UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

VINÍCIUS CARON BRAUN

SEGMENTAÇÃO DE REGIÕES DE INTERESSE EM IMAGENS DE PLANTAS DA SUBESPÉCIE ACER PALMATUM

PATO BRANCO

2023

VINÍCIUS CARON BRAUN

SEGMENTAÇÃO DE REGIÕES DE INTERESSE EM IMAGENS DE PLANTAS DA SUBESPÉCIE ACER PALMATUM

SEGMENTATION OF REGIONS OF INTEREST IN ACER PALMATUM SUBSPECIES PLANT IMAGES

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientadora: Dra. Rúbia Eliza de Oliveira Schultz Ascari.
Coorientador: Dr. Dalcimar Casanova.

PATO BRANCO

2023

VINÍCIUS CARON BRAUN

SEGMENTAÇÃO DE REGIÕES DE INTERESSE EM IMAGENS DE PLANTAS DA SUBESPÉCIE ACER PALMATUM

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 21/junho/2023

Rúbia Eliza de Oliveira Schultz Ascari
Doutorado
Universidade Tecnológica Federal do Paraná - Campus Pato Branco

Dalcimar Casanova
Doutorado
Universidade Tecnológica Federal do Paraná - Campus Pato Branco

Soelaine Rodrigues Ascari
Doutorado
Universidade Tecnológica Federal do Paraná - Campus Pato Branco

Eliane Maria de Bortoli Fávero
Doutorado
Universidade Tecnológica Federal do Paraná - Campus Pato Branco

PATO BRANCO

2023

Dedico este trabalho à minha família, e à minha orientadora.

Agradecimentos

Em primeiro lugar, quero agradecer à minha professora orientadora Dra. Rúbia Eliza de Oliveira Schultz Ascari e ao meu coorientador Dr. Dalcimar Casanova, ambos me auxiliaram a todo o momento na elaboração deste trabalho, depositando sua confiança e atenção.

A todos os colaboradores responsáveis pela construção do banco de imagens utilizado para a realização deste trabalho, sem os recursos fornecidos não seria possível concluir este trabalho.

Aos professores que consultei ao longo da realização deste trabalho, me dando ideias e recomendações de tecnologias que pudessem me auxiliar.

A todos os demais, como minha família e amigos, que me deram suporte com meu crescimento pessoal e profissional.

Não é porque as coisas são difíceis que nós não ousamos; é porque nós não ousamos que elas são difíceis. - Sêneca

RESUMO

A segmentação em imagens de plantas serve para a identificação da planta em questão, diferenciando-a do restante da composição da imagem. O problema da segmentação em plantas geralmente está contido na dificuldade de traçar precisamente os segmentos em volta da folha da planta, a qual muitas vezes é dentada e com elementos de pequenos detalhes. A coloração, tonalidade e iluminação também são fatores que interferem no processo de segmentação e tratamento de imagem. Afinal as etapas de tratamento de imagem baseiam-se na pixelagem da imagem, a qual é composta por fatores como contraste e cor, e isso é também ainda mais dificultado quando trata-se de uma imagem com um fundo colorido e confuso. Sendo assim, o presente trabalho objetiva otimizar a segmentação de imagens da subespécie de planta *Acer* palmatum e suas variedades, escolhida por possuir características de folhagem um pouco mais desafiadoras que o comum, para facilitar no processo de classificação entre suas diferentes variedades, tendo em vista que é uma planta com folhas dentadas e propícias à mudança de cor conforme a época do ano. Para atingir esse objetivo, este trabalho faz uso de tecnologias relacionadas principalmente à linguagem de programação Python, que possui diversas bibliotecas, como *Open Source Computer Vision Library* (OpenCV) e *Remove Background* (RemBg), com recursos direcionados à área de Visão Computacional e Aprendizado de Máquina, que são os dois temas com maior enfoque neste trabalho. Diferentes experimentos foram realizados neste trabalho com imagens da planta capturadas em fundo branco e em fundo confuso, e os resultados obtidos apontam que a segmentação com métodos básicos da biblioteca OpenCV é pouco eficiente em imagens com fundo confuso, apresentando pouca visualização e identificação da imagem da planta após o processamento. Porém no quesito segmentação direta das folhas da planta *Acer* palmatum a biblioteca RemBg, com o algoritmo de *Deep Learning* U2-Net, apresentou uma melhor eficácia de segmentação nas imagens com um fundo confuso, destacando a folha da planta do restante da imagem, enquanto em imagens com fundo branco o algoritmo segmentou de maneira muito mais fácil, como esperado. Também foi empregado o algoritmo de *Deep Learning* Mask R-CNN (*Mask Region-based Convolutional Neural Network*), que apresentou pouca eficácia diante da pouca quantidade de amostras submetidas ao treinamento.

Palavras-chave: segmentação; processamento de imagem; aprendizagem de máquina.

ABSTRACT

Image segmentation in plants serves for the identification of the specific plant, differentiating it from the rest of the image composition. The problem of plant segmentation is often contained in the difficulty of accurately tracing the segments around the plant's leaf, which is often serrated and has small detailed elements. Color, hue, and lighting are also factors that interfere with the segmentation process and image treatment. After all, the steps of image treatment are based on pixelation, which is composed of factors such as contrast and color. This is further complicated when dealing with an image with a colorful and cluttered background. Therefore, the present work aims to optimize the image segmentation of the Acer palmatum subspecie and its varieties, chosen for having foliage characteristics that are slightly more challenging than usual. This optimization will facilitate the classification process among its different varieties, considering that the plant has serrated leaves that change color according to the season. To achieve this goal, this work makes use of technologies mainly related to the Python programming language, which has various libraries such as Open Source Computer Vision Library (OpenCV) and Remove Background (RemBg), with resources focused on Computer Vision and Machine Learning, the two main topics of this work. Different experiments were conducted in this work using images of the plant captured against a white background and a cluttered background. The results obtained indicate that segmentation using basic methods from the OpenCV library is not very efficient in images with a cluttered background, showing poor visualization and identification of the plant image after processing. However, in terms of direct segmentation of Acer palmatum plant leaves, the RemBg library, with the U2-Net Deep Learning algorithm, showed better segmentation efficacy in images with a cluttered background, highlighting the plant leaf from the rest of the image. In images with a white background, the algorithm segmented much more easily, as expected. The Mask R-CNN (Mask Region-based Convolutional Neural Network) Deep Learning algorithm was also employed, but it showed low efficacy due to the limited number of training samples used.

Keywords: segmentation; image processing; machine learning.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

BGR	*Blue, Green, Red*
CNNs	*Convolutional Neural Networks*
COCO	*Common Objects in Context*
GIMP	*GNU Image Manipulation Program*
GPU	*Graphics Processing Unit*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
IoU	*Intersection over Union*
JPG	*Joint Photographic Experts Group*
JPEG	*Joint Photographic Experts Group*
JSON	*JavaScript Object Notation*
OpenCV	*Open Source Computer Vision Library*
PNG	*Portable Network Graphics*
RemBg	*Remove Background*
RGB	*Red, Green, Blue*
R-CNN	*Region-based Convolutional Neural Networks*
RPN	*Region Proposal Network*
UTFPR	Universidade Tecnológica Federal do Paraná
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

A bioinformática e a agroinformática são áreas diretamente relacionadas com a tecnologia, especialmente com a Visão Computacional. Estudos como o de Wolfert et al. (2017) destacam que a agricultura de precisão, impulsionada pela Internet das Coisas (IoT) e *Big Data*, é um exemplo do desenvolvimento futuro. Um trecho de uma publicação postada na biblioteca digital da empresa Embrapa (EMBRAPA, 2020) revela que o número de aplicações da tecnologia nestas áreas está crescendo, como apresentado por exemplo no periódico *Computers and Electronics in Agriculture* (2020), que publica trabalhos voltados para o uso de software e hardware na agricultura. A publicação também revela que 23,7% de todos os trabalhos no ramo da agronomia em 2018 estão relacionados ao tema de Visão Computacional, e subiu para 29,1% em 2019 (EMBRAPA, 2020).

O crescente avanço na pesquisa voltada para Visão Computacional associada a algoritmos de Aprendizado de Máquina mostrou ser possível criar sistemas inteligentes capazes de realizar reconhecimento automático de espécies de plantas com base em imagens (WANG et al., 2017). Contudo, cada espécie de planta possui características específicas, que podem apresentar grande similaridade ou diferença, inclusive dentre as diferentes subespécies de uma mesma espécie. Essas características podem estar ligadas às especificidades das folhas, pecíolos, troncos, flores e frutos, que variam em forma, textura, margem e cor. Assim, a identificação e classificação de espécies nem sempre é uma tarefa trivial, principalmente para pessoas não especializadas.

Plantas do gênero *Acer*, comumente conhecidas como bordo, contêm aproximadamente 200 espécies com folhas que variam em tamanho, forma e cor (JARDINS, 2022). Dentre as espécies de *Acer*, pode-se destacar o *Acer* palmatum, que é original e popular no continente asiático (China, Coreia do Sul e Japão), mas atualmente já encontra-se difundido em vários países, incluindo o Brasil. O *Acer* palmatum subdivide-se em diversas variedades, com características que podem diferir bastante entre si principalmente no tamanho, aparência, e cor das folhas, além de cor e textura de tronco e pecíolos. Por outro lado, há variedades em que essas características são muito parecidas, tornando a tarefa de classificação bastante desafiadora.

Resultados apresentados por Ascari et al. (2023) em um mapeamento sistemático da literatura conduzido com o objetivo de identificar trabalhos voltados para o reconhecimento de plantas da subespécie *Acer* palmatum, indicam a não existência de estudos que utilizaram algoritmos de Aprendizado de Máquina para realizar a identificação e classificação automática de plantas da subespécie *Acer* palmatum. Da mesma forma, o mapeamento sistemático não identificou a existência de conjuntos de dados compostos por imagens da subespécie de planta *Acer* palmatum. Portanto, não é de conhecimento do autor desta proposta a existência de um sistema capaz de reconhecer as diferentes variedades de plantas da subespécie *Acer* palmatum, o que indica que essa é uma área de pesquisa que pode ser melhor investigada e explorada.

Assim, diante do exposto, este trabalho propõe a aplicação de métodos e técnicas de Visão Computacional voltadas para segmentação de imagens de plantas da subespécie *Acer* palmatum e suas diferentes variedades, como forma de viabilizar a criação de um conjunto de dados composto por um número significativo de amostras de diferentes variedades, em que as informações relevantes para reconhecimento da planta são extraídas do plano de fundo.

A partir de um conjunto de dados diversificado, é possível posteriormente desenvolver um classificador automático, construído com base em algoritmos de Aprendizado de Máquina, como *Deep Learning (*Aprendizagem Profunda*)*, treinados a partir dessas imagens.

1.1 OBJETIVO

Aplicar técnicas de Visão Computacional para realizar segmentação de regiões de interesse em imagens de plantas da subespécie *Acer* palmatum como forma de auxiliar no processo de classificação das diferentes variedades desta planta.

1.1.1 Objetivos Específicos

- Construir uma base de dados com imagens diversas de plantas da subespécie *Acer* palmatum, segmentando regiões de interesse.
- Realizar experimentos de segmentação nas imagens aplicando métodos e técnicas clássicas de Visão Computacional.

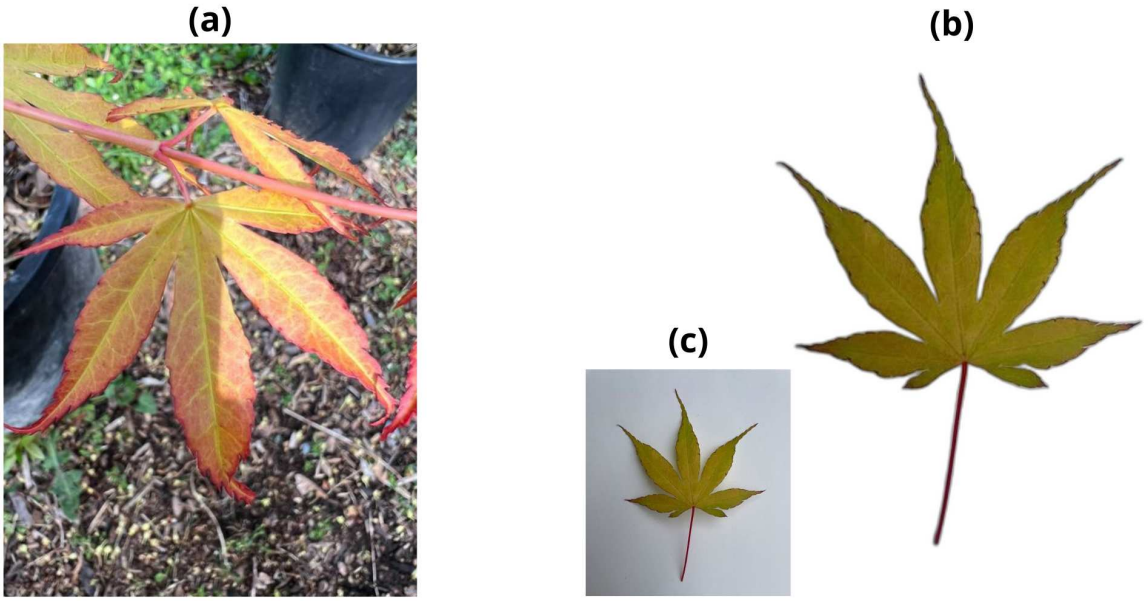
- Realizar experimentos de segmentação nas imagens com técnicas baseadas em *Deep Learning*.
- Avaliar as imagens segmentadas resultantes dos experimentos com visualização qualitativa e métrica de desempenho.

1.2 JUSTIFICATIVA

O *Acer* palmatum, como muitas plantas decíduas (que perdem as folhas em determinada época do ano), sofre alterações na cor de suas folhas, e não apenas relacionadas ao outono. Assim, a cor das folhas pode variar bastante, assumindo tons de vermelho, laranja, entre outros. Essas variações na cor das folhas tornam ainda mais desafiadora a tarefa de classificar as variedades com base nas imagens das folhas e, nesse contexto, a cor é uma característica que não pode ser considerada isoladamente. Assim, existem outras características importantes que precisam ser levadas em consideração, como o formato e a textura das folhas, ou mesmo as características dos troncos e pecíolos.

Para a construção de um conjunto de dados para uso em algoritmos de Aprendizado de Máquina, são necessárias muitas amostras de cada variedade da planta a ser estudada. Muitas vezes a aquisição dessas amostras não segue uma forma padronizada, e pode demandar a aplicação de técnicas de processamento digital de imagens, como por exemplo segmentação, para que a partir de uma imagem com fundo complexo, como apresentado na Figura 1 (a), possa-se chegar a uma imagem com formato mais padronizado, como por exemplo a imagem apresentada na Figura 1 (b), referentes à variação *Acer* palmatum Katsura, onde há também a imagem original de fundo branco (Figura 1(c)) após ser segmentada.

Figura 1 - Amostra de folhas de *Acer* palmatum Katsura com fundo complexo (a) e outra segmentada (b) da imagem original de fundo branco (c)



Fonte: Autoria própria (2023).

No estudo de Yang, Zhong e Li (2020) são aplicadas técnicas de *Deep Learning* para realizar a etapa de segmentação que tem como objetivo extrair de uma imagem apenas as regiões de interesse, como no exemplo apresentado na Figura 2, onde regiões de interesse estão destacadas. Os autores indicam que os métodos empregados podem ser usados para segmentar e classificar a imagem da folha com um fundo complicado de forma eficaz, e poderiam fornecer uma referência para a análise de fenótipo e classificação automática de plantas.

Figura 2 - Exemplo de destaque em partes de interesse de uma imagem, as quais podem ser recortadas por meio de algoritmos voltados para segmentação de imagem

Fonte: Yang, Zhong e Li (2020).

No trabalho de Yang, Zhong e Li (2020), foi proposto um método baseado em redes neurais convolucionais (CNN) para segmentar e classificar folhas em imagens com fundo complexo. O método consistiu em treinar uma CNN para segmentar as folhas, separando-as do fundo complexo das imagens. Em seguida, foram extraídas características das regiões segmentadas das folhas, e essas características foram usadas para classificar as folhas em diferentes classes. Os resultados obtidos mostraram que o método proposto alcançou uma precisão significativa na segmentação e classificação de folhas, mesmo em cenários desafiadores com fundos complexos. Isso indica que o uso de técnicas de *Deep Learning*, como redes neurais convolucionais, pode ser eficaz para melhorar a análise e o entendimento das características das folhas em aplicações agronômicas.

Nesse contexto, o presente trabalho objetiva tratar imagens de plantas especificamente da subespécie *Acer* palmatum, tendo a capacidade de diferenciar regiões de interesse correspondentes à folha, tronco e outros elementos da planta, do restante da imagem, normalmente chamado de fundo. Desta forma pretende-se gerar amostras de melhor qualidade, favorecendo o processo de identificação automática por meio de classificadores baseados em algoritmos de Aprendizado de Máquina.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está organizado em capítulos, como descrito a seguir. O Capítulo 1 mostra as considerações iniciais, apresentando objetivos gerais e específicos deste trabalho, bem como a justificativa. No Capítulo 2 está o referencial teórico que abrange temas e conceitos relacionados ao desenvolvimento do trabalho, como características de plantas do gênero *Acer*, Visão Computacional, Aprendizagem de Máquina e Processamento Digital de Imagens. No Capítulo 3 são apresentados os materiais utilizados para desenvolvimento do trabalho e o método empregado. O Capítulo 4 contém a descrição dos resultados obtidos. Por fim, no Capítulo 5, está a conclusão do trabalho, que é seguido pelas referências utilizadas na composição do texto.

2 REFERENCIAL TEÓRICO

Este capítulo tem como objetivo introduzir um conceito inicial da Botânica e da Visão Computacional como um todo, exibindo métodos aplicados no desenvolvimento deste trabalho. Primeiramente, é mostrado na seção 2.1 um introdutório às plantas *Acer*, gênero botânico, para citar logo em seguida as espécies e subespécies da planta. A seção 2.1.1 apresenta detalhes sobre a subespécie *Acer* palmatum, cujas variedades de plantas serão utilizadas nesse trabalho, revelando pontos a respeito da formação dessa planta, tendo em conta seus elementos naturais para distribuição de folhagem e formato como um todo. Em seguida na seção 2.2 e 2.3 são comentados alguns fatores importantes acerca de classificação, tratamento e processamento de imagens, citando conceitos como *Image Gradient* e também debatendo sobre a segmentação em imagens monocromáticas. Logo na seção 2.3.1, é feito um teste com o algoritmo de Harris para detecção de vértices de folhas de plantas, não deixando de lado também a abordagem de segmentação em imagens coloridas logo na seção 2.3.2, onde é explicado o porquê de haver uma dificuldade maior na realização de segmentação nesse tipo de imagem. Na seção 2.3.3 são disseminados alguns conceitos de *Deep Learning* e YOLO (*You Only Look Once*) *Object Detection*, que trata-se de um método de detecção de objetos.

2.1 *ACER*

Bordo, com o nome científico de *Acer*, é um gênero botânico pertencente à família *Aceraceae*. Possui diversas espécies Áceres[1], usadas comercialmente como ornamento. Uma forma de diferenciar suas várias espécies é olhando a distribuição e a forma das folhas e o ângulo formado pelas sâmaras (GOMES, 2019). Essa diferenciação pode ser bem difícil devido à forte semelhança das folhas, como é possível observar nas figuras 3 e 4, onde há, respectivamente, uma imagem da *Acer* japonicum Vitifolium e da *Acer* japonicum Aconitifolium.

[1] Nome dado ao plural de plantas do gênero botânico *Acer*.

Figura 3 - *Acer japonicum Vitifolium*, Bordo do Japão

Fonte: Gomes (2019).

Figura 4 - *Acer japonicum Aconitifolium*, Bordo do do Japão

Fonte: Gomes (2019).

2.1.1 Acer *palmatum*

Dentre as classificações inferiores do Bordo, há a *Acer* palmatum, a qual possui origens asiáticas e é conhecida popularmente como ácer-japonês ou ácer-palmato. Sua atratividade é caracterizada principalmente pelo fato de apresentar hábito caducifólio (que perde as folhas na estação seca ou no inverno), cujas folhas apresentam tonalidades diferentes conforme a época do ano, ficando vermelhas ou amarelas no final de outono. Plantas da subespécie *Acer palmatum* possuem folhas com geralmente cinco a sete lóbulos em forma de palmeira, suas margens de folhas são pontiagudas e dentadas como exposto na Figura 5.

Figura 5 - Grupos divisores Acer *palmatum* e seus diferentes formatos

Fonte: Ceridono (2011).

A árvore da *Acer* palmatum é enquadrada na categoria ornamental, cuja beleza exótica de florescimento é usada no paisagismo de parques e jardins e na arborização. Encontra-se entre as 80% árvores exóticas usadas como espécies ornamentais na arborização urbana (LORENZI et al., 2003).

O principal pigmento das folhas vegetais é a clorofila, a qual determina a cor da folha e orquestra funções únicas e fundamentais na captação de luz fotossintética (LI et al. 2017). Dito isso, é observado menos clorofila nas mutantes *Acer* palmatum com folhas amareladas, comuns na época de primavera e verão, em comparação às plantas *Acer* palmatum esverdeadas (LI et al. 2015a).

Ainda que a clorofila possua um fator essencial na *Acer* palmatum para seus valores ornamentais e fotossintéticos, a ciência da genética molecular do metabolismo de clorofila nessa espécie ainda foi pouco aprofundada (LI et al. 2017). Com isso, é difícil estimar a diferenciação dessas plantas com métodos empíricos e até científicos, considerando a pouca profundidade que a ciência chegou a respeito da clorofila nessa espécie.

2.2 CLASSIFICAÇÃO AUTOMÁTICA COM BASE EM IMAGENS

Em sistemas voltados para a classificação automática com base em imagens, as imagens provenientes de uma ou mais câmeras são processadas para extrair características que são interpretadas para classificação e reconhecimento por meio de técnicas de Visão Computacional e Aprendizado de Máquina.

O processo de classificação ocorre por meio de etapas específicas que variam de acordo com o contexto da aplicação. A etapa de aquisição de imagem, consiste basicamente na captura de uma imagem real e sua transformação em uma imagem digital, por intermédio de dispositivos que enviam a informação da imagem

adquirida a um conversor analógico digital (BALDNER et al., 2017). Diferentes dispositivos de entrada podem ser utilizados para a aquisição de imagens, e a escolha do dispositivo pode interferir na qualidade da imagem obtida, permitindo o uso de mais ou menos informações nas etapas seguintes.

Em algumas situações, é necessário aplicar técnicas específicas para aprimorar a qualidade da imagem (corrigindo imperfeições, presença de pixels ruidosos, contraste e/ou brilho inadequado) e destacar regiões de interesse. Essa etapa é conhecida como pré-processamento e as técnicas empregadas envolvem métodos que operam no domínio espacial e métodos que operam no domínio da frequência (GONZALEZ; WOODS, 2002).

Para algumas aplicações pode ser necessário segmentar objetos de interesse (veículos, pessoas, dedos da mão, plantas) a partir da imagem de entrada. A segmentação de imagens é uma técnica cujo propósito é dividir a imagem de entrada em regiões semânticas únicas, objetivando tornar a imagem mais simplificada para uma futura análise (RAUT et al., 2009).

Segmentar uma imagem significa, de modo simplificado, separá-la em partes que se diferenciam entre si. É usual denominar "objetos" da imagem os grupos de pixels de interesse, ou que fornecem alguma informação útil para a aplicação. Da mesma forma, a denominação "fundo" da imagem é utilizada para o grupo de pixels que podem ser desprezados ou que não têm utilidade para a aplicação. Ambos os grupos de pixels formam regiões na imagem sem que representem um objeto literalmente presente. A segmentação é considerada essencial e insubstituível (SENTHILKUMARAN et al., 2009), pois quaisquer erros ou distorções geradas nesta etapa refletem negativamente nas etapas seguintes e na eficiência do processo como um todo.

Após segmentada a região de interesse de uma imagem, pode-se executar a etapa de extração de características, resgatando dados sobre forma, silhueta, cores ou outros considerados importantes, para posterior representação. De modo geral, as características podem ser categorizadas em quatro grupos, informações espaço-temporais, informações do domínio da frequência, descritores locais e modelagem corporal (KE et al., 2013).

Depois de extraídas e representadas as características desejadas da imagem, o reconhecimento dessas características por meio de algoritmos de

classificação é a próxima etapa a ser realizada. O objetivo do reconhecimento é realizar, de forma automática, a identificação dos objetos segmentados na imagem.

2.3 TRATAMENTO E PROCESSAMENTO DE IMAGENS

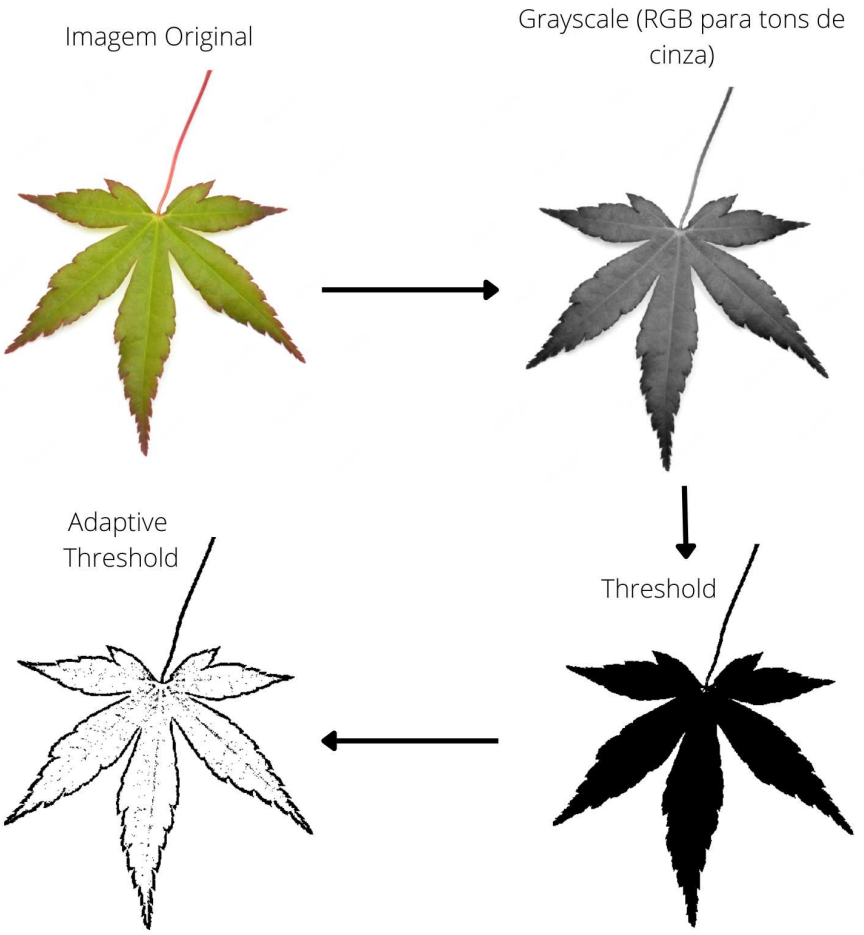
Para que haja qualquer extração de informações de uma imagem, as etapas de pré-processamento da imagem devem ser realizadas para que haja a análise real de seus dados futuramente (SATTI et al., 2013).

No que diz respeito às cores, os humanos a percebem como uma combinação de tristimulus[2] R (*red*/vermelho), G (*green*/verde) e B (*blue*/azul), correspondendo às três cores primárias. Partindo da representação RGB, adveio-se outros tipos de representações de cores com transformações lineares e não lineares (CHENG et al., 2001).

Em relação às plantas em específico, quando utilizadas imagens das folhas para classificação, primeiramente deve haver o processamento inicial da imagem para eliminar ruído e corrigir degradações. Na Figura 6 são apresentados alguns exemplos de etapas realizadas para tratamento de uma imagem, originalmente colorida, onde a aplicação do *Grayscale* (conversão de RGB para tons de cinza) é utilizada para deixar o objeto da imagem cinza, facilitando sua distinção do fundo.

[2] Também chamado de Colorímetro, é um sistema para correspondência visual de uma cor baseando-se nas três cores primárias, sendo expressos como X, Y e Z, correspondendo em valores tristimulus.

Figura 6 - Tipos básicos de tratamento de imagem

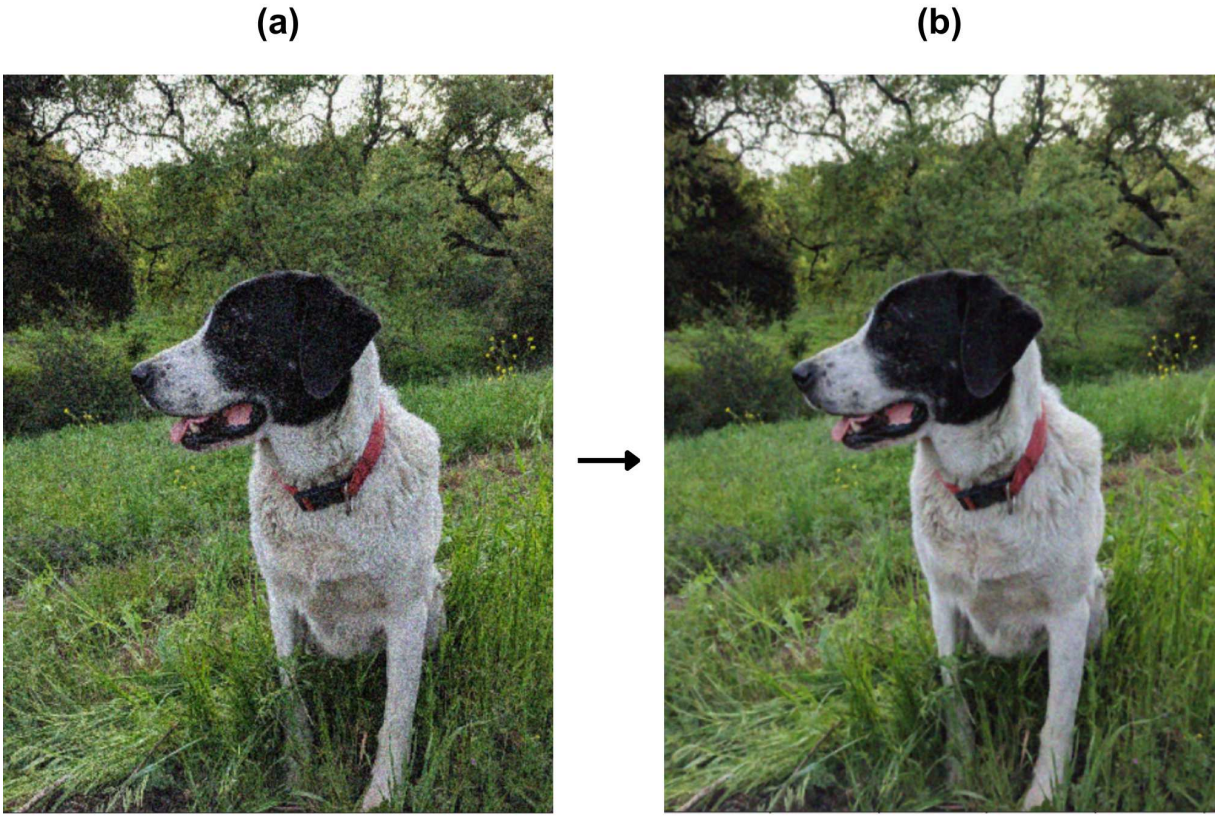


Fonte: Autoria própria (2022).

Os conceitos de *Image Gradient* aplicados à uma imagem referem-se à substituição de intensidades de pixels com a finalidade de solucionar problemas de *image noising*[3], provendo futuramente uma estimação mais confiável de um Aprendizado de Máquina em cima daquela imagem (TZIMIROPOULOS et al., 2012). É possível ver um exemplo de correção de pixelagem de imagem na Figura 7, onde foi realizado alteração do gradiente da imagem da figura (a), com ruído de pixel, para a figura (b), onde o ruído é corrigido.

[3] "Ruído" (da tradução de "noising", algo ruidoso) de imagem é toda e qualquer má difusão de pixelagem de uma imagem.

Figura 7 - Correção de uma imagem com ruído de pixel: (a) imagem original; (b) imagem corrigida



Fonte: Autoria própria (2022).

2.3.1 Segmentação em imagens monocromáticas

Segundo Cheng et al. (2001), a segmentação de imagens monocromáticas é realizada com base na intermitência e semelhança das escalas de cinza em uma determinada região, essa abordagem baseada na intermitência tende a dividir uma imagem detectando bordas, pontos e linhas de acordo com as mudanças súbitas nos níveis acinzentados. Na abordagem de similitude e homogeneidade há um propósito de agrupamento e fusão de regiões (CHENG, 2001).

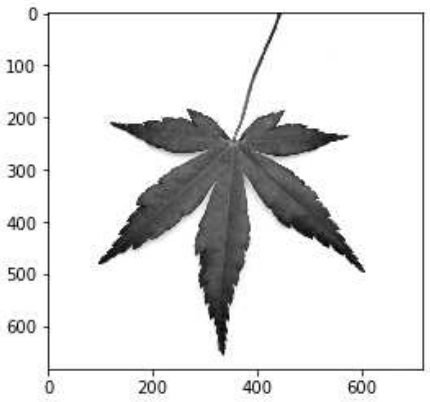
O algoritmo de Harris é usado para identificação de vértices de um objeto na imagem com seus valores de matriz representados por números flutuantes (*float*), diferenciando o objeto do fundo da imagem. No caso de uma imagem com um objeto centralizado e um fundo branco, é facilitado o reconhecimento dos vértices.

Na Figura 8 está uma imagem com o passo a passo de um teste realizado utilizando a imagem de uma folha da planta *Acer* palmatum Katsura, onde os

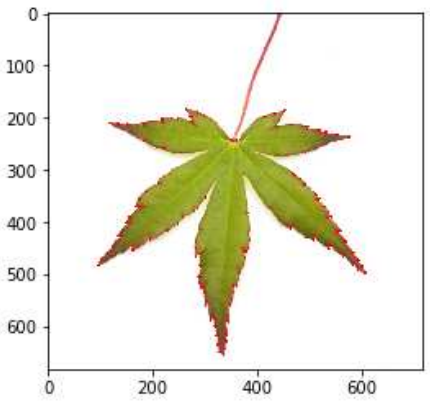
vértices foram facilmente identificados pelo algoritmo de Harris, tendo em conta que o fundo da imagem é branco.

> O algoritmo proposto por Harris utiliza como base a expansão de Taylor, e visa como objetivo uma fórmula, que recebe uma imagem preta e branca e uma coordenada de um ponto, e retorna um número, que se negativo corresponde a um ponto de borda, se positivo a um ponto de superfície plana ou canto, dependendo da magnitude do valor. (BOMBARDELLI, 2014, p.8).

Figura 8 - Teste do algoritmo de Harris em uma imagem da planta *Acer* palmatum Katsura




```
[40]: gray = np.float32(grayAcer) #transofmra os valores da matri

      #deteccção de vértices pelo algoritimo de Harris:
      dst = cv2.cornerHarris(src=gray,blockSize=2,ksize=3,k=0.04)

[41]: dst = cv2.dilate(dst,None) #o resultado do algoritmo é dila

[42]: acerPalmatum[dst>0.01*dst.max()] = [255,0,0] #qualquer resu

[43]: plt.imshow(acerPalmatum) #mostra a imagem

[43]: <matplotlib.image.AxesImage at 0x1a0ab00f780>
```



Fonte: Autoria própria (2023).

No código fonte utilizado para gerar os resultados apresentados na Figura 8, são realizadas as seguintes ações, descritas por bloco:

- Bloco [40]:

    - Primeira linha: Transforma os valores da matriz da imagem em números *float*.
    - Segunda linha: Detecção de vértices pelo algoritmo de Harris. Especificando a imagem a ser utilizada (“gray”), a distância de vértices a serem identificados de um para o outro, o *Sobel Operator*[4] para identificação de bordas e a fórmula de Harris.
- Bloco [41]: O resultado do algoritmo é dilatado para marcar os vértices (traça as linhas vermelhas ao redor da folha).
- Bloco [42]: Qualquer resultado maior que 1% da detecção de Harris, de vértice na imagem, será marcado com um ponto vermelho, indicando que um vértice foi encontrado.
- Bloco [43]: Por fim, o comando que mostra a imagem resultante.

2.3.2 Segmentação em imagens coloridas

A segmentação de uma imagem é o processo de identificação de regiões homogêneas naquela imagem, porém a problemática da segmentação em imagens coloridas é o fato de essas imagens carregarem muitas informações sobre objetos em cena (CECHINEL, 2020). Por conta da semelhança de cores que pode ocorrer em uma imagem, o processo de segmentação em imagens coloridas pode exigir um pouco mais de cautela.

> Segmentação de imagens coloridas é um processo pelo qual se extraem, do domínio da imagem, uma ou mais regiões conectadas que satisfaçam o critério de uniformidade (homogeneidade), o qual é baseado em características derivadas de componentes do espectro. Esses componentes são definidos em um modelo de espaço de cores escolhido. O processo de segmentação pode ser melhorado através de alguns conhecimentos adicionais sobre os objetos em cena tais como as suas propriedades ópticas e geométricas. (CECHINEL, 2020, p. 1).

A segmentação em imagens coloridas é uma técnica essencial em Visão Computacional que visa separar e identificar regiões de interesse ou objetos específicos em uma imagem colorida. O processo de segmentação envolve a atribuição de rótulos ou máscaras a diferentes regiões da imagem, permitindo a análise e extração de informações específicas dessas regiões.

---

[4] Operador utilizado em processamento de imagens para detecção de bordas. Ele consiste em uma máscara de convolução que é aplicada à imagem para realçar as transições de intensidade que ocorrem entre regiões da imagem, permitindo a identificação de bordas ou contornos.

Um dos métodos amplamente utilizados é a antiga segmentação baseada em limiarização adaptativa, proposta por Otsu (1979). Esse método utiliza informações estatísticas da imagem para determinar automaticamente um valor de limiar, separando as regiões de interesse do restante da imagem, objetivando segmentar a imagem em regiões homogêneas com base nas diferenças de intensidade de cor, onde diferentes valores de limiar são calculados para pequenas regiões da imagem.

2.3.3 Conceitos de *Deep Learning*

De acordo com Alzubaidi et al. (2021), *Deep Learning*, ou aprendizado profundo, é a abordagem computacional mais usada no campo do Aprendizado de Máquina, apresentando como principal benefício a capacidade de aprender grandes volumes de dados, sendo o campo de estudo que superou técnicas conhecidas de Aprendizado de Máquina em diversos domínios, como: segurança da informação; processamento de linguagem natural; bioinformática; robótica entre muitas outras.

Por muito tempo a construção de sistemas identificadores de padrões exigiu uma engenharia complexa e uma considerável experiência da mesma para a construção de um extrator de recursos que transformasse dados brutos (pixels de imagens, no caso deste trabalho) em uma representação interna da qual o subsistema pudesse detectar padrões (LECUN et al., 2015).

Como exemplo de uso de algoritmos de Aprendizado de Máquina para reconhecimento de padrões, pode-se citar o YOLO *Object Detection*, que foi empregado neste trabalho para realizar alguns testes de reconhecimento de objetos em uma imagem, incluindo plantas. O YOLO é um sistema de detecção de objetos baseado no *Deep Learning*, apresentando uma única rede neural convolucional para localização e classificação (TAO et al., 2017). Na Figura 9 pode-se observar os resultados obtidos com o uso do YOLO, o qual foi treinado para reconhecer alguns objetos, listados na Figura 10.

Figura 9 - YOLO reconhecendo objetos de uma imagem. Os objetos destacados com bordas na cor azul foram identificados como plantas e um monitor

Fonte: Autoria própria (2022).

Figura 10 - Alguns dos objetos que podem ser identificados em uma imagem por YOLO *Object Detection*.

Fonte: Autoria própria (2022).

Outro exemplo mais específico para segmentação de imagens e identificação de objetos usando redes neurais, é o Mask R-CNN (*Region-based Convolutional Neural Network*), uma rede neural convolucional, usada para tarefas de segmentação de instâncias, que consistem em identificar objetos em uma imagem e separá-los em diferentes instâncias. Essa tarefa é importante em diversas áreas, como Visão Computacional, robótica e processamento de imagens médicas.

O Mask R-CNN é uma extensão do modelo Faster R-CNN, que combina detecção precisa de objetos e classificação em uma arquitetura única. No entanto, além dessas capacidades, o Mask R-CNN também incorpora uma etapa adicional para realizar a segmentação semântica precisa dos objetos em nível de pixel (He et al., 2017).

A arquitetura do Mask R-CNN é composta por três principais componentes: uma rede convolucional de base, uma rede de propostas de região (*Region Proposal*

Network - RPN) e uma *mask head*. A RPN é um componente fundamental para gerar propostas de região que possam conter objetos. Ela examina a saída da rede convolucional de base e gera uma lista de possíveis regiões de interesse na imagem. Essas regiões propostas são obtidas por meio da aplicação de âncoras (*anchors*), que são caixas de diferentes escalas e proporções pré-definidas que cobrem a imagem. A RPN usa a sobreposição dessas âncoras com os objetos reais para determinar a probabilidade de cada região conter um objeto.

Uma vez que as regiões de interesse são obtidas, o Mask R-CNN passa pela *mask head*, que é responsável por gerar máscaras de segmentação precisas para cada região proposta. A *mask head* é uma sub-rede neural totalmente conectada que recebe a região de interesse como entrada e produz uma máscara binária que indica quais pixels pertencem ao objeto em questão.

Além da segmentação de instâncias, o Mask R-CNN também realiza a classificação dos objetos detectados. Após a etapa de região de interesse, cada região proposta é passada por uma sub-rede neural adicional que atribui uma classe ao objeto.

O treinamento do Mask R-CNN ocorre em duas etapas, conforme descrito no trabalho de He et al. (2017). Primeiramente, a rede convolucional de base e a RPN são treinadas conjuntamente usando um conjunto rotulado de dados. Em seguida, a máscara *head* é treinada em um conjunto de dados adicional que inclui máscaras de segmentação para cada objeto.

Os resultados apresentados por Kaiming He et al. (2017) demonstram que Mask R-CNN supera outras abordagens de segmentação de instâncias em diversas métricas, como precisão, velocidade e eficiência. Além disso, a arquitetura é flexível e pode ser adaptada para diferentes tarefas de segmentação, como detecção de objetos em vídeo e segmentação semântica em imagens.

Tendo em vista a importância dos algoritmos de segmentação com *Deep Learning*, outra tecnologia a ser citada é o algoritmo U2-Net. De acordo com Qin et al. (2020), o U2-Net é um algoritmo de *Deep Learning* desenvolvido para detecção de objetos salientes em imagens. A detecção de objetos salientes é uma tarefa crucial em Visão Computacional que envolve identificar e segmentar os objetos visualmente mais distintos em uma imagem.

O treinamento do U2-Net é realizado em um grande conjunto de dados anotados, onde as imagens são rotuladas com informações sobre os objetos

salientes presentes. O modelo é treinado usando a função de perda de entropia cruzada e a técnica de retropropagação do gradiente para ajustar os pesos da rede neural e minimizar a diferença entre as saídas previstas e os rótulos verdadeiros.

O U2-Net utiliza uma arquitetura em forma de “U” chamada U-Net, que é uma rede neural convolucional popularmente usada para tarefas de segmentação. No entanto, o U2-Net aprimora a estrutura original do U-Net com uma abordagem em cascata, conhecida como estrutura em “U” aninhada (*U-Structure*). A estrutura em “U” aninhada do U2-Net consiste em uma série de blocos “U”, em que cada bloco consiste em uma sequência de camadas convolucionais. A parte de codificação captura as características de baixo nível da imagem por meio de camadas convolucionais, enquanto a parte de decodificação reconstrói a imagem saliente por meio de camadas de deconvolução. Essa estrutura em cascata ajuda a preservar informações de detalhes finos e contextuais em diferentes níveis de resolução. A ilustração desta estrutura pode ser conferida na Figura 11.

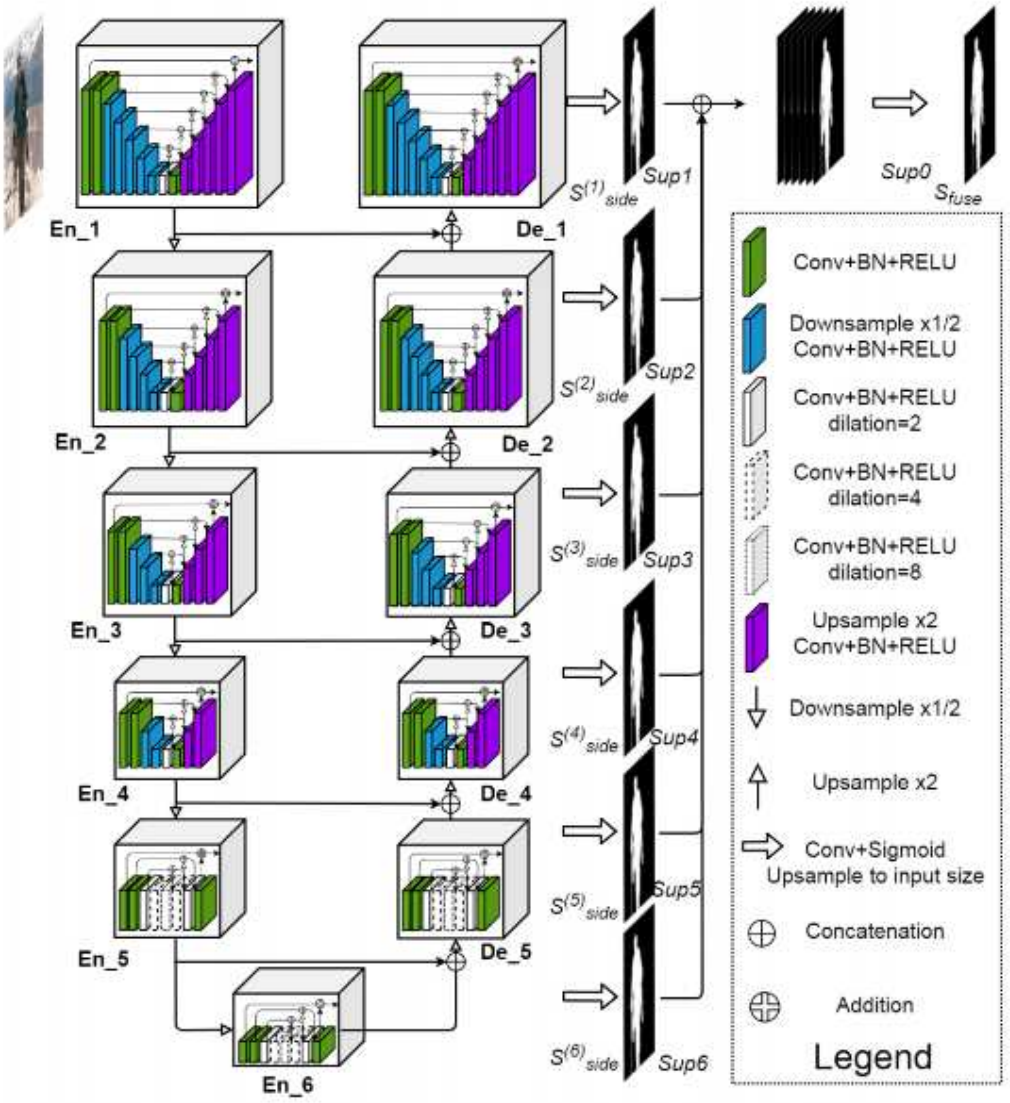
A U-Structure do U2-Net é uma parte fundamental da arquitetura desse modelo de rede neural convolucional. Ela é projetada para capturar informações contextuais em diferentes escalas e ajudar na segmentação precisa de objetos complexos, como folhas de plantas.

A principal característica da U-Structure é a conexão direta entre as camadas do *encoder* (parte "contrativa") e as camadas do *decoder* (parte "expansiva"). Essa conexão direta é conhecida como conexão residual, que permite que as informações sejam propagadas de volta para camadas anteriores, ajudando a preservar detalhes e facilitar o fluxo de informações através da rede (QIN et al. 2020). Como visto na Figura 11, a U-Structure do U2-Net é composta por múltiplos blocos U, que são repetidos em cascata. Qin et al. (2020) descrevem que a *U-Structure* utiliza camadas de *pooling*[5] e convoluções para capturar informações em diferentes escalas. As camadas de *pooling* reduzem a dimensionalidade e a resolução espacial das características, enquanto as convoluções capturam padrões e características em diferentes níveis de abstração. Essa combinação de *pooling* e convoluções em múltiplas camadas permite que a *U-Structure* capture informações contextuais em várias escalas, desde características mais finas até características

[5] Operação comumente usada em redes neurais convolucionais para reduzir a dimensionalidade espacial das representações das características. O *pooling* é aplicado após as camadas convolucionais e tem o objetivo de resumir as informações e reduzir a quantidade de parâmetros da rede.

mais abrangentes. Além disso, os autores mencionam que a *U-Structure* incorpora módulos de atenção residual para aprimorar a precisão e o foco na detecção de objetos salientes. Esses módulos de atenção residual ajudam a destacar regiões relevantes nas imagens, direcionando a atenção para objetos salientes e melhorando a segmentação precisa.

Figura 11 - Ilustração da arquitetura U2-Net (U-Structure)



Fonte: Qin et al. (2020).

O trabalho de Qin et al. (2020) realiza uma comparação da U2-Net com outros métodos de detecção de objetos salientes. Os resultados mostram que a U2-Net supera esses métodos em termos de precisão e capacidade de

segmentação de objetos complexos, demonstrando sua eficácia no contexto de detecção de objetos salientes.

2.3.4 Avaliação de segmentação de imagem

Olhando para uma imagem segmentada é fácil dizer empiricamente se ela está corretamente segmentada ou não, porém é interessante dividir em algumas categorias criadas por Zhang (1996) e descritas por Wang (2020). Dentre as principais, há:

- Avaliação subjetiva: também chamado de método de observação, pode ser reconhecido como o mais conveniente, tendo em vista que os avaliadores são os humanos, porém, como o nome já diz, essa avaliação é muitas vezes subjetiva, onde cada avaliador tem sua própria ótica a respeito da avaliação de uma imagem segmentada. Este método também exige um grande número de imagens e indivíduos para avaliá-las, sendo considerado portanto um processo muito complicado e demorado, nomeado também de visualização qualitativa.

- Avaliação direta e indireta: A avaliação direta trata-se de uma avaliação do próprio algoritmo de segmentação. Enquanto a avaliação indireta trata-se de avaliações que não envolvem o algoritmo de segmentação em si, portanto é feita através de cálculos e experimentos a partir do desempenho do resultado da segmentação.

- Avaliação supervisionada e não supervisionada: Os métodos de avaliação supervisionada, ou métodos de diferença empírica, avaliam o algoritmo de segmentação em relação à comparação de uma imagem segmentada automaticamente com uma imagem segmentada manualmente. Os métodos não supervisionados abordam uma avaliação de imagens de segmentação pelo cálculo de critérios reconhecidos pelo homem que representam bons resultados de segmentação, não exigindo imagens de referência. Um exemplo de avaliação supervisionada é o IoU (*Intersection over Union*), servindo para quantificar a sobreposição entre as máscaras geradas automaticamente por um algoritmo como U2-Net, por exemplo, e as máscaras de referência. Esse

método compara a sobreposição entre a máscara de segmentação predita e a máscara de segmentação de referência, calculando a proporção entre a área em que as duas máscaras se intersectam e a área total da sua união (EVERINGHAM et al. 2015). Quanto maior o valor do IoU, maior é a sobreposição e a similaridade entre as duas máscaras. Com essa métrica, é possível avaliar numericamente a precisão do U2-Net em relação aos rótulos esperados, fornecendo uma medida objetiva da qualidade da segmentação.

3 MATERIAIS E MÉTODO

Os problemas vinculados a este trabalho acerca de segmentação de imagem serão abordados fazendo uso dos materiais e métodos apresentados neste capítulo. Na seção seguinte estão descritos os materiais utilizados, isso inclui tecnologias diversas como: linguagem de programação, bibliotecas e ferramentas.

3.1 MATERIAIS

O Quadro 1 apresenta os materiais que foram utilizados para desenvolvimento do trabalho.

Quadro 1 - Lista de ferramentas e tecnologias

Ferramenta / Tecnologia	Versão	Finalidade
Windows 10	22H2	Sistema Operacional.
Python	3.9.10	Linguagem de Programação.
OpenCV	4.6.0	Biblioteca com funções específicas de Processamento Digital de Imagens e Visão Computacional.
NumPy	1.21.5	Biblioteca para a linguagem de programação Python, que suporta o processamento de grandes arranjos e matrizes.
Jupyter Lab	3.3.2	IDE de desenvolvimento.
PyCharm Community Edition	2022.1.1	IDE de desenvolvimento Python.
RemBg	2.0.25	Biblioteca Python com algoritmos de Aprendizado de Máquina para segmentação de imagem.
Make Sense	--	Website com ferramenta para realização de segmentação manual de imagens
TensorFlow	2.5.0	Biblioteca de Aprendizado de Máquina
Google Colab	2023-05	Ambiente de desenvolvimento baseado em nuvem
GIMP (GNU Image Manipulation Program)	2.10.34	Ferramenta de edição e manipulação de imagens

Fonte: Autoria própria (2023).

Para o desenvolvimento desse projeto, foi utilizada a linguagem de programação Python, juntamente com a biblioteca OpenCV (*Open Source Computer Vision Library*), que possui mais de 2500 algoritmos otimizados de Visão Computacional e Aprendizado de Máquina que podem ser usados para identificação de elementos (OPENCV, 2022).

O Windows foi utilizado como sistema operacional, suportando a instalação dos softwares utilizados, por ter uma fácil instalação e ser amplamente utilizado. Python é uma linguagem de programação própria para este tipo de tarefa, ela dispõe de diversas tecnologias voltadas para o tratamento de dados, automação, Inteligência Artificial, Aprendizado de Máquina, entre outras. Alinhado ao Python, utilizou-se bibliotecas como OpenCV e Numpy, que auxiliam no processamento de imagens e são essenciais para o tratamento, juntamente com a biblioteca RemBg do Python, que possui a tecnologia U2-Net, que trata-se de um modelo de segmentação de imagem. Todos os testes a respeito de processamento de imagem e Aprendizado de Máquina foram feitos na IDE (*Integrated Development Environment)* Jupyter Lab e no *Google Colab*.

3.1.1 Base de Dados

Para a efetivação do objetivo do presente trabalho, é necessário uma base de dados contendo imagens para a realização do processamento. Para isso, foi preparado de forma colaborativa, um banco de imagens contendo diversas amostras de diferentes variedades da subespécie palmatum do gênero *Acer*, com imagens contendo fundo branco e imagens com fundo colorido, abrindo assim a possibilidade de segmentação em ambas condições. As imagens analisadas variam em alguns detalhes como:

- Posição
- Distribuição de folhas
- Luminosidade
- Fundo de imagem
- Sobreposições diversas
- Coloração de folhas

A coleta de imagens da internet não é totalmente segura, tendo em vista que se tratam de diferentes variedades da mesma subespécie de planta. Algumas fontes estão equivocadas quanto à nomeação das plantas contidas nas imagens publicadas, e o processo de classificação é conturbado, portanto, é difícil classificar manualmente para uma separação de imagens via internet. Com isso, o autor deste trabalho juntamente com outros pesquisadores que fazem parte de um projeto voltado para a classificação de plantas por meio de Visão Computacional, buscaram pessoas que cultivam plantas da subespécie *Acer* palmatum e as convidaram a colaborar na montagem de um conjunto de dados, por meio do compartilhamento de fotos de suas plantas, incluindo imagens das folhas, troncos e da planta como um todo. Assim, foram recebidas imagens de 60 variedades diferentes da subespécie *Acer* palmatum. No total, foram adquiridas 1018 imagens variando entre o formato JPG e JPEG, onde há 586 imagens da *Acer* palmatum divididas entre 60 variedades. Também há 373 imagens com fundo branco de algumas dessas variedades, e 59 imagens de 3 outras subespécies de *Acer*, que não a Palmatum.

As imagens recebidas variam muito, pois há amostras correspondentes às folhas das plantas, outras correspondentes aos troncos e outras à planta como um todo. Algumas amostras foram capturadas em ambiente controlado, com fundo branco, enquanto outras foram capturadas com fundo colorido, o que exige algum trabalho de pré-processamento e segmentação antes de serem usadas como entrada para um classificador baseado em Aprendizado de Máquina.

Mediante solicitação ao autor deste trabalho, o conjunto de dados gerado e analisado neste trabalho está disponível para acesso no seguinte link: https://drive.google.com/drive/folders/1hlgfGd70O_c9GQTiAC-mWxLJVJ3dqRtq?usp=sharing.

3.2 MÉTODO

O método aplicado neste trabalho baseia-se em todo o processo envolvendo o tratamento de uma imagem, levando em conta as etapas a serem seguidas para que ocorra a segmentação correta das folhas e demais partes de interesse das imagens de plantas *Acer* palmatum. Os métodos de processamento de imagem mais comuns estão descritos nas seções seguintes, descrevendo alguns conceitos importantes da Visão Computacional, juntamente com técnicas envolvidas no

processo. Isso inclui os aspectos de segmentação com *thresholding* e reconhecimento de padrões, que foram as abordagens utilizadas neste trabalho.

3.2.1 Segmentação por *threshold*

De acordo com Zhu et al. (2007), a segmentação de imagens é o procedimento chave no processamento de imagens, sendo o *thresholding* um dos métodos mais utilizados, porém, como é um método muito básico, é difícil extrair informações complexas de uma imagem com apenas isso.

Neste trabalho, é imprescindível o uso do *thresholding* para os testes de segmentação nas imagens das plantas *Acer*, variando entre os níveis de valores constantes de intensidade para analisar cada pixel da imagem, se a intensidade do pixel for maior que o valor colocado, ficará representado como branco, do contrário, preto, sendo assim a definição de *threshold* binário.

3.2.2 Segmentação por bordas

No mesmo contexto de *threshold*, há ramificações da área de Visão Computacional usando essa técnica que visa segmentar as bordas e contornos contidos em imagens, como é o caso do *Canny Edge Detection*, algoritmo desenvolvido por John F. Canny, contido na biblioteca OpenCV utilizada neste trabalho.

O *Canny Edge Detection* é amplamente utilizado na Visão Computacional, tendo a função de localizar mudanças de intensidade acentuadas e encontrar os cantos dos objetos, seus limites, em uma imagem, classificando um pixel como uma borda se a magnitude gradiente do pixel for maior que a dos pixels em ambos os lados, onde a mudança gradiente se transfere em intensidade máxima (DING e GOSHTASBY, 2001). O Canny será aplicado como forma de testes em plantas *Acer* para avaliar sua eficácia de segmentação.

3.2.3 Segmentação por histogramas

Por definição, histograma é um objeto gráfico de cunho estatístico cujo objetivo é fornecer estimativas consistentes de qualquer função de densidade,

baseando-se em poucas suposições (SCOTT, 2010). Nesta metodologia, um histograma é calculado baseando-se em todos os pixels de uma imagem, sendo os picos e vales do gráfico representando os *clusters*[6] na imagem, sendo utilizadas cor e intensidade luminosa como medida. Como é um tipo de segmentação, também é relevante de ser incluído neste trabalho, mesmo sendo um método que não aborda o objetivo principal do presente estudo.

3.2.4 Segmentação por Aprendizado de Máquina

A segmentação por Aprendizado de Máquina normalmente ocorre por meio de bibliotecas e ferramentas de código aberto para estudo. Neste trabalho, uma das bibliotecas utilizadas é a RemBg, uma biblioteca Python que possui tecnologias e algoritmos envolvidos com Aprendizado de Máquina que são treinados para reconhecimento de imagem para diferenciação de primeiro e segundo plano, ou seja, diferenciação entre objetos de interesse e fundo de imagem. O RemBg, mais precisamente, faz a remoção do fundo da imagem por meio da segmentação do objeto de interesse em primeiro plano automaticamente.

A biblioteca RemBg utiliza o modelo de segmentação U2-Net que, segundo Shao et al. (2022), se comparado a métodos de segmentação manual e a outros métodos automáticos, possui uma grande vantagem de desempenho, promovendo resultados que indicam que a precisão de segmentação do U2-Net é superior, concluindo também que é um método interessante e foi empregado neste trabalho para segmentação de imagens.

Também foi utilizado o algoritmo Mask R-CNN, que recebe imagens pré-segmentadas, aprende seus padrões, e reconhece em uma outra imagem aquele objeto exposto no treinamento, segmentando-o com uma máscara colorida para mostrar a identificação.

Para utilizar o Mask R-CNN, foram utilizadas as bibliotecas de processamento de imagens OpenCV e NumPy e a biblioteca de Aprendizado de Máquina TensorFlow. Além disso, foi utilizado o ambiente de desenvolvimento integrado Google Colab, que fornece acesso a *Graphics Processing Unit* (GPUs) para acelerar o treinamento do modelo.

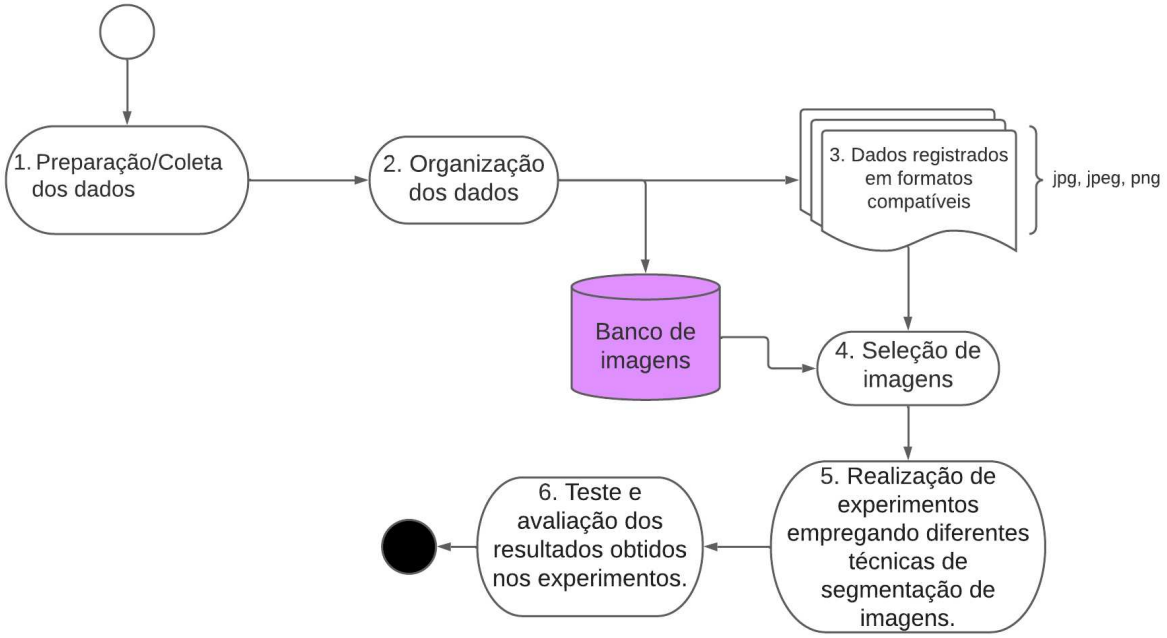
---

[6] Clusters são ‘agrupamentos’, ou partições, de dados individuais contidos na imagem de acordo com suas características mais evidentes de intensidade, RGB, dimensões, entre outras.

Para a obtenção dos resultados desejados neste trabalho, foram usados dois algoritmos de tecnologias diferentes de Aprendizado de Máquina com as imagens do conjunto de dados disponível. Detalhes dos conjuntos de dados e testes realizados em cada experimento serão descritos posteriormente. Desta maneira, foi possível construir um modelo capaz de reconhecer os parâmetros da *Acer* palmatum e segmentá-la na imagem, diferenciando-a do fundo da imagem.

A Figura 12 mostra o fluxograma da metodologia proposta para desenvolvimento do presente trabalho, apresentando as etapas executadas a fim de atender os objetivos estabelecidos.

Figura 12 - Fluxograma da metodologia proposta



Fonte: Autoria própria (2023).

1. Preparação e coleta de dados: A primeira etapa a ser seguida é a de coleta de dados, onde neste trabalho as imagens foram solicitadas de alguns cultivadores da planta *Acer* palmatum e suas variedades. Os contribuintes registraram diversos ângulos e diferentes partes das plantas.
2. Organização dos dados: Com os dados coletados, é necessário a limpeza dos mesmos, separando-os por subespécie e variedade, também separando aquelas imagens cujo registro fora estabelecido com um fundo colorido (Figura 13) das imagens com fundo branco (Figura 14).

Figura 13 - *Acer* palmatum Hogyoku com fundo colorido

Fonte: Autoria própria (2023).

Figura 14 - *Acer* palmatum Hogyoku com fundo branco

Fonte: Autoria própria (2023).

3. Dados registrados em formatos compatíveis: Para a viabilidade e funcionalidade do código, as imagens devem ser incluídas com formatos: JPG, JPEG ou PNG. Qualquer outro formato pode ocasionar problemas relacionados ao código, acarretando o não funcionamento da execução.
4. Seleção de imagens: Com o banco de imagens montado, essa etapa consiste em selecionar as imagens condizentes com o propósito da execução desejada, nesse caso o presente trabalho objetiva segmentar as regiões de interesse das *Acer* palmatum, portanto é necessário selecionar imagens dessa subespécie que apresentem suas folhas e demais características visíveis ao algoritmo, possibilitando o reconhecimento de seus padrões e aprendendo sua distribuição.

5. Realização de experimentos empregando diferentes técnicas de segmentação de imagens: Após selecionar as imagens, nelas serão aplicadas técnicas de segmentação para viabilizar o treinamento de máquina seguinte, facilitando o reconhecimento da planta e diferenciando-a do fundo de imagem.
6. Teste e avaliação dos resultados obtidos nos experimentos: Como etapa final, o que resta é testar se o algoritmo está capacitado a segmentar a folha e demais características da planta, distinguindo-a do restante da imagem. Para avaliar as segmentações feitas neste trabalho, há alguns princípios a serem seguidos, os métodos selecionados para avaliar a qualidade dos testes foram: métrica de segmentação IoU e visualização qualitativa.

No próximo capítulo, os resultados obtidos por meio da realização de diferentes experimentos, sendo utilizados algoritmos de segmentação distintos.

4 RESULTADOS

Neste capítulo, são apresentados os resultados obtidos com o desenvolvimento da proposta deste trabalho. Primeiramente, é mostrada a organização estabelecida para a separação das imagens. Em seguida, logo nas seções seguintes, são exibidos os resultados obtidos por meio do uso de segmentações básicas e métodos de segmentação baseados em Aprendizado de Máquina, os quais estão contidos nas bibliotecas utilizadas neste trabalho. Vale ressaltar que a ordem seguida dessas etapas foi feita de acordo com o fluxograma estabelecido na Figura 12.

4.1 ORGANIZAÇÃO DA BASE DE DADOS

Seguindo da primeira à quarta etapa do fluxograma da Figura 12, na Figura 15 é mostrado o banco de imagens completo, composto pelos 586 registros das variedades da subespécie Acer palmatum, separadas por nome. Logo em seguida, na Figura 16 é mostrado apenas as variedades cujas imagens foram utilizadas nos experimentos. Este trabalho faz parte de um projeto que está sendo desenvolvido por um grupo de alunos e pesquisadores do Departamento Acadêmico de Informática da UTFPR Campus Pato Branco, então esta etapa de organização tem sido realizada de forma colaborativa por todos os participantes. Detalhes sobre esse projeto colaborativo e a proposta deste trabalho foram publicados em um artigo (ASCARI et al., 2023), no qual o autor deste trabalho é coautor. As imagens originais variam nos formatos JPG e JPEG, e nos testes, foram geradas algumas imagens no formato PNG.

Figura 15 - Todas as imagens das diferentes variedades da subespécie *Acer* palmatum, organizadas por seus diferentes nomes

Fonte: Autoria própria (2023).

Figura 16 - Imagens das variedades utilizadas nos experimentos

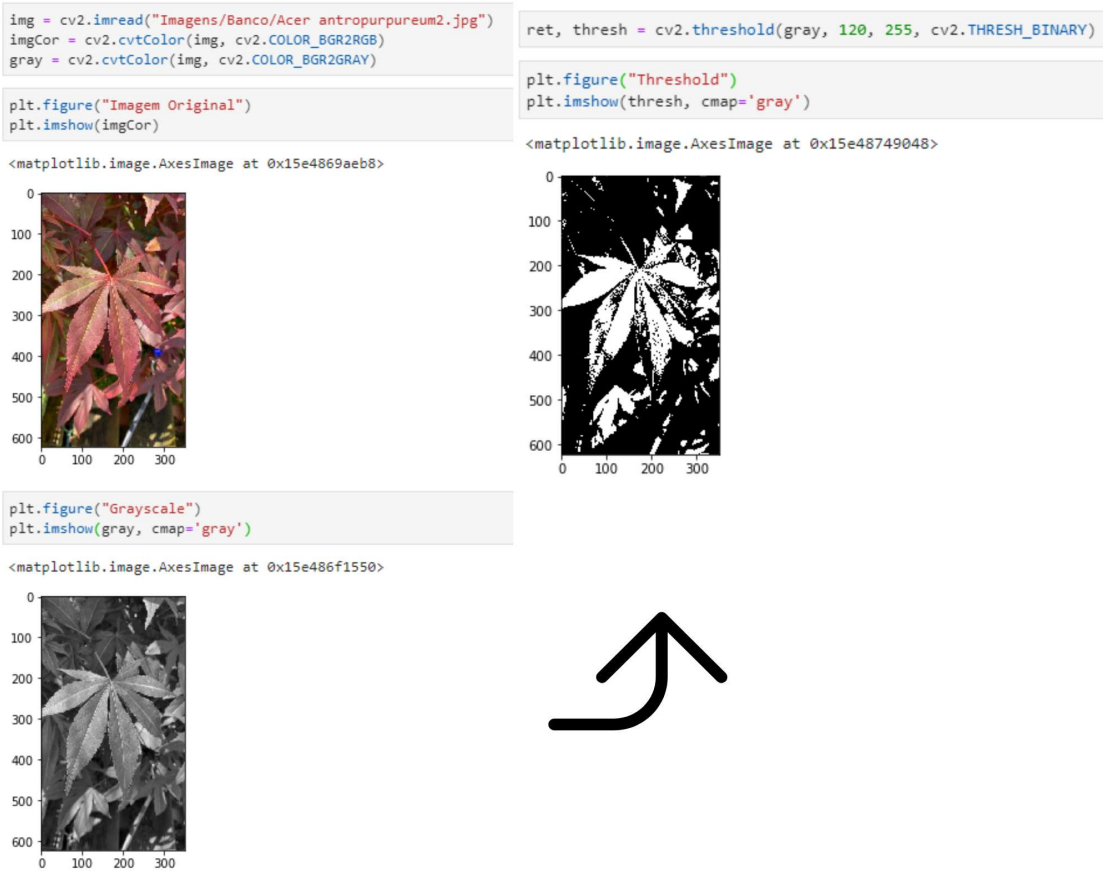
Fonte: Autoria própria (2023).

4.2 IMAGEM SUBMETIDA AO *THRESHOLD* BINÁRIO

A segmentação por *threshold* binário não é completamente precisa em imagens com muita informação, cores e elementos diversos. Esse método visa identificar os valores constantes de intensidade dos pixels da imagem, separando-os em preto e branco, alterando a escala de cinza, montando a imagem segmentada. Na Figura 17 está exposta uma exemplificação com código em Python realizado no Jupyter Lab da segmentação básica feita por *threshold* binário em uma imagem da planta *Acer* palmatum Atropurpureum, retirada do banco de imagens utilizado neste trabalho, passando também pelo processo monocromático de *Grayscale*, onde a imagem é submetida à coloração cinza para facilitar o reconhecimento de pixel. Percebe-se que, nesta figura (Figura 17), foi difícil traçar o *threshold* da imagem,

pois a pixelagem das folhas mais próximas se mescla com a pixelagem dos elementos do fundo, porém ainda é possível observar a silhueta das folhas. Já na Figura 6 da seção 2.3, é evidente que a imagem contém uma folha de *Acer* palmatum Katsura com um fundo branco, o que notoriamente facilita muito mais o *thresholding* binário.

Figura 17 - *Threshold* Binário em imagem de uma *Acer* palmatum Atropurpureum com fundo complexo



Fonte: Autoria própria (2022).

O método de *threshold* binário é eficiente apenas em imagens mais limpas, com o objeto mais bem distribuído e com um fundo de imagem um pouco menos complexo. No teste é possível observar que, ainda que a imagem gerada esteja com partes segmentadas não desejadas, é possível identificar a folha da *Acer* palmatum Atropurpureum, mas não eficientemente.

A Listagem 1 mostra o código utilizado para a aplicação deste *threshold* binário, em seguida a descrição das funções de cada linha.

Listagem 1 - Lógica aplicada para *Threshold* Binário

```python
img=cv2.imread("Imagens/Banco/Acer antropurpureum2.jpg")
imgCor = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

plt.figure("Imagem Original")
plt.imshow(imgCor)

plt.figure("Grayscale")
plt.imshow(gray, cmap='gray')

ret,thresh=cv2.threshold(gray,120,255,
cv2.THRESH_BINARY)

plt.figure("Threshold")
plt.imshow(thresh, cmap='gray')
```

Fonte: Autoria própria (2023).

Esse bloco de código em Python utiliza a biblioteca OpenCV (cv2) para carregar uma imagem de um arquivo no disco e, em seguida, converter a imagem carregada em diferentes formatos de cores.

A primeira linha carrega a imagem "Acer antropurpureum2.jpg" que está localizada no diretório "Imagens/Banco/" e armazena-a na variável "img" utilizando o método "cv2.imread()". Esse método retorna uma matriz NumPy que representa a imagem carregada em formato BGR (*Blue-Green-Red*).

Na segunda linha, o código converte a imagem BGR carregada em RGB utilizando o método "cv2.cvtColor()". Isso é necessário porque muitas bibliotecas, como matplotlib, usam o formato RGB em vez do formato BGR.

A terceira linha converte a imagem carregada em escala de cinza (*grayscale*) utilizando novamente o método "cv2.cvtColor()". Isso é útil para trabalhar com imagens em tons de cinza ou para reduzir a quantidade de dados necessários para armazenar a imagem. A variável "gray" contém a imagem em escala de cinza resultante após a conversão.

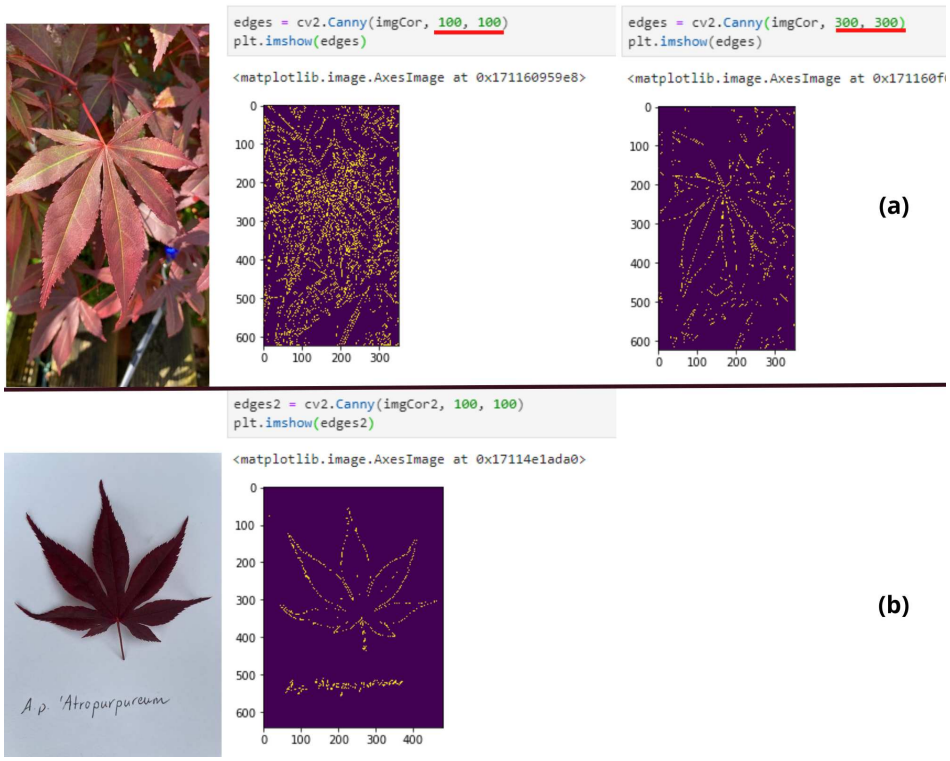
Em seguida, o código em Python utiliza a biblioteca "matplotlib" para exibir uma imagem em uma janela de plotagem com "plt.figure()" e "plt.imshow()". O método “cv2.threshold” do OpenCV que converte a imagem "gray" em uma imagem

binária, onde os pixels com valores acima de 120 são definidos como brancos (255) e os pixels com valores abaixo de 120 são definidos como pretos (0).

4.3 *CANNY EDGE DETECTION* APLICADO EM IMAGENS

O *Canny Edge Detection* foi aplicado na mesma imagem contida na Figura 17 da seção 4.2, usando técnicas de threshold. Na Figura 18 há um exemplo onde ele é usado, referente ao algoritmos de *Canny Edge Detection* contido na biblioteca OpenCV que foi aplicado em duas imagens da planta *Acer* palmatum Atropurpureum. Pode ser observado que na primeira imagem da figura, com um fundo colorido, é mais difícil aplicar o *thresholding* binário para coletar detalhes, sendo necessário aumentar o nível de *threshold*, grifado em vermelho no código do lado (a) da Figura 18, para otimizar a diferenciação de pixels, provendo uma segmentação menos detalhada, porém que diferencia mais os contornos. Diferente da imagem com fundo branco, apresentada também na Figura 18, que é facilmente aplicável a um nível menor de *threshold*, provendo um maior detalhamento de segmentação e uma facilidade maior de processamento.

Figura 18 - Comparativo do *Canny Edge Detection* aplicado em imagem de uma *Acer* palmatum Atropurpureum com fundo colorido (a), e outra com fundo branco (b)



Fonte: Autoria própria (2023).

É observado que esse método é eficiente para segmentação de objetos sem detalhes no fundo de imagem, tendo em conta que seu objetivo é segmentar as bordas e o contorno de uma imagem, não havendo o intuito de preenchimento de objeto, mas sim a identificação de seus limites.

A Listagem 2 mostra o código utilizado para a aplicação da segmentação do *Canny Edge Detection*.

Listagem 2 - Lógica aplicada para Canny Edge Detection

```python
edges = cv2.Canny(gray, 300, 300)
plt.imshow(edges)

edges2 = cv2.Canny(thresh2, 100, 100)
plt.imshow(edges2)

edges = cv2.Canny(gray, 100, 100)
plt.imshow(edges)
```

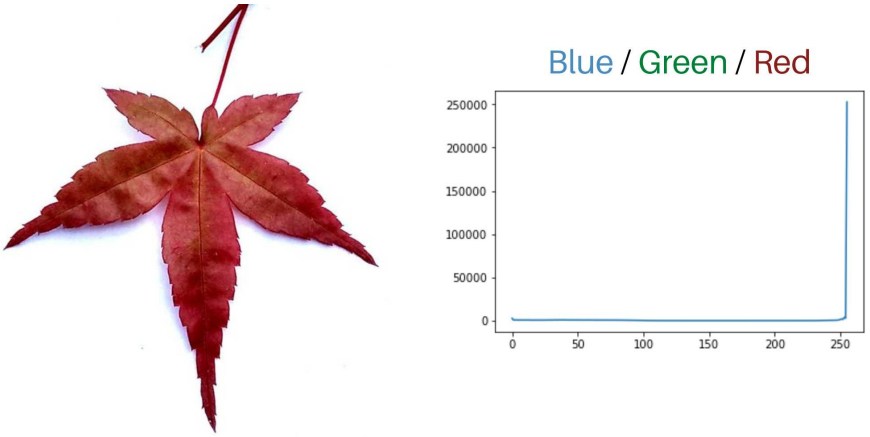
Fonte: Autoria própria (2023).

A primeira linha define a variável "edges", que é usada para armazenar a imagem com as bordas detectadas. A função "cv2.Canny()" é usada para detectar as bordas da imagem "gray" com base em dois limiares: um limiar inferior de 300 e um limiar superior de 300. Sendo “edges” a primeira imagem, e “edges2” a segunda imagem, havendo uma correção do limiar de 300 para 100 na primeira imagem.

4.4 EQUALIZAÇÃO DE HISTOGRAMA

Servindo como uma forma de visualizar estatísticas referentes à pixelagem e valores de BGR de uma imagem, um registro de uma *Acer* palmatum Deshojo foi submetido à uma segmentação de Histograma. A imagem é apresentada na Figura 19, ao lado de um gráfico em que é possível visualizar um pico elevado na cor vermelha da imagem, que condiz com seu tom avermelhado de folha.

Figura 19 - Histograma dividido respectivamente em Azul, Verde e Vermelho da planta *Acer* palmatum Deshojo

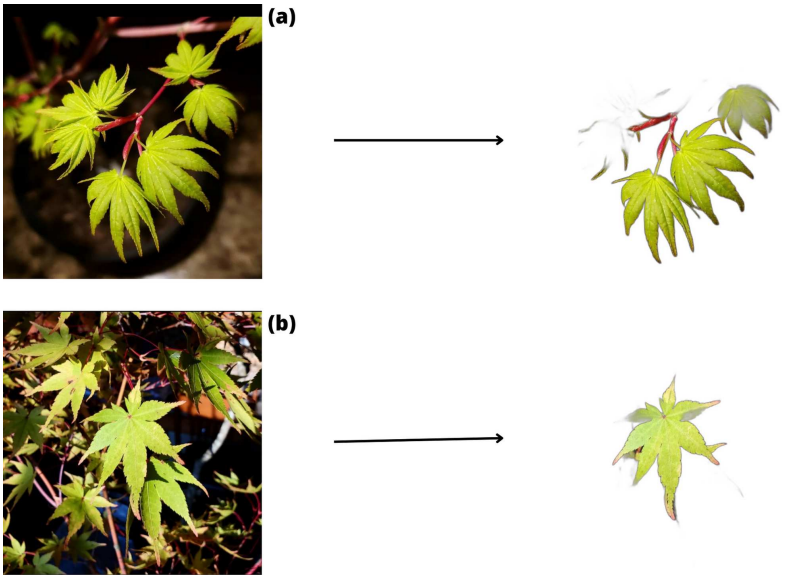


Fonte: Autoria própria (2022).

4.5 TESTES DE SEGMENTAÇÃO COM REMBG E U2-NET

Com o treinamento próprio do modelo U2-Net já é possível segmentar alguns objetos de uma imagem, segmentando suas arestas e diferenciando sua forma do restante da imagem. Na Figura 20 está contido um teste realizado em duas imagens da subespécie *Acer* palmatum Beni-Kawa que possuem um fundo colorido. Suas bordas de folhas foram segmentadas por meio do modelo de segmentação U2-Net, e seu fundo removido pela biblioteca RemBg.

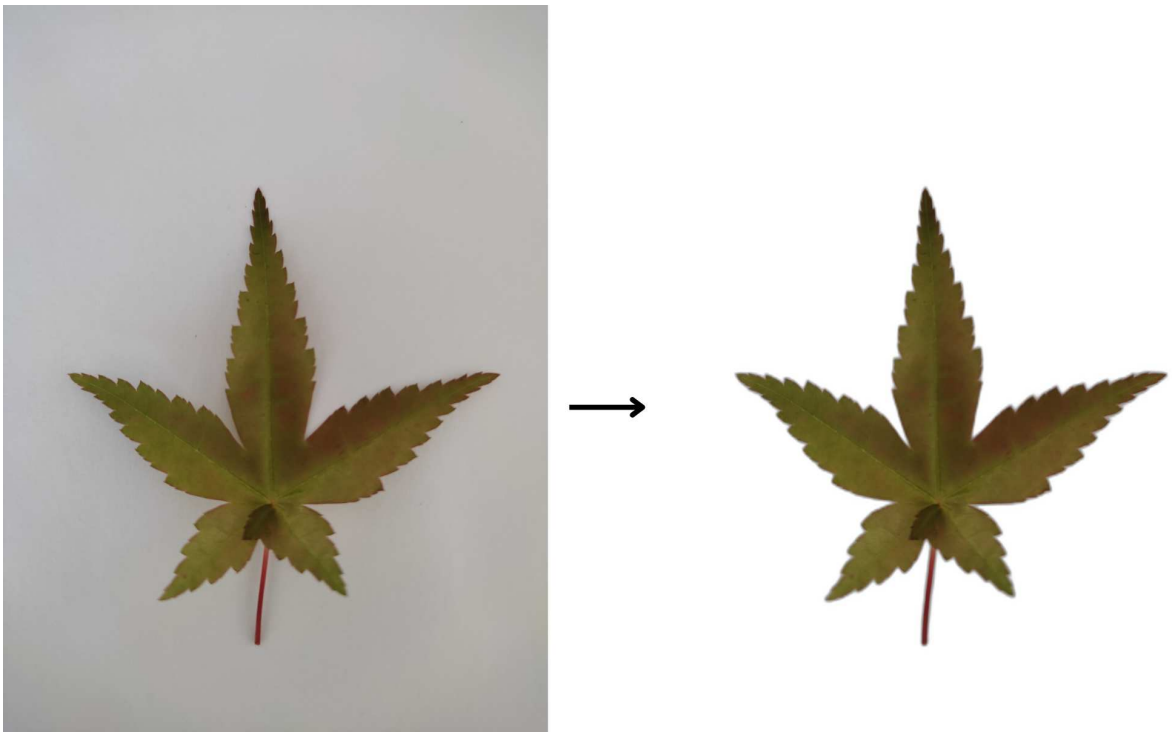
Figura 20 - Biblioteca RemBg com modelo de segmentação U2-Net aplicado às imagens da plantas *Acer* palmatum Beni-Kawa com fundo confuso



Fonte: Autoria própria (2023).

No teste apresentado na Figura 20 é possível observar que o algoritmo é eficiente destacando o objeto em primeiro plano, diferenciando-o do fundo. Na Figura 20 (a), isso deve-se ao fato de que o fundo está levemente embaçado, facilitando a identificação das folhas, mas ainda assim o algoritmo não pôde segmentar completamente todas as folhas daquele galho, segmentando apenas a metade mais destacável. Na Figura 20 (b), o algoritmo se saiu bem identificando a folha do meio da imagem, a qual não se localiza exatamente em primeiro plano, tendo em vista que as outras folhas ao redor não estão com um aspecto embaçado, porém a folha segmentada possui mais luminosidade e contraste, consequentemente fora mais facilmente identificada pelo algoritmo. Visualmente, a Figura 20 não apresentou testes com recortes totalmente precisos da folha, como comparativo há na Figura 21 um teste com uma folha de *Acer* palmatum Deshojo, cuja imagem possui a folha distribuída em um fundo branco, facilitando assim a segmentação do algoritmo U2-Net, sendo perfeitamente recortado do fundo pelos recursos contidos na biblioteca RemBg.

Figura 21 - Biblioteca RemBg com modelo de segmentação U2-Net aplicado a uma imagem de uma *Acer* palmatum Deshojo com fundo branco



Fonte: Autoria própria (2022).

Para ambos os testes foi usado o seguinte bloco de código incluído na Listagem 3.

Listagem 3 - Lógica aplicada para o uso da biblioteca RemBg

```python
from rembg import remove
from PIL import Image

input_path = 'Beni Kawa10.jpg'
output_path = 'resultado3.png'

input = Image.open(input_path)

output = remove(input)
output.save(output_path)
```

Fonte: Autoria própria.

Esse pequeno bloco de código extrai a imagem ‘jpg’, e converte-a em ‘png’ com o fundo removido. A primeira linha importa o método "remove" da biblioteca "rembg" e o módulo "Image" da biblioteca "Pillow", usada para manipular arquivos de imagens. A segunda e a terceira linha definem as variáveis "input_path" e "output_path", que armazenam os caminhos do arquivo de entrada e de saída, respectivamente. O arquivo de entrada é a imagem que será processada e o arquivo de saída é onde a imagem com o fundo removido será salva. A quarta linha carrega a imagem de entrada utilizando o método "Image.open()". A imagem é armazenada na variável "input". A quinta linha utiliza o método "remove" para remover o fundo da imagem "input". O resultado é uma nova imagem que contém apenas o objeto de interesse, sem o fundo. Essa imagem é armazenada na variável "output". E por fim a imagem de resultado é salva normalmente, já podendo ser visualizada.

O U2-Net foi desenvolvido com base na arquitetura U-Net, que é amplamente utilizada para tarefas de segmentação de imagens. A arquitetura U-Net consiste em uma rede neural com uma parte "*encoder*" e uma parte "*decoder*". O *encoder* é responsável por extrair características das imagens em diferentes níveis de abstração, enquanto o *decoder* realiza a reconstrução da imagem segmentada com base nessas características. Ele utiliza uma série de camadas convolucionais e de *pooling* para capturar características de diferentes escalas, permitindo a detecção precisa de bordas e detalhes finos. Além disso, há módulos incorporados de atenção residual para ajudar na preservação de detalhes importantes durante o processo de segmentação. Com isso, o procedimento automático que o algoritmo fez para gerar as imagens com o fundo removido e a segmentação das folhas foi:

1. Pré-processamento: A imagem original foi carregada e preparada para a segmentação. Isso pode incluir redimensionamento, normalização de valores de pixel ou aplicação de outras transformações necessárias.
2. Segmentação: A imagem pré-processada foi fornecida ao modelo U2-Net para segmentação. O modelo analisa a imagem e produz um mapa de segmentação, onde cada pixel é classificado como pertencente à folha da planta ou ao fundo.
3. Pós-processamento: O mapa de segmentação é refinado para melhorar a qualidade e a precisão da segmentação. Isso pode envolver a remoção de ruído, a suavização das bordas ou a aplicação de técnicas avançadas de processamento de imagem.
4. Extração das folhas: Com base no mapa de segmentação pós-processado, as áreas classificadas como folhas da planta são extraídas da imagem original, enquanto o fundo é removido.

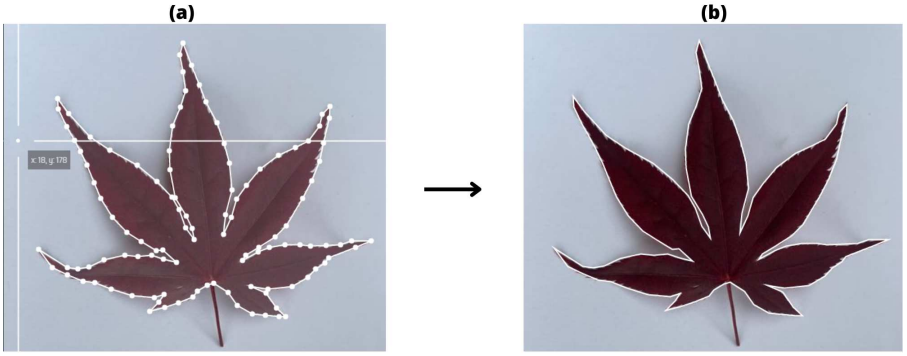
A biblioteca RemBg facilitou todo esse processo, fornecendo uma interface simples para carregar imagens, executar a segmentação usando o modelo U2-Net e obter os resultados segmentados. Como o U2-Net se trata de um algoritmo que detecta o objeto saliente da imagem, o desempenho e a precisão da segmentação dependem da qualidade e da natureza das imagens de entrada, portanto foi de extrema importância que as fotos selecionadas para o teste tenham sido registradas da maneira adequada para o modelo, onde o objeto a ser detectado estava destacado na imagem. Vale ressaltar também que elementos como fundo com *blur* mais alto (desfocado) são de forte influência para uma identificação bem-sucedida, tendo em vista que o *blur* "embaça" o formato dos elementos de fundo, facilitando a distinção do objeto saliente do restante da imagem.

4.6 TESTES COM O ALGORITMO MASK R-CNN E SEGMENTAÇÃO MANUAL

Os testes feitos com o algoritmo Mask R-CNN foram realizados com um total de 33 imagens, as quais foram segmentadas manualmente pelo autor deste trabalho, fazendo uso da ferramenta Make Sense, disponível gratuitamente pelo próprio site, sem a necessidade de fazer alguma instalação. É possível ver registros do processo de segmentação manual na Figura 22, onde a imagem da parte (a)

possui os pontos poligonais que segmentam a imagem, posicionados estrategicamente com traços vetorizados para acompanhar os vértices e arestas da folha, e a imagem da parte (b) é a segmentação traçada em si, podendo ser visualizada ao redor da folha.

Figura 22 - Processo de segmentação manual através da ferramenta Make Sense



Fonte: Autoria própria (2023).

As imagens selecionadas foram da variedade Acer palmatum Atropurpureum, todas selecionadas com fundo branco, para que o algoritmo conseguisse identificar da melhor maneira os padrões da folha, por estar mais claro e disposto na imagem. O objetivo foi avaliar se o algoritmo, absorvendo os padrões da folha em imagens de fundo branco, conseguiria identificar e segmentar a planta em imagens com fundo colorido e diferente.

4.6.1 Aplicando o algoritmo Mask R-CNN

Logo após a segmentação manual, a ferramenta Make Sense disponibiliza uma opção de download da segmentação poligonal em formato *COCO* (*Common Objects in Context*) *JSON*[7] (*JavaScript Object Notation)*, para que possa haver a interpretação dessas segmentações pelo algoritmo. Os testes de segmentação do algoritmo foram feitos nas imagens de fundo branco referentes à variedade Acer palmatum Atropurpureum e também nas de fundo colorido.

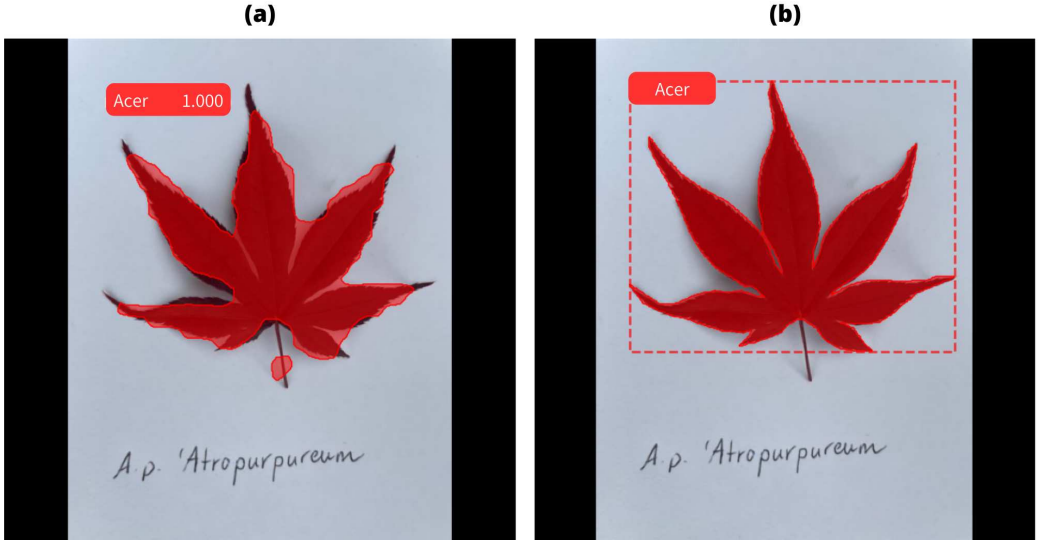
A Listagem 4 apresenta o uso da tecnologia de detecção de objetos em imagens chamada Mask R-CNN, que é uma variante do modelo R-CNN (*Region-based Convolutional Neural Networks*) que permite a detecção de objetos e

[7] O formato COCO JSON é um padrão de arquivo para representar dados de detecção e anotação de objetos em imagens. Ele é baseado em um arquivo JSON e é usado para avaliar e treinar algoritmos de detecção e segmentação de objetos.

segmentação de imagem de alta precisão em uma única etapa. O modelo é baseado em redes neurais convolucionais profundas (*Convolutional Neural Networks* - CNNs) e pode ser usado em tarefas de detecção de objetos e segmentação de imagens em geral.

Há dois tipos de testes presentes nesta seção, que é o teste de reconhecimento do algoritmo, onde é verificado se o algoritmo reconhece a folha *Acer*, e o teste de segmentação do algoritmo, onde o algoritmo treinado verifica imagens de fundo branco e imagens de fundo colorido, buscando segmentar adequadamente as folhas da planta *Acer* na imagem. Os resultados do teste de reconhecimento podem ser conferidos na Figura 23 em uma imagem de fundo branco, mostrando na parte (a) a precisão do algoritmo reconhecendo aquele objeto como um objeto de seu treinamento, e na parte (b) identificando-o na imagem. É possível notar que o algoritmo identificou com 100% de certeza uma planta *Acer* na imagem (a), não segmentando a folha de maneira eficiente, porém o objetivo do primeiro teste era averiguar se o algoritmo foi devidamente treinado para reconhecer a folha. Também conclui-se inicialmente, por meio de visualização qualitativa, que o algoritmo reconheceu e segmentou a folha perfeitamente na imagem (b), também identificando a folha *Acer* na imagem. Este primeiro teste foi apenas para avaliar se o algoritmo sabia como segmentar a folha e identificá-la com certeza em uma imagem.

Figura 23 - Teste de precisão e identificação da folha da planta *Acer* palmatum Atropurpureum em uma imagem com fundo branco: (a) - Precisão de certeza do algoritmo na identificação da *Acer* na imagem. (b) - Identificação da presença da *Acer* na imagem



Fonte: Autoria própria (2023).

Os códigos contendo os métodos para a realização dos testes mostrados na Figura 23 podem ser conferidos a seguir na Listagem 4

Listagem 4 - Lógica aplicada para avaliar previamente o modelo

Python

```
# Parte 1 - Dataset da imagem

# Bloco 1
images_path = "dataset.zip"
annotations_path = "annotations.json"

extract_images(os.path.join("/content/",images_path),
"/content/dataset")

# Bloco 2
dataset_train =
load_image_dataset(os.path.join("/content/",
annotations_path), "/content/dataset", "train")

dataset_val =
load_image_dataset(os.path.join("/content/",
annotations_path), "/content/dataset", "val")

class_number = dataset_train.count_classes()
print('Train: %d' % len(dataset_train.image_ids))
print('Validation: %d' % len(dataset_val.image_ids))
print("Classes: {}".format(class_number))

display_image_samples(dataset_train)


# Parte 2 - Treinamento

# Bloco 3
config = CustomConfig(class_number)
model = load_training_model(config)

train_head(model, dataset_train, dataset_train,
config)


# Parte 3 - Detecção

# Bloco 4
test_model, inference_config =
load_test_model(class_number)
test_random_image(test_model, dataset_val,
inference_config)
```

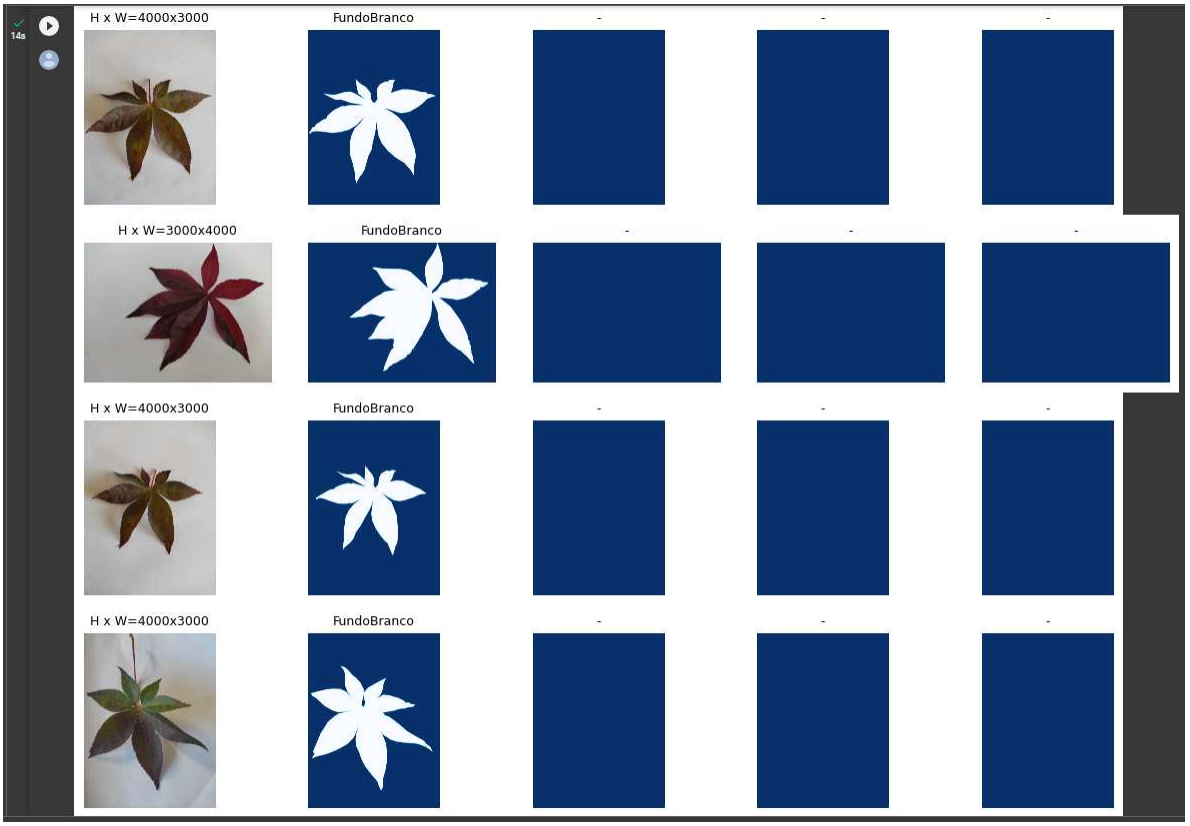
Fonte: Sergio Canu (2021).

O código contido no primeiro bloco faz parte de um processo de pré-processamento de dados para um modelo de segmentação de imagem usando o Mask R-CNN. O primeiro passo foi definir as variáveis “images_path” e “annotations_path” que contêm o caminho para o arquivo zip que contém as imagens e o caminho para o arquivo json que contém as anotações das imagens, respectivamente. Em seguida, a função "extract images" é chamada, que recebe como argumentos o caminho completo para o arquivo zip contendo as imagens, e o diretório onde as imagens serão extraídas. A função extrai as imagens do arquivo zip para o diretório especificado.

Em seguida, no Bloco 2, a função “load_image_dataset” é chamada duas vezes com argumentos diferentes, para carregar o conjunto de treinamento e validação. A primeira chamada salva o conjunto de treinamento na variável “dataset_train” e a segunda chamada salva o conjunto de validação na variável “dataset_val”. Também é chamada a função “count_classes” no objeto “dataset_train” para obter o número de classes no conjunto de treinamento, e depois armazena esse número na variável “class_number”. Por fim, a função “display_image_samples” recebe o objeto “ImageDataset”, que contém as informações sobre as imagens, anotações[8] e classes[9]. A função então seleciona aleatoriamente algumas imagens do conjunto de treinamento e exibe essas imagens, juntamente com suas anotações, em uma grade, sendo possível visualizar algumas imagens do conjunto de treinamento com suas anotações para que seja possível ter uma ideia de como as imagens são rotuladas e se há algum problema na anotação das imagens que precisam ser corrigidos. Por conta de haver apenas uma classe, provinda da variável “class_number”, que no caso do presente trabalho é a *Acer* selecionada para o teste, a imagem exemplo gerada mostra amostras cujas classes não foram definidas, portanto estão vazias. A amostra gerada pode ser visualizada na Figura 24.

[8] No contexto do uso de modelos de segmentação de imagens, como o Mask R-CNN, as "anotações" se referem às informações que descrevem a localização e forma dos objetos de interesse em uma imagem.
[9] As "classes" referem-se às diferentes categorias ou tipos de objetos que o modelo é capaz de identificar e segmentar.

Figura 24 - Amostras das máscaras de algumas imagens fornecidas no treinamento do modelo



Fonte: Autoria própria (2023).

No Bloco 3 (Listagem 4), a classe “CustomConfig” recebe como argumento o número de classes detectadas nas imagens. A classe define os hiperparâmetros do modelo de treinamento, como tamanho de imagem de entrada, número de etapas de treinamento, taxa de aprendizado, entre outros. A função “load_training_model” é usada para carregar o modelo de treinamento do Mask R-CNN, com base nas configurações definidas pela classe “CustomConfig”. Essa função retorna uma instância do modelo, que é armazenada na variável “model”. Em seguida a função “train_head” é usada para treinar as camadas finais que são específicas para a tarefa de detecção de objetos. A função recebe como argumentos o modelo a ser treinado, o conjunto de dados de treinamento e validação, e a configuração do modelo definida pela classe CustomConfig. Com isso, os *epochs*[10] são gerados e o modelo é avaliado no conjunto de dados de validação. Isso permite monitorar o

[10] Um epoch é uma iteração completa por todo o conjunto de dados de treinamento. Durante cada epoch, o modelo é alimentado com um lote (batch) de amostras de treinamento, ele realiza o cálculo da função de perda (loss) e atualiza os pesos das camadas por meio do algoritmo de otimização. Esse processo é repetido para cada lote até que todas as amostras do conjunto de treinamento tenham sido usadas.

desempenho do modelo e detectar se ele está sofrendo *overfitting*, que é quando o modelo se ajusta muito bem aos dados de treinamento, mas não generaliza bem para novos dados.

Por fim, no último bloco (Listagem 4), o modelo treinado é carregado na variável "test_model" juntamente com a configuração de inferência "inference_config". Essa configuração é usada para definir parâmetros como o tamanho das imagens de entrada, a confiança mínima para detectar objetos (*threshold*) e outras configurações específicas para a fase de inferência. Em seguida a função "test_random_image" é chamada para testar o modelo em uma imagem aleatória do conjunto de validação "dataset_val". Essa função seleciona uma imagem aleatória do conjunto de validação e usa o modelo carregado "test_model" e a configuração de inferência "inference_config" para detectar objetos na imagem. O resultado é então exibido, mostrando a imagem original e a imagem segmentada com os objetos detectados, como exposto anteriormente na Figura 23.

Com o modelo treinado, o passo seguinte é a aplicação do algoritmo Mask R-CNN nas imagens, para testar se o modelo identifica o objeto submetido com base no aprendizado adquirido durante o treinamento. Na Listagem 5 consta a parte do código que utiliza o modelo previamente treinado para detectar a folha *Acer* em imagens.

Listagem 5 - Lógica aplicada para testar o modelo treinado em uma imagem Acer palmatum Atropurpureum de fundo confuso

```python
# Parte 1 - Executando o Mask R-CNN nas imagens

# Bloco 1
img = cv2.imread("/content/images.jpg")

test_model, inference_config = load_inference_model(1,
"/content/mask_rcnn_object_0005.h5")
image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Bloco 2
r = test_model.detect([image])[0]
colors = random_colors(80)


# Parte 2 - Exibição da imagem com as máscaras

```

```
# Bloco 3
from google.colab.patches import cv2_imshow

object_count = len(r["class_ids"])
for i in range(object_count):
    mask = r["masks"][:, :, i]
    contours = get_mask_contours(mask)
    for cnt in contours:
        cv2.polylines(img, [cnt], True, colors[i], 2)
        img = draw_mask(img, [cnt], colors[i])

cv2_imshow(img)
```

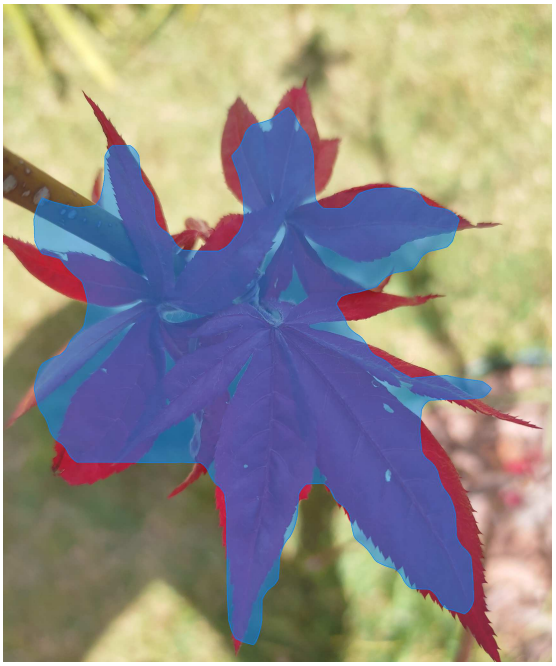
Fonte: Sergio Canu (2021).

Na primeira parte da listagem, o primeiro bloco de código carrega uma imagem e utiliza o modelo treinado previamente para detectar objetos nessa imagem. Inicialmente, o modelo treinado é carregado por meio da função “load_inference_model” que recebe como argumento o número de classes e o caminho para o arquivo do modelo treinado (/content/mask_rcnn_object_0005.h5). Em seguida, a imagem é lida com a função “cv2.imread” e convertida de BGR para RGB através da função "cv2.cvtColor". Por fim, no segundo bloco, a imagem é passada para o modelo por meio do método “detect”, que recebe uma lista com as imagens e retorna as detecções encontradas. Neste caso, como há apenas uma imagem, a lista contém apenas essa imagem. Com isso, o resultado da detecção é armazenado na variável “r”, e as cores para desenhar as máscaras dos objetos detectados são geradas através da função “random_colors”.

O terceiro bloco (Listagem 5), já na segunda parte do código, é responsável por exibir a imagem original e desenhar sobre ela as máscaras que foram geradas pelo modelo treinado usando o Mask R-CNN. Primeiro, a imagem é coletada usando a biblioteca OpenCV, e então o modelo de inferência é carregado usando a função “load_inference_model”. A imagem é convertida de BGR para RGB e, em seguida, é passada para o modelo usando a função “detect”, que retorna um dicionário contendo informações sobre as regiões da imagem onde objetos foram detectados, bem como máscaras binárias que indicam a localização exata desses objetos. O bloco de código itera sobre cada uma das máscaras e desenha as regiões correspondentes na imagem original, usando a função “cv2.polylines” para desenhar os contornos da região e a função “draw_mask” para preencher a região com uma

cor aleatória. A imagem final com as regiões detectadas é exibida usando a função “cv2_imshow”, como mostra a Figura 25 a seguir de uma das imagens testadas.

Figura 25 - Imagem final de um dos testes com as regiões identificadas pelo algoritmo que correspondem às folhas da *Acer*



Fonte: Autoria própria (2023).

4.7 AVALIAÇÃO DOS RESULTADOS

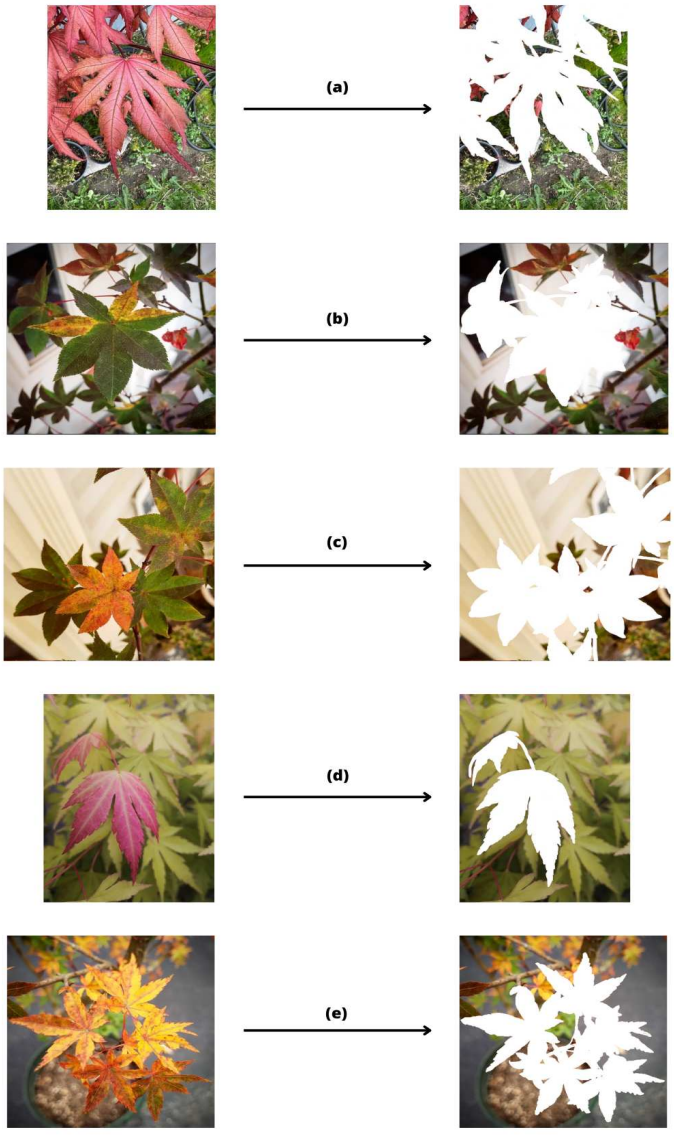
A avaliação dos resultados obtidos por um algoritmo de segmentação de imagens desempenha um papel fundamental na análise e compreensão da eficácia do método aplicado. Seguindo o fluxograma apresentado na Figura 12, será apresentado uma análise abrangente dos resultados obtidos com os testes realizados usando Mask R-CNN e principalmente o modelo U2-Net. O objetivo é avaliar a qualidade e precisão das máscaras geradas pelo modelo, bem como comparar seu desempenho com outros modelos amplamente utilizados na área de segmentação de imagens.

Para realizar essa avaliação, foi adotada uma abordagem multifacetada, utilizando uma combinação de métricas de segmentação IoU e visualização qualitativa ao longo dos testes. Essa abordagem permite obter uma visão mais completa dos resultados e proporcionar *insights* valiosos sobre a efetividade dos testes realizados neste contexto específico.

4.7.1 Métrica de avaliação de segmentação IoU para os testes com U2-Net

Referente aos testes e resultados do U2-Net, o código na Listagem 6 teve como objetivo carregar uma das imagens segmentadas manualmente, as quais podem ser conferidas na Figura 26, e uma das imagens segmentadas pelo algoritmo U2-Net, para então convertê-las em máscaras binárias e armazenar essas máscaras para posterior análise e comparação.

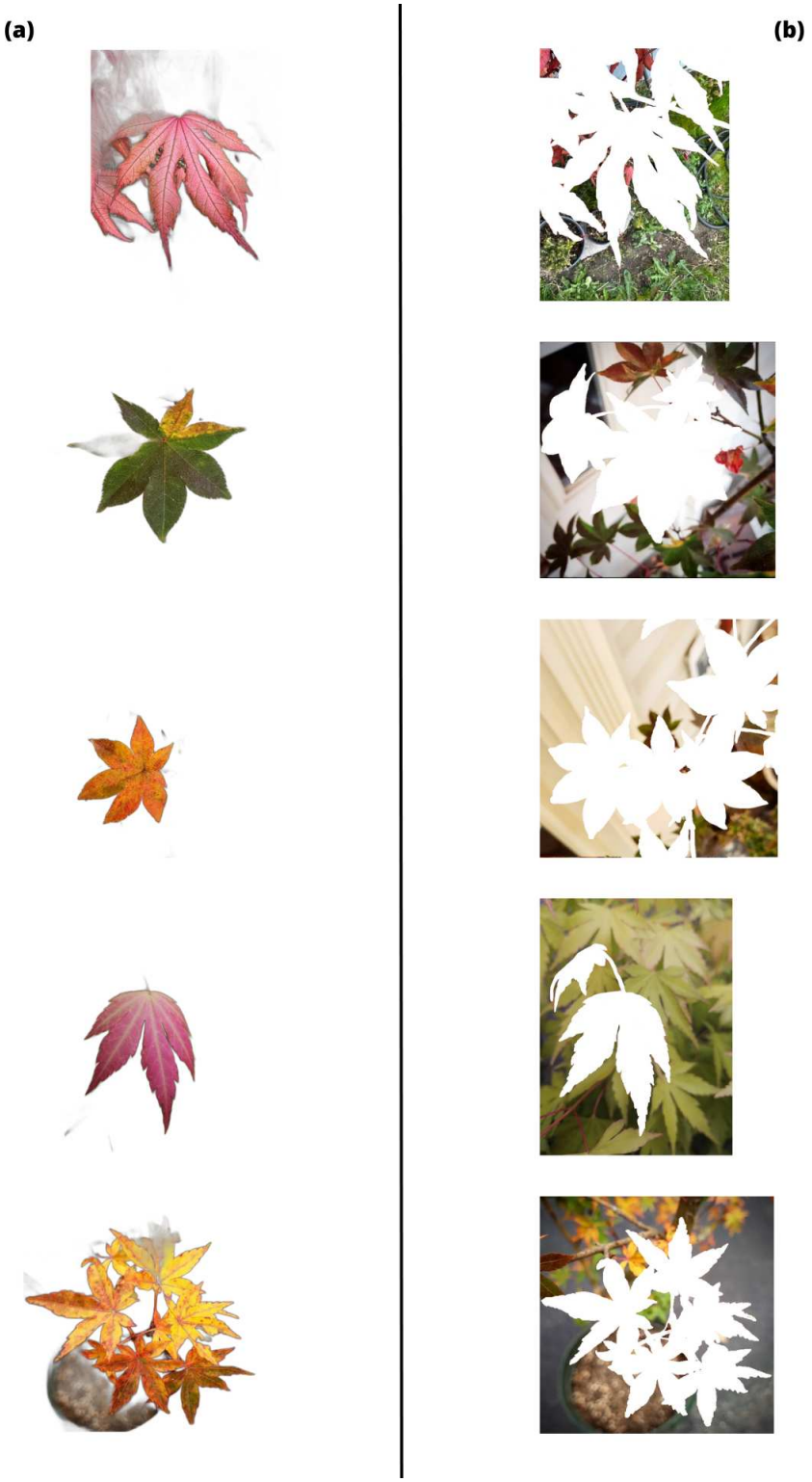
Figura 26 - Imagens originais x Máscaras segmentadas manualmente. (a) *Acer* palmatum Amber Ghost; (b) e (c) *Acer* palmatum Hogyoku; (d) e (e) *Acer* palmatum Katsura



Fonte: Autoria própria (2023).

A seguir na Figura 27 há um comparativo entre as imagens segmentadas pelo algoritmo U2-Net e as imagens das máscaras segmentadas manualmente.

Figura 27 - (a) Imagens segmentadas pelo algoritmo U2-Net. (b) Imagens com as máscaras segmentadas manualmente



Fonte: Autoria própria (2023).

Na Listagem 6 é apresentado o código utilizado para conversão de imagens em máscaras binárias.

Listagem 6 - Conversão de imagens em máscaras binárias

```python
from PIL import Image
import numpy as np

# Bloco 1
def load_image(path):
    image = Image.open(path)
    image = image.convert('L')

    return image

def image_to_binary_mask(image):
    array = np.array(image)
    mask = array > 0

    return mask

# Bloco 2
manual_mask = load_image('semgementacao_manual.png')
segmented_image = load_image('imagem_segmentada.png')

original_mask = image_to_binary_mask(manual_mask)
segmented_mask = image_to_binary_mask(segmented_image)
```

Fonte: Autoria própria (2023).

No Bloco 1 (Listagem 6), a função “load_image(path)” é definida para carregar uma imagem a partir de um caminho fornecido como parâmetro. A imagem é aberta usando a biblioteca “PIL” e, se necessário, convertida para escala de cinza. Em seguida, a função retorna a imagem carregada. A função “image_to_binary_mask(image)” é usada para converter uma imagem em uma máscara binária. A imagem é convertida em um array numpy, e em seguida, uma máscara binária é criada com base nos valores dos pixels da imagem. Pixels que possuem valores maiores que zero são considerados como não-nulos e definidos como “*True*” na máscara binária, enquanto os pixels com valor zero são considerados como nulos e definidos como “*False*”. A função retorna a máscara binária resultante.

Em seguida no Bloco 2 (Listagem 6), é carregada a imagem segmentada manualmente e a imagem segmentada pelo algoritmo e, logo em seguida, são convertidas em máscaras binárias através da função “image_to_binary_mask”.

Com isso, é possível realizar o cálculo do IoU entre duas máscaras binárias: “original_mask” (segmentada manualmente pelo autor deste trabalho) e “segmented_mask” (segmentada automaticamente por um algoritmo). Onde o código contido na Listagem 7 avalia a sobreposição ou similaridade entre duas regiões ou segmentações.

Listagem 7 - Cálculo do Índice de Jaccard (IoU) entre duas máscaras binárias

```python
intersection = np.logical_and(original_mask, segmented_mask)
union = np.logical_or(original_mask, segmented_mask)
iou = np.sum(intersection) / np.sum(union)

resultado = 'O valor do IoU é: {}'.format(iou)
print(resultado)
```

Fonte: Autoria própria (2023).

O código da Listagem 7 utiliza a função “np.logical_and” para calcular a interseção elemento a elemento entre as máscaras “original_mask” e “segmented_mask”. A operação “logical_and” retorna “*True”* para os elementos em que ambas as máscaras possuem valor “*True*”, e “*False*” caso contrário. O resultado dessa operação é armazenado na variável “intersection”.

A função “np.logical_or” é utilizada para calcular a união elemento a elemento entre as máscaras “original_mask” e “segmented_mask”. A operação “logical_or” retorna “*True*” para os elementos em que pelo menos uma das máscaras possui valor “*True*”, e ”*False*“ caso contrário. O resultado dessa operação é armazenado na variável “union”.

Em seguida, o valor do IoU é calculado dividindo a soma dos elementos “*True*” na máscara de interseção (“intersection”) pela soma dos elementos “*True*” na máscara de união (“union”). Por fim, o valor do IoU calculado é formatado em uma *string* com o uso da função “format” e armazenado na variável “resultado”, que é impresso logo em seguida com a função “print”.

O valor do IoU obtido pode ser utilizado para avaliar a qualidade da segmentação obtida em relação à máscara de referência (original). Quanto mais próximo de “1” for o valor do IoU, maior é a sobreposição entre as máscaras e melhor é a segmentação.

Para que fosse possível visualizar não apenas os resultados e os valores avulsos de porcentagem provindos do IoU, foi feito um código, podendo ser conferido na Listagem 8, para visualização da imagem original, sua máscara, e sua segmentação de acordo com a máscara, mostrando como funciona o processo que o o U2-Net faz para realizar a segmentação com máscara binária.

Listagem 8 - Apresentação das máscaras binárias e suas segmentações na imagem

Python

```
import cv2
import matplotlib.pyplot as plt

# Bloco 1
mask = cv2.imread("imagem_segmentada.png",
cv2.IMREAD_GRAYSCALE)
_, binary_mask = cv2.threshold(mask, 65, 255,
cv2.THRESH_BINARY)

# Bloco 2
imagem_original = cv2.imread("'imagem_original.jpg")

imagem_segmentada = cv2.bitwise_and(imagem_original,
imagem_original, mask=binary_mask)

# Bloco 3
plt.figure(figsize=(10, 10))
plt.subplot(1, 3, 1)
plt.imshow(imagem_original[:, :, ::-1])
plt.title("Imagem Original")
plt.subplot(1, 3, 2)
plt.imshow(binary_mask, cmap='gray')
plt.title("Máscara")
plt.subplot(1, 3, 3)
plt.imshow(imagem_segmentada[:, :, ::-1])
plt.title("Imagem Segmentada")
plt.show()
```

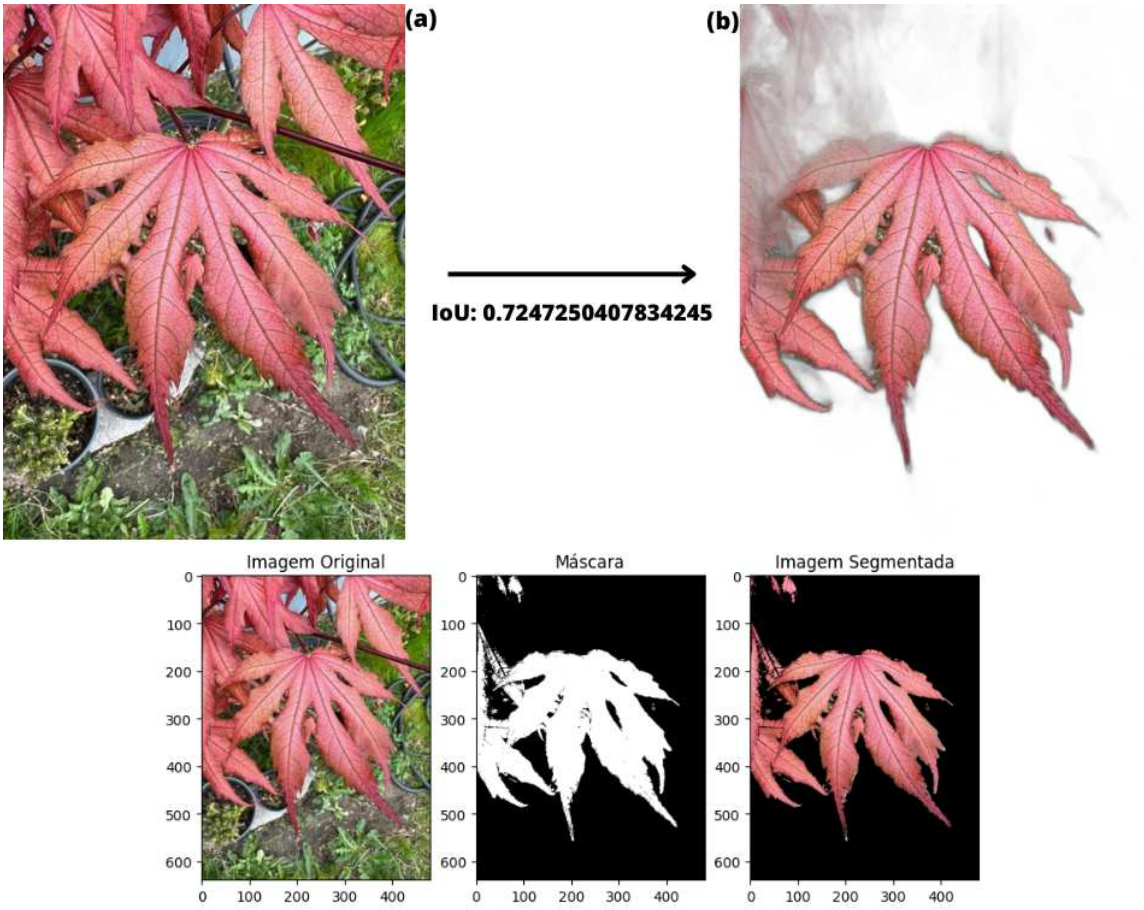
Fonte: Autoria própria (2023).

O código inicialmente importa as bibliotecas OpenCV para manipulação de imagem e “matplotlib.pyplot” para criação de gráficos e visualizações em Python. Em seguida no Bloco 1 a imagem segmentada pelo U2-Net é carregada e convertida para uma máscara binária utilizando a função “cv2.threshold, definindo que pixels com intensidade menor que 65 são definidos como 0 (preto), e pixels com intensidade maior ou igual a 65 são definidos como 255 (branco). A máscara binária resultante é armazenada na variável “binary_mask”.

Em seguida no Bloco 2 é carregada a imagem original que será segmentada, utilizando a função “cv2.imread”, sendo armazenada na variável “imagem_original”, e é aplicada a máscara na imagem original usando a função “cv2.bitwise_and”. Essa operação realiza uma operação *bit* a *bit* entre a imagem original e a máscara binária, preservando somente os pixels que correspondem a regiões brancas na máscara. O resultado é armazenado na variável “imagem_segmentada”. Por fim, no Bloco 3 são exibidos os resultados com as personalizações de gráficos da biblioteca “matplotlib”.

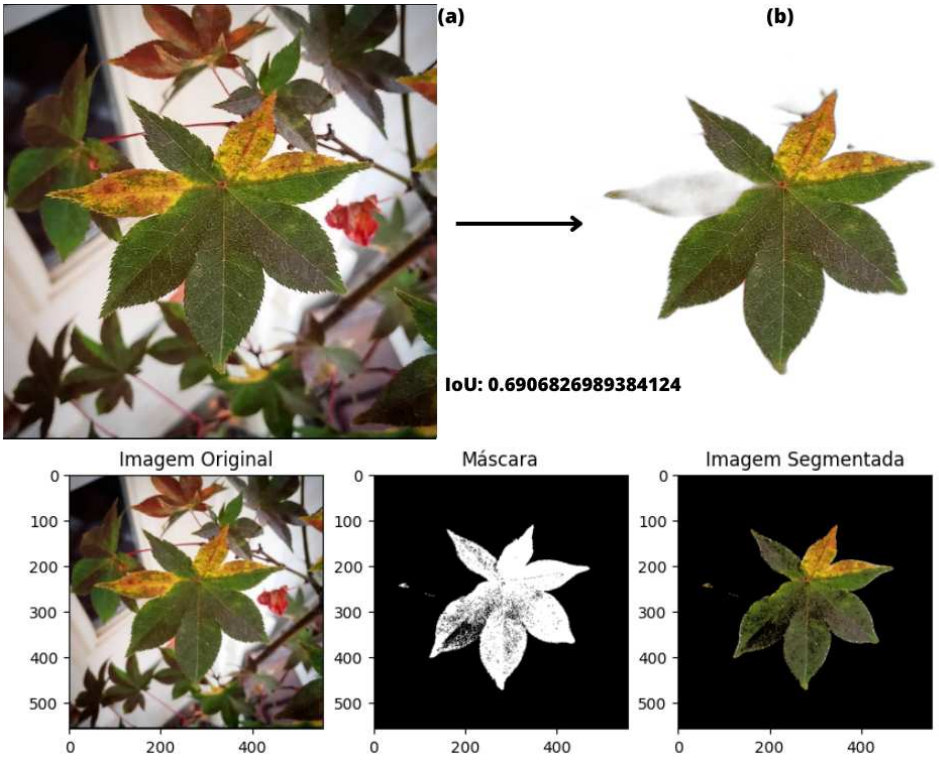
Os resultados finais dos testes de IoU podem ser conferidos a seguir nas Figuras 28 a 33, juntamente com as imagens mostrando as máscaras binárias provindas das segmentações do U2-Net. É importante ressaltar que o valor máximo do IoU nestes testes é 1, sendo 0 o valor mínimo.

Figura 28 - Processo de binarização de máscara e segmentação U2-Net na *Acer* palmatum Amber Ghost. (a) Imagem original; (b) Imagem segmentada pelo U2-Net



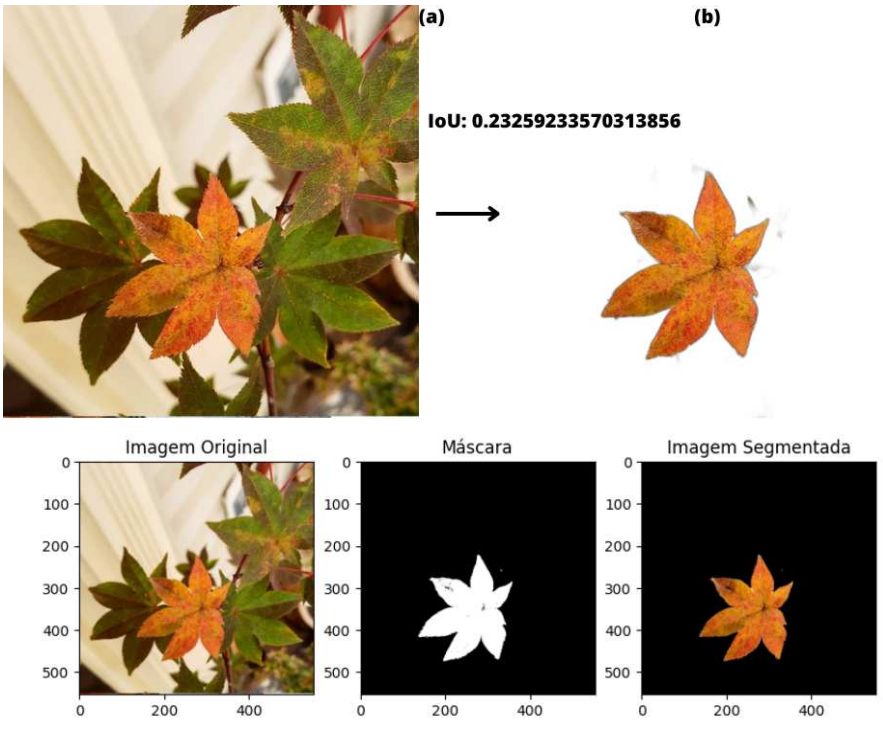
Fonte: Autoria própria (2023).

Figura 29 - Processo de binarização de máscara e segmentação U2-Net na *Acer* palmatum Hogyoku com folhas verdes. (a) Imagem original; (b) Imagem segmentada pelo U2-Net



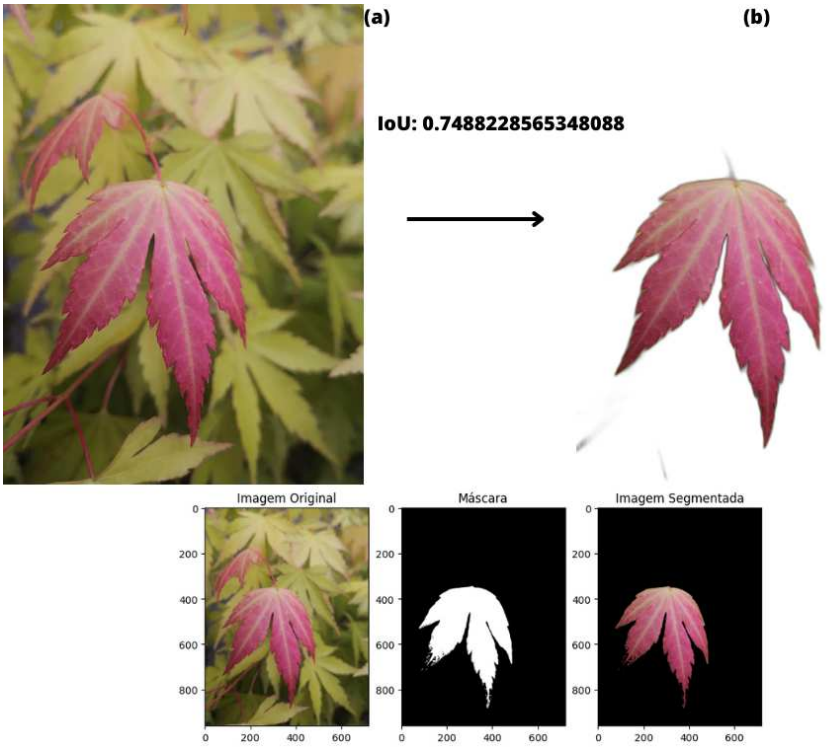
Fonte: Autoria própria (2023).

Figura 30 - Processo de binarização de máscara e segmentação U2-Net na *Acer* palmatum Hogyoku com folhas verdes e uma amarelada. (a) Imagem original; (b) Imagem segmentada pelo U2-Net



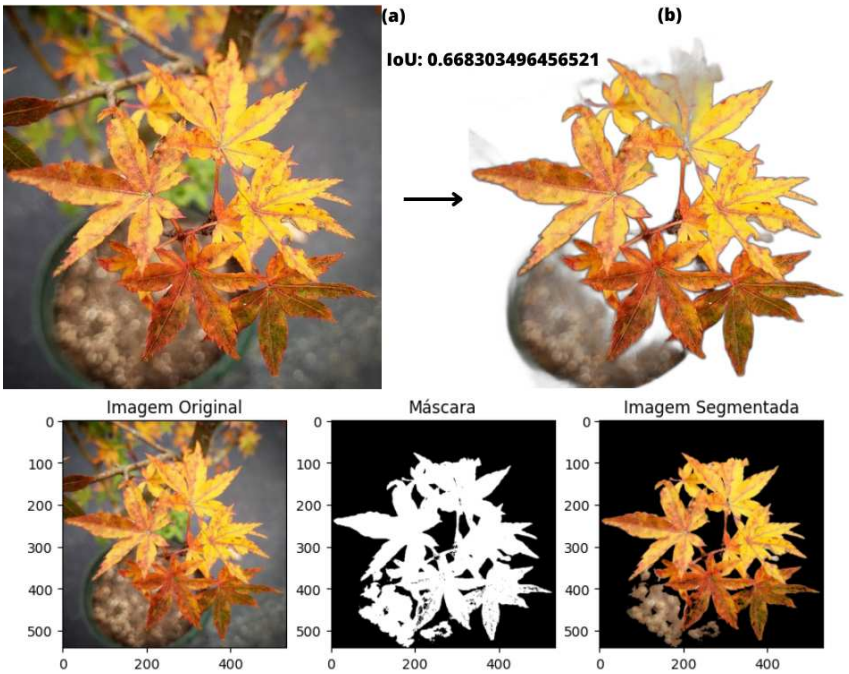
Fonte: Autoria própria (2023).

Figura 31 - Processo de binarização de máscara e segmentação U2-Net na *Acer* palmatum Katsura com folhas rosadas. (a) Imagem original; (b) Imagem segmentada pelo U2-Net



Fonte: Autoria própria (2023).

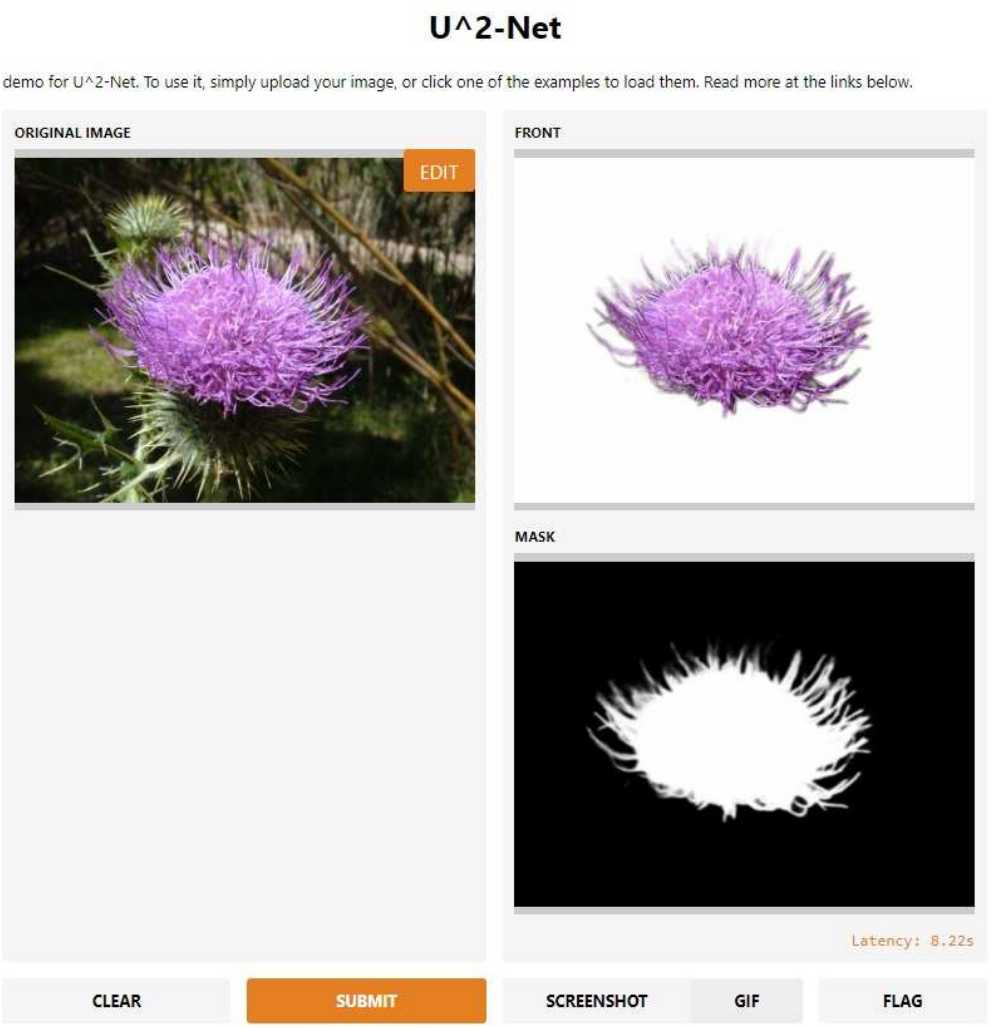
Figura 32 - Processo de binarização de máscara e segmentação U2-Net na *Acer* palmatum Katsura com folhas amareladas. (a) Imagem original; (b) Imagem segmentada pelo U2-Net



Fonte: Autoria própria (2023).

Como comparativo para os resultados de IoU dos testes com as plantas *Acer* palmatum, foi extraída uma imagem de um trabalho da comunidade postado na página do código-fonte do autor do algoritmo U2-Net, Xuebin Qin (2021). O trabalho postado buscava utilizar o algoritmo U2-Net para segmentar o objeto principal e, logo em seguida, converter a imagem segmentada em uma máscara binária, assim como também foi feito no presente trabalho. Com isso, essa imagem coletada serviu de referência para ter o teste de IoU de uma imagem que fez parte do trabalho do autor do algoritmo U2-Net, comparada com os testes de IoU realizados no presente trabalho. O registro da postagem pode ser conferido na Figura 33, contendo a imagem extraída para a realização da comparação.

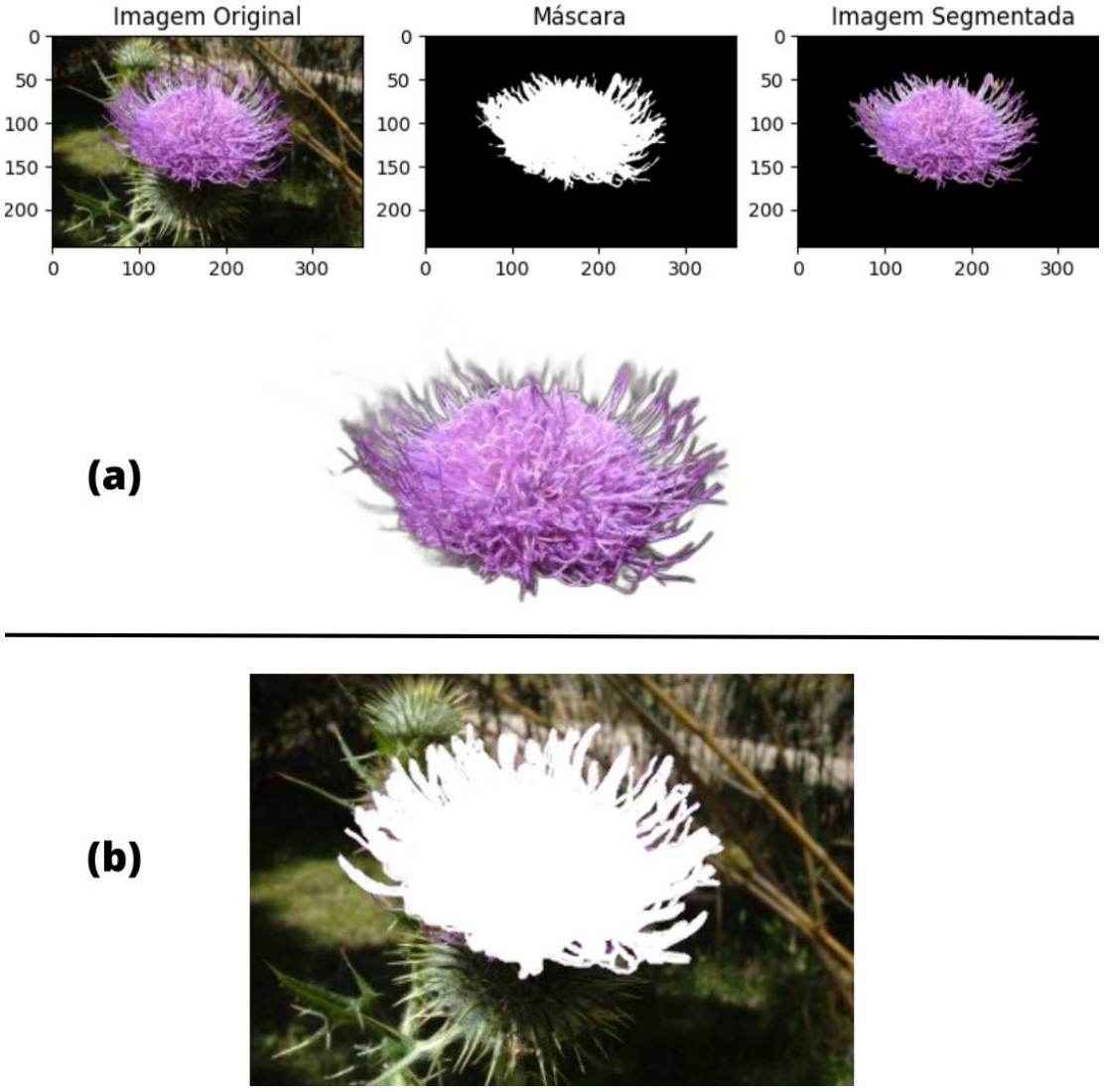
Figura 33 - Registro de um dos trabalhos da comunidade, postado na página do código-fonte do U2-Net



Fonte: Qin, X., Khaliq, A. (2021).

Os testes realizados em cima da imagem coletada do registro forneceram um valor de IoU de 0.8229560541130997, servindo como um comparativo para os valores de IoU dos testes realizados com as plantas *Acer* palmatum. A seguir, na Figura 34, pode ser visto o mesmo processo realizado, onde as imagens do lado (a) representam o procedimento realizado pelo U2-Net, enquanto o lado (b) representa a máscara feita manualmente.

Figura 34 - Resultados do U2-Net na imagem usada como comparação. (a) Imagem original, máscara binária, imagem segmentada pelo U2-Net; (b) Imagem segmentada manualmente



Fonte: Autoria própria (2023).

Comparando o IoU da imagem de referência da Figura 34 com os resultados de IoU obtidos dos testes com a *Acer*, percebe-se que a *Acer* palmatum Hogyoku da Figura 30 foi a que menos obteve êxito em alcançar a segmentação ideal (com base na máscara manual traçada). Isso pode ter ocorrido possivelmente por conta de

haver apenas uma folha amarelada, e esta por estar em primeiro plano na imagem, o algoritmo U2-Net segmentou-a apenas, deixando de identificar as folhas vizinhas, não alcançando o resultado esperado, que seria identificar o máximo possível de folhas *Acer* palmatum na imagem. Os outros testes foram relativamente satisfatórios no que diz respeito ao nível de IoU, e no quesito de visualização qualitativa, pode-se dizer que a qualidade da segmentação é notável, tendo em vista que alguns detalhes únicos da *Acer* palmatum foram respeitados na segmentação do algoritmo U2-Net.

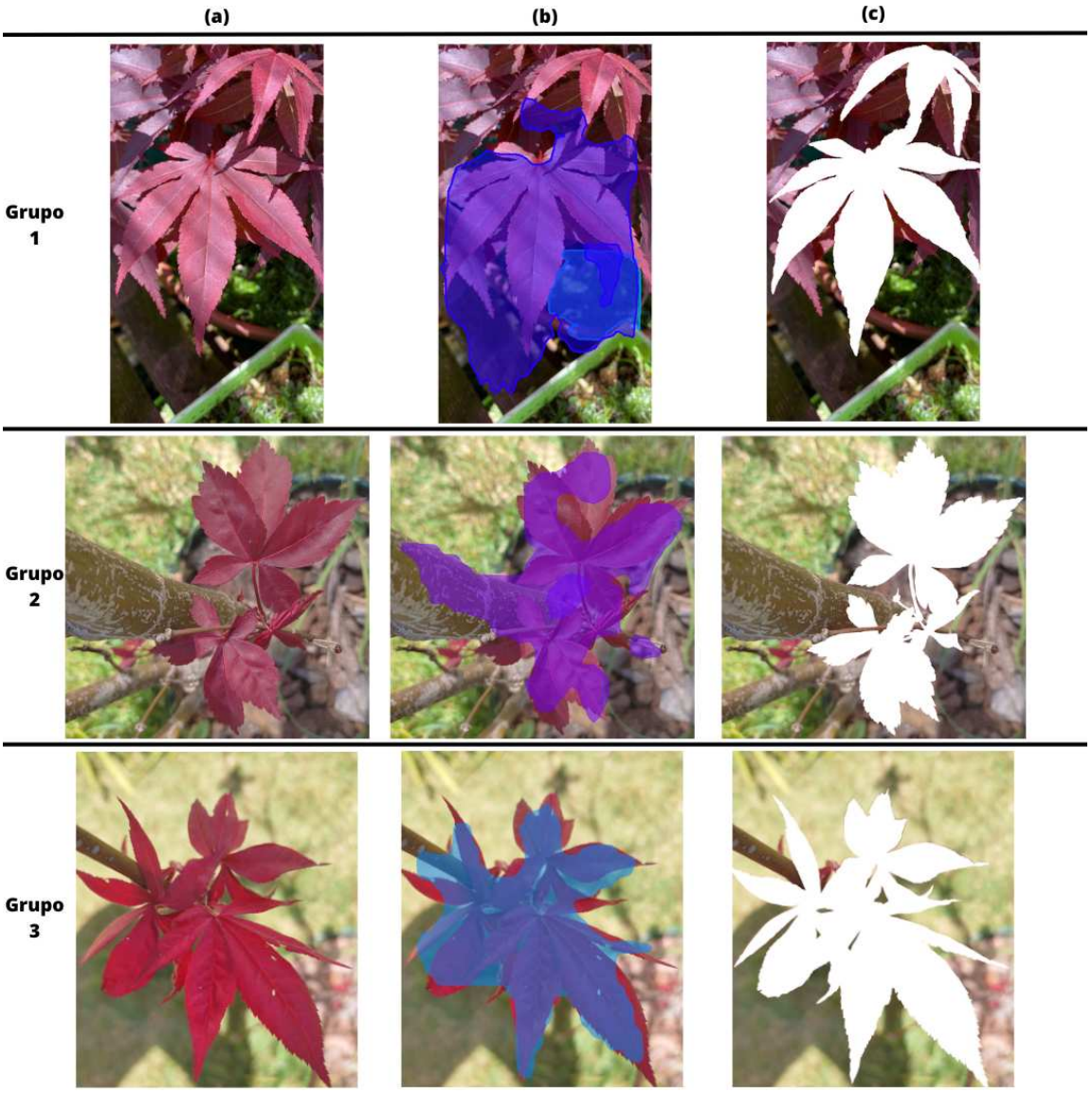
4.7.2 Métrica de avaliação de segmentação IoU para os testes com Mask R-CNN

Assim como nos testes com as imagens segmentadas pelo U2-Net, as do Mask R-CNN também foram testadas em cima dos códigos apresentados na Listagem 6 e na Listagem 7, ambos relacionados com o método de avaliação IoU.

O modelo foi treinado com imagens de plantas *Acer* palmatum da variedade Atropurpureum, e submetido a testes para reconhecer outras imagens distintas dessa mesma variedade. A seguir, juntamente com os valores de IoU de cada grupo de testes, há as máscaras geradas pelo Mask R-CNN visíveis na coluna (b) da Figura 35, as imagens originais na coluna (a), e na coluna (c) há as máscaras traçadas manualmente, as quais, no teste de IoU, foram sobrepostas às máscaras feitas pelo Mask R-CNN.

- Grupo 1 - IoU: 0.5301188750781597
- Grupo 2 - IoU: 0.4064202303551175
- Grupo 3 - IoU: 0.7359775700953535

Figura 35 - (a) Imagens originais da *Acer* palmatum Atropurpureum; (b) Máscaras traçadas pelo algoritmo Mask R-CNN; (c) Máscaras feitas manualmente



Fonte: Autoria própria (2023).

Visivelmente, no quesito qualitativo, as máscaras (b) traçadas pelo Mask R-CNN de todos os grupos não atenderam às expectativas. Tendo em vista que o conjunto de dados utilizado no treinamento do modelo totalizou apenas 33 imagens, é possível inferir que a causa dos resultados ruins tenha sido o fato que a quantidade de amostras precisa ser maior para treinar um modelo de Mask R-CNN. Como apresentado, o Grupo 3 obteve o maior número de IoU, e visivelmente é possível observar que, dentre os testes do Mask R-CNN realizados, sua máscara gerada pelo algoritmo foi a que mais seguiu a folha *Acer* de fato, possivelmente devido a qualidade da imagem e também pelo fato das amostras apresentarem as folhas como elemento principal.

5 CONCLUSÃO

Com base nos objetivos propostos, o presente trabalho buscou aplicar técnicas de Visão Computacional para a segmentação de regiões de interesse em imagens de plantas da subespécie Acer palmatum, com o intuito de auxiliar no processo de classificação das diferentes variedades desta planta. Para atingir esse objetivo, foram fundidos conceitos e práticas das áreas de Visão Computacional e Botânica, visando contribuir para a construção de um conjunto de dados diversificado contendo imagens desta espécie.

Por meio dos experimentos realizados, foram aplicadas diferentes técnicas de segmentação de imagens, com o objetivo de identificar as mais adequadas para destacar as características das plantas, distinguindo-as do fundo da imagem. Inicialmente, as técnicas aplicadas envolveram práticas com *Thresholding* Binário, de pré-processamento de imagem, e *Canny Edge Detection*, de processamento de imagem em si, ambas contidas na biblioteca de manipulação de imagem OpenCV do Python. Para práticas um pouco mais avançadas, foi utilizado a biblioteca RemBg, a qual utiliza o algoritmo de *Deep Learning* U2-Net, que visa segmentar o objeto saliente na imagem, diferenciando-o do fundo. Por fim, houve o uso de outro algoritmo de *Deep Learning*, o Mask R-CNN, com o propósito de testar sua qualidade de segmentação de objetos.

Com base nos resultados obtidos com o U2-Net, conclui-se que, no quesito qualidade de segmentação, o algoritmo U2-Net se saiu melhor nos experimentos realizados. Porém, para manipular o modo como o U2-Net funciona, é necessário alterar alguns parâmetros do código-fonte, pois seu funcionamento básico depende bastante de como a foto da imagem é registrada, a qual deve buscar deixar o objeto saliente na imagem e, de preferência, com o fundo embaçado, destacando melhor ainda o objeto. Com isso, é possível obter resultados satisfatórios com o U2-Net, apesar da qualidade da imagem influenciar bastante na qualidade da segmentação que o algoritmo realiza. Os valores de IoU, quando comparados com o valor da imagem extraída da página do autor do U2-Net, revelaram níveis aceitáveis ao fazer a sobreposição de uma máscara segmentada manualmente com a máscara gerada pelo U2-Net.

Os resultados obtidos com o Mask R-CNN não foram visivelmente positivos, e também os valores de IoU sobrepondo a máscara manualmente traçada com a

máscara gerada pelo algoritmo não foram altos, revelando que o treinamento do modelo deve ser realizado com um número maior de amostras para reconhecer a folha da *Acer* palmatum Atropurpureum na imagem e segmentá-la devidamente, tendo em vista que apenas 33 imagens da variedade Atropurpureum foram utilizadas no treinamento, com suas folhas segmentadas manualmente para treinar o algoritmo.

Apesar do resultado obtido ter sido parcialmente positivo em relação ao objetivo proposto, foi possível obter conclusões e sugestões válidas para trabalhos futuros envolvendo a segmentação em imagens de folhas de plantas, tais como:

- Os algoritmos de *Deep Learning* U2-Net e Mask R-CNN são válidos, quando manipulados da maneira correta, para segmentação de imagem envolvendo folhas de plantas.
- Técnicas de pré-processamento de *Thresholding* Binário são úteis em alguns casos como: segmentação de objetos, binarização de imagens, remoção de ruído e detecção de bordas.
- A segmentação manual é um método em potencial para preparação de dados para treinar um algoritmo.
- O método estatístico IoU é um meio estatístico válido para chegar a conclusões da qualidade de segmentação, onde ele faz a sobreposição da máscara ideal (geralmente traçada manualmente), com a máscara gerada por um algoritmo.
- Tendo em vista que o trabalho se trata de imagens, a visualização qualitativa também é útil, em alguns casos é possível dizer por meio de observação humana se a segmentação da imagem ficou ideal ou não.

É importante ressaltar que o curso de formação do autor deste trabalho não trata das informações e estudos aqui apresentados, portanto o conhecimento adquirido com as tecnologias utilizadas, e o entorno do assunto, foram válidas para um aprendizado a mais.

REFERÊNCIAS

ALZUBAIDI, L., ZHANG, J., HUMAIDI, A.J. et al. **Review of deep learning: concepts, CNN architectures, challenges, applications, future directions**. J Big Data 8, 53 (2021).

ASCARI, Rúbia Eliza de Oliveira Schultz; CASANOVA, Dalcimar; FÁVERO, Eliane Maria De Bortoli; SAGIORATTO, Patrick de Souza; BRAUN, Vinicius Caron. **Developing an automatic classifier of different plant genera of the subspecies Acer Palmatum**. Anais do Computer on the Beach, v. 14, p. 166-173, 2023.

BALDNER, F., Costa, P., Gomes, J. F. S. e Leta, F. R. (2017). **Metrologia por Imagem**. Elsevier Brasil.

BOMBARDELLI, F. G. **Localização de robôs móveis por aparência visual** (2014).

CANU, S. ***Train Mask R-CNN for Image Segmentation - Online (Free GPU)***, 2021. Disponível em: https://pysource.com/2021/08/10/train-mask-r-cnn-for-image-segmentation-online-free-gpu/. Acesso em: 09 abr. 2023.

CECHINEL, Christian. **Técnicas de Segmentação de Imagens a Cores**. 2000.2. Seminário Visão Computacional - CPGCC/UFSC. Disponível em: Técnicas de Segmentação de Imagens a Cores - Visão Computacional - Aldo von Wangenheim's HomePage (ufsc.br), Acesso em: 27 out. 2022.

CERIDONO, Suzana Galvão. **Acer palmatum - Japanese maple.** 5 mai. 2011. Disponível em: Acer palmatum – Japanese maple | Cultivando Elegância (wordpress.com). Acesso em: 28 set. 2011.

CHENG, H., et al.: ***Color Image Segmentation: Advances and Prospects***. *Pattern Recognition* 34(12), 2259–2281 (2001).

DING, L. GOSHTASBY, A. ***Pattern Recognition***, Volume 34, *Issue* 3, 2001, *Pages* 721-725.

EMBRAPA. **Agricultura digital: perspectivas e desafios para a inovação tecnológica.** In: EMBRAPA (Org.). Agricultura digital. Brasília, DF: Embrapa, 2020. Cap. 6, p. 119-136. Disponível em: https://www.alice.cnptia.embrapa.br/bitstream/doc/1126261/1/LV-Agricultura-digital-2020-cap6.pdf. Acesso em: 17 de mar. de 2023.

EVERINGHAM, M., ESLAMI, S.M.A., VAN GOOL, L. et al. ***The PASCAL Visual Object Classes Challenge: A Retrospective.*** Int J Comput Vis 111, 98–136 (2015). Disponível em: https://doi.org/10.1007/s11263-014-0733-5 Acesso em: 01 jun. 2023.

GENÉ-MOLA, J.; GREGORIO, E.; CHEEIN, F. A.; GUEVARA, J.; SANZ-CORTIELLA, R.; ESCOLÀ, A.; ROSELL-POLO, J.
R. Fruit detection, yield prediction and canopy geometric characterization using LiDAR with forced air flow.
Computers and Electronics in Agriculture, v. 168, 105121, Jan 2020. DOI: 10.1016/j.compag.2019.105121.

GOMES, Joana Alves. **Como identificar as várias espécies de Áceres**. 4 de jul. de 2019. Disponível em: As várias espécies de Áceres (jagarquiteturapaisagista.com). Acesso em: 03 out. 2022.

GONZALEZ, R. C. e Woods, R. E. (2002). **Digital image processing**. Prentice hall.

G. Tzimiropoulos, S. Zafeiriou e M. Pantic, ***Subspace Learning from Image Gradient Orientations***, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2454-2466, Dez 2012, doi: 10.1109/TPAMI.2012.40.

HE, Kaiming et al. Mask R-CNN. In: ***Proceedings of the IEEE International Conference on Computer Vision***. 2017. p. 2980-2988.

JARDINS. **Ácer: para um outono espetacular**. Disponível em: https://revistajardins.pt/acer-um-outono-espetacular/. Acesso em: 01/11/2022.

KE, S.-R., Thuc, H. L. U., Lee, Y.-J., Hwang, J.-N., Yoo, J.-H. e Choi, K.-H. (2013). **A review on video-based human activity recognition**. Computers, 2(2):88–131.

LECUN, Y., BENGIO, Y. & HINTON, G. ***Deep learning***. *Nature* 521, 436–444 (2015).

LI S.S., LI Q.Z., RONG L.P., TANG L., WANG J.J., e ZHANG B. 2015*a*. ***Analysis of the transcriptome of green and mutant golden-yellow leaves of Acer palmatum Thunb. using high-throughput RNA sequencing***. J. Hortic. Sci. Biotech. 90: 388–394.

LORENZI, H. et al. **Árvores Exóticas do Brasil:** Madeireiras, Ornamentais e Aromáticas. Nova Odessa: Instituto Plantarum, 2003. p 79-80.

OpenCV - ***ABOUT OPENCV***. Disponível em: About - OpenCV. Acesso em: 29 ago. 2022.

OTSU, N. (1979). ***A threshold selection method from gray-level histograms.*** *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62-66.

QIN, X. et al. (2020). U2-Net: ***Going Deeper with Nested U-Structure for Salient Object Detection. Pattern Recognition***, 106, 107404.

QIN, X., KHALIQ, A. (2021). U-2-Net: *Going Deeper with Nested U-Structure for Salient Object Detection. - Source Code.* ***Gradio Web Demo of U2-Net.*** Disponível em: https://github.com/xuebinqin/U-2-Net. Acesso em: 01 jun. 2023.

SATTI, V. SATYA, A. SHARMA, S. (2013). ***An automatic leaf recognition system for plant identification using machine vision technology***. International Journal of Engineering Science and Technology (IJEST). 5. 874-879.

S. A. Raut, M. Raghuvanshi, R. Dharaskar and A. Raut, ***"Image Segmentation – A State-Of-Art Survey for Prediction"*** 2009 *International Conference on Advanced Computer Control*, 2009, pp. 420-424, doi: 10.1109/ICACC.2009.78.

SCOTT, D.W. (2010), ***Histogram***. *WIREs Comp Stat*, 2: 44-48. Disponível em: https://doi.org/10.1002/wics.59, Acesso em: 09 nov. 2022.

SENTHILKUMARAN, N., RAJESH, R., "***Image Segmentation - A Survey of Soft Computing Approaches***" 2009 *International Conference on Advances in Recent*

Technologies in Communication and Computing, Kottayam, India, 2009, pp. 844-846, doi: 10.1109/ARTCom.2009.219.

SHAO, J. ZHOU, K. CAI Y. GENG, D. ***Application of an Improved U2-Net Model in Ultrasound Median Neural Image Segmentation***, *Ultrasound in Medicine & Biology, Volume* 48, *Issue* 12, 2022, *Pages* 2512-2520.

S.-S. Li, Q.-Z. Li, L. Tang, J. Wen. 2017. ***Pigment comparison and expression of chlorophyll metabolism genes in yellow and green Acer palmatum Thunb. ex Murray leaves****. Canadian Journal of Plant Science*. v. 97.

TAO, J., WANG, H., ZHANG, X., LI, X., & YANG, H. (2017). ***An object detection system based on YOLO in traffic scene****. In 2017 6th International Conference on Computer Science and Network Technology* (ICCSNT) (pp. 315-319). doi: 10.1109/ICCSNT.2017.8343709.

YANG, K., ZHONG, W., & LI, F. (2020). ***Leaf segmentation and classification with a complicated background using deep learning***. *Agronomy*, *10*(11), 1721.

WANG, Z. LI, H. ZHU Y. XU, T. ***Review of plant identification based on image processing, Archives of Computational Methods in Engineering*** 24(3) (2017) 637–654.

WANG, Z., WANG, E. & ZHU, Y. ***Image segmentation evaluation: a survey of methods.*** *Artif Intell Rev* 53, 5637−5674 (2020).

Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M. J. (2017). ***Big Data in Smart Farming: A Review.*** *Agricultural Systems*, 153, 69-80.

ZHU S., XIA X., ZHANG Q., BELLOULATA K., ***An Image Segmentation Algorithm in Image Processing Based on Threshold Segmentation***, 2007 *Third International IEEE Conference on Signal-Image Technologies and Internet-Based System*, 2007, pp. 673-678, doi: 10.1109/SITIS.2007.116.

ZHANG, Yu Jin. ***A survey on evaluation methods for image segmentation***. Pattern recognition, Elsevier, v. 29, n. 8, p. 1335–1346, 1996