e detecção de ervas daninhas em plantações de soja.

Outro aspecto crucial a ser considerado é que nenhum dos trabalhos mencionados oferece uma abordagem explícita quanto ao desempenho em tempo real de seus modelos. O trabalho em andamento visa preencher essa lacuna, priorizando a criação de um modelo eficiente em dispositivos embarcados. Essa ênfase é de extrema importância, pois está diretamente relacionada à necessidade prática de realizar o reconhecimento das ervas daninhas e a aplicação do inseticida em tempo real nas plantações. A capacidade de operar em tempo real é um requisito crítico para garantir a eficácia e a viabilidade do sistema em campo, e esse é um diferencial fundamental que destacará o presente trabalho em relação às referências anteriores. Portanto, o foco na eficiência e na operação em tempo real é uma característica distintiva essencial deste projeto de pesquisa.

Portanto, essas distinções ressaltam a contribuição única e relevante do presente

trabalho, que se concentra na análise da aplicação de redes generativas condicionais na segmentação de plantações de soja, bem como na busca por soluções eficazes com um conjunto de dados mais limitado, visando aprimorar a eficiência e acessibilidade dessa abordagem na agricultura moderna.

# 4 METODOLOGIA

Neste capítulo, serão descritos os métodos e técnicas empregados na condução deste estudo. A metodologia apresentada a seguir delineará os passos seguidos para o treinamento de um modelo de aprendizado de máquina destinado à segmentação de imagens de plantações de soja. Para a execução bem-sucedida do experimento, dividimos o processo em quatro etapas distintas: coleta de dados de plantações de soja, pré-processamento das imagens, aplicação de técnicas de aumento de dados (*data augmentation*) e, por fim, o treinamento propriamente dito do modelo.

## 4.1 Coleta dos dados

Para tornar o treinamento viável, a coleta de imagens nas plantações foi realizada por técnicos agrícolas da Accore[1], uma startup brasileira especializada no desenvolvimento de sistemas para a indústria agrícola, em geral. Optamos por focar nossos experimentos na plantação de soja, conforme detalhado anteriormente. A soja desempenha um papel crucial no agronegócio brasileiro e representa um dos principais produtos de importação alimentar no país.

A coleta de dados foi realizada manualmente durante o turno da tarde, utilizando um conjunto de quatro câmeras distintas. A altura padrão utilizada para as capturas foi de 70 centímetros em relação ao solo. A resolução das imagens, medida em megapixels (MPixels), variou de 5 a 20 MPixels. Essa escolha de altura tinha como objetivo simular a posição da câmera em um veículo de dispersão de inseticidas. Além disso, a estratégia de diversificar a resolução das imagens e escolher várias câmeras foi cuidadosamente planejada para minimizar qualquer viés (bias) durante o treinamento do modelo. Capturando os mesmos objetos em imagens com diferentes resoluções, buscamos capacitar a rede neural a se adaptar a variações de informações.

Esse processo de coleta meticulosa e estratégica resultou em um conjunto de dados diversificado e robusto, fundamental para apoiar nossos esforços de treinamento e pesquisa. No entanto, o conjunto de dados usado para treinar o modelo final continha um total de 400 imagens. Isso ocorreu devido à criação simultânea do *dataset*, que exigiu a marcação manual das imagens com os respectivos rótulos. Esse é um processo trabalhoso que requer a expertise de profissionais da área.

[1] <https://accore.com.br>

As imagens do *dataset* continham informações sobre cinco estados possíveis encontrados na plantação de soja. O primeiro estado representava a própria planta de soja, enquanto o segundo era o Caruru. O terceiro estado era a Folha-Estreita, o quarto era o Folha-Larga, sendo que os três últimos eram diferentes espécies de plantas daninhas. O quinto estado representava uma situação em que o solo não continha nada. Cada uma dessas situações foi rotulada com cores diferentes, sendo a soja marcada em verde, o caruru marcado em laranja, a Folha-estreita marcada em azul, a Folha-Larga marcada em vermelho e o solo vazio em cinza.

## 4.2 Pré-processamento

O pré-processamento de dados é uma etapa fundamental no campo de aprendizado de máquina, uma vez que a qualidade dos dados e as informações úteis que podem ser extraídas deles têm um impacto direto na capacidade de aprendizado do modelo. Portanto, é de suma importância preparar os dados para ficarem em um estado que o algoritmo possa analisar com facilidade e eficácia. Nas Subseções a seguir, detalho cada uma das etapas de pré-processamento aplicadas às imagens.

### 4.2.1 Separação das imagens

Para simplificar o processo de carregamento das imagens no modelo, é realizado uma fusão (*merge*) entre as imagens de *ground truth* (verdade de referência) e as imagens de entrada. Nesse contexto, a imagem de entrada representa aquela que será fornecida ao modelo no início do processo, enquanto o *ground truth* é a imagem alvo que o modelo tentará reproduzir.

No entanto, devido a essa fusão das imagens, torna-se necessário realizar uma etapa de separação dos dados, a fim de extrair tanto a imagem original quanto a imagem de *target* para serem utilizadas pelo modelo. No nosso caso, as imagens foram combinadas horizontalmente, o que significa que a separação das duas partes pode ser realizada de maneira direta e eficiente, simplesmente dividindo a imagem ao meio.

### 4.2.2 Recolorização

A criação da base de dados envolveu a marcação das imagens por meio de uma ferramenta que aplicou o método SLIC (*Simple Linear Iterative Clustering*) (JAIN, 2019) para segmentação das imagens. No entanto, esse processo permitiu uma variação significativa e, em muitos casos, desnecessária nas cores. Um exemplo notável é a introdução de uma grande variedade de tons de verde, o que poderia potencialmente prejudicar o treinamento do modelo.

Para mitigar esse problema, foi proposto uma técnica de clusterização que difere da abordagem tradicional, onde não são calculados os centroides, pois os mesmos já foram previamente definidos. São feitas modificações nas cores das regiões segmentadas com base na proximidade ao centroide correspondente. Na prática, esses centroides são tratados como as cores de referência usadas na marcação inicial.

Para cada píxel (i, j) na imagem I, onde i e j variam de 0 a N-1 e de 0 a M-1, sendo M e N as dimensões da imagem, atribuímos ao pixel a cor do centroide mais próximo entre o conjunto de centroides c.

Essa abordagem nos permitiu eliminar variações desnecessárias e reduzir a quantidade de possíveis erros nas etiquetas introduzidos durante o processo de marcação das imagens.

### 4.2.3 Redimensionamento

Um dos desafios significativos ao trabalhar com redes totalmente conectadas é a relação direta entre o número de neurônios e o tempo necessário para processar resultados. Esse problema se torna especialmente evidente ao lidar com imagens, onde, em teoria, seria necessário ter um neurônio para cada pixel da imagem. No entanto, essa questão pode ser mitigada com o uso de Redes Neurais Convolucionais (CNNs), que são altamente eficazes na tarefa de processamento de imagens.

No entanto, mesmo com o uso de CNNs, ainda podemos enfrentar problemas de tempo de processamento, especialmente quando se trata de camadas densas, conhecidas como camadas totalmente conectadas. Uma maneira de abordar esse problema é redimensionando *resize* as imagens antes de alimentá-las na rede.

No experimento realizado aqui, optamos por redimensionar as imagens para um tamanho de 255 x 255 píxeis, seguindo a abordagem proposta pelos autores no trabalho

(ISOLA et al., 2017). As imagens originais do *dataset* tinham dimensões de 1280 x 720 píxeis e, ao redimensioná-las, conseguimos reduzir a quantidade de neurônios necessários nas camadas internas da rede. Isso resultou em um treinamento mais eficiente, pois o processamento tornou-se mais ágil e a rede foi capaz de aprender com eficácia.

### 4.2.4 Normalização

Um desafio significativo que pode surgir durante o treinamento de modelos é a presença de dados discrepantes no conjunto de dados, onde os valores variam consideravelmente, por exemplo, com valores mínimos próximos a 1 e valores máximos em torno de 2.000.000. Isso pode introduzir uma alta taxa de erro, pois a variação nos valores dos dados é muito grande. Uma estratégia eficaz e muito comum para mitigar esse problema é a normalização dos dados. Essa técnica permite mapear os valores do conjunto de dados para uma faixa menor, reduzindo assim a taxa de erro do modelo.

No caso do *dataset* utilizado no modelo proposto, a normalização foi realizada da seguinte forma: inicialmente, todos os píxeis das imagens foram divididos por 127,5 (representando a metade de 255, que é o valor máximo de intensidade de cor em uma imagem) e, em seguida, subtraiu-se 1. Essa operação foi aplicada tanto para às imagens originais quanto para às imagens de destino *target*. Ao final desse processo, todos os píxeis nas imagens ficaram dentro da faixa de valores de -1 a +1.

Essa normalização contribuiu significativamente para tornar os dados mais adequados ao treinamento do modelo, reduzindo a amplitude das variações e, assim, melhorando a estabilidade do processo de aprendizado.

## 4.3 Separação dos dados

Após a aplicação do processo de aumento de dados *data augmentation*, procedeu-se com a técnica de *holdout* para dividir o conjunto de dados em dois subconjuntos: um de treinamento e outro de teste. Originalmente, o conjunto de dados consistia apenas em 400 imagens. Optar por uma configuração de holdout com uma divisão em três conjuntos poderia ser prejudicial, uma vez que cada conjunto resultante seria de tamanho muito reduzido.

Para o conjunto de teste, foi decidido alocar 80% dos dados, reservando 20% para

o conjunto de treinamento. A separação dos conjuntos foi realizada de forma completamente aleatória. Essa escolha foi feita para garantir uma quantidade razoável de dados no conjunto de treinamento, considerando a limitação no tamanho do conjunto de dados original, visto que o usado por (ANTHONIRAJ; KARTHIKEYAN; VIVEK, 2022) usa um *dataset* com 17k imagens.

## 4.4 *Data Augmentation*

Devido à limitação na disponibilidade de dados de treinamento, optamos por adotar uma abordagem baseada em GAN. Para contornar essa escassez de dados, implementamos estratégias robustas de aumento de dados, totalizando nove técnicas aplicadas.

Rotação Horizontal e Vertical: Efetuamos reflexões horizontais e verticais das imagens originais, criando versões espelhadas que enriqueceram a diversidade do conjunto de dados.

Rotações de 90° e 180°: aplicamos rotações nas imagens originais de 90 graus e 180 graus, introduzindo variações na orientação das imagens.

Zoom (Aproximação e Afastamento): Utilizamos zoom nas imagens, tanto para aproximá-las quanto para afastá-las, com um fator de escala de 20%, expandindo a variabilidade das imagens.

Deslocamentos em X e Y (20%): Implementamos deslocamentos de 20% tanto no eixo X quanto no eixo Y das imagens, criando novas versões deslocadas das imagens originais.

Adicionalmente, realizamos uma etapa de recorte *crop* em cada uma das imagens geradas, dividindo-as em nove novas imagens. Essa abordagem ampliou ainda mais o volume de dados de treinamento disponível. Essas estratégias de aumento de dados desempenharam um papel fundamental na melhoria da robustez e na ampliação da diversidade do nosso conjunto de dados de treinamento.

Ao final deste processo, conseguimos expandir o número de imagens para mais de 19 mil, enriquecendo significativamente nosso conjunto de dados para fins de treinamento.

## 4.5 Arquitetura

Como mencionado anteriormente, a rede escolhida como base para nosso projeto foi a pix2pix, devido à sua notável capacidade de aprender com um número limitado de exemplos, conforme discutido em detalhes por Isola et al. em seu trabalho (ISOLA et al., 2017). Essa característica tornou o treinamento muito mais viável, especialmente considerando a quantidade limitada de dados disponíveis para o treinamento, visto que o próprio trabalho tem um exemplo de 400 imagens onde os autores mostram um bom resultado.

No processo de treinamento das redes, utilizamos a arquitetura da UNet-255 como gerador e uma versão modificada do PatchGAN como discriminador. Essa combinação de arquiteturas é estratégica para a nossa tarefa específica, pois permite que o gerador crie imagens de alta qualidade e o discriminador avalie de forma eficaz a autenticidade das imagens geradas em relação às imagens reais.

Essas escolhas arquiteturais foram fundamentais para o sucesso do nosso projeto, pois nos permitiram alcançar resultados satisfatórios mesmo com um conjunto limitado de dados de treinamento.

### 4.5.1 Unet-256

A Figura 4.1 representa a rede utilizada como geradora no modelo escolhido para nosso experimento. O funcionamento dessa rede é projetado para superar desafios relacionados à perda de informações ao longo das camadas. Para resolver isso, incorporamos conexões de salto *skip connections*, seguindo o princípio geral da arquitetura 4.1, conforme proposto em seu trabalho (RONNEBERGER; FISCHER; BROX, 2015). Essas conexões de salto são um componente essencial da U-Net e são fundamentais para melhorar a capacidade da rede de preservar detalhes cruciais durante tarefas como a segmentação de imagens.

O conceito por trás das conexões de salto é simples: conectamos cada camada $i$ à camada correspondente $n - i$, onde $n$ é o número total de camadas na rede. Cada conexão de salto realiza uma operação de concatenação, combinando todos os canais de informação na camada $i$ com os canais da camada $n - i$. Isso permite que informações de baixo nível, detalhes de alta resolução e características de alto nível sejam transmitidas eficazmente pelas camadas da rede.

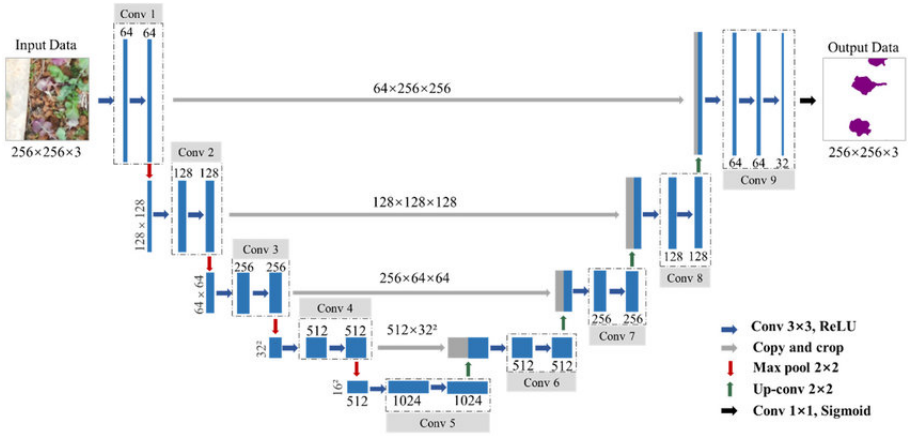



Figura 4.1 – Arquitetura Unet-256 (ZHANG et al., 2020)
Fonte: (GARDNER; DORLING, 1998)

Essas conexões de salto desempenham um papel fundamental em diversas aplicações de visão computacional, como a segmentação de objetos em imagens médicas e a detecção de características em imagens de satélite. Elas possibilitam que o modelo capture informações contextuais em grande escala, mantendo ao mesmo tempo a resolução espacial e a precisão na segmentação, tornando a arquitetura U-Net uma escolha eficaz para tais tarefas.

### 4.5.2 PatchGAN

A arquitetura PatchGAN, representada na Figura 4.2, desempenha um papel crucial no modelo escolhido para o nosso experimento. Essa abordagem difere das redes convencionais, como as redes neurais convolucionais tradicionais (CNNs), ao focar na avaliação da autenticidade de pequenas regiões (patches) de uma imagem em vez da imagem inteira. Isso é especialmente útil em tarefas como a geração de imagens condicionais, como no nosso caso.

O funcionamento do PatchGAN é relativamente simples, mas altamente eficaz. A ideia principal é subdividir a imagem em pequenos patches, geralmente de tamanho fixo, e avaliar se cada patch é real ou falso. Essa avaliação é realizada por meio de convoluções locais em cada patch, em vez de em toda a imagem. O resultado é uma matriz de saída que contém pontuações de autenticidade para cada patch.

A contribuição significativa do PatchGAN é a capacidade de fornecer feedback detalhado sobre a autenticidade em várias regiões da imagem. Isso permite ao gerador criar imagens mais realistas, pois ele é incentivado a focar em detalhes importantes de

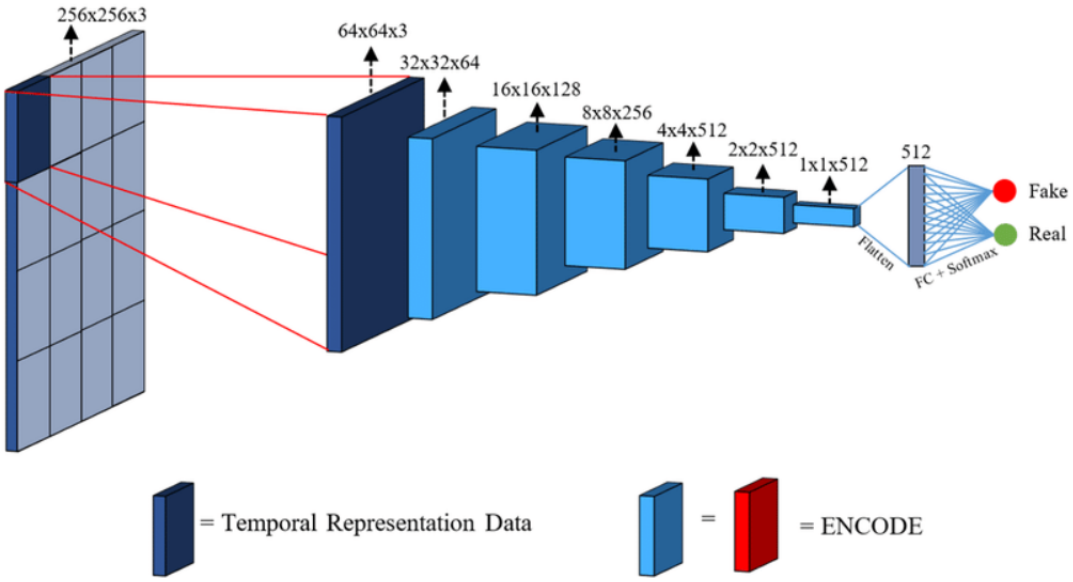



Figura 4.2 – Arquitetura PatchGAN (GANOKRATANAA; ARAMVITH; SEBE, 2020)
Fonte: (GARDNER; DORLING, 1998)

uma imagem e aprimorar essas áreas. Além disso, o discriminador PatchGAN é menos suscetível a problemas de "modo colapso", nos quais o gerador produz imagens sem diversidade.

Em resumo, a arquitetura PatchGAN desempenha um papel crucial no nosso modelo, permitindo a avaliação eficaz da autenticidade de patches individuais de imagens, o que é fundamental para a geração de imagens condicionais de alta qualidade. Essa abordagem contribui para a melhoria da qualidade e diversidade das imagens geradas.

# 5 ANALISE EXPERIMENTAL

Este capítulo, abordaremos os resultados obtidos com base na metodologia descrita anteriormente. Apresentaremos gráficos que mostrarão os resultados alcançados e, adicionalmente, realizaremos uma comparação visual entre as imagens esperadas e as imagens obtidas pelo modelo. Explicaremos detalhadamente esses resultados, destacando todos os aspectos relevantes descobertos durante a elaboração deste trabalho. Além disso, forneceremos tabelas comparativas que facilitarão a visualização do desempenho dos modelos propostos em um ambiente de tempo real.

## 5.1 Ambiente de execução

O experimento foi implementado e testado em um ambiente de execução único. A máquina utilizada para os experimentos possui as seguintes especificações: uma CPU Ryzen 5 - 2600 de 3,4 GHz, uma GPU NVIDIA GeForce 1650 Super com 4 GB de VRAM e 16 GB de memória RAM com frequência de 3200 MHz. Para melhorar a eficiência, foram empregados alguns frameworks de aceleração por hardware, como Numba e cuDNN.

O sistema está configurado com a distribuição Mint do Linux, Python 3, bibliotecas como NumPy, OpenCV e PyTorch foram instaladas para suportar o desenvolvimento e execução dos experimentos. Os códigos utilizados no experimento estão disponíveis em github(https://github.com/kovaleski100/TCC).

## 5.2 Resultados

Para uma análise mais aprofundada dos dados, propomos dividir a seção atual em dois tópicos distintos. O primeiro tópico se concentrará na avaliação dos modelos criados utilizando o conjunto de dados original. O segundo tópico abordará a análise dos modelos treinados com o conjunto de dados aumentado por meio da técnica de *data augmentation*. Além disso, apresentaremos uma subseção dedicada à comparação do desempenho do nosso modelo em tempo real com outros modelos similares.

### 5.2.1 Treinamento e Teste

Para realizar o treinamento da rede com o conjunto de dados da Accore, seguimos os procedimentos detalhados no Capítulo 4. Primeiramente, aplicamos um processo de pré-processamento nas imagens, visando eliminar o máximo de ruído que ainda poderia estar presente nas marcações. Em seguida, implementamos o aumento de dados *data augmentation* para enriquecer nosso conjunto de treinamento. Após essas etapas, dividimos os dados em dois conjuntos: treino e teste. Optamos por uma proporção de 80/20, ou seja, 80% das imagens foram destinadas ao conjunto de treinamento e 20% para o conjunto de teste.

Para avaliar o desempenho do modelo durante o treinamento, utilizamos a métrica de acurácia, que nos permite determinar a taxa de acerto da rede em relação às imagens. Além disso, monitoramos o valor da função de perda *loss* do modelo para entender como a taxa de erro evolui ao longo do treinamento.

A cada execução de um modelo de rede, geramos gráficos para representar visualmente a evolução da rede a cada época de treinamento. Esses gráficos são fundamentais para proporcionar uma compreensão clara do desempenho e do progresso do modelo ao longo do processo de treinamento.

Para a criação dos modelos foi decidido por 2 abordagens. Primeiramente foi feita uma "recriação"dos experimentos propostos por (ISOLA et al., 2017) onde os mesmos comentam os parâmetros usados para cada *dataset* em termos de épocas de treinamento e em *batch size*, representado pelas Figuras 5.1, 5.2, 5.3, 5.4 e 5.5. Tendo feito isso, decidiu se testar diferentes valores de *batch size* a fim de tentar encontrar um valor que maximize a acurácia do modelo, visto que o problema tratado nesse trabalho não foi abordado no trabalho de (ISOLA et al., 2017), para isso foi usado os valores de 2, 8, e 32.

Durante o treinamento, a variação no número de épocas para a criação de cada modelo não pareceu ter um impacto significativo. Em algumas classes, observamos uma convergência do modelo, apesar de algumas instabilidades que provavelmente ocorreram devido à tentativa do modelo de explorar diferentes valores dos pesos. Uma observação notável foi que os modelos treinados por mais épocas geralmente apresentaram uma convergência mais sólida. No entanto, é importante notar que a melhoria na convergência não necessariamente se traduziu em uma melhor capacidade do modelo de generalizar casos.

Quando avaliamos os modelos com os dados de teste, torna-se evidente que há outros aspectos a serem considerados na busca por um modelo de alta qualidade. Ao

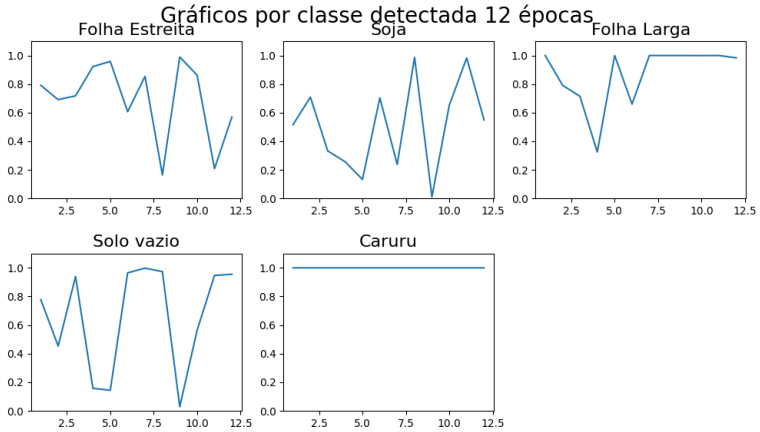



Figura 5.1 – Gráficos por classe detectada 12 épocas

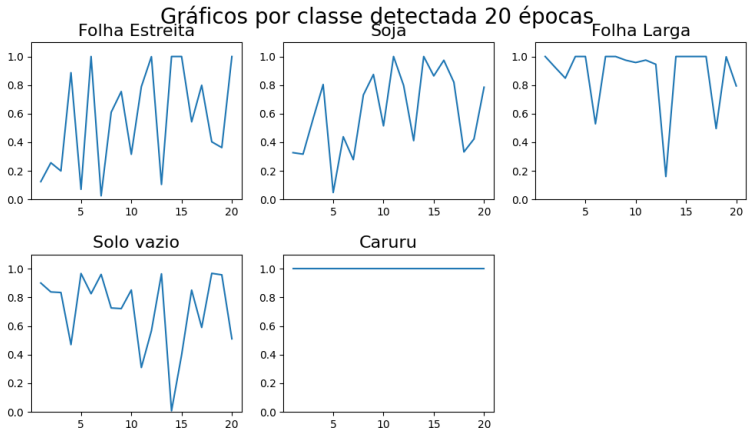



Figura 5.2 – Gráficos por classe detectada 20 épocas

analisarmos a Figura 5.6, podemos perceber que a variação no número de épocas de treinamento resultou em poucos benefícios visíveis no desempenho do modelo. No entanto, ao examinarmos as outras duas figuras, a Figura 5.7 e a Figura 5.8, torna-se aparente que o valor ideal de épocas de treinamento é 34. Isso se evidencia nos gráficos, já que é o ponto em que a precisão atinge seu pico e o valor do F1-score é maximizado, indicando que esse é o ponto ótimo para o desempenho do modelo. Nesse caso, é possível dizer que o melhor valor de epocas para se usar para o treinamento é desses modelos é 34.

Como mencionado anteriormente, também foram feitos 3 testes, deixando o número de época em 200 e variando o número do *batch size*. Para esse treinamento, foi usado o conjunto de dados original, no caso, sem o aumento de dados, visto que, em geral, um treinamento com o *data augmentation* tende a aumentar o valor de acurácia.

Após os treinamentos descritos acima, foi rodado um treinamento com as métricas do modelo com *batch size* 32 e com 200 épocas para averiguarmos como o *data augmentation* melhorou a acurácia do modelo. Conforme evidenciado na Figura 5.9, é possível

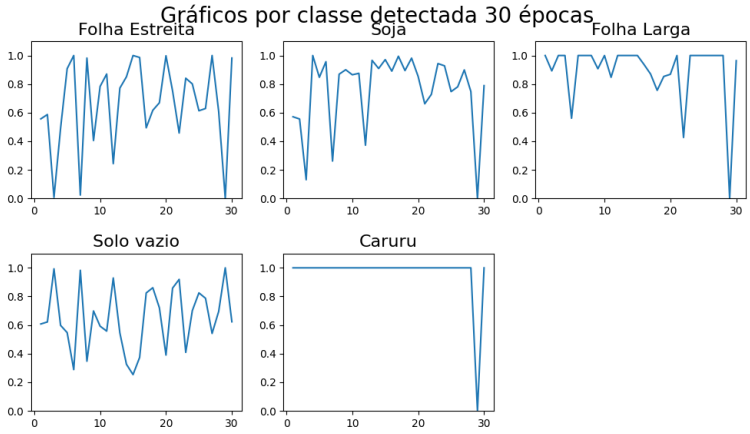



Figura 5.3 – Gráficos por classe detectada 30 épocas

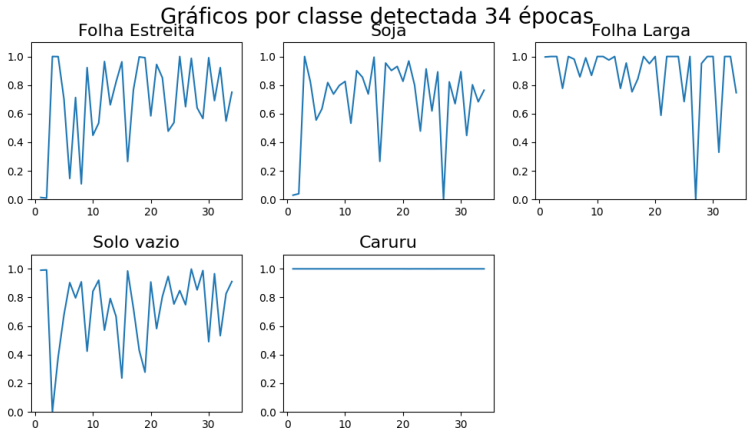



Figura 5.4 – Gráficos por classe detectada 34 épocas

observar que a utilização do método de *data augmentation* resultou em uma melhoria significativa na acurácia do modelo. Isso é notável ao considerar que a acurácia média por época é consistentemente mais elevada quando o aumento de dados é aplicado.

Conforme evidenciado pelas figuras 5.14, 5.15 e 5.16, observamos um fenômeno semelhante ao que ocorreu quando variamos o número de épocas de treinamento. A acurácia não apresentou variação significativa quando avaliada no conjunto de teste. No entanto, é notável que o modelo treinado com um tamanho de lote *batch size* igual a 2 exibiu um desempenho superior em relação aos outros modelos. Portanto, nesse contexto, podemos concluir que o tamanho de lote igual a 2 é o que melhor representa o modelo em termos de desempenho.

Esse achado sugere que o tamanho do lote desempenha um papel crucial no treinamento do modelo, e um tamanho de lote menor, como 2, pode ser mais eficaz para o problema em questão, levando a resultados mais favoráveis em comparação com outros tamanhos de lote testados.

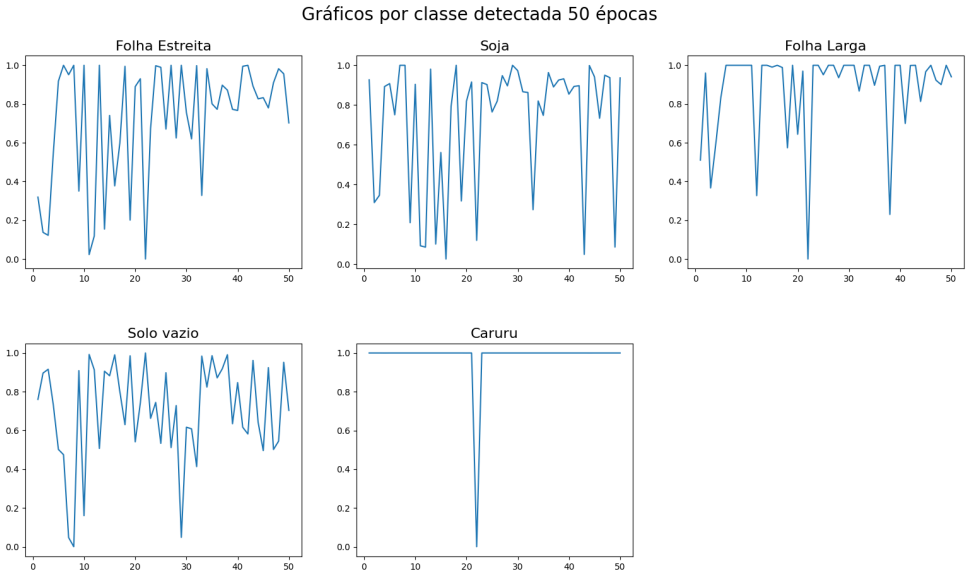



Figura 5.5 – Gráficos por classe detectada 50 épocas

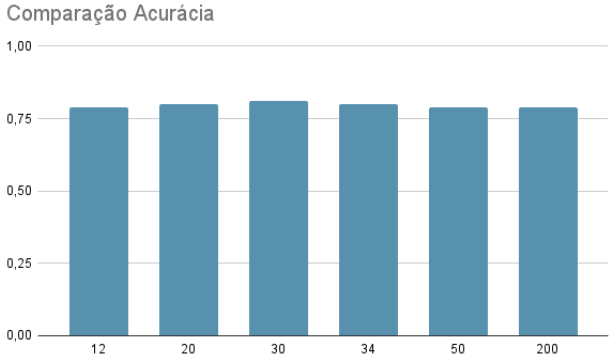



Figura 5.6 – Comparação da acurácia de cada modelo

Ao analisarmos os resultados com o conjunto de testes, como ilustrado na figura 5.13, observamos que, embora a diferença na acurácia entre os dois modelos não tenha sido significativa, houve um aumento notável de quase 20% na precisão e no F1-score, indicando que a técnica de *data augmentation* proporcionou um ganho considerável de desempenho.

Além dos experimentos mencionados, também conduzimos um estudo que considerou todos os dados coletados nos experimentos anteriores. Neste estudo, utilizamos *data augmentation*, realizamos 34 épocas de treinamento e definimos o tamanho do lote *batch size* como 2. O objetivo deste experimento foi avaliar se haveria benefícios ao empregar valores tão elevados para o treinamento, especialmente quando comparados ao modelo de *data augmentation*, que foi treinado com 200 épocas e um tamanho de lote de 32.

Esse experimento visa examinar se o aumento na duração do treinamento e no

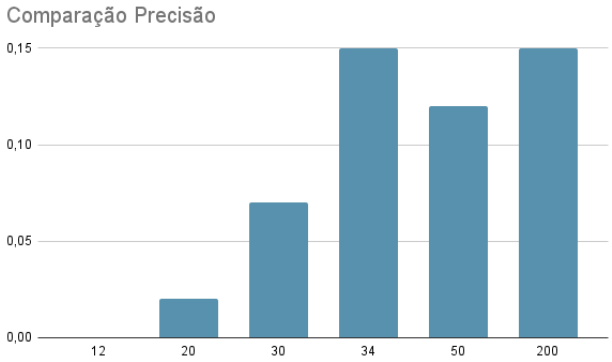



Figura 5.7 – Comparação da precisão de cada modelo

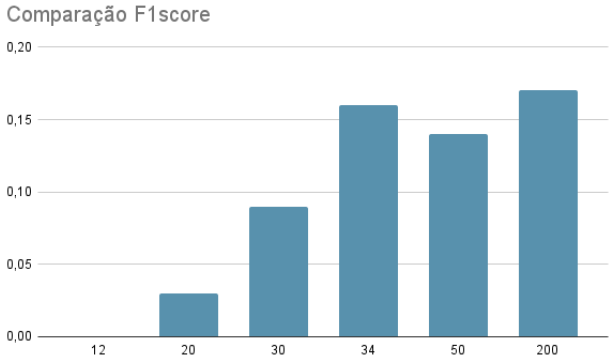



Figura 5.8 – Comparação de F1-score de cada modelo

tamanho do lote resulta em melhorias significativas em relação ao modelo de *data augmentation*, que é uma referência comum para comparação.

Conforme podemos observar, ambos os modelos demonstram uma performance bastante semelhante, o que sugere que o treinamento com valores mais elevados do que os utilizados no modelo anterior não é realmente necessário, uma vez que os valores de precisão e F1-score são comparáveis.

Embora ambos os modelos tenham obtido resultados sólidos em termos de acurácia em comparação com os modelos anteriores, é importante ressaltar que nenhum deles foi especificamente projetado com foco em ambientes embarcados. Portanto, optamos por realizar o treinamento usando um modelo com um número menor de parâmetros, visando criar um modelo mais ágil e mais adequado para dispositivos embarcados. Para fins de discussão dos resultados dos modelos, chamaremos o modelo de menor complexidade de "tiny-GAN", enquanto o modelo anterior, com dimensões maiores, será referido como "o modelo GAN original".

No novo modelo, realizamos reduções nas camadas internas tanto do gerador quanto do discriminador. Especificamente, no gerador, optamos por cortar pela metade o número de filtros gerados na última camada. Isso proporcionou benefícios significa-

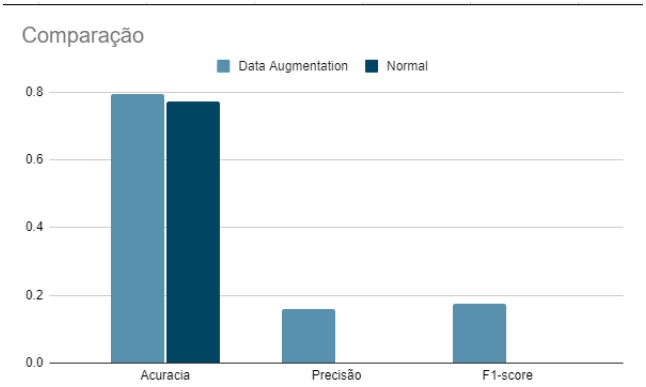



Figura 5.9 – Comparação da acurácia entre os modelos testados

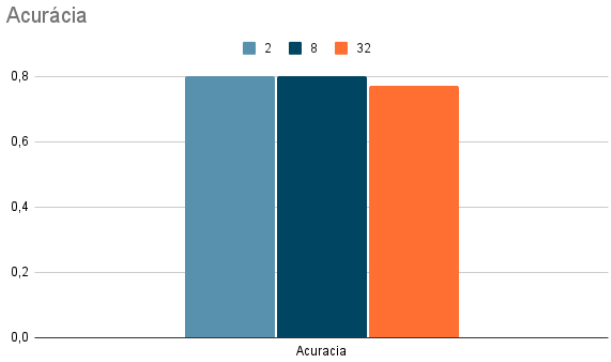



Figura 5.10 – Comparação da acurácia entre os modelos testados

tivos, pois a redução resultou em economia de memória e, ao mesmo tempo, aumentou a velocidade de inferência do modelo. Vale destacar que o gerador desempenha um papel crucial na estratégia GAN, pois é responsável pela geração da imagem segmentada, tornando essa modificação particularmente relevante para obter os resultados desejados.

Conforme evidenciado nos gráficos apresentados nas referências 5.17, 5.18 e 5.19, é notável que o modelo treinado exibe uma sólida taxa de acurácia geral. No entanto, quando consideramos a precisão por classe, observamos que o desempenho do modelo deixa algo a desejar. Essa discrepância provavelmente se deve ao fato de que, mesmo após a avaliação do modelo com a aplicação de técnicas de aumento de dados *data augmentation*, a variabilidade intrínseca do modelo não aumentou substancialmente. Em outras palavras, a distribuição das classes no conjunto de dados de treinamento permaneceu inalterada, com uma prevalência significativamente maior de ocorrências de soja e áreas vazias em comparação com ervas daninhas, que são uma ocorrência menos comum na plantação e, portanto, consideradas uma anomalia.

Ao conduzir uma análise das distribuições de classes no conjunto de dados, verificou-se que aproximadamente 30% das amostras pertencem à classe de folha larga, 7% à classe de folha estreita, 0% à classe de caruru, 36% à classe de solo vazio e 25% à classe de soja.

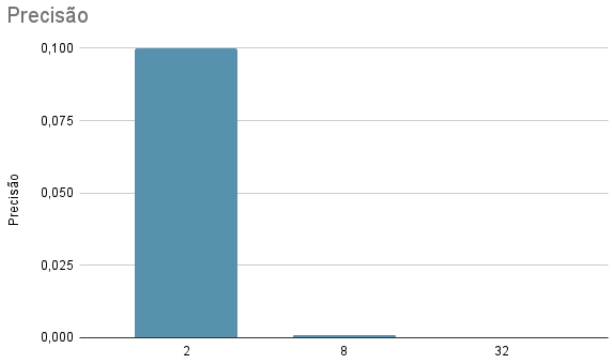



Figura 5.11 – Comparação da precisão entre os modelos testados

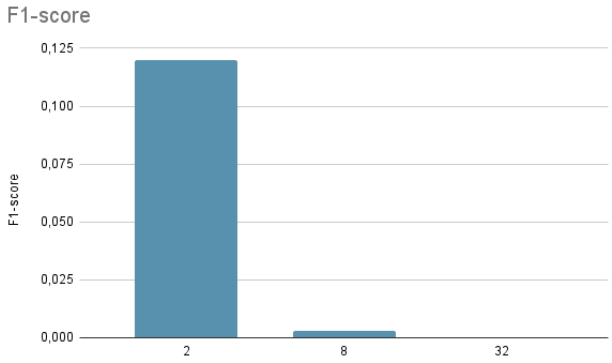



Figura 5.12 – Comparação de F1-score entre os modelos testados

Com base nessa observação e na avaliação dos resultados obtidos nas análises, pode-se concluir que a alta acurácia do modelo ocorre devido ao seu desempenho sólido em situações envolvendo soja, folhas largas e solo vazio, mas apresenta desafios substanciais quando se trata de identificar folhas estreitas.

No próximo capítulo, aprofundaremos a discussão sobre o desempenho do modelo, denominado "tiny-GAN", ao compará-lo com outros modelos empregados com o mesmo propósito de classificação.

### 5.2.2 Real-time

Os experimentos em tempo real envolvendo a aplicação e teste dos modelos em dispositivos finais seguiram um procedimento meticuloso. Inicialmente, as aplicações e bibliotecas necessárias foram instaladas para permitir a execução dos modelos. Em seguida, os modelos foram executados em dois ambientes distintos: a Jetson AGX Orin, a máquina topo de linha da NVIDIA na família Jetson, e a Jetson Nano, a plataforma mais limitada da mesma família. Cada experimento foi realizado sequencialmente, com

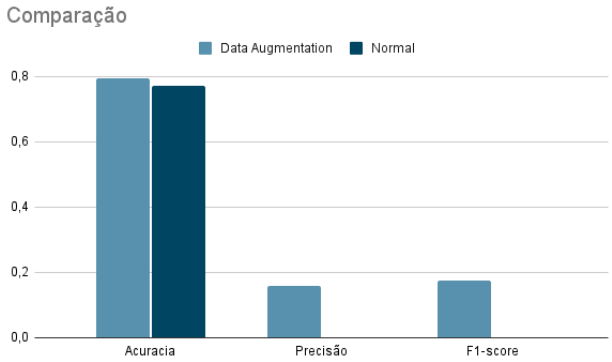



Figura 5.13 – Gráfico comparativo modelo dados aumentados vs modelo original

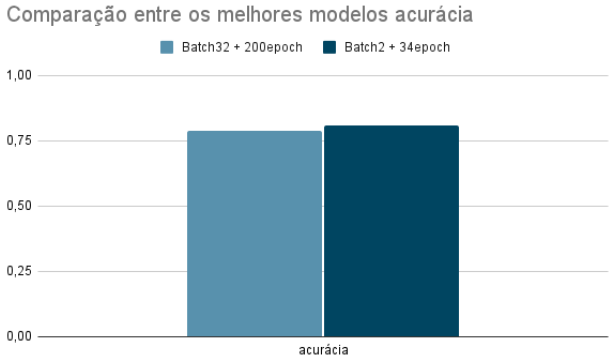



Figura 5.14 – Comparação da acurácia entre os modelos que tiveram melhor desempenho

reinicialização das máquinas entre eles para garantir que vazamentos de memória não afetassem os resultados dos modelos subsequentes. Os dados apresentados nas Tabelas 5.1 foram obtidos por meio do cálculo da média de FPS (Quadros por segundo), que conta quantas vezes o loop principal do programa é executado em um segundo. Por outro lado, os dados na Tabela 5.2 foram coletados usando um programa de perfilamento fornecido pela NVIDIA.

Para viabilizar os testes nas plataformas, foi necessário adaptar as saídas dos modelos para serem compatíveis com o sistema já existente na empresa. Nesse sentido, foi implementado um pós-processamento das imagens geradas. Primeiramente, as imagens passaram por um redimensionamento usando interpolação cúbica para as dimensões [36, 64]. Essas dimensões foram escolhidas com base nas especificações da empresa. Além disso, cada cor presente na imagem foi convertida para sua representação em binário correspondente, variando de 0 a 5. Também foram feitas modificações na forma como as imagens eram transmitidas entre os módulos relevantes. Em particular, dois módulos, o de pré-processamento e o de difusão, precisavam se conectar com o modelo de IA. Foi criado um pequeno *socket* utilizando memória compartilhada para transmitir dados entre esses módulos e o modelo de IA.

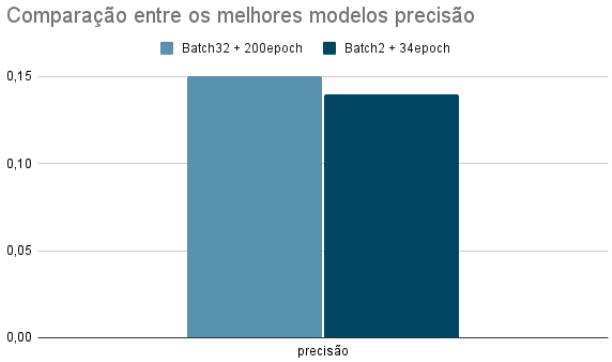



Figura 5.15 – Comparação da precisão entre os modelos que tiveram melhor desempenho

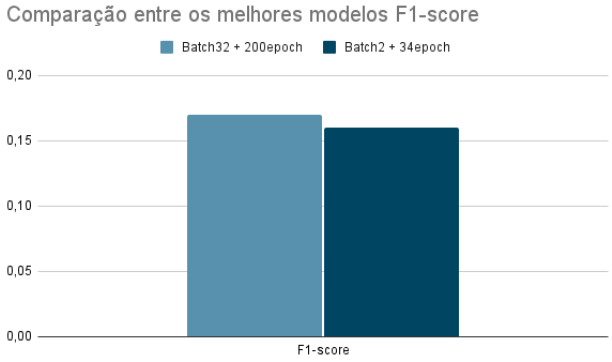



Figura 5.16 – Comparação de F1-score entre os modelos que tiveram melhor desempenho

Como pode ser visto nas Tabelas 5.1 e 5.2, os dois modelos treinados apresentaram desempenho comparável aos modelos já desenvolvidos para uso em tempo real, como o YOLOv7, que é uma abordagem amplamente reconhecida para detecção de objetos usando caixas delimitadoras.

A Tabela 5.1 mostra que a abordagem GAN é mais lenta em comparação com os modelos já utilizados pela empresa, como a YOLOv7, sendo mais de 6 vezes mais lenta. Isso sugere que a utilização de um modelo mais robusto pode oferecer resultados promissores quando há limitação de dados, mas também requer mais recursos computacionais para processar as imagens. No entanto, é importante observar que, quando não há restrições de recursos, a abordagem GAN se equipara ao desempenho do YOLOv7, sugerindo que nosso problema pode ser devido ao modelo possuir um grande número de parâmetros. Essa hipótese é confirmada na Tabela 5.2, que indica a necessidade de fazer trocas de memória *swap* devido à escassez de recursos.

Diante disso, decidimos testar o desempenho do modelo menor, denominado Tiny-GAN, nos mesmos ambientes. Foi observado que, com a redução da memória utilizada, o Tiny-GAN apresentou um desempenho muito melhor, aproximando-se do desempenho da Tiny-YOLO, que é o modelo mais rápido da Tabela, sendo apenas 60% mais lenta que

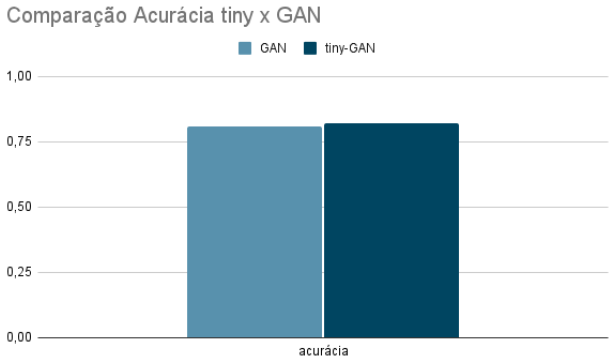



Figura 5.17 – Comparação da acurácia entre os modelos que tiveram melhor desempenho

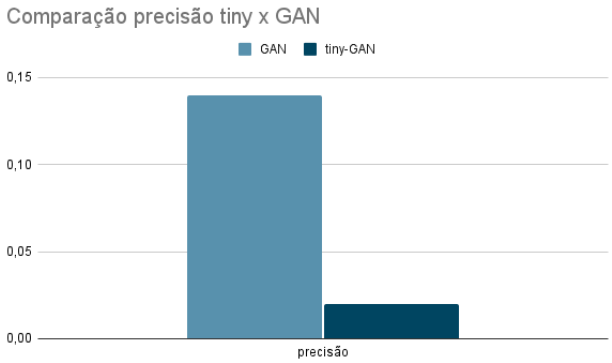



Figura 5.18 – Comparação da precisão entre os modelos que tiveram melhor desempenho

o Tiny-GAN.

Tabela 5.1 – Tabela comparativa de desempenho nas plataformas embarcadas.

| Developer Kit | AI | RTT |
|---|---|---|
| Jetson Nano | GAN | 55 sec |
| Jetson Nano | ResNet-50 | 38 sec |
| Jetson AGX Orin | GAN | 2.5 sec |
| Jetson AGX Orin | ResNet-50 | 26 sec |
| Jetson AGX Orin | MobileNet | 26 sec |
| Jetson AGX Orin | ResNet-50 | 6 sec |
| Jetson AGX Orin | MobileNet | 6 sec |
| Jetson AGX Orin | YOLOv7 | 2.5 sec |
| Jetson Nano | YOLOv7 | 9.5 sec |
| Jetson AGX Orin | Tiny-YOLO | 0.35 sec |
| Jetson Nano | Tiny-YOLO | 2 sec |
| Jetson Nano | Tiny-GAN | 5 sec |

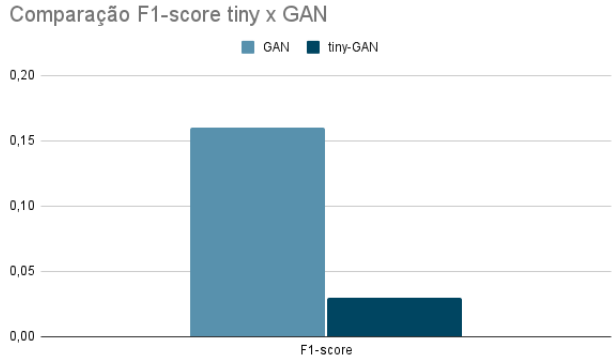



Figura 5.19 – Comparação de F1-score entre os modelos que tiveram melhor desempenho

Tabela 5.2 – Tabela comparativa de uso de memória nas plataformas embarcadas.

| Developer Kit | AI | RAM (no AI) | RAM (with AI) | Swap (with AI) |
|---|---|---|---|---|
| Jetson Nano | GAN | ? | 3.4 GiB | 1013 MiB |
| Jetson Nano | ResNet-50 | ? | ? | ? |
| Jetson AGX Orin | GAN | 4.8 GiB | 7.0 GiB | 0 |
| Jetson AGX Orin | ResNet-50 | 2.6 GiB | 3.3 GiB | 0 |
| Jetson AGX Orin | MobileNet | 2.6 GiB | 3.3 GiB | 0 |
| Jetson AGX Orin | ResNet-50 | 3.5 GiB | 5.4 GiB | 0 |
| Jetson AGX Orin | MobileNet | 3.4 GiB | 5.4 GiB | 0 |
| Jetson AGX Orin | YOLOv7 | 2.0 GiB | 4.1 GiB | 0 |
| Jetson Nano | YOLOv7 | 952 MiB | 3.5 GiB | 1.9 GiB |
| Jetson AGX Orin | Tiny-YOLO | 0 | 0 | 0 |
| Jetson Nano | Tiny-YOLO | ? | ? | ? |
| Jetson Nano | Tiny-GAN | ? | ? | ? |

# 6 CONCLUSÃO

Este trabalho empregou técnicas de GAN (Redes Generativas Adversariais) e segmentação semântica visando identificar a presença de ervas daninhas em plantações de soja. Além disso, buscou-se criar um modelo baseado no framework pix2pix que pudesse ser executado em dispositivos embarcados. Conforme mencionado em Isola et al. (2017) [1], o modelo desenvolvido por eles demonstrou grande eficiência na identificação e recriação de estruturas. Entretanto, ficou claro que a obtenção de um conjunto de dados equilibrado é essencial para garantir o desempenho adequado do modelo.

Conforme evidenciado no capítulo 5.2, os modelos treinados alcançaram uma alta acurácia, mesmo quando fornecidos com um número limitado de imagens de treinamento. Isso sugere que o modelo teve um desempenho superior ao lidar com instâncias mais frequentes no conjunto de dados do que as que não eram, como, por exemplo, folha estreita que só tinha 7% de ocorrência no *dataset*. Também é notável que a técnica desenvolvida pelos autores do pix2pix foi capaz de produzir resultados satisfatórios com poucas iterações, embora os resultados de precisão por classe não fossem tão impressionantes.

Considerando a composição do conjunto de dados, onde mais de 61% das instâncias representam situações em que a aplicação de pesticidas não é necessária, as figuras de 5.1 a 5.5 sugerem que, em sua maioria, o modelo opera como um classificador binário. Ele tenta identificar situações em que há presença de soja ou solo vazio, mas enfrenta dificuldades na classificação precisa das situações de dispersão do produto. Em vez disso, o modelo consegue apenas reconhecer a presença da dispersão, sem uma classificação refinada das situações específicas. Considerando essa situação, o modelo ainda pode trazer um certo beneficio em seu uso, pois acabaria conseguindo reduzir o uso de produtos agrícolas no plantio.

Ao analisarmos a adaptação do modelo para uso em sistemas embarcados, observamos que a acurácia se manteve comparável ao modelo original, mas houve uma clara degradação na precisão e no F1-score. Essa degradação sugere que o modelo reduzido pode não ser capaz de reconhecer eficazmente situações raras ou pouco representadas no conjunto de dados. No entanto, é importante notar que o modelo Tiny-GAN demonstrou um desempenho considerável em termos de tempo de processamento, aproximando-se do desempenho do modelo Tiny-YOLO. Nesse contexto, podemos especular sobre como o modelo se sairia com um conjunto de dados maior para treinamento. Assim, a escolha entre o modelo completo e o modelo reduzido pode depender da disponibilidade de dados

e dos requisitos de tempo de processamento.

## 6.1 Trabalhos futoros

Algumas extensões possíveis para este trabalho incluem a aplicação de técnicas de *transfer learning*. Poderíamos treinar um modelo inicial com alta precisão e, em seguida, transferir o conhecimento de segmentação aprendido por ele para um novo modelo, visando aprimorar a segmentação de plantas. Outra direção interessante seria explorar a segmentação de instâncias, que representa uma evolução da segmentação semântica. Nesse contexto, seria possível estimar não apenas a presença de objetos, mas também a quantidade de instâncias individuais, o que poderia ser valioso em várias aplicações práticas.

Essas extensões podem contribuir para melhorar ainda mais o desempenho e a versatilidade do modelo, tornando-o mais eficaz em cenários onde a precisão e a capacidade de reconhecer situações raras são críticas.

# REFERÊNCIAS

ANTHONIRAJ, S.; KARTHIKEYAN, P.; VIVEK, V. Weed detection model using the generative adversarial network and deep convolutional neural network. **Journal of Mobile Multimedia**, p. 275–292, 2022.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006.

CORDTS, M. et al. The cityscapes dataset for semantic urban scene understanding. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 3213–3223.

CRESWELL, A. et al. Generative adversarial networks: An overview. **IEEE signal processing magazine**, IEEE, v. 35, n. 1, p. 53–65, 2018.

EKMAN, M. **Learning Deep Learning: Theory and Practice of Neural Networks, Computer Vision, Natural Language Processing, and Transformers Using TensorFlow**. [S.l.]: Addison-Wesley Professional, 2021.

ENGELEN, J. E. V.; HOOS, H. H. A survey on semi-supervised learning. **Machine learning**, Springer, v. 109, n. 2, p. 373–440, 2020.

GANOKRATANAA, T.; ARAMVITH, S.; SEBE, N. Unsupervised anomaly detection and localization based on deep spatiotemporal translation network. **IEEE Access**, PP, p. 1–1, 03 2020.

GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric Environment**, v. 32, n. 14, p. 2627–2636, 1998. ISSN 1352-2310. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1352231097004470>.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais**. [S.l.]: Editora Blucher, 2000.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

IBERDROLA. **Ervas daninhas: quais são e porque não são tão terríveis — iberdrola.com**. 2023. <https://www.iberdrola.com/sustentabilidade/ervas-daninhas>. [Accessed 10-09-2023].

ISOLA, P. et al. Image-to-image translation with conditional adversarial networks. In: **Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on**. [S.l.: s.n.], 2017.

JAIN, D. **Superpixels and SLIC — darshita1405.medium.com**. 2019. <https://darshita1405.medium.com/superpixels-and-slic-6b2d8a6e4f08>. [Accessed 07-09-2023].

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. **CoRR**, abs/1411.4038, 2014. Disponível em: <http://arxiv.org/abs/1411.4038>.

OLSEN, A. et al. Deepweeds: A multiclass weed species image dataset for deep learning. **Scientific reports**, Nature Publishing Group UK London, v. 9, n. 1, p. 2058, 2019.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. **arXiv preprint arXiv:1511.08458**, 2015.

RONNEBERGER, O.; FISCHER, P.; BROX, T. **U-Net: Convolutional Networks for Biomedical Image Segmentation**. 2015.

RUSSELL, S. J. **Artificial intelligence a modern approach**. [S.l.]: Pearson Education, Inc., 2010.

SZELISKI, R. **Computer vision: algorithms and applications**. [S.l.]: Springer Nature, 2022.

TECNOLOGIA, c. e. m. Maycol de A. Diferença entre computação gráfica e processamento de imagens/Differences between graphic computer and image processing — maalencar.wordpress.com. 2012/03/22. <https://maalencar.wordpress.com/2012/03/22/differences-between-graphic-computer-and-image-processing/>. [Accessed 10-09-2023].

TYLECEK, R.; SARA, R. Spatial pattern templates for recognition of objects with regular structure. In: . [S.l.: s.n.], 2013. v. 8142. ISBN 978-3-642-40601-0.

YU, A.; GRAUMAN, K. Fine-grained visual comparisons with local learning. In: **2014 IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 192–199.

ZHANG, J. et al. Segmenting purple rapeseed leaves in the field from uav rgb imagery using deep learning as an auxiliary means for nitrogen stress detection. **Remote Sensing**, v. 12, p. 1403, 04 2020.

ZHU, J.-Y. et al. **Generative Visual Manipulation on the Natural Image Manifold**. 2018.