UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

CARLOS ROBERTO MODINEZ JUNIOR

MARCOS VINICIUS MAZEPA FRIDRISCVSKI

METODOLOGIA DE CLASSIFICAÇÃO E SEGMENTAÇÃO SEMÂNTICA DE PLANTAS DANINHAS COM AUXÍLIO DE CONJUNTOS DE DADOS SINTÉTICOS

CURITIBA

2024

CARLOS ROBERTO MODINEZ JUNIOR
MARCOS VINICIUS MAZEPA FRIDRISCVSKI

METODOLOGIA DE CLASSIFICAÇÃO E SEGMENTAÇÃO SEMÂNTICA DE PLANTAS DANINHAS COM AUXÍLIO DE CONJUNTOS DE DADOS SINTÉTICOS

Methodology for Classification and Semantic Segmentation of Weeds with the Aid of Synthetic Datasets

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônca da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. André Eugenio Lazzaretti

Coorientador: Prof. Anderson Brilhador

CURITIBA
2024

CARLOS ROBERTO MODINEZ JUNIOR
MARCOS VINICIUS MAZEPA FRIDRISCVSKI

METODOLOGIA DE CLASSIFICAÇÃO E SEGMENTAÇÃO SEMÂNTICA DE PLANTAS DANINHAS COM AUXÍLIO DE CONJUNTOS DE DADOS SINTÉTICOS

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Eletrônica do Curso de Bacharelado em Engenharia Eletrônca da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 25/junho/2024

André Eugenio Lazzaretti
Doutor em Engenharia Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná

Anderson Brilhador
Mestre profissional em Programa de Pós-Graduação em Informática - PPGI
Universidade Tecnológica Federal do Paraná

Gustavo Benvenutti Borba
Doutor em Engenharia Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná

Rafael Eleodoro de Góes
Doutor em Engenharia Elétrica e Informática Industrial
Universidade Tecnológica Federal do Paraná

CURITIBA

2024

RESUMO

O setor agrícola desempenha um papel crucial na segurança alimentar e no crescimento econômico global, exigindo constantemente avanços tecnológicos para otimizar o processo de cultivo. Um desafio significativo enfrentado nas lavouras é a presença de plantas daninhas, que competem com as culturas por recursos vitais, como nutrientes, água e luz, podendo causar sérios prejuízos à colheita se não forem identificadas e tratadas rapidamente. Neste contexto, este trabalho propõe o desenvolvimento de um algoritmo de Deep Learning com o objetivo de identificar e classificar as principais espécies de plantas daninhas encontradas no oeste paranaense. Para alcançar esse objetivo, serão empregadas técnicas avançadas de visão computacional e processamento de imagens, como segmentação semântica, morfologia matemática, otimizadores e funções de custo. Para a tarefa de classificação, utilizou-se um conjunto de dados contendo as principais plantas daninhas presentes no oeste paranaense, alcançando uma precisão de 0.98. Uma abordagem inovadora deste projeto consiste na criação de conjuntos de dados sintéticos a partir da modelagem de plantas 3D como metodologia para data augmentation, fundamentais para um treinamento mais robusto e eficiente do algoritmo, permitindo que ele reconheça uma ampla variedade de cenários e condições encontradas nas lavouras brasileira, mesmo havendo poucas imagens que representem a realidade. Esses conjuntos de dados sintéticos serão utilizados na tarefa de segmentação semântica, substituindo a etapa de aquisição de imagens de infestação de daninhas no meio à cultura de milho. A métrica de avaliação escolhida para essa tarefa foi a MIoU (Mean Intersection over Union), que avalia o acerto do modelo a nível de pixel. O modelo apresentou um resultado de 0.7698, demonstrando a viabilidade do projeto.

Palavras-chave: visao computacional; deep learning; plantas daninhas; aprendizado de maquina.

ABSTRACT

The agricultural sector plays a crucial role in food security and global economic growth, constantly requiring technological advancements to optimize the cultivation process. A significant challenge faced in crop fields is the presence of weeds, which compete with crops for vital resources such as nutrients, water, and light, potentially causing serious damage to the harvest if not identified and treated promptly. In this context, this work proposes the development of a Deep Learning algorithm aimed at identifying and classifying the main species of weeds found in western Paraná. To achieve this goal, advanced techniques in computer vision and image processing will be employed, such as semantic segmentation, mathematical morphology, optimizers, and cost functions. For the classification task, a dataset containing the main weeds present in western Paraná was used, achieving an accuracy of 0.98. An innovative approach of this project is the creation of synthetic datasets, which are fundamental for a more robust and efficient training of the algorithm, allowing it to recognize a wide variety of scenarios and conditions found in Brazilian crop fields. These synthetic datasets will be used in the task of semantic segmentation, replacing the step of acquiring images of weed infestations in maize crops. The chosen evaluation metric for this task was the MIoU (Mean Intersection over Union), which measures the model's accuracy at the pixel level. The model achieved a result of 0.7698.

Keywords: computer vision; agriculture; segmentation; deep learning; cnn.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

De acordo com Alan Bojanic, representante da FAO (Organização das Nações Unidas para Alimentação e Agricultura), o Brasil desempenha um papel significativo no contexto global de segurança alimentar. A demanda por alimentos está em constante crescimento no mundo, de forma que, projeções apresentadas durante o 31° Congresso Nacional de Milho e Sorgo, estimou-se que até 2050, a população mundial deverá se estabilizar em torno de 9,3 bilhões de habitantes, trazendo assim, uma necessidade no aumento da produção mundial de alimentos em até 70%, de forma a suprir a necessidade desta população (BOJANIC, 2016).

A produtividade na agricultura brasiliera vem destacando-se em relação aos outros países da América do Sul. Segundo dados do reporte (ARIAS *et al.*, 2017), nas últimas décadas, o Brasil quadruplicou sua produtividade no campo. No entanto, para que a produtividade continue avançando de maneira sustentável, é necessário repensar os modelos de inovações agrícolas, trazendo mais produtividade para não somente os grande produtores mas sim as fazendas menos produtivas. Isso exige pesquisa contínua e otimização dos diversos problemas encontrados na colheita (ARIAS *et al.*, 2017).

Um dos grandes fatores que interferem negativamente na produção de commodities, são os prejuízos causados por plantas daninhas nas lavouras (VASCONCELOS; SILVA; LIMA, 2012). As plantas daninhas são plantas que surgem espontaneamente nas lavouras, competindo por luz, água e nutrientes, estabelecendo um processo competitivo entre os commodities e as daninhas, causando prejuízos significativos à colheita conforme pode ser observado na figura 1. Alem dessa competição, as plantas daninhas podem atuar como hospedeiras de pragas e doenças, reduzir a bioversidade, dificultar o manejo de água no agroecossitema, além dos efeitos prejudiciais causados pelos métodos de controle necessários (VASCONCELOS; SILVA; LIMA, 2012). O conhecimento das infestações é um procedimento fundamental para a implementação de medidas preventivas no controle das ervas daninhas onde destaca-se a importância de métodos que possibilitem a quantificação e análise rápida da distribuição da infestação de ervas daninhas (RIZZARDI; FLECK, 2004).

Figura 1 – Foto infestação de plantas daninhas em lavoura de soja

Fonte: (Rural Press,).

Dessa forma, uma área que vem apresentando destaque no controle das lavouras é a agricultura digital, particulamente o campo da Visão Computacional (SANTOS *et al.*, 2020). Este ramo da Inteligência Artificial se dedica á extração de informações de imagens digitais capturadas no nível de solo ou seja via drone. Aplicações com essa tecnologia conseguem extrair informações das imagens de diversos fatores presentes na lavoura, como detecção de doenças e pragas, estimativa de safra e qualificação de cultivares (SANTOS *et al.*, 2020). Essa é uma aréa abrangente e que vem apresentando fortes investimentos por parte do mercado internacional. Um controle efetivo das pragas nas lavouras pode elevar a produtividade e abir espaços para novas tecnologias que visam ajudar no manejo dessas pragas (DUCKETT *et al.*, 2018).

Nos últimos anos, diversas abordagens foram desenvolvidas para a detecção e manejo de plantas daninhas utilizando tecnologias variadas. Por exemplo, Viliotti (2002) propôs um sistema eletrônico com sensores ópticos para correlacionar a radiação global incidente com a radiação refletida de superfícies cobertas por plantas daninhas. Embora tenha alcançado resultados relevantes, essa abordagem ainda requer muita manutenção e apresenta alto custo. Abordagens mais recentes utilizam redes neurais para a classificação e identificação de plantas daninhas. O trabalho de (BELETE *et al.*, 2019), por exemplo, empregou algoritmos que identificam padrões nas imagens, enquanto (MILIOTO; LOTTES; STACHNISS, 2018) propuseram uma rede neural capaz de detectar plantas daninhas em tempo real.

Nossa metodologia busca integrar o melhor das técnicas de ponta, utilizando redes neurais convolucionais (CNNs) para a correta identificação das diversas espécies de plantas daninhas no Paraná, além de realizar a segmentação semântica para detectar plantas daninhas dispersas nas culturas de soja. Para auxiliar nesse projeto, propomos uma abordagem inovadora para alimentar o conjunto de dados de treinamento do modelo, criando datasets sintéticos. Isso permite abranger um volume maior de dados e trazer mais variedade para o aprendizado, tornando o modelo mais robusto e eficaz na detecção de plantas daninhas.

1.1 Objetivos

Nessa seção, serão abordados os principais objetivos a serem alcançados com este trabalho.

1.1.1 Objetivo geral

Elaborar uma metodologia fundamentada em visão computacional para a detecção de plantas daninhas em meio à cultura de milho.

1.1.2 Objetivos específicos

- Preparar um conjunto de dados para pipleline de classificação composto por um dataset com imagens reais de plantas daninhas.
- Desenvolver um modelo de classificação capaz de identificar as espécies de ervas daninha, utilizando técnicas de aprendizado de máquina.
- Criar um conjunto de dados sintéticos por meio de técnicas de manipulação de imagem, complementando o conjunto de dados real para enriquecer o treinamento do modelo.
- Gerar um conjunto de dados sintéticos utilizando métodos de modelagem e simulação 3D, ampliando a representatividade dos dados utilizados no treinamento.
- Implementar um modelo de segmentação semântica para a detecção precisa de plantações de milho e plantas daninhas em imagens.
- Apresentar uma metodologia robusta para calcular as métricas de desempenho do modelo.

1.2 Justificativa

Conforme afirmado em Ayaz, Aytekin e Akgün (2019), a robótica de campo deve capacitar o desenvolvimento de novos equipamentos agrícolas com o propósito de reduzir o desperdício e o impacto ambiental, ao mesmo tempo em que mantém a viabilidade econômica. Isso resultaria em um aumento da sustentabilidade na produção de alimentos.

Desta maneira, este trabalho visa estudar os algoritmos de deep learning para a detecção autônoma de plantas daninhas, dando os primeiros passos em direção às tecnologias de controle e manejo dessas infestações. Algumas startups e empresas já exploram a remoção dessas plantas por meio de lasers ou pulverização guiada, reduzindo assim o impacto ambiental causado pelos agrotóxicos. Um exemplo notável é a startup americana Carbon Robotics, que utiliza lasers para eliminar plantas daninhas detectadas por imagens e com precisões milimétricas. Sua tecnologia, denominada LaserWeeder™, identifica plantas invasoras e as elimina por meio de lasers, reduzindo a necessidade de herbicidas e mão de obra intensiva (Carbon Robotics, 2023). Outro exemplo é o See Spray™ Ultimate da John Deere, que utiliza visão computacional e aprendizado de máquina para direcionar a pulverização apenas para as plantas daninhas em culturas de milho, soja e algodão, permitindo a aplicação precisa de herbicidas e o uso de misturas de tanques mais avançadas graças à sua configuração de tanque duplo (John Deere, 2023). A correta identificação e distinção entre pragas e culturas são fundamentais para o sucesso dessas tecnologias, garantindo eficiência e sustentabilidade no manejo agrícola.

Desta maneira, este projeto inicialmente será capaz de classificar diferentes espécies de plantas daninhas. Na etapa seguinte, será construída a pipeline de segmentação semântica para classificar diferentes plantas daninhas no meio da cultura de milho em uma mesma imagem, além de extrair as informações de posição das plantas invasoras.

Além disso, este estudo investigará o uso de conjuntos de dados sintéticos no treinamento dos modelos de aprendizado de máquina. A aquisição de imagens é uma parte crucial do treinamento dos algoritmos de inteligência artificial e, muitas vezes, é custosa. Considerando os devidos aspectos de iluminação e distribuição das plantas, a criação de conjuntos de dados sintéticos pode auxiliar, agilizar, ou até mesmo substituir por completo a a necessidade da criação de conjunto de dados com imagens 100% reais, além de trazer mais variedade para os dados utilizados.

1.3 Estrutura do Trabalho

Este trabalho está organizado em três partes principais, a primeira parte destina-se a explicar os conceitos fundamentais necessários para a compreensão completa do trabalho. Aqui, abordamos as noções de aprendizado de máquina e *deep learning*, conceitos que formam a base teórica deste estudo.

Na segunda parte, discutimos detalhadamente as metodologias aplicadas. Primeiro, apresentamos a pipeline de classificação, onde explicamos como diferentes espécies de plantas daninhas são identificadas. Em seguida, exploramos a pipeline de segmentação semântica, que não só classifica múltiplas plantas daninhas em uma única imagem, mas também extrai informações sobre a localização exata dessas plantas invasoras. Além disso, destacamos a inovação deste trabalho: a utilização de conjuntos de dados sintéticos para o treinamento dos modelos de aprendizado de máquina.

Por fim, na terceira parte, apresentamos os resultados obtidos e as considerações para projetos futuros. Esta seção é de grande importância, pois não só demonstra a eficácia da metodologia proposta por meio de métricas e avaliações detalhadas, mas também oferece insights sobre possíveis melhorias e direções para futuras pesquisas. Discutimos como os resultados contribuem para o avanço da tecnologia de detecção de plantas daninhas e sugerimos novas abordagens e aplicações que podem ser exploradas a partir deste estudo.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os principais conceitos que embasam o estudo, proporcionando uma compreensão aprofundada do tema e contextualizando a relevância dos objetivos propostos.

2.1 Visão Computacional

Visão computacional é a área da inteligência Artificial (IA) que tem como foco emular a visão humana para resolver problemas complexos, tendo a capacidade de tomar decisões com base nas informações extraidas de imagens digitais (GONZALEZ; WOODS, 2008). A área tem caráter multidisciplinar, englobando diversas áreas do conhecimento, possibilitando a aplicação da tecnologia em uma gama de setores. Os problemas envolvem, por exemplo análise de imagens, reconhecimento de padrões e controle inteligente.

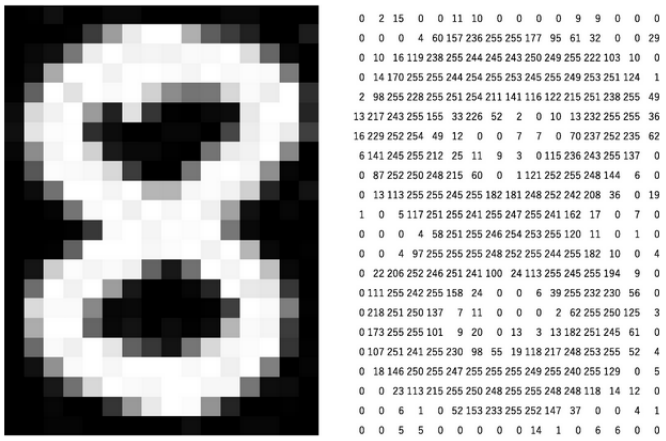
2.2 Fundamentos de imagens digitais

Uma imagem digital pode ser conceituada como uma representação digital da percepção visual humana. Ela é expressa através de uma função f(x,y), a qual busca capturar e retratar as informações de um ambiente visual. Podemos equipará-la ao processo pelo qual o olho humano interpreta e processa o mundo ao seu redor, convertendo a luz em padrões de informação que o cérebro interpreta (GONZALEZ; WOODS, 2008). Nas imagens digitais, as coordenadas x e y correspondem a posições espaciais em um plano, enquanto o valor de f em cada ponto (x,y) codifica a intensidade da luz. Estes pontos gerados por f(x,y) são denominados *pixels*, sendo a menor unidade para representar uma imagem. Os *pixels*, distribuídos em um plano, possuem valores únicos de intensidade e localização, os quais, juntos, formam uma matriz de elementos constituintes de uma imagem digital.

A função f(x,y), que descreve uma imagem digital, pode variar em complexidade, desde simples representações binárias em preto e branco até representações coloridas de alta resolução. Cada pixel, portanto, carrega não apenas informações sobre intensidade de luz, mas também pode conter informações sobre cor, textura e até mesmo profundidade, dependendo do tipo de imagem e do contexto em que é utilizada.

O nível de intensidade de um pixel varia entre 0 e 255 em uma imagem digital de 8 bits, resultando em tons de cinza e apenas uma camada de profundidade. No entanto, podemos empregar outros modelos para representar imagens, como o modelo RGB (*Red, Green e Blue*), que consiste em *pixels* de 24 bits distribuídos em três camadas distintas [(R, G, B)]. Cada camada possui *pixels* de 8 bits, representando a intensidade do pixel para cada cor. A com-

Figura 2 – Estrutura de uma Imagem Digital



Fonte: (Analytics Vidhya,).

binação dessas três camadas é capaz de reproduzir imagens digitais coloridas (BRILHADOR; SERRARENS; LOPES, 2015).

Dessa maneira, é possível utilizar múltiplas camadas para adicionar mais detalhes aos modelos de representação de imagens, onde cada pixel representa valores únicos que descrevem sua intensidade em uma matriz.

2.3 Processamento de imagens digitais

O processamento de imagens digitais é definido como qualquer processo que envolva a entrada e a saída de uma imagem digital (GONZALEZ; WOODS, 2008). Este campo abrange uma ampla variedade de operações, desde correção de exposição e balanço de cores até redução de ruído, aumento de nitidez e retificação da imagem através de rotação (SZELISKI, 2022). Tais operações são essenciais para a extração e identificação de informações contidas nas imagens, melhoria da qualidade visual e realce de características específicas que facilitam a compreensão tanto por humanos quanto por computadores (PEDRINI; SCHWARTZ, 2008).

2.3.1 Operações Espaciais

As operações no domínio do espaço são caracterizadas pela manipulação direta dos *pixels* da imagem e podem ser subdivididas conforme o escopo de ação, como pontuais ou locais (QUEIROZ; GOMES, 2001). As operações pontuais, também conhecidas como operadores de ponto, são as mais simples transformações de processamento de imagem, onde o valor de cada *pixel* de saída depende apenas do *pixel* correspondente na imagem de entrada. Exemplos incluem ajustes de brilho e contraste, bem como correção de cores e transformações (SZELISKI, 2022). Essas operações podem ser expressas matematicamente como $g(x) = h(f(x))$, onde

$g(x)$ é a imagem de saída, $f(x)$ é a imagem de entrada e h é a função de transformação aplicada (SZELISKI, 2022).

Por outro lado, nas operações locais, o valor de saída em uma coordenada específica depende dos valores de entrada de um grupo de *pixels*, ou seja, do *pixel* em questão e seus vizinhos. Tais operações são importantes para tarefas que envolvem suavização, nitidez e filtragem de bordas. Exemplos incluem a aplicação de filtros lineares e não lineares, como o filtro bilateral e métodos de difusão anisotrópica (SZELISKI, 2022). Essa abordagem permite que a operação considere a informação do entorno do *pixel*, proporcionando resultados que refletem a relação entre *pixels* adjacentes e melhorando a qualidade e a compreensão da imagem processada.

2.3.2 Operações com Histograma

O histograma de uma imagem representa a distribuição estatística dos *pixels* conforme a intensidade que eles possuem. Trata-se de uma representação gráfica que mostra o número de *pixels* associados a cada nível de intensidade em uma imagem (SZELISKI, 2022).

Diversas operações podem ser realizadas sobre o histograma, como a equalização do histograma. Esta técnica visa melhorar o contraste da imagem ao redistribuir os *pixels* entre os níveis de intensidade, resultando em uma distribuição mais uniforme.

2.3.3 Morfologia Matemática

A morfologia digital ou matemática (QUEIROZ; GOMES, 2001) parte do princípio de que a imagem é um conjunto de pontos elementares que formam subconjuntos bi ou tridimensionais. Com isso, é possível executar operações de morfologia utilizando conceitos de álgebra booleana e teoria dos conjuntos. As operações básicas da morfologia matemática são:

- Erosão: Remoção de *pixels* que não atendem a um padrão específico. Esta operação encolhe os objetos na imagem, removendo pequenas irregularidades e detalhes.
- Dilatação: Alteração de uma pequena área ao redor de um pixel para um padrão específico. Esta operação expande os objetos na imagem, preenchendo lacunas e conectando componentes próximos.

2.3.4 Índices de Vegetação

Os índices de vegetação em imagens digitais são métricas calculadas a partir da transformação de imagens no espaço de cores RGB, resultando em imagens em tons de cinza que destacam os elementos de vegetação. Para isolar os *pixels* que representam vegetação dos demais, é necessário aplicar um limiar ajustável, como o método de Otsu (BRILHADOR; SER-

RARENS; LOPES, 2015), representado na Tabela 1. Nesta aplicação específica, ao se coletar imagens de vegetação como nicho de processamento, esse método facilita a identificação das áreas de interesse e a aplicação do limiar ajustável.

Tabela 1 – Principais índices vegetativos encontrados na literatura

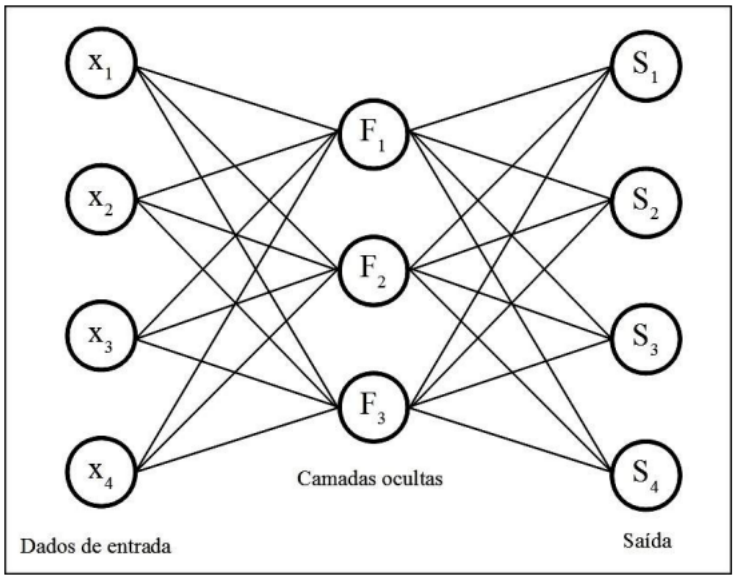
Nome	Fórmula	Observação
Verde excessivo (ExG)	$ExG = 2g - r - b$	Moorthy (2015) estudou diversos índices de vegetação com o objetivo de distinguir as plantas do solo. O índice de verde excessivo apresentou desempenho satisfatório em várias condições de iluminação solar. Alguns anos depois, Ribeiro et al. (2005) realizou algumas modificações no índice, baseando-se em estudos feitos no espaço de cores CYMK (Cyan-Magenta-Yellow-Black).
Vermelho excessivo (ExR)	$ExR = 1.4r - g$	O índice de vermelho excessivo foi descrito em Meyer, Hindman e Laksmi (1999). O autor também testou o índice verde excessivo, porém, os resultados foram inferiores ao índice proposto.
Índice de diferença normalizada (NDI)	$NDI = \frac{(g-r)}{(g+r)}$	O índice de diferença normalizada foi proposto por Woebbecke *et al.* (1993). O índice é semelhante à infravermelho, utilizando a luz vermelha de reflexão para separar as plantas do solo. Recentemente, o índice foi utilizado em (PEREZ et al., 2000), para separar ervas daninhas e plantas de cereais do solo.
Índice de cor de vegetação (CIVE)	$CIVE = 0.441r - 0.811g + 0.385b + 18.78745$	O método foi utilizado para distinguir plantas de soja Kataoka *et al.* (2003).
Verde excessivo menos vermelho excessivo (ExGR)	$ExGR = ExG - ExR$	Em Neto (2004) o autor combinou dois índices de vegetação, verde excessivo e vermelho excessivo, criando o índice denominado de Verde excessivo menos vermelho Excessivo (ExG - ExR). O índice proposto apresentou resultados superiores ao índice de verde excessivo e o índice de diferença normalizada.
Índice vegetativo (VEG)	$VEG = \frac{g}{(r^a b^{1-a})}, a = 0.667$	Esse índice foi proposto por Hague, Tillett e Wheeler (2006). Duas vantagens do método são destacadas a insensibilidade em relação à amplitude de iluminação e a invariância em relação à experimentos fora do campo. No entanto, o trabalho não indica como ajustar a constante a.
Índices combinados (COM)	$COM = 0.25ExG + 0.30ExGR + 0.33CIVE + 0.12VEG$	Guijarro *et al.* (2011) propôs a combinação de vários índices de vegetação afim de aumentar o contraste entre as plantas e o solo, melhorando o resultado da segmentação. Após experimentos foram definidos quatro índices mais relevantes, que foram atribuídos pesos a cada um deles conforme a taxa de erro de cada segmentação. O CIVE foi considerado o método mais relevante.

Fonte: (BRILHADOR; SERRARENS; LOPES, 2015).

2.4 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são modelos computacionais inspirados no funcionamento do cérebro humano, projetados para aprender a partir de dados. Elas são compostas por unidades chamadas neurônios artificiais, organizados em camadas e conectados entre si por meio de pesos (GOODFELLOW; BENGIO; COURVILLE, 2016). A arquitetura básica de uma RNA consiste em uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio em uma camada está conectado a todos os neurônios da camada seguinte, e cada conexão possui um peso que determina a influência de um neurônio sobre outro. A figura 3 representa a estrutura de uma RNA composta por três camadas principais: A camada de entrada, representada pelas letras X_i, a camada oculta, representada pelas letras F_i e, por fim, a camada de saída, representada pelas letras S_i.

Figura 3 – Estrutura de uma RNA



Fonte: (PEDRINI; SCHWARTZ, 2008).

Cada neurônio recebe um conjunto de entradas x_i, cada uma associadas a pesos correspondentes w_i, com a adição de um bias b. A saída z é o resultado obtido do somatório da operação exercida sobre todas as entradas, conforme descrita na equação 1:

$$z = \sum_{i=1}^{n} w_i x_i + b. \tag{1}$$

A saída do neurônio é obtida aplicando uma função de ativação sobre a saída z. Ela funciona como um “interruptor“ que pode ativar ou desativar a atividade dos neurônios, destacando a importância das entradas que tem maior influência no resultado final e permitindo que a rede aprenda a reconhecer padrões complexos nos dados. Existem diversas funções de ativação co-

munete usadas, como a sigmoide, a tangente hiperbólica (tanh), a ReLU (*Rectified Linear Unit*), entre outras.

No aprendizado supervisionado, a rede é treinada fornecendo os valores de entrada e comparando a saída com os valores reais conhecidos, chamados de *ground truth*. Esse processo envolve ajustar iterativamente os pesos das conexões da rede para minimizar o erro entre as previsões da rede e os valores reais. Os dados de entrada são processados à medida que se propagam pela rede, em um processo conhecido como *forward propagation*. Durante o treinamento, um algoritmo como o *backpropagation* é utilizado para calcular o erro e ajustar os pesos de forma a melhorar a precisão da rede. Esse método permite que as RNAs aprendam a partir de dados complexos e realizem diversas tarefas, incluindo classificação, regressão, clustering e reconhecimento de padrões, sendo amplamente utilizadas em áreas como visão computacional, processamento de linguagem natural e reconhecimento de fala.

2.5 Backpropagation

O *backpropagation* é um algoritmo utilizado para treinar redes neurais artificiais, permitindo que elas aprendam a partir dos dados. Seu objetivo é ajustar os pesos da rede de forma que a diferença entre as saídas previstas e os valores reais (o erro) seja minimizada (GOODFELLOW; BENGIO; COURVILLE, 2016).

Na propagação para trás, o erro é propagado de volta através da rede, da camada de saída para a camada de entrada. O objetivo é calcular quanto cada peso contribuiu para o erro da saída. Isso é feito através do cálculo do gradiente da função de perda em relação aos pesos da rede, usando a regra da cadeia da derivada.

Uma vez que o gradiente é conhecido, os pesos da rede são atualizados usando um algoritmo de otimização, como o gradiente descendente. Esse processo é repetido para muitos exemplos de treinamento, ajustando gradualmente os pesos da rede para minimizar o erro.

Neste trabalho, serão abordadas duas das principais funções de custo utilizadas em problemas de classificação e segmentação. Essas funções possuem parametrizações que permitem ajustes para lidar com problemas de classes desbalanceadas.

2.5.1 Cross Entropy Loss

A *Cross Entropy* (MAO; MOHRI; ZHONG, 2023) é uma função de perda amplamente utilizada em tarefas de classificação. Ela mede a discrepância entre a a probabilidade predita pelo modelo $\mathbf{p}$ com a distribuição de probabilidade verdadeira y. Ela penaliza fortemente previsões que atribuem baixa probabilidade à classe verdadeira. Como resultado, a minimização da *Cross Entropy* durante o treinamento faz com que o modelo preveja distribuições de probabilidade que se aproximam das distribuições verdadeiras. Para uma classificação multiclasse com C classes,

a Cross Entropy é estendida da seguinte forma:

$$\text{CE}(p) = -\sum_{i=1}^{C} y_i \log(p_i). \tag{2}$$

De forma que y_i é um valor binário indicando se a classe i é a classe verdadeira (1 para verdadeiro, 0 para falso). p_i é a probabilidade predita pelo modelo para a classe i.

Em muitos casos de classificação, especialmente quando lidamos com classes desbalanceadas, a utilização da *Cross Entropy* padrão pode não ser ideal. Nestes cenários, a *Weighted Cross Entropy* é uma variação que atribui diferentes pesos às classes, permitindo que o modelo penalize a as classes menos frequentes e se ajuste melhor ao desequilíbrio dos dados:

$$\text{WCE}(p) = -\sum_{i=1}^{C} w_i y_i \log(p_i). \tag{3}$$

2.5.2 Focal Loss

A *Focal Loss* (LIN *et al.*, 2018) é uma função de perda desenvolvida para abordar o problema de desequilíbrio de classes, comum em tarefas de detecção de objetos e segmentação semântica. Nesse contexto, classes como *background* são muito mais representadas do que classes menores, que podem estar sub-representadas. Para mitigar esse desequilíbrio, a *Focal Loss* introduz o termo $(1 - p_t)^\gamma$, que atua como um peso. Esse termo diminui o impacto dos exemplos bem classificados (p_t próximo de 1) e aumenta o impacto dos exemplos mal classificados (p_t próximo de 0). Esse ajuste dinâmico faz com que a rede foque mais nos exemplos difíceis durante o treinamento.

A *Focal Loss* é definida da seguinte forma para uma única amostra:

$$\text{FL}(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t), \tag{4}$$

sendo p_t é a probabilidade predita pelo modelo para a classe verdadeira. Se a classe verdadeira é $y = 1$, então $p_t = p$; se $y = 0$, então $p_t = 1 - p$. Os termos α_t e γ são parâmetros ajustáveis, sendo α_t um fator de ponderação para lidar com o desequilíbrio de classes, que pode ser definido de maneira diferente para cada classe e γ um fator de modulação que reduz o peso dos exemplos bem classificados, concentrando o treinamento nos exemplos que se encontra uma maior dificuldade, muitas vezes aqueles que apresenta menos amostras. Valores comuns de γ variam entre 0 e 5.

2.6 Deep Learning

Deep learning permite que modelos computacionais com múltiplas camadas aprendam representações atravás dos dados em diferentes níveis de abstração, buscando identificar padrões significativos (LECUN; BENGIO; HINTON, 2015). No processamento de imagens digitais, o reconhecimento de padrões refere-se à rotulação de objetos com base em seus descritores, ou *features*, que são extraídos dos dados de entrada de forma a caracterizar os objetos e distinguir uns dos outros da maneira mais eficaz eficaz (GONZALEZ; WOODS, 2008).

2.6.1 Classificação

A tarefa de classificação de imagens é um dos problemas fundamentais em visão computacional, que consiste em atribuir rótulos ou classes às imagens com base em suas características visuais (HE *et al.*, 2016). Esse processo é realizado por modelos de aprendizado de máquina, como redes neurais convolucionais (CNNs), que aprendem a mapear imagens de entrada para classes de saída durante o treinamento.

Para realizar a classificação, um modelo de aprendizado de máquina, como uma rede neural convolucional (CNN), é treinado em um conjunto de dados rotulados. Durante o treinamento, o modelo aprende a associar padrões visuais nas imagens aos rótulos correspondentes, ajustando seus pesos para minimizar a diferença entre as previsões e os rótulos reais por meio de um mecanismo chamado *backpropagation* (GOODFELLOW; BENGIO; COURVILLE, 2016).

Para realizar a classificação de imagens, um modelo de aprendizado de máquina precisa receber as imagens como entrada. Geralmente, essas imagens são representadas como tensores multidimensionais, onde as dimensões correspondem à largura, altura e canais de cor da imagem. Por exemplo, uma imagem colorida em formato RGB (*Red, Green, Blue*) teria uma representação em tensor de forma (largura, altura, 3), onde 3 representa os três canais de cor, enquanto uma imagem em tons de cinza, teria um representação em tensor de forma (largura, altura, 1) (SZELISKI, 2021).

Antes de serem fornecidas ao modelo, as imagens geralmente são pré-processadas onde aplica-se operações de normalização (para terem valores de pixel entre 0 e 1, por exemplo) e redimensionamento para um tamanho específico, garantindo que todas as imagens tenham as mesmas dimensões de entrada.

As saídas de um modelo de classificação de imagens são as classes ou rótulos atribuídos a cada imagem de entrada. Dependendo do problema, essas classes podem ser binárias, onde é possível identificar apenas uma classe, como gato ou cachorro, ou ainda multiclasse como por exemplo, classificação de dígitos de 0 a 9 (KRIZHEVSKY; SUTSKEVER; HINTON, 2012).

Durante a inferência, o modelo recebe uma imagem de entrada e produz uma previsão de classe. Na saída da rede é obtida uma probabilidade para cada classe (GOODFELLOW;

BENGIO; COURVILLE, 2016). A classe com a maior probabilidade é então escolhida como a previsão final do modelo.

2.6.2 Segmentação semântica

A segmentação semântica é uma técnica de visão computacional que consiste em atribuir um rótulo a cada *pixel* de uma imagem, de modo que *pixels* pertencentes ao mesmo objeto tenham o mesmo rótulo. Diferentemente da classificação de imagens, onde o objetivo é atribuir um rótulo à imagem como um todo, a segmentação semântica permite uma compreensão mais detalhada da cena, identificando não apenas os objetos presentes, mas também suas localizações precisas (LONG; SHELHAMER; DARRELL, 2015).

Existem várias abordagens para realizar a segmentação semântica, sendo as redes neurais convolucionais (CNNs) uma das mais utilizadas devido à sua capacidade de aprender representações hierárquicas de características visuais (CHEN *et al.*, 2018). As CNNs são capazes de capturar informações espaciais e contextuais em diferentes escalas, o que as torna particularmente eficazes para tarefas de segmentação (BADRINARAYANAN; KENDALL; CIPOLLA, 2017).

Para treinar um modelo de segmentação semântica, são necessárias imagens de entrada e suas máscaras correspondentes, onde cada *pixel* na máscara é rotulado com o objeto ao qual pertence. Durante o treinamento, o modelo aprende a associar características visuais nas imagens aos rótulos nas máscaras, ajustando seus pesos para minimizar a diferença entre as previsões e as máscaras reais. Após o treinamento, o modelo pode ser usado para segmentar novas imagens, atribuindo um rótulo a cada *pixel* com base em suas características visuais (RONNEBERGER; FISCHER; BROX, 2015).

2.7 Otimizadores

Otimizadores, ou *Optimizers*, (GOODFELLOW; BENGIO; COURVILLE, 2016) são algoritmos ou metodologias usadas no treinamento de redes neurais para ajustar os pesos do modelo visando minímizar a função de custo. Durante a etapa de *backpropagation*, a rede calcula o gradiente da função de custo para encontrar os mínimos locais e o minimo global onde a função converge para um resultado ideal. Os pesos são então ajustados de acordo com esse gradiente, e é nesse momento que as funções de otimização são utilizadas. A função dos otimizadores é encontrar a melhor e mais eficiente maneira de ajustar os pesos das conexões entre os neurônios para reduzir o erro nas previsões do modelo.

Dessa forma, esses algorítimos buscam convergir para o resultado de maneira estável, com o menor número de iterações possível. Alguns otimizadores conhecidos são o Stochastic Gradient Descent (SGD), Momentum, RMSprop e Adam.

2.7.1 ADAM

O otimizador Adam (*Adaptive Moment Estimation*) (KINGMA; BA, 2014) é um dos métodos mais utilizados e eficazes para otimização de redes neurais. O Adam combina as melhores características de dois outros otimizadores amplamente utilizados: o Adagrad e o RMSprop. Ele é projetado para adaptar as taxas de aprendizado (*learning rate*) que determina o tamanho dos passos que o algoritmo de otimização dá na direção dos mínimos da função de perda. Isso é feito utilizando momentos de primeira e segunda ordens dos gradiente, de forma que ele leva em consideração tanto a média dos gradientes quanto a sua variabilidade, resultando em atualizações mais estáveis e rápidas em comparação com métodos que utilizam apenas a média ou apenas a variância dos gradientes.

A primeira etapa desse metodo é iniciar os parametros média móvel dos gradientes m_0 e a variância dos gradientes v_0 como vetores de zeros. Definir os hiperparâmetros β_1, β_2, α e ϵ. Embora o Adam ainda tenha hiperparâmetros como β_1, β_2 e a taxa de aprendizado α, ele é geralmente menos sensível as suas configurações iniciais do que outros otimizadores. Isso facilita o processo de ajuste e *tuning* durante o treinamento.

Após isso é possível calcular o gradiente da função de custo. O gradiente de uma função retorna um vetor que aponta na direção do maior declive descendente, indicando a direção para o mínimo da função. Aplicando essa operação iterativamente, é possível ajustar os parâmetros do modelo para convergir para um mínimo local da função de custo. A equação 5 ilustra a operação de gradiente sobre a função de custo, essencial para a atualização dos pesos no treinamento de redes neurais:

$$g_t \leftarrow \nabla_\theta J(\theta_t). \tag{5}$$

As equações 6 e 7 representam a média móvel e a variância dos gradientes, respectivamente. Essas operações ajudam a entender a trajetória dos gradientes durante o treinamento. Observando a média móvel e a variância, podemos determinar se os gradientes estão se tornando mais agudos, o que indica uma descida mais rápida em direção ao mínimo da função de custo. Essa análise é fundamental para ajustar os parâmetros do modelo de forma eficiente e garantir a convergência adequada durante o treinamento:

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{6}$$

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2. \tag{7}$$

As equações 8 e 9 representam o ajuste de bias. Esse ajuste é crucial para corrigir a tendência inicial das estimativas de primeira e segunda ordens (média a variância) dos gradientes. Sem essa correção, as estimativas tendem a ser enviesadas para valores próximos de zero, especialmente nas primeiras iterações. A correção de bias garante que as estimativas sejam

mais precisas e robustas ao longo do tempo, permitindo que o otimizador Adam ajuste os pesos de forma mais eficiente e estável. Isso resulta em uma convergência mais rápida e consistente para o mínimo da função de custo.

$$\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}, \tag{8}$$

$$\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}. \tag{9}$$

Ao final do processo, os parâmetros são atualizados utilizando as estimativas corrigidas de primeira e segunda ordens. A equação 10 representa esta atualização final, onde α é a taxa de aprendizado, $\hat{m}_t$ é o momento corrigido, $\hat{v}_t$ é a variância corrigida, e ϵ é um termo de estabilidade para evitar divisão por zero. Esta atualização combina a direção do gradiente com a escala da variância, ajustando os pesos de forma mais precisa.

$$\theta_t \leftarrow \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \tag{10}$$

2.8 Redes Neurais de Convolução (CNN)

As Redes Neurais Convolucionais (CNNs) são um tipo especializado de rede neural projetada para processar dados que possuem uma estrutura de grade, como imagens (GOODFELLOW; BENGIO; COURVILLE, 2016). Elas são amplamente utilizadas em tarefas de visão computacional, incluindo classificação de imagens, detecção de objetos, segmentação semântica e muito mais, devido à sua capacidade de capturar padrões espaciais em dados de entrada.

Uma característica chave das CNNs são as camadas convolucionais, que aplicam operações de convolução nas entradas. A convolução permite que a rede extraia automaticamente características hierárquicas das imagens, como bordas, texturas e padrões mais complexos, aprendendo a partir dos dados durante o treinamento.

Além das camadas convolucionais, as CNNs geralmente incluem camadas de *pooling*, que reduzem a dimensionalidade dos mapas de características, preservando as informações mais importantes. Isso ajuda a tornar a rede mais eficiente computacionalmente e a melhorar a generalização para novos dados.

Outra característica importante das CNNs é a utilização de camadas totalmente conectadas no final da rede, que agregam as informações das camadas convolucionais e de *pooling* para produzir uma saída final, como a probabilidade de cada classe em uma tarefa de classificação de imagens. Durante o treinamento, as CNNs utilizam o algoritmo de backpropagation para ajustar os pesos da rede e minimizar a diferença entre as previsões e os rótulos reais.

2.8.1 Convolução

Convolução é uma operação matemática realizada entre duas funções, ao efetuar essa operação de duas funções $x(a)$ e $w(a)$, estamos essencialmente calculando um somatório ponderado dos produtos dos valores das duas funções, equanto deslocamos a função $w(a)$ no espaço conforme apresentado na equação 11:

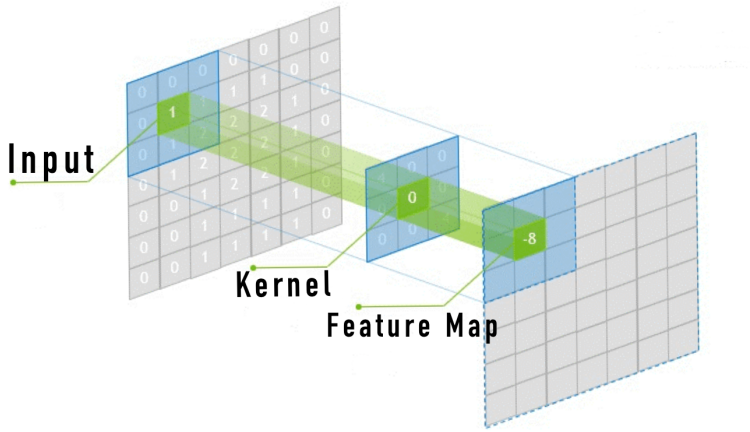
$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{11}$$

O resultado obtido representa a “semelhança“ das duas funções naquele ponto do espaço (GOODFELLOW; BENGIO; COURVILLE, 2016). Se os valores de $x(a)$ e $w(a)$ coincidem fortemente em uma certa posição, a soma dos produtos será grande, indicando uma alta similaridade entre as funções. De forma análoga, se os valores de $x(a)$ e $w(a)$ não coincidirem bem, a soma dos produtos será pequena, indicando uma baixa similaridade.

No contexto de CNNs, o primeiro argumento $x(a)$ da convolução é chamado de *input*e o segundo argumento $w(a)$ é referido como *kernel*. Em aplicações de aprendizado de máquina, o *input* geralmente é um array multidimensional de dados, e o *kernel* é um array multidimensional de parâmetros que são aprendidos pela rede neural. O output é conhecido como *feature map* e guarda as informações correlatas entre a função de entrada e o *kernel*.

A estrutura desta operação pode ser observada na figura 4.

Figura 4 – Operação de Convolução de de dados multidimensionais



Fonte: Adaptado de (NATHAN; WAMPLER, 2023).

A convolução matematicamente inverte a segunda função, conforme pode ser observado na equação 11, mas essa inversão é útil somente para propriedades comutativas da operação. No contexto de redes neurais, essa propriedade não é relevante; em vez disso, é utilizada a *cross-correlation*, que executa o mesmo processo da convolução, porém sem inverter a função do *kernel* (GOODFELLOW; BENGIO; COURVILLE, 2016). A equação 12 representa uma

convolução de duas dimensões comumente usada nas CNNs:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{12}$$

As CNNs são especialmente eficazes no processamento de imagens devido à sua capacidade de aprender *kernels* que capturam características hierárquicas de maneira eficiente (HE *et al.*, 2016). Para entender o que ocorre na convolução entre o *input* e o *kernel*, podemos analisar um filtro específico como o filtro de Sobel (GOODFELLOW; BENGIO; COURVILLE, 2016). As regiões onde o filtro apresenta um valor alto indicam a presença de uma borda. Esses filtros são aprendidos pela rede utilizando funções de custo e otimização, que identificam os *kernels* que melhor discriminam os dados, trazendo maior separabilidade para a classificação. Em camadas mais rasas os filtros podem detectar bordas e texturas simples, enquanto em camadas mais profundas, podem reconhecer formas, objetos e até mesmo contextos completos.

2.8.2 Arquitetura CNN

Podemos dividir a arquitetura de uma CNN em três etapas: a camada de entrada, as camadas escondidas e as camadas de saída (GOODFELLOW; BENGIO; COURVILLE, 2016). A camada de entrada recebe os dados originais. Em uma imagem digital, ela é representada como uma matriz de *pixels* que será processada pela rede.

Nas camadas escondidas, temos diferentes tipos de camadas. As camadas convolucionais são responsáveis por aplicar os *kernels* na imagem e extrair as características locais, como bordas, texturas e outros padrões. Durante o processamento, a rede utiliza diversos *kernels*, cada um convoluindo sobre a imagem e produzindo um mapa de características (*feature map*).

Após as operações de convolução, é aplicada uma função de ativação, como a ReLU, que introduz não linearidades na rede, destacando os mapas de features que melhor representam o conjunto de entrada. A função retorna o próprio valor x de entrada para x>0 e 0 para:

$$f(x) = \max(0, x). \tag{13}$$

Em seguida, os dados passam por uma camada de *Pooling* que reduz as dimensões espaciais dos *feature maps*. No contexto de processamento de imagem isso se torna essencial visto que a quantidade de *pixels* em uma imagem de alta resolução torna o processamento muito custoso. O *pooling* mais comum é o *max pooling*, que toma o valor máximo de uma região de *pixels*, como por exemplo, 2x2, sobre o mapa de características. Isso reduz a resolução espacial dos dados, mas preserva as características mais importantes

As camadas ocultas consistem em múltiplas camadas de convolução, ReLU e *pooling*. À medida que avançamos para camadas mais profundas, a CNN é capaz de detectar características mais específicas dos dados.

As camadas de saída preparam os dados para a classificação. A camada de *flattening* transforma os *feature maps* bidimensionais em um vetor unidimensional, preparando assim os dados para uma rede neural artificial. Esse vetor de características, gerado pelas etapas anteriores, contém todas as informações necessárias para a identificação e classificação da imagem.

Em seguida, temos as camadas totalmente conectadas, que representam uma RNA. A camada de classificação é a última camada da CNN, onde é possivel utilizar uma função *softmax* para converter a saída das camadas totalmente conectadas em um vetor de probabilidades. Dessa forma, a saída representa a probabilidade dos dados representarem cada classe ou rótulo. A função *softmax*

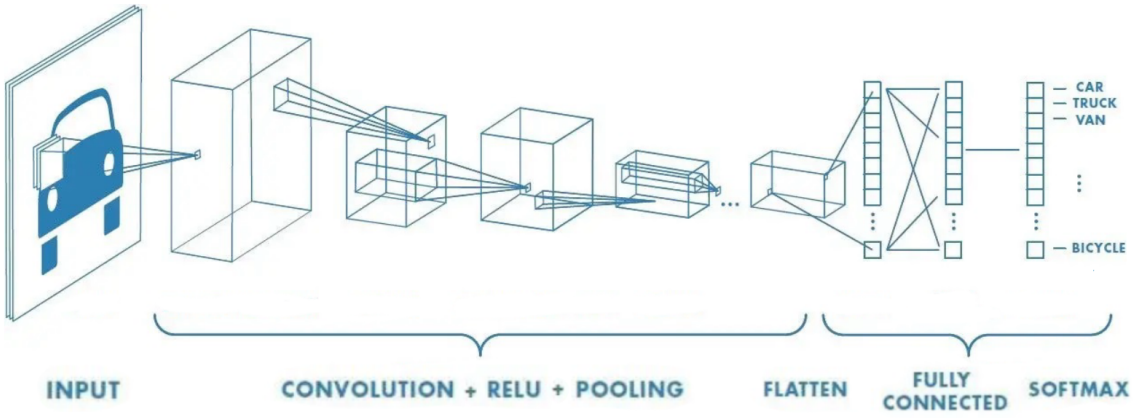
A função softmax da classe j, considerando K classes, é definida como:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}. \tag{14}$$

Nesta definição, $\sigma(z)_j$ representa a probabilidade da classe j após a aplicação da função softmax, z_j é o valor da entrada correspondente à classe j, K é o número total de classes, e e é a base do logaritmo natural.

A função softmax transforma os valores de entrada z_j em uma distribuição de probabilidade, garantindo que a soma das probabilidades para todas as classes seja igual a 1.

Figura 5 – Arquitetura CNN



Fonte: Adaptado de (SEZER; OZBAYOGLU, 2018).

2.9 Métricas de avaliação

Conseguir avaliar os modelos é fundamental para assegurar a precisão e a robustez dos algoritmos de aprendizagem. As métricas de avaliação fornecem uma base quantitativa para comparar diferentes modelos e selecionar o que melhor se ajusta a uma tarefa específica. Essas métricas são ferramentas matemáticas que evidenciam os acertos e erros do modelo (GOODFELLOW; BENGIO; COURVILLE, 2016).

Para compreender as métricas de avaliação, é crucial entender os conceitos de verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN):

- **Verdadeiros Positivos (VP):** São os casos em que o modelo previu corretamente a classe positiva. Por exemplo, um modelo que indica há presença de uma planta em uma imagem que realmente contém uma planta.
- **Falsos Positivos (FP):** São os casos em que o modelo previu erroneamente a classe positiva. Por exemplo, um modelo que indica há presença de uma planta em uma imagem que na verdade não contém uma planta.
- **Verdadeiros Negativos (VN):** São os casos em que o modelo previu corretamente a classe negativa. Por exemplo, um modelo que identifica corretamente uma imagem sem planta como negativa.
- **Falsos Negativos (FN):** São os casos em que o modelo previu erroneamente a classe negativa. Por exemplo, um modelo que não detecta uma planta em uma imagem que na verdade contém uma planta.

2.9.1 Acurácia

A acurácia é definida como a proporção de previsões corretas (tanto verdadeiros positivos quanto verdadeiros negativos) em relação ao total de casos:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{15}$$

Embora seja uma métrica intuitiva, a acurácia pode levar a interpretações errôneas, especialmente em casos de desbalanceamento de classes. Isso ocorre porque um modelo pode prever corretamente as classes majoritárias, indicando uma alta acurácia, mas ainda assim apresentar um desempenho baixo nas classes minoritárias (MüLLER; SOTO-REY; KRAMER, 2022).

2.9.2 Recall

O recall (MüLLER; SOTO-REY; KRAMER, 2022) é a razão entre os verdadeiros positivos e a soma dos verdadeiros positivos e falsos negativos. Essa métrica avalia a eficiência do modelo em detectar corretamente as instâncias positivas, e penalizar casos de falsos negativos:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{16}$$

2.9.3 Precisão

A precisão (MüLLER; SOTO-REY; KRAMER, 2022) é a razão entre os verdadeiros positivos e a soma dos verdadeiros positivos e falsos positivos. Ela indica a proporção de instâncias preditas como positivas que são realmente positivas. Alta precisão significa que a maioria das predições positivas do modelo são corretas. Isso é crucial em situações onde os falsos positivos têm um alto custo:

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{17}$$

2.9.4 F1-score

O F1-score oferece uma abordagem balanceada para a avaliação de modelos de classificação (POWERS, 2020). Esta métrica torna-se particularmente relevante em cenários com classes desbalanceadas, onde a distribuição de amostras entre as classes é assimétrica. Nessas situações, é crucial considerar tanto os falsos positivos quanto os falsos negativos para evitar avaliações distorcidas do desempenho do modelo:

$$\text{F1-Score} = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}}. \tag{18}$$

2.9.5 Matriz de confusão

A matriz de confusão é uma ferramenta fundamental para a avaliação de modelos de classificação (GOODFELLOW; BENGIO; COURVILLE, 2016), especialmente quando se trata de problemas com múltiplas classes. Ela proporciona uma visão clara de como o modelo está se saindo em termos de classificações corretas e incorretas em comparação com cada classe. Para um problema de classificação com N classes, a matriz de confusão será uma matriz NxN. Cada elemento M_{ij} na matriz representa o número de amostras que pertencem à classe i (verdadeira) e foram classificadas como classe j (prevista).

Dessa maneira, a diagornal principal (Mii) representa as amostras corretamente classificadas para cada classe i. Os elementos fora da diagonal indicam os erros de classificação. Por

Tabela 2 – Matriz de Confusão para Múltiplas Classes

Classe Verdadeira	Classe Predita			
	Classe Predita 1	Classe Predita 2	...	Classe Predita N
Classe Verdadeira 1	M_{11}	M_{12}	...	M_{1N}
Classe Verdadeira 2	M_{21}	M_{22}	...	M_{2N}
...	...	...	...	...
Classe Verdadeira N	M_{N1}	M_{N2}	...	M_{NN}

exemplo, M_{12} representa o número de amostras da classe 1 que forma incorretamente classificadas como classe 2. Para melhor visualização e interpretação, especialmente em conjuntos de dados desbalanceados, a matriz de confusão pode ser normalizada, resultando em proporções ou porcentagens de acerto e erros para cada classe i.

2.9.6 IoU

O Índice de Jaccard (REZATOFIGHI *et al.*, 2019), também conhecido como IoU (do inglês *Intersection over Union*), é uma métrica comum usada em problemas de segmentação de imagens para avaliar o quão bem as predições de um modelo se sobrepõem o *ground truth*. Ele é calculado como a razão entre a área de interseção e a área de união entre a predição e o *ground truth*:

$$\text{IoU} = \frac{\text{Área de Interseção}}{\text{Área de União}}. \tag{19}$$

- **Área de interseção** representa a região onde a predição e o *ground truth* se sobrepõem. Em outras palavras, é a parte da predição que coincide com o ground truth.
- **Área de união** é a combinação de todas as regiões cobertas tanto pela predição quanto pelo *ground truth*, sem duplicações. Isso inclui a área que é exclusiva da predição, a área que é exclusiva do *ground truth* e a área de sobreposição.

Quanto mais próximo o valor do IoU estiver de 1, melhor a sobreposição entre a predição e o *ground truth*, indicando uma segmentação mais precisa. Essa métrica apresenta a precisão em nível de *pixel* para modelos de segmentação semântica. Para problemas de segmentação com múltiplas classes é comumente utilizado a média do IoU (*MIoU*). Ela calcula o IoU para cada classe e tira a média dessas pontuações onde N é o número de classes e IoU_i é o IoU para a classe i:

$$\text{Mean IoU} = \frac{1}{N} \sum_{i=1}^{N} \text{IoU}_i. \tag{20}$$

2.10 Arquitetura de redes para classificação

Para a classificação de imagens, diversas arquiteturas de rede neural têm sido desenvolvidas ao longo dos anos, cada uma trazendo avanços significativos em termos de precisão e velocidade. O objetivo desse tipo de rede é, dado uma imagem ao modelo, ele ser capaz de reconher uma classe pré-definida, ou seja, as entradas são imagens e as saídas um número que representa uma das classes de treinamento. Esse tipo de abordagem só permite que sejam classificados um único objeto dentro de uma imagem.

2.10.1 VGG19

A arquitetura VGG19 (SIMONYAN; ZISSERMAN, 2014), é uma variante mais profunda da rede VGG, consistindo de 19 camadas ponderadas. A principal característica das redes VGG é a utilização de filtros de convolução de tamanho fixo (3x3) com *strides* e *paddings* que preservam as dimensões das imagens após cada convolução. Essa simplicidade na escolha dos hiperparâmetros facilita a implementação e permite um aumento progressivo da profundidade da rede. A VGG19 possui um total de 16 camadas convolucionais, 3 camadas totalmente conectadas e utiliza funções de ativação ReLU. A arquitetura termina com uma camada *softmax* para a classificação.

2.10.2 ResNet

As redes ResNet (HE *et al.*, 2016) introduziram a ideia de *residual learning* para mitigar o problema de *gradient descent* em redes muito profundas. Em vez de aprender mapeamentos diretamente, as ResNets aprendem funções residuais com referência às entradas da camada. Esse método permite a construção de redes muito mais profundas sem a degradação da precisão que normalmente acompanha o aumento da profundidade. A estrutura básica de uma ResNet é o “bloco residual“, que consiste em múltiplas camadas convolucionais com atalhos que contornam uma ou mais camadas, adicionando a entrada inicial ao *output* final. Existem várias variantes da ResNet, cada uma com um número diferente de camadas e os mais populares implementados são a ResNet50, ResNet101 e ResNet152.

2.10.3 MobileNets

As arquiteturas MobileNet (HOWARD *et al.*, 2017) são projetadas para serem leves e eficientes, visando dispositivos móveis e aplicações com recursos limitados. A principal inovação da MobileNet é a utilização de convoluções separáveis em profundidade, que reduzem significativamente o número de parâmetros e operações computacionais. Um exemplo a MobileNetV2

introduz a ideia de blocos inversos residuais e convoluções *depthwise* seguidas por convoluções *pointwise*. Já a MobileNetV3 é uma versão aprimorada da MobileNetV2, incorporando técnicas de otimização neural, como busca de arquitetura de rede (NAS) e técnicas de aprimoramento de eficiência como SE (Squeeze-and-Excitation) e h-swish. MobileNetV3 vem em duas variantes, *Small* e *Large*, adaptadas para diferentes *trade-offs* entre precisão e eficiência computacional.

2.11 Arquitetura de redes para segmentação semântica

Assim como as arquiteturas de rede com a finalidade de classificação, as CNN's com a finalidade de segmentação têm sido amplamente exploradas. Tratando-se de segmentação, há diversas formas como uma mesma imagem pode ser segmentada, mas, nesse trabalho, o foco será dado unicamente há segmentação semântica, àquela capaz de não só segmentar os objetos dentro de uma imagem, mas também de dar sentido a cada um deles por meio de uma classificação dos *pixels*. Nesse tipo de abordagem, a entrada da rede é uma imagem e a saída, uma máscara de identificadores onde cada pixel representa uma classe pré-definida. Esse tipo de abordagem permite segmentar e classificar diversas classes dentro de uma mesma imagem.

2.11.1 Unet

A U-Net (RONNEBERGER; FISCHER; BROX, 2015) é uma arquitetura de rede neural convolucional amplamente utilizada para segmentação de imagens. A principal característica da U-Net é a sua forma de "U" simétrica, que consiste em uma fase de contração (*downsampling*) seguida por uma fase de expansão (*upsampling*).

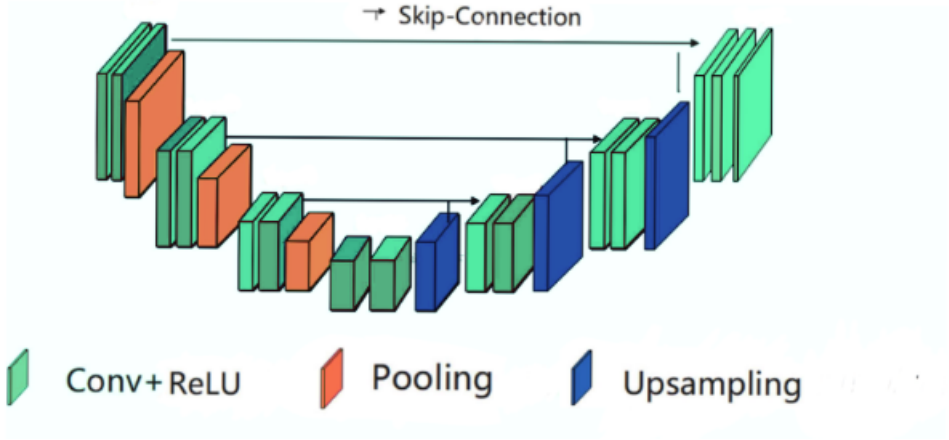
Na fase de contração, também conhecida como *encoder*, a U-Net utiliza várias camadas convolucionais seguidas por operações de *pooling*. Essas operações de pooling, geralmente *max-pooling*, reduzem a dimensionalidade espacial da imagem de entrada enquanto aumentam a profundidade do mapa de características. Esse processo permite que a rede aprenda a capturar contextos de imagem em diferentes escalas.

A fase de expansão, ou *decoder*, reverte o processo de contração aumentando a resolução espacial dos mapas de características. Isso é feito através de camadas de convolução transposta, também conhecidas como deconvoluções, que realizam o *upsampling*. O objetivo dessa fase é recuperar a resolução original da imagem e produzir uma segmentação detalhada.

Uma das principais inovações da U-Net são as conexões de *skip* (atalhos) presentes no modelo. Essas conexões ligam camadas correspondentes das fases de contração (*encoder*) e expansão (*decoder*), permitindo uma transferência direta de informações. Durante a fase de contração, a resolução espacial da imagem é reduzida progressivamente através de operações de *pooling*, enquanto a profundidade do mapa de características aumenta. Este processo, em-

bora útil para capturar contextos globais e características de alto nível, pode resultar na perda de detalhes e informações locais.

Figura 6 – Arquitetura clássica da rede U-Net



Fonte: Adaptado:(CAI; WU; CHEN, 2022).

As conexões de *skip* armazenam os *feature maps* do *encoder* antes das operações de *downsampling* (*pooling*) e concatenam essa informação com os feature maps das camadas equivalentes no *decoder*. Essa operação permite que as informações detalhadas de alta resolução sejam combinadas com as informações mais abstratas aprendidas durante a contração.

Desta forma, as conexões de *skip* garantem que os detalhes finos e precisos da imagem original sejam preservados até a fase de reconstrução.

2.12 Data augmentation

A técnica de *data augmantation* (aumento de dados) é uma estratégia que permite aumentar a quantidade e a diversidade dos dados de treinamento sem a necessidade de coletar novas amostras, o que pode ser caro e demorado (SHORTEN; KHOSHGOFTAAR, 2019).

2.12.1 Adição de espaços em volta da imagem

A adição de espaços em volta da imagem, também conhecida como *padding*, é uma técnica usada para garantir que todas as imagens do conjunto de dados tenham dimensões mínimas específicas. Isso é especialmente útil quando se trabalha com imagens de diferentes tamanhos. O *padding* adiciona bordas à imagem original até que ela atinja as dimensões desejadas e é usada para evitar a distorção das imagens durante o redimensionamento e para assegurar que características importantes não sejam cortadas ou perdidas.

2.12.2 Redimensionamento

O redimensionamento de imagens é uma etapa essencial para preparar os dados de entrada para a rede neural. Ao uniformizar o tamanho das imagens para uma dimensão específica (por exemplo, 256x256 *pixels*), garantimos que o modelo receba entradas consistentes.x

2.12.3 Brilho e Contraste Aleatório

Variar o brilho e o contraste das imagens de entrada de forma aleatória é uma técnica eficaz para aumentar a robustez do modelo contra diferentes condições de iluminação. Ao expor o modelo a uma gama de intensidades de brilho e contraste, ele se torna mais adaptável a variações nas condições de iluminação que podem ocorrer em ambientes reais.

2.12.4 Virar a imagem horizontalmente e verticalmente

Ao criar imagens espelhadas a partir das originais, essas técnicas aumentam a diversidade do conjunto de dados sem modificar o conteúdo semântico das imagens. Essas transformações ajudam a garantir que o modelo se torne invariável à mudanças de orientação, melhorando sua robustez.

2.12.5 Desfoque mediano

A aplicação de desfoque mediano é uma técnica utilizada para reduzir o ruído nas imagens e suavizar detalhes menores. O desfoque mediano substitui cada *pixel* pelo valor mediano dos *pixels* em torno dele, o que ajuda a preservar as bordas enquanto reduz o ruído. Ao aplicar desfoque mediano de forma seletiva e aleatória, aumentamos a capacidade do modelo de aprender características mais robustas e menos suscetíveis a variações de ruído, melhorando a qualidade da classificação e da segmentação.

2.13 Trabalhos Correlatos

Nesta seção, serão apresentados e discutidos alguns trabalhos correlatos encontrados na literatura, que abordam a detecção e classificação de plantas daninhas utilizando diferentes abordagens e tecnologias. A análise desses trabalhos visa identificar as metodologias empregadas, seus resultados, bem como as limitações encontradas.

Viliotti (VILIOTTI, 2002) desenvolveu um sistema eletrônico equipado com sensores ópticos para correlacionar a radiação global incidente com a radiação refletida por superfícies cobertas com plantas daninhas. Este método mostrou-se eficiente na detecção de plantas da-

ninhas, porém, a estrutura do sistema requer manutenção frequente e cuidados específicos, o que pode aumentar os custos operacionais a longo prazo. Testes em condições simuladas não apresentaram anomalias e os testes de campo mostraram uma eficácia perto dos 100% no controle das plantas daninhas .

Ferreira (FERREIRA, 2017) utilizou algoritmos de classificação, como *Random Forest* e *Support Vector Machine* (SVM), para classificar plantas daninhas a partir de atributos extraídos de imagens, como matrizes de coocorrência, histogramas de gradientes, espaço de cores e padrões binários locais. Embora esta metodologia permita a detecção de plantas daninhas apenas com o uso de imagens, a fase de extração de atributos é custosa e pode exigir recursos computacionais significativos.

Souza Belete (BELETE *et al.*, 2019) e colaboradores propuseram um sistema de visão computacional que utiliza o método de segmentação SLIC superpixels e atributos visuais para extrair características físicas das folhas, como cor, gradiente, textura e forma. A metodologia demonstrou alta precisão (91,34%) utilizando o algoritmo SVM, mas ainda não é totalmente confiável para suporte a agricultores pois apresentou dificuldades de detecção em cenários com iluminação precária.

Milioto (MILIOTO; LOTTES; STACHNISS, 2018) desenvolveu redes neurais convolucionais totalmente conectadas (FCN) para a segmentação semântica em tempo real das plantas daninhas em meio a cultura de soja. Essa metodologia apresentou alta precisão e robustez na detecção de plantas daninhas, mas a variabilidade nas condições de luz ainda representa um desáfio .

O estudo (KIM; PARK, 2022) apresentou o MTS-CNN (Multi-task Semantic Segmentation-Convolutional Neural Network) para a detecção de culturas e plantas daninhas, utilizando uma abordagem de treinamento de uma etapa com segmentação semântica multitarefa baseada em duas redes U-Net com backbone VGG-16. A primeira rede realiza a segmentação dos objetos (cultura e plantas daninhas), enquanto a segunda gerencia a segmentação específica de culturas e plantas daninhas, utilizando atenção aos mapas de características para melhorar a precisão. Testado em três conjuntos de dados públicos (BoniRob, CWFID e um conjunto de mudas de arroz), o MTS-CNN alcançou valores de MIoU de 0,9164, 0,8372 e 0,8260, respectivamente. No entanto, a precisão da segmentação depende fortemente da segmentação inicial dos objetos, a abordagem requer hardware de alto desempenho e a sensibilidade ao desequilíbrio de classes pode impactar o desempenho.

3 MATERIAIS E MÉTODOS

3.1 Materiais

Diversas fontes de dados foram fornecidas por trabalhos anteriores e, principalmente, por professores parceiros. Este subtópico dedica-se a descrever cada um deles e também falar sobre como foram utilizado neste trabalho.

3.1.1 Dataset

Para construir os conjuntos de dados sintéticos, foi necessário utilizar duas bases de imagens distintas. A primeira contém imagens das principais plantas daninhas encontradas em território nacional. Já o segundo conjunto de dados consiste em fotos retiradas de um cultivo de milho.

Ao longo desta seção, serão apresentadas essas bases de imagens, além das metodologias empregadas para a detecção e classificação, bem como as abordagens utilizadas para gerar os conjuntos de dados sintéticos.

3.1.1.1 Western Paraná Weeds

Western Paraná Weeds é uma base de imagens privada de ervas daninhas localizadas na região oeste do Paraná, construído em 2020[1]. A base é composta por 752 fotografias das 8 principais famílias e 14 principais espécies distribuídas em território nacional. As imagens estão organizadas por familias e espécies e apresentam a evolução das plantas daninhas no decorrer de 2 meses, todas elas foram cultivadas em vasos e as fotografias foram tiras de uma câmera que estava a 90 cm da base do vaso. Elas são fornecidas no formato JPG (*Joint Photographic Experts Group*) com espaço de cores RGB de 24 bits onde a resolução das imagens é de 480 x 360 pixels.

Os gráficos das figuras 7 e 8 expressam a distribuição de imagens deste conjunto de dados em relação as famílias e espécies. Note que as espécies Sorghum Arundinaceum, Cenchus Echinatus, Rhynchelytrumrepens, Digitaria Insularis e Choris Barbata pertencem a família Poaceae e as espécies Tridax Procumbens, Bidens-pilosa e Sonchus Oleraceus à Asteraceae. Além disso, a figura 9 mostra alguns exemplos das imagens deste conjunto de dados.

[1] Este conjunto de dados foi desenvolvido e disponibilizado pelos professores Anderson Brilhador, Alessandra Matte, e Cintia Maria Teixeira Fialho da Universidade Tecnológica Federal do Paraná - Campus Santa Helena.

Figura 7 – Distribuição imagens do Conjunto Western Paraná Weeds por família

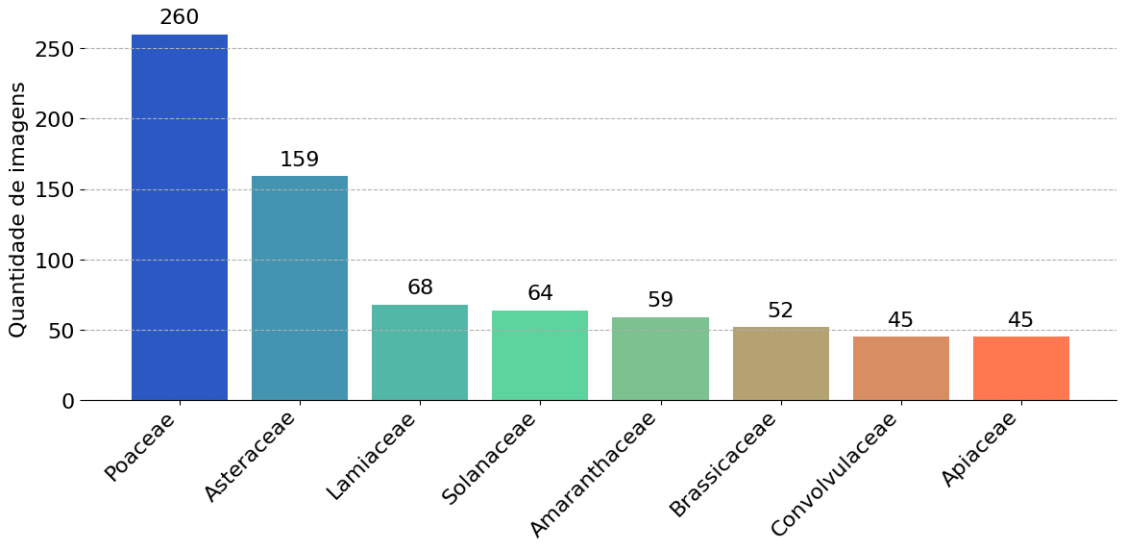
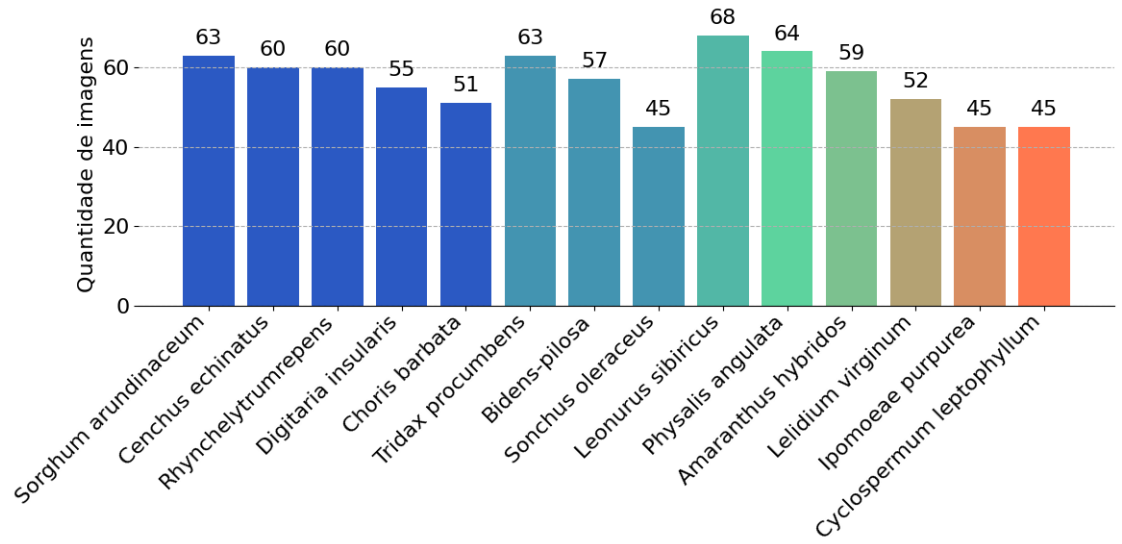


Figura 8 – Distribuição imagens do Conjunto Western Paraná Weeds por espécie

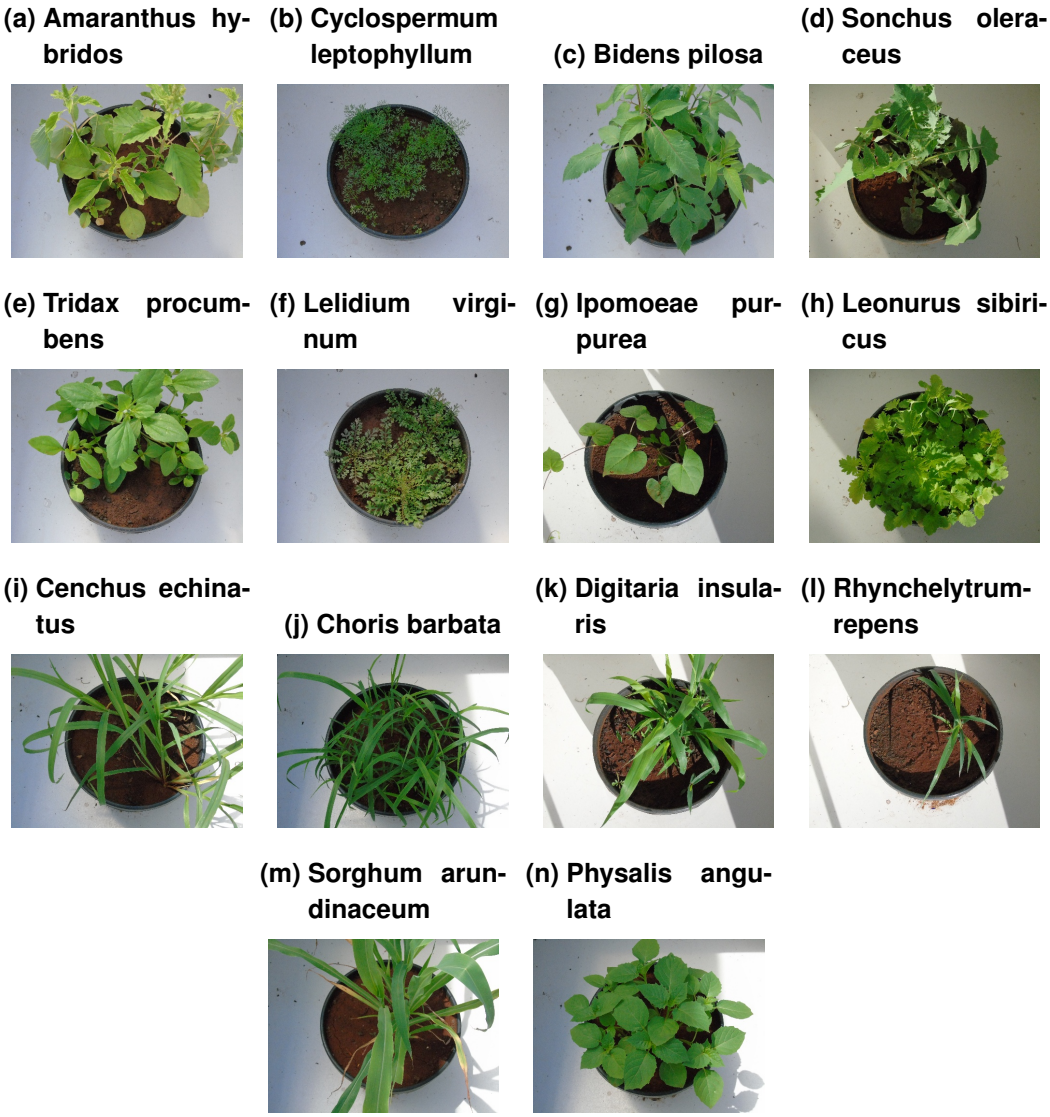


3.1.1.2 Cornspacing

Cornspacing é uma base pública construída para análise de áreas de plantio da cultura de milho (BRILHADOR; SERRARENS; LOPES, 2015), que consiste de fotos panorâmicas de campos de milho, capturadas por um dispositivo móvel.

Para a aquisição das imagens, uma pessoa segurou a câmera a uma altura média de 1,2 metros do solo, capturando diversas imagens da plantação de milho. A base possui 188 imagens e estão disponibilizadas no formato de arquivo JPG. O processo de captura se deu a partir da escolha de um ponto inicial feito de forma aleatória e a com o movimento do fotográfo por aproximadamente 3 metros para composição da foto panorâmica. Devido ao método utilizado, a resolução das imagens não são padronizadas, mas possuem medidas de aproximadamente 3000 x 600 pixels.

Figura 9 – Examplos de imagens do conjunto de dados Western Paraná Weeds



Fonte: Western Paraná Weeds (2024).

A figura 10 possui duas amostras das imagens presentes no conjunto de dados para diferentes cenários: O de plantio convencional e outro após aplicação de herbicidas. O conjunto é composto por 66 imagens de plantio convencional e 122 imagens capturadas após a aplicação de herbicidas.

3.1.2 Blender

O Blender [2] é uma ferramenta de criação de elementos gráfico e simulações 3D, tendo seu código fonte disponível sob a licença GNU GPL (*General Public License*), tornando-o assim um software livre. Esta ferramenta oferece uma variedade de funcionalidades como modelagem, renderização, animação, pós-produção, criação e visualização de conteúdo 3D interativo.

[2] Disponível gratuitamente em https://www.blender.org/

Figura 10 – Exemplo imagens do conjunto de dados cornspacing
(a) Plantio convencional

(b) Plantio direto após aplicação de herbicidas

Fonte: (BRILHADOR; SERRARENS; LOPES, 2015).

Destacando-se entre suas características está sua impressionante capacidade de renderização. O *software* proporciona uma extensa gama de opções para criar imagens de alta qualidade e realistas. Desde técnicas avançadas de iluminação e sombreamento até simulação de materiais complexos e oferece aos usuários um leque diversificado de ferramentas para alcançar resultados visualmente impressionantes.

Outra funcionalidade que o *software* oferece é uma bilioteca em python capaz de executar comandos complexos dentro do blender, possiblilitando a automatização de processos e simulações.

3.1.3 GIMP

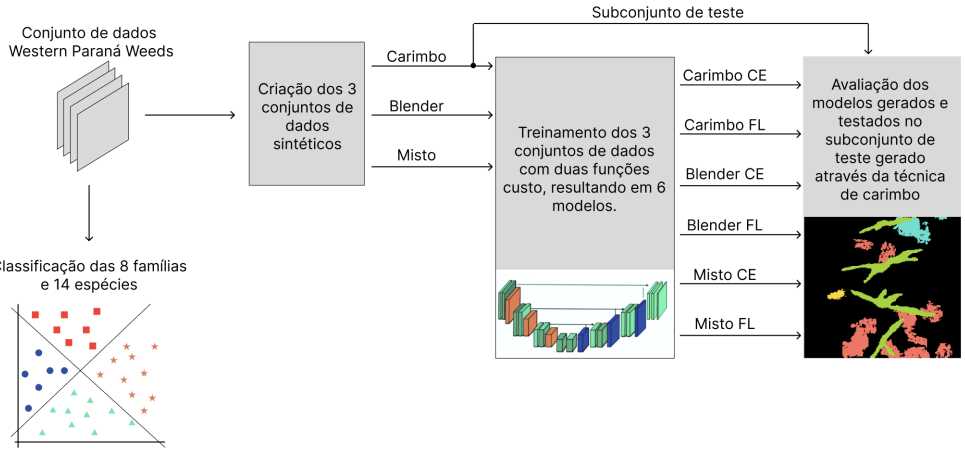
O *software* conhecido comumente pela sua sigla GIMP [3], é um editor de imagens gratuito e de código aberto, utilizado para manipulação de imagens e edição de imagens, desenho livre, transcodificação entre diferentes formatos de arquivo de imagem, e tarefas mais especializadas.

3.2 Métodos

Neste trabalho, foram empregadas diversas técnicas e tecnologias, e esta seção é dedicada a descrever as metodologias utilizadas em cada etapa. Isso inclui a classificação do conjunto de dados Western Paraná Weeds, a criação de um conjunto de dados sintético gerado por meio da técnica de carimbo com os conjuntos de dados Cornspacing e Western Paraná Weeds, o desenvolvimento de um conjunto de dados gerado por simulações utilizando o Blender e

[3] Disponível gratuitamente em https://www.gimp.org/

Figura 11 – Visualização da metodologia utilizada para a realização do experimento. A distinção das funções de perda é feita a partir das siglas FL (*Focal Loss*) e CE (*Cross Entropy*)



Fonte: Autoria própria (2024).

a segmentação das plantas daninhas em meio às plantações de milho. A metodologia aplicada neste trabalho pode ser visualzada na figura 11 e que será discutida em maior profundidade nos tópicos seguintes.

3.2.1 Metodologia para a construção da pipeline de classificação

A classificação das plantas daninhas será um processo importante para verificar se é de fato possível, com os modelos existentes hoje, classificar plantas daninhas que são extremamente semelhantes entre elas. Será necessário verificar a possibildiade de diferenciá-las de acordo com suas 8 famílias e 14 espécies.

Todos os modelos mencionados anteriormente permitem a realização de *fine-tuning* a partir do conjunto de dados ImageNet. Portanto, a proposta é criar um *benchmark* que compare os modelos VGG19, ResNet50, ResNet101, ResNet152, MobileNetV2 e MobileNetV3. Também serão comparadas as diferentes funções de perda para esse problema, a *FocalLoss* e a *Weighted Cross Entropy*, uma vez que estaremos tratando dados desbalanceados. A comparação levará em consideração a precisão, o recall e o F1-score.

Cada um dos modelos será treinado por 35 épocas usando o conjunto Brazilian Weeds 2020 com as imagens redimensionadas para 224x224 pixels. Os *augmentations* aplicados para variação de dados serão o *flip* horizontal aleatório, a rotação aleatória, o zoom aleatório e o contraste aleatório. Todos eles com a probabilidade de 10% de acorrerem. O Adam será utilizado como otimizador em todos os experimentos.

3.2.2 Criação semi-automática de máscaras

Para desenvolver um modelo de segmentação semântica, é essencial ter não apenas as imagens a serem segmentadas, mas também as máscaras binárias correspondentes a cada imagem. Uma máscara binária é uma imagem em preto e branco que destaca as regiões de interesse, como as plantas daninhas em uma imagem agrícola. Nessa máscara, os pixels brancos representam as áreas que deseja-se segmentar (no caso, as plantas), enquanto os pixels pretos representam o fundo ou outras áreas que não são de interesse. Uma abordagem semi-automática pode ser empregada para criar essas máscaras de referência, especialmente quando há uma distinção clara entre a planta e o fundo.

Figura 12 – Exemplo de máscara binária obtida com um dos índices de vegetação e ajuste manual

(a) Planta daninha **(b) Máscara binária**

Fonte: Autoria própria.

No caso do conjunto de dados Western Paraná Weeds, é essencial aplicar diferentes índices de vegetação, como Excessive Green, CIVE e NDI, dependendo da imagem. Contudo, os valores obtidos por esses métodos podem não abranger toda a cobertura da planta daninha, uma vez que as plantas apresentam uma variedade de cores, incluindo tons amarelados e amarronzados, além das tonalidades de verde. Portanto, é muitas vezes necessário ajustar manualmente o valor do *threshold* para obter resultados mais precisos.

Uma possível solução para agilizar a criação dessa máscara seria criar uma ferramenta que permite aplicar diferentes índices de vegetação e ajustar manualmente o *threshold* conforme necessário. Utilizando a ferramenta Streamlit, foi desenvolvida uma aplicação capaz de realizar este trabalho e gerar máscaras binárias a partir do *threshold* obtido pelo limiar de Otsu, simplificando e acelerando a criação de conjuntos de dados. Na interface da ferramenta será possível escolher entre diferentes índices de vegetação e aplicar pós-processamentos, como erosão e dilatação.

As imagens resultantes desses processamentos geralmente correspondem bem ao objeto de interesse. No entanto, em algumas situações, é necessário ajustar manualmente o *threshold*. A ferramenta também permite esse ajuste, além de oferecer a visualização da máscara

sobre a planta, uma imagem em preto e branco mostrando apenas a máscara e o histograma da imagem para auxiliar no ajuste manual do threshold se necessário.

A ferramenta poderá auxiliar outros projetos que envolvam a criação de conjunto de dados que podem ser semi-automatizados utilizando os índices de vegetação que ressaltam a cor verde, o que é muito comum em problemas que envolvem plantio.

3.2.3 Metodologia para a construção do conjunto de dados de carimbo

A necessidade de criar um conjunto de dados sintético surge devido ao tempo disponível para a realização do trabalho e a disponibilidade de diferentes plantas daninhas com diferentes tamanhos. Um dataset completo, que abrange diferentes épocas do ano, com plantas de diferentes tamanhos e em diversos tipos de solo, poderia levar meses ou até anos para ser construído. No entanto, com a técnica de carimbo, é possível construir um conjunto de dados sintético de plantas daninhas infestando uma plantação de milho de maneira rápida, utilizando os dois conjuntos apresentados anteriormente: o Western Paraná Weeds e o Cornspacing.

Para cada planta daninha, foi aplicada sua máscara específica para isolar apenas os pixels que representam a planta de interesse. As plantas mascaradas foram então “carimbadas“ em uma imagem do Cornspacing que representa a cultura de milho. Em seguida, foram obtidos apenas os pixels que representam a cultura do milho na imagem selecionada do Cornspacing. Como as máscaras das plantas de milho já foram fornecidas no dataset, bastou aplicá-las na imagem. Após essa aplicação, as plantas de milho mascaradas foram sobrepostas às imagens contendo as plantas daninhas, de modo que o milho sempre fique por cima das ervas, considerando que a cultura geralmente ultrapassa a altura das plantas daninhas.

Depois de segmentar cada erva daninha de maneira semi-automática, foi possível distribuí-las nas imagens do cornspacing. Para obter uma configuração mais realista, as ervas daninhas devem ser espalhadas obedecendo a uma distribuição normal dentro de elipses que não se sobrepõem. Além disso, alguns pontos serão distribuídos de forma aleatória para inserir ervas daninhas, aproximando-se ainda mais de uma configuração real.

Em cada imagem, foram geradas até seis elipses de distribuição normal, com um tamanho mínimo de 12% da menor dimensão da imagem do Cornspacing e um tamanho máximo de 100% da mesma largura. O desvio padrão da distribuição é de 50% do valor do raio da elipse. Foi configurada uma variável de densidade de plantas daninhas, representando a quantidade de ervas em relação à área da elipse, obedecendo a seguinte equação:

$$n^{\circ} \text{ de pontos} = \frac{\text{área da elipse} \times \text{densidade}}{100{,}000}. \tag{21}$$

Os pontos aleatórios fora das áreas sombreadas pelas elipses foram distribuídos de forma randômica, podendo aparecer até 12 pontos aleatórios. Os intervalos de valores de cada parâmetro permitem a criação de imagens com grande variedade, desde imagens com poucas

unidades de plantas daninhas e poucas espécies, até imagens de plantações completamente infestadas com diferentes espécies.

Contudo, a grande variedade no tamanho das plantas pode causar problemas de escala na segmentação, onde plantas maiores são mais facilmente identificadas do que as menores, e algumas podem até ser perdidas no processo de redimensionamento. Estudos sugerem dividir essas imagens de alta resolução em imagens menores, propondo a superposição das extremidades para criar redundância em certas áreas, facilitando a identificação do contexto do corte da imagem pelo modelo.

Dessa forma, as três mil imagens geradas a partir das plantas daninhas e das imagens do conjunto Cornspacing foram divididas em imagens menores, dependendo do comprimento da imagem, visto que cada uma possui dimensões distintas. Quando necessário, adicionou-se *paddings* nas imagens para que todos os cortes tenham o mesmo tamanho, sem distorcer as imagens.

3.2.4 Metodologia para construção do conjunto de dados sintético com simulação 3D

Para a construção de uma simulação para construção do conjunto de dados sintético, é essencial desenvolver um ambiente digital que seja capaz de replicar fielmente as características do ambiente real. No âmbito deste projeto, o foco foi dado no desenvolvimento de um conjunto de dados sintéticos em um modelo digital 3D, com o intuito de utilizar como complemento nas imagens de treinamento do modelo.

A utilização de um ambiente virtual em 3D proporciona uma vantagem significativa, permitindo uma melhoria substancial no conjunto de dados de treinamento ao explorar diversas características presentes em cenários reais, tais como variações na iluminação e em diferentes condições adversas (POTTIER *et al.*, 2023).

Para atingir esse objetivo, o Blender foi utilizado para realizar a simulação. Neste processo, vários fatores foram levados em considereçãom incluindo o cultivo de milho, a iluminação ambiente, as características do solo e a presença de plantas daninhas. Detalhes sobre a metodologia adotada para cada grupo de atributos serão discutidos nas seções posteriores.

3.2.4.1 Folhas das plantas

O modelo de treinamento considera todas as características extraídas das daninhas, sendo essencial trazer as características físicas e de texturas das imagens reais para a simulação.

A primeira etapa executada é a separação de algumas folhas de cada espécie de planta daninha para serem utilizadas na simulação 3D. Inicialmente, foram utilizadas duas folhas de cada espécie para os testes iniciais. As imagens das folhas estam em formato png, sem fundo. Em seguida, foi necessário converter a imagem da folha para somente contorno em formato

SVG, que contém apenas informações do contorno da folha. Desta forma, o contorno será inserido para dentro do ambiente de simulação, onde passou por pré-processamentos para se tornar um molde 3D, chamado de *Mesh*. Esse processo utiliza algoritmos para conectar todos os pontos do contorno, formando faces entre eles, e o objeto resultante já apresenta características de uma figura 3D, como contorno e sombras. Foi fundamental garantir que o contorno das folhas das plantas fossem equivalentes aos modelos reais nesta etapa.

Em seguida, se tornou necessário trazer as informações de textura para o ambiente de simulação. Para isso, foi realizado um pré-processamento na imagem da folha original. O Blender requer alguns arquivos para acoplar uma textura a uma *Mesh*, sendo um deles uma imagem do tipo Normal, que contém informações de profundidade da imagem. Isso é essencial para o cálculo de sombras e reflexão da luz na folha da daninha. Para obter esse formato de imagem, foi necessário realizar um processo específico na imagem, esse processo é feito por diversos softwares que já tem funcionalidade implementada. Um dos algoritimos padrões que esses softwares usam é a detecção de gradientes de luminosidade.

O software selecionado para executar essa operação foi o GIMP, uma ferramenta livre para manipulação de imagens digitais, que já possui uma ferramenta para extração de mapas normais de imagens. Após essa etapa, é possível carregar o mapa normal e a imagem original das folhas em suas respectivas meshes no Blender, que realizou a conexão da textura com o objeto criado. Dessa maneira, foi possível simular as folhas das plantas daninhas, mantendo suas informações de textura e dimensionamento conforme as fotos retiradas de um ambiente real.

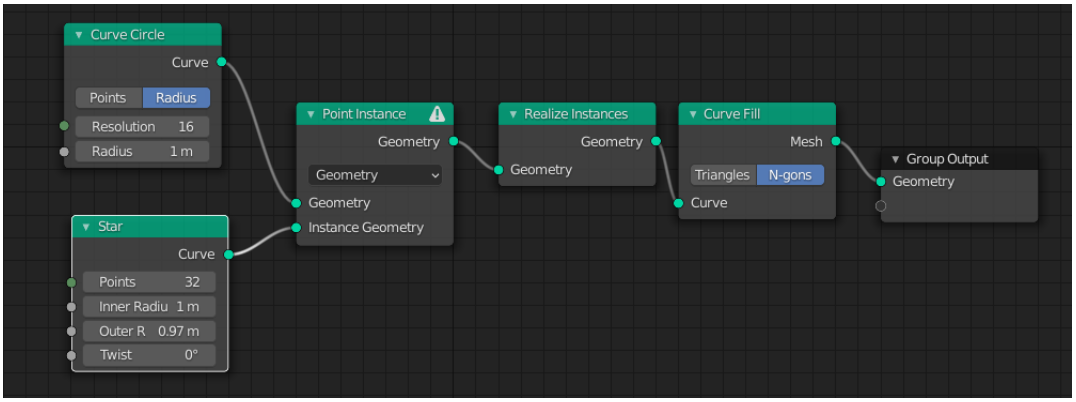
3.2.4.2 Estrutura das plantas

A estrutura da planta é outra parte fundamental do projeto, pois nessa etapa é possivel trazer parâmetros de construção para as plantas. As plantas podem apresentar diferentes tamanhos, quantidade de folhas e diferentes estrutura de caules.

Essa etapa foi modelada pelo Blender utilizando a feramenta conhecida como Geometry Nodes, presentes dentro da plataforma. A visualização desta ferramente está demonstrada na figura 13. O Geometry Nodes do Blender é uma ferramenta poderosa que permite aos usuários criar e manipular geometria de forma procedural. Usando nós visuais, é possível realizar uma ampla gama de operações, desde a geração até a modificação e distribuição da geometria.

A correta execução desta etapa permitiu a criação de uma estrutura base que serviu para as demais, sendo possível a modificação de alguns parâmetros como altura, diâmetro do caule, número de folhas e espaçamento das folhas.

Figura 13 – Exemplo Geometry Nodes Blender



Fonte: Autoria Propria.

3.2.4.3 Iluminação e Sombreamento

Um HDRI, abreviação de Imagem de Alta Faixa Dinâmica, é uma representação panorâmica de 360° que envolve uma imagem, proporcionando iluminação e fundo. Esse tipo de imagem é gerado ao combinar várias exposições diferentes da mesma cena, capturando desde as sombras mais escuras até os destaques mais brilhantes. O arquivo então é salvo no formato .hdr.

Com precisão de ponto flutuante, ele armazena os valores de cor RGB de cada pixel, permitindo registrar o brilho de forma detalhada. Uma imagem de alta faixa dinâmico típica possui 32 bits por pixel por canal, muito mais do que os 8 bits das imagens JPEG tradicionais.

Essas imagens são amplamente utilizadas na visualização 3D, pois ao adicionar um ambiente HDRI a um modelo 3D, é possível alcançar renderizações extremamente detalhadas e realistas, incluindo sombras e reflexos de iluminação. Sem o uso de HDRIs, modelos e cenas 3D podem parecer simples e menos profissionais.

Neste projeto, os HDRI utilizados foram extraídos da plataforma *Poly Heaven*[4], abrangendo uma variedade de HDRIs com foco em fazendas. Essas imagens foram escolhidas especificamente para capturar informações de iluminação de forma realista, visando aprimorar a simulação 3D. Ao utilizar diferentes HDRIs, foi possível obter uma gama diversificada de ambientes e condições de iluminação, o que contribui para uma representação mais precisa e envolvente na simulação 3D.

3.2.4.4 Simulação

Após a conclusão das etapas anteriores, será possível criar cenários completos que incluem todos os modelos das estruturas, plantas daninhas, além de conter todas as texturas e HDRI necessários para a renderização. Para alcançar uma simulação realista, é ideal utilizar o renderizador *Cycles* que oferece resultados de alta qualidade baseados em *ray tracing*. O

[4] Acessado em junho de 2024 pelo endereço https://polyhaven.com/hdris

processo começa com a modelagem 3D das estruturas no cenário, seguida pela aplicação de texturas para adicionar detalhes visuais. A iluminação desempenha um papel crucial na garantia da realidade da simulação, e para isso, deve-se utilizar imagens de alta faixa dinâmica (HDRI). O Cycles utiliza essas informações para renderizar a cena, produzindo imagens finais realistas e coesas. A técnica de *ray tracing* calcula a propagação dos elementos de luz no cenário, levando em consideração os normal maps das texturas trazendo mais realismo nos reflexos e sombras.

Para otimizar o processo de renderização no Blender, é possível aproveitar uma poderosa ferramenta: o interpretador Python embutido. O Blender oferece um ambiente que suporta a execução de scripts Python, proporcionando acesso a uma gama de módulos Python que interagem com os dados, classes e funções do *software*. Dessa forma, uma metodologia eficaz para otimizar as simulações deve considerar a capacidade de renderizar um grande número de imagens, enquanto varia os principais parâmetros. Estes incluem o número e variedades de ervas daninhas, as diferentes variedades de milho, e os cenários de iluminação. A integração do Blender com Python possibilita automatizar esse processo de forma eficiente, permitindo a criação de scripts que ajustam automaticamente os parâmetros de renderização para gerar as imagens desejadas.

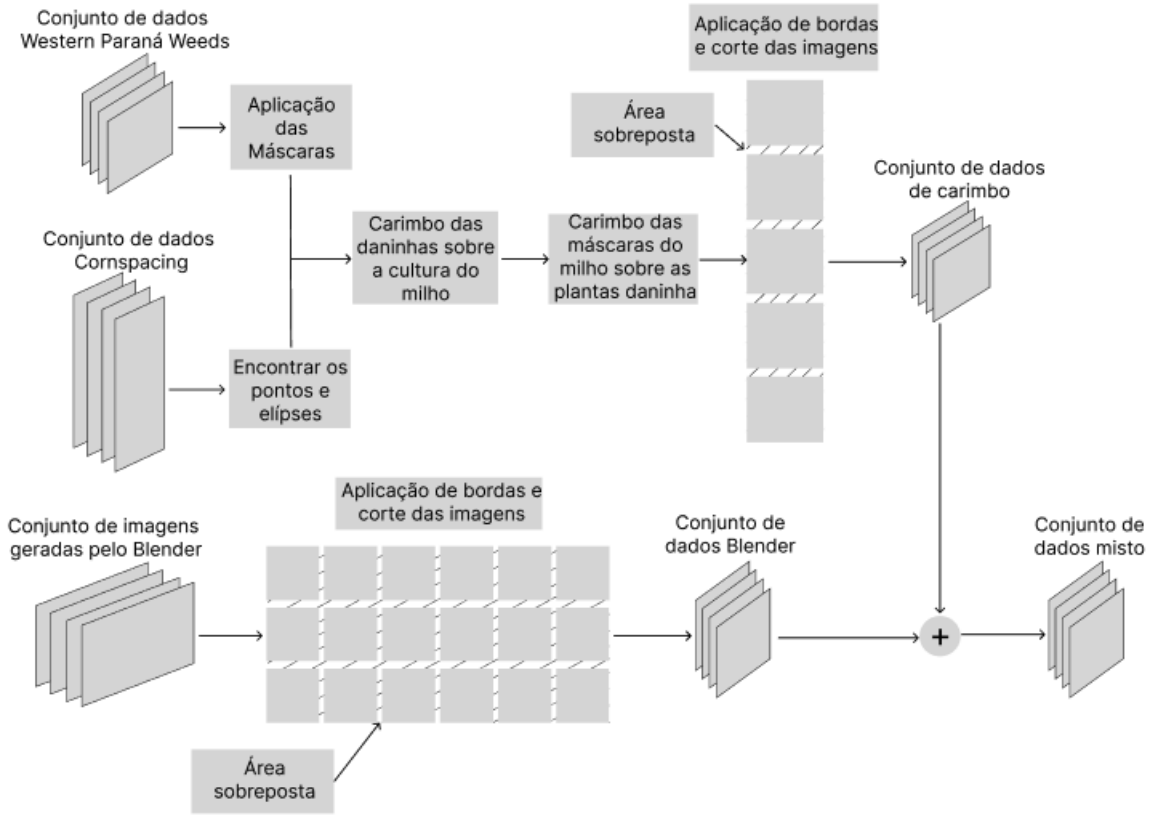
3.2.4.5 Máscaras

Para preparar as imagens simuladas para o treinamento dos modelos de segmentação semântica, é fundamental criar máscaras correspondentes a cada imagem. Essa etapa deve ser integrada ao algoritmo de automação para garantir eficiência e consistência no processo. Portanto, é necessário realizar renderizações adicionais para capturar essas informações, considerando que as anotações das máscaras devem conter detalhes das diferentes espécies de plantas daninhas presentes nas imagens simuladas. Essa abordagem assegura que as máscaras geradas sejam precisas e contemplem a diversidade das plantas presentes, o que é crucial para o treinamento eficaz dos modelos de segmentação semântica.

3.2.5 Metodologia para a construção do conjunto de dados misto

O conjunto de dados misto será gerado a partir da concatenação dos outros dois conjuntos de dados anteriormente descritos. O conjunto de treinamento conterá imagens geradas das duas maneiras propostas neste trabalho. No entanto, o conjunto de teste conterá apenas imagens criadas a partir da técnica de carimbo. O fluxo para criação do conjunto de dados pode ser melhor observado na figura 14.

Figura 14 – Fluxo para construção dos conjuntos de dados de Carimbo, gerados através do Blender e o conjunto de dados Misto.



Fonte: Autoria própria (2024).

3.2.6 Metodologia para construção de pipeline de segmentação

A pipeline de segmentação será projetada e configurada para testar diferentes abordagens na solução do problema de segmentação de plantas daninhas e culturas de milho. As variações incluem mudanças nas funções de perda com o objetivo de comparar o desempenho de diferentes configurações.

Em seguida, serão aplicadas técnicas de aumento de dados (*data augmentation*) ao conjunto de treinamento. As técnicas de aumento de dados discutidas anteriormente incluem rotação, variação de brilho e contraste, espelhamento horizontal e desfoque gaussiano (*blur*). Essas técnicas são realizadas de forma aleatória, com uma probabilidade de 50% para a maioria das transformações e uma probabilidade de 10% para o desfoque gaussiano. O objetivo dessas técnicas é aumentar a robustez do modelo, expondo-o a diversas variações das imagens de entrada.

Para definir a função de perda que melhor segmenta as plantas presentes nas imagens, serão configuradas duas funções de perda: *Focal Loss* e *Cross Entropy*. A Focal Loss é particularmente eficaz para lidar com datasets desbalanceados, focando mais nos exemplos difíceis, enquanto a Cross Entropy é amplamente utilizada em tarefas de classificação e segmentação devido à sua simplicidade e eficácia.

O otimizador escolhido para ajustar os pesos do modelo é o Adam, que utiliza o algoritmo de gradiente descendente para minimizar a função de perda. A taxa de aprendizado foi configurada para 1×10^{-4}, um valor que equilibra a velocidade de convergência com a estabilidade do treinamento.

O modelo será treinado por 35 épocas. Durante o treinamento, e como *encoder* será utilizado o melhor classificador da etapa de classificação de plantas daninha. Além disso, métrica MIoU (mean Intersection over Union) será calculada em cada época para avaliar o desempenho do modelo. As configurações da época com o maior valor de MIoU serão salvas, garantindo que a melhor versão seja armazenada.

Este processo se repetirá por 3 vezes com os 3 conjuntos de dados criados: o gerado através da técnica de carimbo, o gerado digitalmente pelo Blender e o conjunto misto, de forma que ao final do treinamento dos modelos, sejam comparadas 6 possibilidades de configuração para a segmentação do subconjunto de teste do conjunto de dados por carimbo que representa o nosso conjunto de dados reais. Essa metodologia é visualmente descrita na figura 14.

A tabela 3 resume as variações de cada um dos experimentos. Haverá diferenças na criação dos conjunto de dados, mas todos eles serão testados no mesmo subconjunto de teste que será gerado através da técnica de carimbo.

Tabela 3 – Descrição dos experimentos realizados na pipeline de segmentação

Experimento	Descrição
Carimbo com a Cross Entropy (Carimbo CE)	Neste experimento, o conjunto de dados é gerado através da técnica de carimbo e a função de perda utilizada é a Cross Entropy. O modelo gerado é testado com o subconjunto de teste também gerado através da técnica de carimbo.
Carimbo com a Focal Loss (Carimbo FL)	Neste experimento, o conjunto de dados é gerado através da técnica de carimbo e a função de perda utilizada é a Focal Loss. O modelo gerado é testado com o subconjunto de teste também gerado através da técnica de carimbo.
Blender com a Cross Entropy (Blender CE)	Neste experimento, o conjunto de dados gerado através do Blender é utilizado para treinar o modelo de segmentação com a Cross Entropy como função de perda. O modelo gerado é testado no subconjunto de teste gerado através da técnica de carimbo.
Blender com a Focal Loss (Carimbo FL)	Neste experimento, o conjunto de dados gerado através do Blender é utilizado para treinar o modelo de segmentação com a Focal Loss como função de perda. O modelo gerado é testado no subconjunto de teste gerado através da técnica de carimbo.
Misto com a Cross Entropy (Misto CE)	O conjunto de dados de treinamento é gerado através da concatenação dos dois conjuntos de dados gerados de forma sintética: O de carimbo e o gerado através do Blender. A função de perda utilizada é a Cross Entropy e o modelo gerado é testado no subconjunto de teste contento apenas imagens geradas através da técnica de carimbo.
Misto com a Focal Loss (Misto FL)	O conjunto de dados de treinamento é gerado através da concatenação dos dois conjuntos de dados gerados de forma sintética: O de carimbo e o gerado através do Blender. A função de perda utilizada é a Focal Loss e o modelo gerado é testado no subconjunto de teste contento apenas imagens geradas através da técnica de carimbo.

Fonte: Autoria própria (2024).

4 RESULTADOS

Esse capítulo dedica-se a mostrar os resultados obtidos em cada uma das etapas propostas, evidenciando as dificuldades e problemas encontrados durante sua execução.

4.1 Resultados da classificação

A tarefa de classificação foi executada para a identificação entre as 8 espécies de plantas daninhas contidas no conjunto de dados, bem como para a classificação entre as 14 famílias. Para essa tarefa, foram utilizadas seis das principais arquiteturas de classificação: VGG19, ResNet50, ResNet101, ResNet152, MobileNetV2 e MobileNetV3. Todas estas arquiteturas foram previamente treinadas com o conjunto de dados ImageNet, o que proporciona uma base sólida e acelera o processo de treinamento ao reutilizar pesos já ajustados em um grande e diversificado conjunto de imagens.

Para maximizar a precisão da classificação, foram testadas duas funções de perda distintas: *Focal Loss* e *Weighted Cross Entropy*. A *Focal Loss* foi escolhida devido à sua capacidade de lidar com conjunto de dados desbalanceados, focando mais nos exemplos difíceis. Por outro lado, a *Weighted Cross Entropy* atribui pesos maiores às classes menos representadas, equilibrando a contribuição de cada classe na função de perda. Todos os modelos foram treinados por 20 épocas.

Tabela 4 – Resultados da classificação de famílias das plantas daninha

Loss	Modelo	Precisão	Recall	F1-score	Precisão Ponderada	Recall Ponderada	F1-score Ponderada
Focal loss	VGG19	0,83	0,84	0,83	0,88	0,87	0,87
	Resnet50	0,94	0,96	0,94	0,96	0,96	0,96
	Resnet101	0,97	0,95	0,96	0,97	0,97	0,97
	Resnet152	0,94	0,95	0,94	0,96	0,96	0,96
	MobileNetV2	0,91	0,90	0,90	0,92	0,91	0,91
	MobileNetV3	0,96	0,96	0,96	0,98	0,97	0,97
Weighted CE	VGG19	0,86	0,88	0,86	0,90	0,89	0,89
	Resnet50	1,00	0,98	0,99	0,99	0,99	0,99
	Resnet101	0,92	0,95	0,93	0,96	0,95	0,95
	Resnet152	0,96	0,96	0,96	0,97	0,97	0,97
	MobileNetV2	0,77	0,66	0,81	0,80	0,81	0,77
	MobileNetV3	0,94	0,98	0,96	0,97	0,97	0,97

Fonte: Autoria própria (2024).

Entre os modelos propostos, todos apresentaram comportamentos semelhantes para a classificação quando comparados seus resultados entre famílias e espécies. Para a atividade mais complexa, que é a classificação de espécies, destacaram-se as arquiteturas ResNet (50, 101 e 152), com média geral de 0,97 para a classificação de famílias e espécies, e MobileNetsV3, também com média de 0,97 para ambas as atividades de classificação.

Tabela 5 – Resultados da classificação de espécies das plantas daninha

Loss	Modelo	Precisão	Recall	F1-score	Precisão Ponderada	Recall Ponderada	F1-score Ponderada
Focal loss	VGG19	0,76	0,67	0,68	0,81	0,69	0,71
	Resnet50	0,96	0,97	0,96	0,97	0,97	0,97
	Resnet101	0,96	0,95	0,96	0,97	0,97	0,97
	Resnet152	0,95	0,97	0,95	0,96	0,96	0,97
	MobileNetV2	0,83	0,81	0,81	0,85	0,84	0,84
	MobileNetV3	0,96	0,96	0,96	0,95	0,97	0,97
Weighted CE	VGG19	0,82	0,78	0,79	0,85	0,82	0,83
	Resnet50	0,96	0,96	0,96	0,97	0,97	0,97
	Resnet101	0,97	0,98	0,97	0,98	0,98	0,98
	Resnet152	0,97	0,98	0,97	0,98	0,98	0,98
	MobileNetV2	0,81	0,79	0,75	0,86	0,76	0,75
	MobileNetV3	0,98	0,97	0,97	0,98	0,98	0,98

Fonte: Autoria própria (2024).

Não houve grandes variações ao se comparar as duas funções de perda. A *Focal Loss* apresentou uma média geral de 0,91 para a classificação de famílias e 0,90 para a classificação de espécies, enquanto com *Weighted Cross Entropy* obtivemos resultados semelhantes, com média geral de 0,90 para a classificação de famílias e 0,92 para a classificação de espécies.

A matriz de confusão (Tabela 6) evidencia a facilidade com que a rede ResNet152, previamente treinada com o conjunto de imagens ImageNet, classifica objetos extremamente semelhantes. Entretanto, há um certo nível de confusão ao classificar a espécie Cenchus echinatus, pois em 20% das tentativas, o modelo previu erroneamente a classe Rhyncheytrumrepens, uma espécie pertencente à mesma família. Ao analisar as imagens destas duas famílias é evidente o quão semelhantes elas são, principalmente na fase adulta.

Tabela 6 – Matriz de confusão do modelo ResNet152 para classificação das 14 espécies

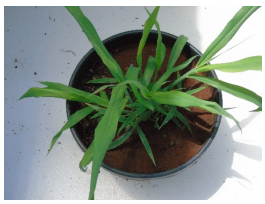
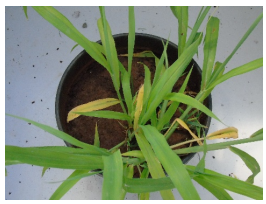
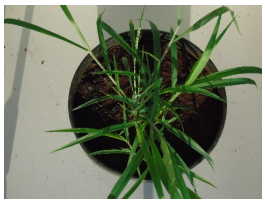
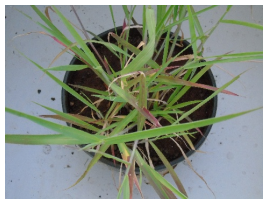
	Amaranthus hyb.	Bidens-pilosa	Cenchus echin.	Choris barbata	Cyclospermum lept.	Digitaria ins.	Ipomoea purp.	Leidium virg.	Leonurus sib.	Physalis angul.	Rhynchelytrum.	Sonchus oler.	Sorghum arun.	Tridax proc.
Amaranthus hyb.	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Bidens-pilosa	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cenchus echin.	0.00	0.00	0.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00
Choris barbata	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Cyclospermum lept.	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Digitaria ins.	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Ipomoea purp.	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Leidium virg.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
Leonurus sib.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00
Physalis angul.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
Rhynchelytrum.	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00	0.00	0.00
Sonchus oler.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00
Sorghum arun.	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.00
Tridax proc.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Fonte: Autoria própria (2024).

Além disso, houve erros também na classificação da classes Sorghum arundinaceum, que previu erroneamente 7% do conjunto de teste como pertencente à espécie Digitaria insularis, e Rhyncheytrumrepens que previu erronemanete também 7% do conjunto de teste como pertencente à espécie Cenchus echinatus.

Todas as espécies onde houve erros de classificação pertencem à família Poaceae, que representa a maior parte do conjunto de dados e que possui a maior quantidade de espécies dentro da mesma família, totalizando 260 imagens e 5 diferentes espécies.

Tabela 7 – Visualização das espécies pertencentes à família Poaceae nos três primeiros meses de vida.

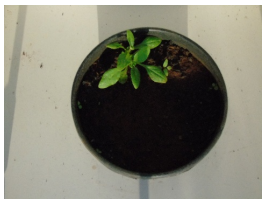
Espécie	Primeiro mês	Segundo mês	Terceiro mês
Cenchus echinatus			
Choris barbata			
Digitaria insularis			
Rhynchelytrum-repens			
Sorghum arundinaceum			

Fonte: Autoria própria (2024).

Os erros de classificação desta família podem ser explicados pela extrema semelhança entre suas 5 espécies, o que pode ser visualizado na tabela 7 e que apresentam uma estrutura folicular extremamente semelhante em todas as fases de vida. A única outra família que possui mais de uma espécie além da família Poaceae é a família Asteraceae e as 3 espécies contidas no conjunto de dados são facilmente diferenciáveis visualmente, pois há uma clara distinção até

mesmo para não profissionais da área. Essas diferenciações podem ser melhor observadas nas tabelas 7 e 8.

Tabela 8 – Visualização das espécies pertencentes à família Asteraceae nos três primeiros meses de vida.

Espécie	Primeiro mês	Segundo mês	Terceiro mês
Bidens Pilosa			
Sonchus Oleraceus			
Tridax Procumbens			

Fonte: Autoria própria (2024).

Apesar dos erros de classificação do modelo, é evidente que todas as arquiteturas com as diferentes funções de perda apresentaram bons resultados. Os modelos já previamente treinados com conjuntos de dados extensos como no caso do ImageNet não tiveram dificuldades em se adaptar para a classificação de objetos semelhantes como o caso de plantas daninha. Nesta etapa do projeto, onde buscávamos a confirmação de que seria possível classificar todas as plantas daninha, obtivemos sucesso sem grandes dificuldades.

4.2 Conjunto de dados sintéticos

Foram criados dois conjunto de dados sintéticos para fins de comparação. O primeiro deles foi criado com a técnica de carimbo, ou seja, as plantas daninha presentes no conjunto de dados *Western Paraná Weeds* foram “carimbadas“ nas imagens do conjunto *Cornpacing*. Já o segundo foi criado a partir de uma simulação 3D com auxílio da ferramente Blender.

4.2.1 Conjunto de dados de carimbo

A partir dos conjunto de dados *Cornspacing* e *Western Paraná Weeds 2020*, foi criado um novo conjunto de dados onde as 14 famílias de plantas daninha ficaram distribuídas na cultura de milho com dois diferentes solos. Utilizando a metodologia descrita para a criação

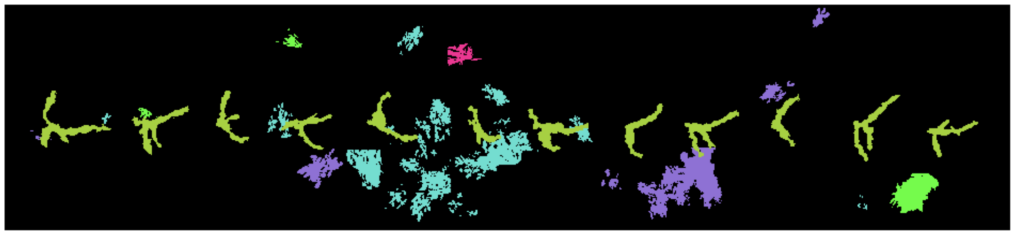
deste novo conjunto de dados, foi possível criar imagens semelhantes a que obteríamos em um ambiente real a fim de executar a atividade de segmentação. Um exemplo deste conjunto de dados sintético pode ser observado na figura 15.

As sombras e luzes se tornaram um fator de dificuldade durante este processo. Tentamos algumas abordagens para minimizar este problema, como a normalização dos histogramas das plantas daninhas e da imagem do *Cornspacing* e obtivemos algumas melhorias. No entanto, essa abordagem se mostrou ineficiente a nível de *hardware*, visto que esta normalização deveria ser feita em cada uma das imagens considerando todas as plantas daninhas que seriam adicionadas sobre a imagem, o que elevou o tempo de criação do novo conjunto de dados em mais de 30 segundos por imagem e, então, resolvemos abandonar esta ideia.

A falta da normalização do histograma pode ser um fator que facilitou o treinamento da rede, uma vez que há uma diferenciação clara na coloração das plantas daninha em relação ao restante da imagem. Além desta clara diferenciação de cores, a falta de tratamento de luz e sobra também são fatores que deixaram as imagens menos naturais.

Figura 15 – Exemplo de imagens do conjunto de dados gerado pela técnica de carimbo
(a) Imagem RGB

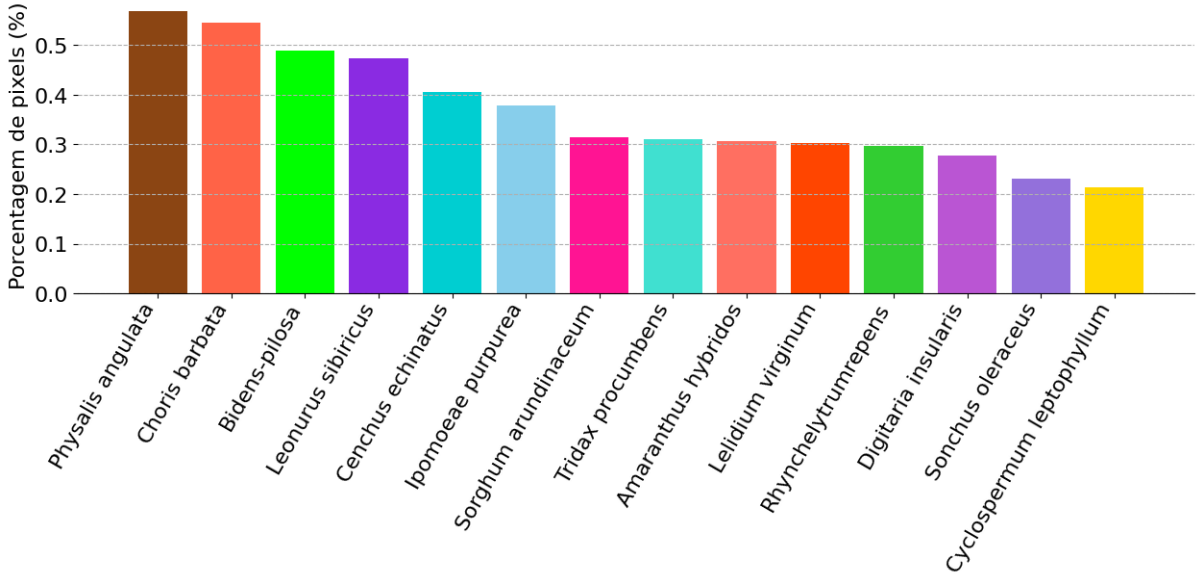
(b) Anotação da imagem



Fonte: Autoria própria (2024).

No total, foram criadas 3500 imagens com as imagens originais do conjunto de dados *Cornspacing* e que posteriormente foram cortadas respeitando a técnica de sobreposição, onde a área sobreposta representa 1/3 da largura total da imagem, gerando ao final um total de 17.476 imagens cortadas de dimensão 256x256 com uma variação extremamente alta de diferentes plantas daninha, nível de infestação e estilos de distribuição. A distribuição de classes dos pixels pode ser visualizada na figura 16.

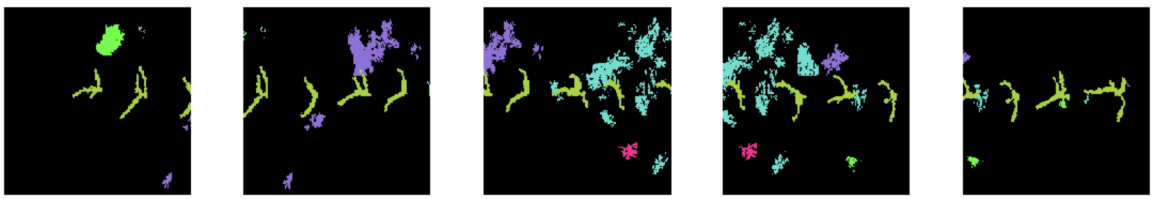
Figura 16 – Distribuição de pixels por classe: os valores estão organizados de maneira decrescente, com as cores representando cada uma das classes nas imagens anotadas. Devido aos seus valores discrepantes, as classes de fundo e milho foram ofuscadas, representando 90,5% e 4,3%, respectivamente.



Fonte: Autoria própria (2024).

Figura 17 – Exemplo de imagens após serem cortadas
(a) Imagem RGB cortada

(b) Anotação cortada



Fonte: Autoria própria (2024).

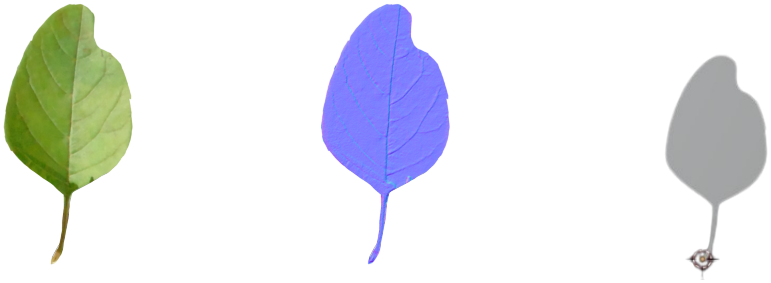
4.2.2 Simulação 3D

O processo de preparação do cenário para renderização é abrangente e requer conhecimentos variados sobre a ferramenta Blender. Na primeira etapa, realizamos a separação das folhas das plantas daninhas utilizando o GIMP. Posteriormente, avançamos para a construção das *meshes* com base nos contornos das folhas. Este procedimento é essencial para criar modelos tridimensionais precisos que serão utilizados na simulação e renderização dentro do

Blender. Além da imagem com a textura original, foi preciso criar os *normal maps* de cada folha. Com essa etapa concluída, tornou-se possível atribuir a textura correspondente à sua respectiva *mesh*. A figura 18 exemplifica visualmente a folha, o mapa normal desta folha e, por fim, a *mesh*.

Figura 18 – Exemplo folha, mapa normal e Mesh

(a) Folha **(b) Mapa Normal** **(c) Mesh**



Fonte: Autoria própria (2024).

Esse procedimento é repetido para cada folha de todas as espécies de plantas daninhas. Ao concluir esta etapa, obtivemos um conjunto de objetos 3D que representam as folhas individuais de cada planta. Para criar os modelos 3D das plantas, adotamos uma abordagem baseada em uma metodologia que emprega os *geometry nodes*. Nesse processo, são utilizados nós que realizam operações fundamentadas na distribuição de pontos. Isso nos permite estabelecer parâmetros controláveis que influenciam a aparência de cada planta.

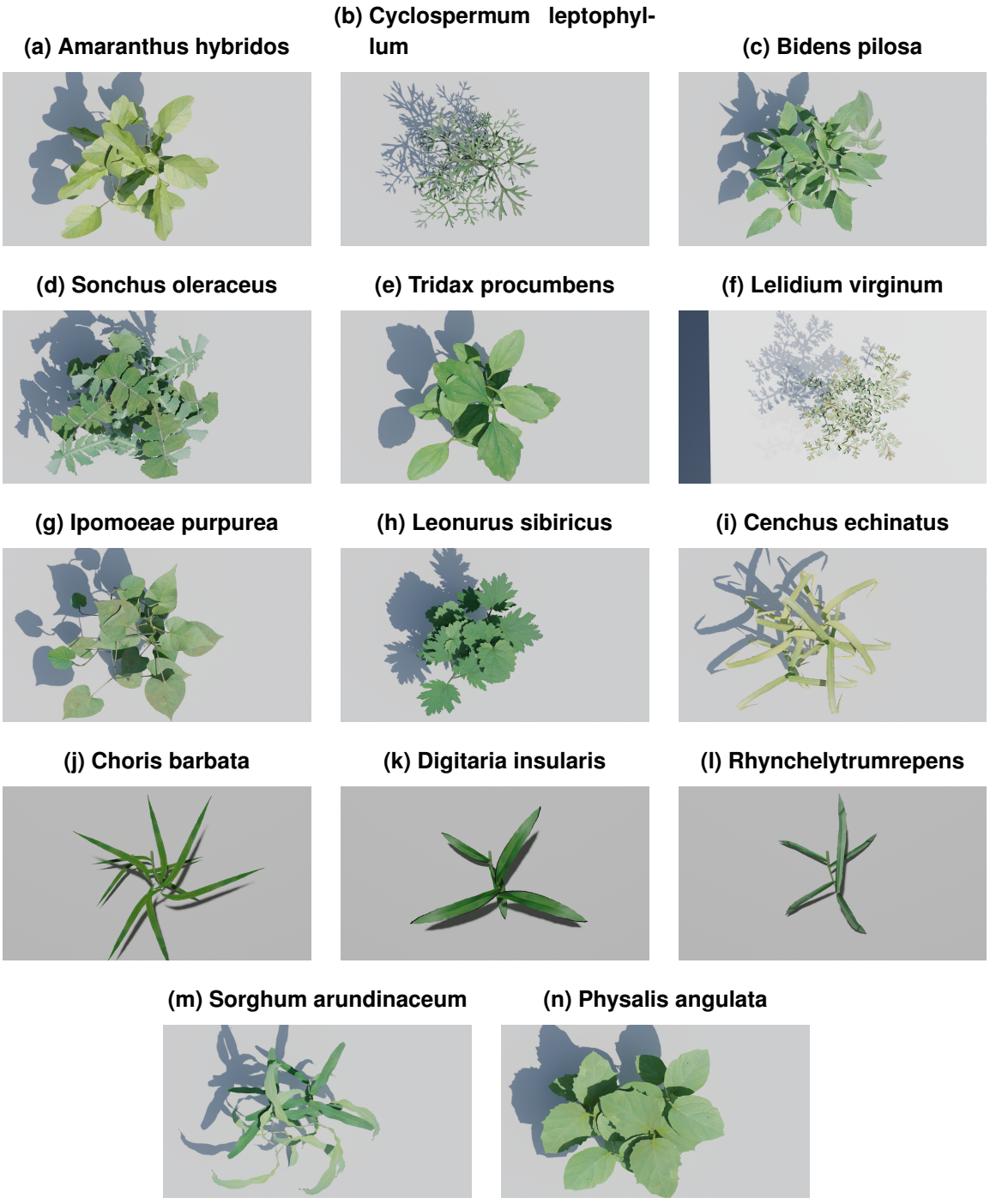
Por meio dessa metodologia, foi elaborado um caule que conecta as folhas de cada planta a ele. Esse caule pode ser ajustado em relação a parâmetros como espessura, textura, altura, posição e quantidade de folhas no ramo. Ao fim dessa etapa ja se tornou possível renderizar alguns exemplos de plantas.

Para a simulação, foi desenvolvido um script em Python utilizando a biblioteca do Blender. O script automatiza a renderização das imagens, criando todo o cenário da simulação, incluindo o solo, a iluminação por HDRIs, e a distribuição de milhos e plantas daninhas.

Alguns parâmetros ajudaram a trazer diversidade ao novo conjunto de dados: Para o milho, é possível configurar a distância entre os cultivares, a altura, o número de folhas e sua orientação; para as plantas daninhas, os parâmetros incluem a seleção da espécie, quantidade e distribuição. Outros parâmetros de simulação, como tipo de solo e iluminação, ajudam a aumentar a diversidade do dataset. O intervalo desses parâmetros foi definido utilizando medidas encontradas na vida real.

Após configurar a cena, o script renderiza a imagem e gera as máscaras correspondentes às classes presentes nela, deletando os elementos da cena em seguida. O algoritmo ajusta os parâmetros e repete o processo, resultando imagens sintéticas com diferentes tipos de plantas, solos e iluminações.

Figura 19 – Exemplo de imagens sintéticas geradas pelo Blender possuindo as mesmas espécies do conjunto Western Paraná Weeds

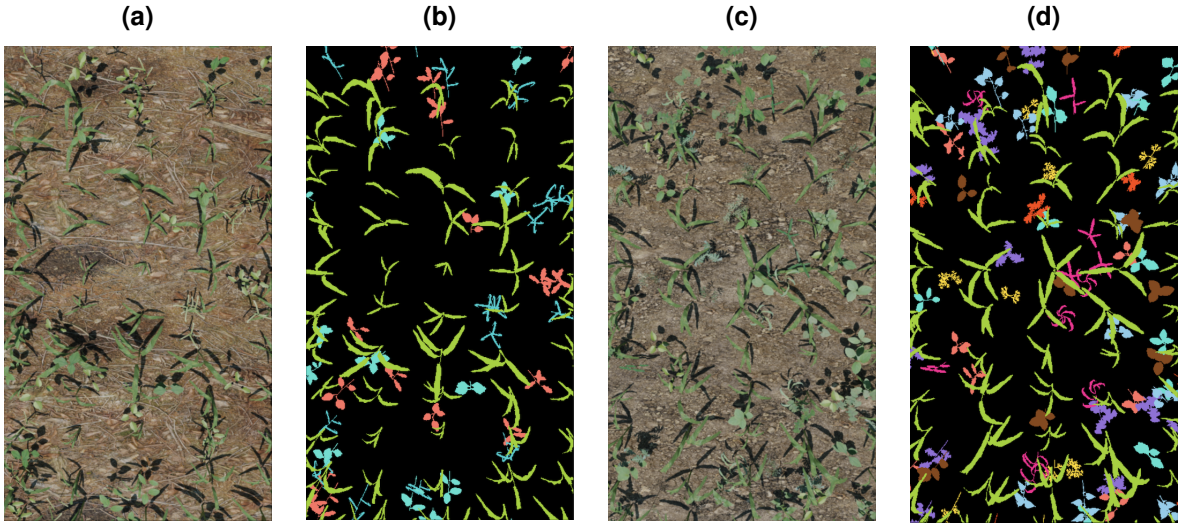


Fonte: Autoria própria (2024).

Em testes de benchmark, utilizando uma placa de vídeo NVIDIA 1060 3GB, foi apresentado um tempo de renderização de 40 segundos por imagem. Embora isso possa parecer custoso computacionalmente, há possibilidade de melhoria de tempo com hardware melhor.

Foram criadas ao todo 1000 imagens que foram posteriormente cortadas assim como as imagens geradas por carimbo. No entanto, por se tratarem de imagens maiores e com mais filheiras de milhos, foram cortadas em 3 linhas e 6 colunas. Logo, ao final deste processo,

Figura 20 – Exemplo da Simulação 3D e suas respectivas máscaras de segmentação



Fonte: Autoria própria (2024).

obtivemos um total de 18.000 imagens simuladas e cortadas, respeitando a sobreposição de 1/3 da imagem veriticalmente e horizontalmente.

As renderizações apresentaram ser semelhantes aos conjunto de dados originais, principalmente na fase adulta. No entanto, há plantas que variam suas cores durante a maturação e o algorítimo criado não abrangeu essa diferenciação durante o crescimento da planta. Além disso, há plantas que mudam sua distribuição dentro do vaso durante a maturação, como no caso das plantas pertencentes à família Poaceae que, em sua fase mais jovem, parece haver várias plantas dentro de um mesmo vaso, enquanto em sua fase adulta, é possível ver uma única unidade. Esta variação também não foi incluída dentro desta abordagem.

Apesar das dificuldades em abranger todas as fases da vida da planta, as renderizações criadas são semelhantes às plantas daninha e foi possível gerar imagens que se assemelham às originais inclusive em suas texturas e formatos, além de ser possível escalar a quantidade de imagens se necessário para um treinamento mais robusto.

4.3 Segmentação semântica

A rede Unet foi treinada com os três conjuntos de dados construídos de maneira sintética: o por carimbo, o gerado através da simulação do Blender e o conjunto de dados misto, gerado através da junção dos dois anteriores. Além disso, as funções de perda também foram alternadas para fins de comparação, de forma a executar todos os experimentos anteriormente descritos na tabela 3.

Com a realização dos experimentos descritos na tabela 3, foram extraídas as métricas de avaliação onde o MIoU pode ser visualizado na tabela 9.

Tabela 9 – Tabela de médias de IoU por classe em diferentes configurações de conjunto de dados e funções de perda

Classes	Carimbo (CE)	Carimbo (FL)	Blender (CE)	Blender (FL)	Misto (CE)	Misto (FL)
Background	0,9891	0,9895	0,9669	0,9655	0,9838	0,9889
Milho	0,8207	0,8258	0,3829	0,3609	0,7723	0,8220
Amaranthus Hybridos	0,7631	0,7202	0,0818	0,0616	0,6373	0,7767
Cyclospermum Leptophyllum	0,7128	0,7013	0,0203	0,0208	0,6099	0,7218
Tridax Procumbens	0,7381	0,6796	0,0000	0,0000	0,5637	0,7327
Sonchus Oleraceus	0,7247	0,7155	0,0400	0,0423	0,5686	0,7552
Bidens-pilosa	0,7990	0,8010	0,0002	0,0018	0,7078	0,8326
Lelidium Virginum	0,7616	0,7579	0,0241	0,0285	0,6407	0,7788
Ipomoeae Purpurea	0,7675	0,7104	0,0040	0,0015	0,6673	0,7907
Leonurus Sibiricus	0,8120	0,8041	0,0065	0,0070	0,7129	0,8226
Sorghum Arundinaceum	0,3458	0,2858	0,0000	0,0000	0,1758	0,2851
Rhynchelytrumrepens	0,7707	0,7605	0,0239	0,0463	0,6550	0,7833
Digitaria Insularis	0,7745	0,7345	0,0000	0,0000	0,6634	0,7782
Choris Barbata	0,8280	0,8117	0,0009	0,0065	0,7254	0,8325
Cenchus Echinatus	0,7656	0,7572	0,0543	0,0521	0,6496	0,7799
Physalis Angulata	0,8403	0,8069	0,0000	0,0001	0,7372	0,8361
Todos	0,7633	0,7414	0,1004	0,0997	0,6544	0,7698

Fonte: Autoria própria (2024).

As classes “Milho“, “Bidens-pilosa“, e “Choris Barbata“ apresentam consistentemente boas médias de IoU no conjunto de dados de Carimbo com ambas as funções de perda. No conjunto de dados Blender, várias classes como “Sorghum Arundinaceum“, “Digitaria Insularis“ e “Physalis Angulata“ têm médias de IoU próximas de zero, indicando que os modelos não conseguiram segmentar essas classes adequadamente. No conjunto de dados Misto, a classe“Physalis Angulata“ tem uma das maiores médias de IoU com 0,8361 para Focal Loss.

Os modelos treinados com o conjunto de dados Carimbo apresentaram resultados acima de 0,74 de MIoU considerando todas as classes. Para o conjunto de dados Misto, apesar de terem sido treinados com mais imagens e maior custo computacional, já que se trata da concatenação dos dois conjuntos de dados anteriores, não houve melhora significativa se comparados com os resultados para o conjunto de dados Carimbo.

O MIoU foi consideravelmente melhorado pela função de perda *focal loss* ao se comparar os resultados dos modelos do conjunto de dados Misto, no entanto, não houve melhoria ao se comparar os resultados para os modelos treinados com o conjunto de dados Carimbo. Entretanto, o resultado das duas funções de perda para esse conjunto de dados apresentaram um valor alto. Como aumentamos o grau de dificuldade durante o treinamento do conjunto de dados Misto, a função de perda *focal loss* foi essencial para que o modelo obtivesse melhor resultado em uma gama ainda maior de imagens. Contudo, essa robustez não foi justificada, dado que o modelo treinado apenas com o conjunto de dados de carimbo já obteve resultados altos se comparado ao estado da arte.

Como citados anteriormente, um fator que torna a segmentação mais fácil para o modelo, é a falta da normalização das cores e a falta de tratamento de luz e sombra, o que, em ambientes reais, seria diferente, uma vez que as cores seriam normalizadas e os aspectos de luz e sombra mais realistas.

Tabela 10 – Matriz de confusão do experimento Blender CE considerando apenas as classes Daninhas, Background e Milhos

	Back.	Milhos	Daninhas
Back.	0.98	0.01	0.02
Milhos	0.10	0.57	0.34
Weeds	0.16	0.30	0.54

Tabela 11 – Matriz de confusão do experimento Blender FL considerando apenas as classes Daninhas, Background e Milhos

	Back.	Milhos	Daninhas
Back.	0.98	0.01	0.01
Milhos	0.12	0.61	0.27
Weeds	0.16	0.27	0.57

O modelo treinado apenas com o conjunto de dados do blender se mostrou promissor, uma vez que os resultados de segmentação aparentaram capturar corretamente a existência de um objeto, mas o modelo erra ao tentar classificá-lo, o que influencia negativamente para a métrica proposta.

Tabela 12 – Tabela de médias de IoU por classe considerando todas as plantas daninha como uma única classe

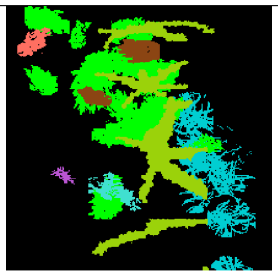
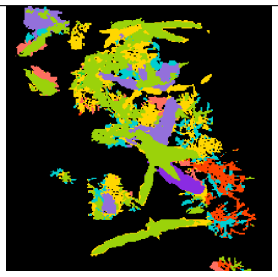
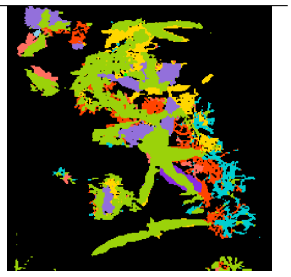
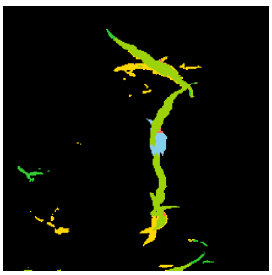
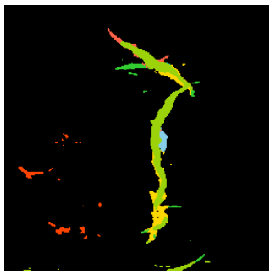
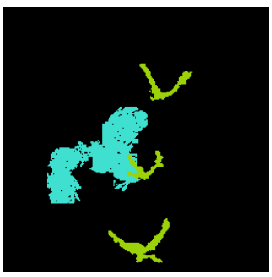
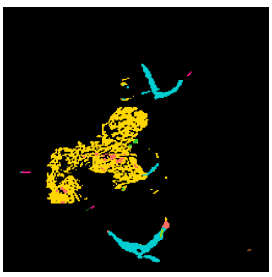
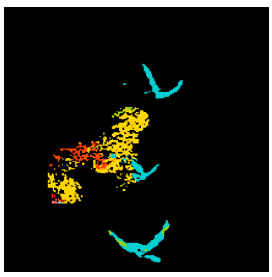
Classes	Carimbo (CE)	Carimbo (FL)	Blender (CE)	Blender (FL)	Misto (CE)	Misto (FL)
Background	0,9891	0,9895	0,9669	0,9655	0,9838	0,9889
Milhos	0,8207	0,8258	0,3829	0,3609	0,7723	0,8220
Daninhas	0,9521	0,9500	0,4078	0,3442	0,8975	0,9469
Todos	**0,9206**	**0,9218**	**0,5859**	**0,5568**	**0,8845**	**0,9193**

Fonte: Autoria própria (2024).

Para fins de comparação, geramos alguns resultados considerando todas as plantas daninha como uma única classe e observamos que o modelo treinado unicamente com o conjunto de dados do Blender acerta mais do que 0,5 das classificações dos pixels dos milhos e das plantas daninhas, além de apresentar um IoU superior a 0,36 para a classe de milho e superior a 0,30 para a classe de daninhas unificadas com ambas configurações de funções de perda, o que é considerado um bom resultado, dado o cenário.

Ao se analisar as imagens contidas nas tabela 10, 11 e 13, é evidente que o modelo gerado a partir dos dados gerados pelo Blender é promissor, mas ainda há alguns aspectos que podem ser melhorados em trabalhos futuros. Todas as simulações criadas foram feitas apenas a partir da vista superior das plantas daninhas e das imagens do conjunto *Cornspacing*, o que limita a fidelidade da simulação e que pode ter contribuido negativamente para os resultados deste experimento. Além disso, em capturas de imagens futuras, a documentação pode incluir o horário do dia e condições do tempo para uma simulação mais fidedigna.

Tabela 13 – Resultados da inferência dos modelos treinados com o conjunto de dados gerados pelo Blender com diferentes funções de perda e testados no subconjunto de teste do conjunto criado através da técnica de carimbo

Imagens	Máscaras	Blender (CE)	Blender (FL)
			
			
			

Fonte: Autoria própria (2024).

A abordagem de conjuntos de dados sintéticos para treinamentos de modelos que se aplicam em ambientes reais ainda está emergindo e estudos para a aplicação desta técnica na agricultura utilizando simulações 3D é inexistente. Procuramos abordar métodos de geração de conjunto de dados que poderão ser utilizados para treinar modelos e, posteriormente, estudar suas aplicações em ambientes reais para que assim se crie maior variedade de dados e se economize tempo para a criação de conjunto de dados que abrangem diferentes condições climáticas, de luz, solo e tempo de cultivo.

Tabela 14 – Resultados da inferência dos 6 modelos com diferentes funções de perda e treinados com diferentes conjuntos de treino e testados no subconjunto de teste do conjunto criado através da técnica de carimbo

Imagens	Máscaras	Carimbo (CE)	Carimbo (FL)	Blender (CE)	Blender (FL)	Misto (CE)	Misto (FL)
							

Fonte: Autoria própria (2024).

5 CONCLUSÃO

Este projeto teve como objetivo desenvolver uma metodologia baseada em visão computacional para a classificação das principais ervas daninhas presentes na região oeste do estado do Paraná, além da identificação dessas pragas na cultura de milho, considerando um cenário onde não se dispõe de um conjunto de dados orgânico para o treinamento do modelo.

O projeto introduz a utilização de conjuntos de dados sintéticos para a implementação em cultivos agrículas e conta com a *pipeline* de classificação e segmentação semântica de plantas daninhas e da cultura de milho, empregando as principais metodologias presentes na literatura para a execução destas atividades, como a utilização das funções de custo *Cross-Entropy* e *Focal Loss*.

A metodologia para construção de conjunto de dados sintéticos foi um estudo inovador, buscando formas de treinar modelos que não têm acesso a fontes de dados orgânicos. Tanto o modelo de carimbo quanto o de simulação 3D têm espaço para melhorias, mas já apresentaram resultados promissores para a simulação de cenários adversos utilizando elementos das imagens reais.

A metodologia de avaliação do trabalho também propôs *benchmarks* entre diferentes modelos difundidos na literatura para a classificação de plantas visualmente bastante parecidas (portanto de difíssil classificação), obtendo resultados de precisão de acima 0,95 em 5 modelos da literatura. Lembrando que esses modelos já foram previamente treinado com um grande volume de dados, trazendo assim já muito entendimento sobre diversas carcterísitcas como contorno, bordas etc.

A metodologia da segmentação semântica buscava avaliar os modelos utilizando diferentes funções de custo, além de investigar se a utilização de imagens sintéticas poderia ajudar a trazer mais diversidade aos dados, como um tipo de data augmentation, visando melhorar o entendimento do modelo. Essa abordagem ainda não se mostrou eficiente, dado o tempo gasto na criação do conjunto de dados de simulação 3D.

Ao final do projeto, observou-se que o modelo treinado com a função de perda *Focal Loss* e o conjunto de dados misto foi o que apresentou o melhores resultados. Porém, o acréscimo das imagens construídas no Blender como data augmentation apresentou uma melhoria de menos de 0,01% em relação ao modelo treinado apenas com o conjunto de carimbo e função de perda *Cross Entropy*. Dessa forma, não houve melhoria significativa no modelo com a utilização de simulação 3D como aumento de dados. Isso pode ser derivado de diversos fatores, como os muitos parâmetros ajustados na renderização que partem da interpretação humana das características importantes para distinguir plantas, como o contorno, ou textura. Entretanto, como o modelo aprende os kernels que mais diferenciam as classes de forma empírica, as renderizações podem não estar trazendo informações locais importantes que apareceriam em imagens reais. Além disso, as simulações foram feitas completamente a partir da vista superior das plantas daninhas, o que impossibilita a representação fidedigna das mesmas.

Apesar dos resultados da simulação 3D, há parâmetros que ainda podem ser ajustados e melhorados para obtenção de melhores resultados no problema abordado neste trabalho. No entanto, a construção de conjunto de dados sintéticos através desta abordagem ainda é um campo em construção e trata-se de um campo promissor e que facilitará pesquisas e treinamentos de modelos complexos no futuro.

O projeto abordou um estudo de uma área em expansão no território nacional e global. Algoritimos de visão computacional como os abordados neste projeto são a base para tecnologias emergentes que buscam otimizar o manejo desses causadores de prejuízo. Este trabalho constitui a base necessária para a implementação de projetos mais complexos envolvendo agricultura de precisão, além de trazer os conhecimentos para aplicar em qualquer projeto de classificação e segmentação semântica.

As ferramentas criadas para a segmentação semi-automática das plantas, assim como os algoritimos criados em cada uma das *pipelines* descritas estão disponibilizados no github através dos links https://github.com/CarlosModinez/easy-plant-segmentation e https://github.com/CarlosModinez/brazilian_weeds. Com a publicação deste trabalho em um artigo mais aprofundado sobre o tema, as bases de dados também serão disponibilizadas no futuro.

REFERÊNCIAS

Analytics Vidhya. **Analytics Vidhya: Data Science and Machine Learning Knowledge Hub**. https://www.analyticsvidhya.com. Acessado em: 19 de junho de 2024.

ARIAS, D. *et al.* **Agriculture Productivity Growth in Brazil: Recent Trends and Future Prospects**. 2017. Acesso em: 17 jun. 2024. Disponível em: https://openknowledge.worldbank.org/handle/10986/32202.

AYAZ, A.; AYTEKIN, A.; AKGüN, F. Intelligent and natural agriculture with industry 4.0. **Bartın Orman Fakültesi Dergisi**, v. 21, p. 938–944, 12 2019.

BADRINARAYANAN, V.; KENDALL, A.; CIPOLLA, R. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 39, n. 12, p. 2481–2495, 2017.

BELETE, N. A. de S. *et al.* Classification of weed in soybean crops using unmanned aerial vehicle images. **Universidade Federal de Rondônia, Universidade Católica Dom Bosco**, Cacoal/RO, Campo Grande/MS, Brasil, 2019.

BOJANIC, A. **As crescentes demandas da população mundial sobre a agricultura e sobre os recursos naturais do planeta, além do papel que o Brasil deverá exercer nesse cenário no futuro**. 2016. Conferência apresentada no 31° Congresso Nacional de Milho e Sorgo, Bento Gonçalves, RS, 26 de setembro de 2016. Disponível em: https://www.fao.org/brasil/noticias/detail-events/en/c/436508/. Acesso em: 17 jun. 2024.

BRILHADOR, A.; SERRARENS, D. A.; LOPES, F. M. A computer vision approach for automatic measurement of the inter-plant spacing. *In*: PARDO, A.; KITTLER, J. (Ed.). **Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications**. Cham: Springer International Publishing, 2015. p. 219–227. ISBN 978-3-319-25751-8.

CAI, S.; WU, Y.; CHEN, G. A novel elastomeric unet for medical image segmentation. **Frontiers in Aging Neuroscience**, v. 14, 2022. ISSN 1663-4365. Disponível em: https://www.frontiersin.org/articles/10.3389/fnagi.2022.841297.

Carbon Robotics. **About Carbon Robotics**. 2023. https://carbonrobotics.com/. Accessed: 2023-06-18.

CHEN, L.-C. *et al.* Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 40, n. 4, p. 834–848, 2018.

DUCKETT, T. *et al.* **Agricultural Robotics: The Future of Robotic Agriculture**. [*S.l.*], 2018.

FERREIRA. **Classificação de Plantas Daninhas em Lavouras de Soja Usando Imagens de Veículos Aéreos Não Tripulados**. 2017. Tese (Doutorado) — Universidade de Brasília, Brasília, Brasil, 2017.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 3rd. ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2008. ISBN 9780131687288.

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

GUIJARRO, M. *et al.* Automatic segmentation of relevant textures in agricultural images. **Computers and Electronics in Agriculture**, v. 75, n. 1, p. 75–83, 2011. ISSN 0168-1699.

HAGUE, T.; TILLETT, N.; WHEELER, H. Automated crop and weed monitoring in widely spaced cereals. **Precision Agriculture**, Kluwer Academic Publishers, v. 7, n. 1, p. 21–32, 2006. ISSN 1385-2256.

HE, K. *et al.* Deep residual learning for image recognition. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*S.l.*: *s.n.*], 2016. p. 770–778.

HOWARD, A. G. *et al.* Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

John Deere. **See & Spray Ultimate**. 2023. https://www.deere.com/en/sprayers/see-spray-ultimate/. Accessed: 2023-06-18.

KATAOKA, T. *et al.* Crop growth estimation system using machine vision. *In*: IEEE. **Advanced Intelligent Mechatronics, 2003. AIM 2003. Proceedings. 2003 IEEE/ASME International Conference on**. [*S.l.*], 2003. v. 2, p. b1079–b1083.

KIM, Y. H.; PARK, K. R. Mts-cnn: Multi-task semantic segmentation-convolutional neural network for detecting crops and weeds. **Computers and Electronics in Agriculture**, Elsevier, v. 199, p. 107146, 2022.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, arXiv, 2014.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *In*: **Advances in neural information processing systems**. [*S.l.*: *s.n.*], 2012. v. 25, p. 1097–1105.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, p. 436–444, 2015. Disponível em: https://doi.org/10.1038/nature14539.

LIN, T.-Y. *et al.* **Focal Loss for Dense Object Detection**. 2018.

LONG, J.; SHELHAMER, E.; DARRELL, T. Fully convolutional networks for semantic segmentation. *In*: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [*s.n.*], 2015. p. 3431–3440. Disponível em: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Long_Fully_Convolutional_Networks_2015_CVPR_paper.html.

MAO, A.; MOHRI, M.; ZHONG, Y. **Cross-Entropy Loss Functions: Theoretical Analysis and Applications**. 2023.

MEYER, G. E.; HINDMAN, T. W.; LAKSMI, K. Machine vision detection parameters for plant species identification. *In*: **Proceedings. SPIE**. [*S.l.*: *s.n.*], 1999. v. 3543, p. 327–335.

MILIOTO, A.; LOTTES, P.; STACHNISS, C. Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. **2018 IEEE International Conference on Robotics and Automation (ICRA)**, p. 2229–2235, 2018.

MOORTHY, S. Effective segmentation of green vegetation for resource-constrained real-time applications. **Wageningen academic publishers**, Precision agriculture, p. 93–98, 07 2015.

MüLLER, D.; SOTO-REY, I.; KRAMER, F. Towards a guideline for evaluation metrics in medical image segmentation. **BMC Research Notes**, Springer, v. 15, n. 1, p. 210, 2022.

NATHAN, P.; WAMPLER, D. **Hardware > Software > Process: Data Science in a Post-Moore's Law World**. [*S.l.*: *s.n.*], 2023. With contributions from Jim Scott.

NETO, J. A. C. **A combined statistical-soft computing approach for classification and mapping weed species in minimum-tillage systems**. 2004. Tese (Doutorado) — Universidade de Nebraska, 2004.

PEDRINI, H.; SCHWARTZ, W. **Análise de imagens digitais: princípios, algoritmos e aplicações**. [*S.l.*]: Thomson Learning, 2008. ISBN 9788522105953.

POTTIER, C. *et al.* Developing digital twins of multi-camera metrology systems in blender. **Measurement Science and Technology**, v. 34, n. 7, p. 075001, 2023. Accessed on 1 Jun. 2024. Disponível em: https://doi.org/10.1088/1361-6501/acc59e.

POWERS, D. M. W. **Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation**. 2020.

QUEIROZ, J. E. R. d.; GOMES, H. M. Introdução ao processamento digital de imagens. **Revista RITA**, VIII, n. 1, 2001.

REZATOFIGHI, H. *et al.* Generalized intersection over union: A metric and a loss for bounding box regression. *In*: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [*S.l.*: *s.n.*], 2019. p. 658–666.

RIZZARDI, M. A.; FLECK, N. G. Métodos de quantificação da cobertura foliar da infestação de plantas daninhas e da cultura da soja. **Ciência Rural**, Ciência Rural, Santa Maria, v. 34, n. 1, p. 13–18, 2004. ISSN 0103-8478.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. *In*: SPRINGER. **Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18**. [*S.l.*], 2015. p. 234–241.

Rural Press. **Rural Press: Agricultura e Pecuária**. https://www.ruralpress.com.br. Acessado em: 19 de junho de 2024.

SANTOS, T. T. *et al.* Visão computacional aplicada na agricultura. *In*: ____. **Agricultura Digital: Pesquisa, Desenvolvimento e Inovação nas Cadeias Produtivas**. Campinas, SP: Embrapa Informática Agropecuária, 2020. cap. 6, p. 146–164. ISBN 978-85-7035-941-0.

SEZER, O. B.; OZBAYOGLU, M. Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion. **Applied Soft Computing**, Elsevier, v. 70, p. 525–538, 2018.

SHORTEN, C.; KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. **Journal of Big Data**, Springer, v. 6, n. 1, p. 60, 2019.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 2nd. ed. Springer, 2021. Final draft, September 30, 2021. Disponível em: https://szeliski.org/Book.

SZELISKI, R. **Computer Vision: Algorithms and Applications**. 2nd. ed. Springer, 2022. Final draft, September 30, 2021. This electronic draft was downloaded Jun 19, 2024 for the personal use of Marcos. Disponível em: https://szeliski.org/Book.

VASCONCELOS, M. da Conceição Costa de; SILVA, A. F. A. da; LIMA, R. da S. Interferência de plantas daninhas sobre plantas cultivadas. **Agropecuária Científica no Semiárido (ACSA)**, v. 8, n. 1, p. 1–6, 2012. Disponível em: http://www.cstr.ufcg.edu.br/acsa/.

VILIOTTI, C. A. **Desenvolvimento de um sistema eletrônico para detecção da presença de plantas daninhas e controle da aplicação de herbicidas**. 2002. Tese (Doutorado) — Universidade Federal de Viçosa, Viçosa, Minas Gerais, Brasil, 2002.

WOEBBECKE, D. M. *et al.* Plant species identification, size, and enumeration using machine vision techniques on near-binary images. *In*: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. **Applications in Optical Science and Engineering**. [*S.l.*], 1993. p. 208–219.