UNIVERSIDADE FEDERAL FLUMINENSE

PAULO HENRIQUE BUENO LOPES

ASRDNet - Detecção de Ferrugem Asiática em Imagens de Folhas Próximas

NITERÓI

2024

PAULO HENRIQUE BUENO LOPES

ASRDNet - Detecção de Ferrugem Asiática em Imagens de Folhas Próximas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Orientador:
Leandro Augusto Frata Fernandes

NITERÓI

2024

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

L864a Lopes, Paulo Henrique Bueno
ASRDNET - DETECÇÃO DE FERRUGEM ASIÁTICA EM IMAGENS DE FOLHAS PRÓXIMAS / Paulo Henrique Bueno Lopes. - 2024.
82 f.

Orientador: Leandro Augusto Frata Fernandes.
Dissertação (mestrado)-Universidade Federal Fluminense, Instituto de Computação, Niterói, 2024.

1. Computação visual. 2. Segmentação de imagem. 3. Processamento de imagem. 4. Produção intelectual. I. Fernandes, Leandro Augusto Frata, orientador. II. Universidade Federal Fluminense. Instituto de Computação. III. Título.

CDD - XXX

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Paulo Henrique Bueno Lopes

ASRDNet - Detecção de Ferrugem Asiática em Imagens de Folhas Próximas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Aprovada em fevereiro de 2024.

BANCA EXAMINADORA

Prof. Dr. Leandro Augusto Frata Fernandes - Orientador, UFF

Prof. Dr. João Paulo Papa, UNESP

Profa. Dra. Aura Conci, UFF

Niterói

2024

Dedicatória: Dedico este trabalho à minha esposa Késsia e aos meus filhos Mateus e Tiago.

Agradecimentos

Agradeço primeiramente à Deus, por me manter são para concluir este trabalho. À minha esposa Késsia, minha eterna companheira, que durante todo este tempo, suportou uma carga inimaginável, para que eu pudesse concluir os meus estudos. Aos meus filhos, Mateus e Tiago, que foram minha luz no fim do túnel. Aos meus pais, Samuel e Creoneci, que sempre se sacrificaram e apoiaram para que eu tivesse bons estudos. Ao meu irmão Murillo, por me ajudar quando precisei. Aos meus sogros, Moracyr e Bernadeth, por me apoiar e auxiliar minha esposa durante os meus afastamentos. Ao meu orientador Leandro que me ajudou e nunca me abandonou por todo este tempo. Ao Alessandro (Xandão), que gentilmente me cedeu a ideia e várias informações para o trabalho. Aos colegas do PROGRAF, Eduardo, Raphael, Altobelli, Yanexis, dentre vários outros, que sempre que precisei de ajuda, prontamente me auxiliaram. Aos colegas de MINTER que durante a nossa primeira jornada me ajudaram a não desistir. À Elifas, por me ajudar quando mais precisei. E finalmente ao IFMT como um todo, por me disponibilizar os afastamentos quando precisei e compreender a situação da minha primeira jornada.

Resumo

Neste trabalho é proposto um modelo de rede neural convolucional que realiza a identificação de lesões em folhas de soja, doença esta chamada de ferrugem asiática, que é causada pelo fungo Phakopsora pachyrhizi. Para realizar a identificação das lesões, utiliza-se um modelo de rede neural convolucional para segmentar e extrair as características das imagens de treinamento e algoritmos para a escolha dos melhores hiper-parâmetros da rede. Um conjunto de imagens de folhas doentes é utilizado como entrada no treinamento da rede neural convolucional, a qual foi desenvolvida para suportar a multiresolução de diversos tamanhos de lesão, para treinamento e ajuste dos pesos da rede. Os dados de entrada são submetidos à aumento de dados, visto que o conjunto de dados possui somente 63 imagens de amostras. Como contribuição deste trabalho, está a criação de uma rede neural com ajuste nos pesos capaz de identificar a ferrugem asiática em folhas de soja sem o auxílio de um técnico especialista ou fitopatologista, utilizando uma menor quantidade de parâmetros treináveis e com um tamanho de modelo menor, quando comparado com redes de segmentação clássicas (U-Net, SegNet e DeepLab). A validação da técnica foi realizada utilizando imagens de folhas doentes anotadas por técnicos especializados na detecção da ferrugem asiática. A detecção foi avaliada utilizando a métrica F1-score, obtendo 0,9077 pontos, no dataset sem aumento de dados e 0,9251 pontos no dataset com aumento de dados.

Palavras-chave: ferrugem asiática da soja, *Phakopsora pachyrhizi*, redes neurais convolucionais.

Abstract

In this work, a convolutional neural network model is proposed to identify lesions on soybean leaves, a disease called Asian rust, caused by the fungus Phakopsora pachyrhizi. To perform lesion identification, a convolutional neural network model is used to segment and extract features from training images and algorithms for selecting the best network hyperparameters. A set of images of diseased leaves is used as input to train the convolutional neural network, which was developed to support multi-resolution of various lesion sizes for network training and weight adjustment. Input data is subject to data augmentation, as the dataset only has 63 sample images. As a contribution of this work, the creation of a neural network with weight adjustment capable of identifying Asian rust on soybean leaves without the assistance of a specialist technician or phytopathologist, using fewer trainable parameters and a smaller model size when compared to classic segmentation networks (U-Net, SegNet, and DeepLab). The technique's validation was performed using images of diseased leaves annotated by technicians specialized in Asian rust detection. Detection was evaluated using the F1-score metric, achieving 0.9077 points in the dataset without data augmentation and 0.9251 points in the dataset with data augmentation.

Keywords: Asian rust of soybean, Phakopsora pachyrhizi, convolutional neural networks.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Adam Adaptive Moment Estimation

ASPP Atrous Spatial Pyramid Pooling

ASR Asian Soybean Rust

ASRD Asian Soybean Rust Detection

CNN Convolutional Neural Network

FCN Fully Convolutional Networks

MF^{s} R-CNN Multi-Feature Fusion Faster R-CNN

OpenCV Open Source Computer Vision Library

ReLU Rectified Linear Unit

RGB Red Green Blue

RMSprop Root Mean Square Propagation

RPN Region Proposal Network

SGD Stochastic Gradient Descent

SLIC Simple Linear Iterative Clustering

SSNN Semantic Segmentation Neural Networks

SVM Support Vector Machine

VANTs Veículos Aéreos Não Tripulados

W&B Weights and Biases

Sumário

1 Introdução

A soja, *Glycine max (L.) Merrill*, é uma oleaginosa rica em proteínas. É considerada como um "feijão dourado" por causa das suas características e uso diversos (PUJARI et al., 2016, p. 75). Somente no ano de 2021 foram produzidos 362.947 milhões de toneladas em uma área plantada de 127.842 milhões de hectares, em todo o mundo, sendo que desta parcela, o Brasil é responsável pela colheita de 135.409 milhões de toneladas em 38.502 milhões de hectares (CONAB, 2022). A soja em forma de grão vegetal normalmente não é vista pelos consumidores finais, porém, os seus derivados estão presentes na alimentação diariamente. Uma grande porcentagem destes grãos estão presentes no cotidiano em forma de óleos vegetais, leite, proteína vegetal, ração animal e até na produção de biocombustíveis (THIAGO et al., 2003).

Com a popularização do plantio extensivo na forma de monocultura de soja, algumas doenças começaram a ser detectadas nas lavouras, sendo uma delas a ferrugem asiática, que é uma doença que afeta as plantas de soja e é considerada uma das principais ameaças para a produção de soja em várias partes do mundo. Essa doença é caracterizada pelo aparecimento de manchas de cor marrom-alaranjada nas folhas das plantas, semelhantes a pequenas lesões de ferrugem, daí o nome "ferrugem asiática".

A doença é causada pelo fungo *Phakopsora pachyrhizi*, um patógeno biotrófico, o que significa que ele depende da planta hospedeira para sobreviver e se reproduzir. O fungo passa por diferentes estágios de desenvolvimento, incluindo a produção de esporos que são liberados no ambiente e podem se espalhar para outras plantas de soja (GODOY et al., 2023). Devido à sua capacidade de causar danos significativos às plantações e à sua natureza altamente contagiosa, a ferrugem asiática é uma preocupação constante para os agricultores e cientistas agrícolas em regiões onde a soja é uma cultura importante.

A identificação precoce da ferrugem asiática é um fator crítico para minimizar os prejuízos causados por essa doença nas plantações de soja. Quanto mais cedo a presença da doença for detectada, mais rapidamente os agricultores podem implementar medidas

eficazes de controle, reduzindo o potencial de disseminação e os danos econômicos. Essas medidas podem incluir a aplicação estratégica de fungicidas, a remoção de plantas infectadas e a adoção de práticas agrícolas que reduzam a probabilidade de infecção e propagação da doença (EMBRAPA, 2023).

O papel desempenhado pelos fitopatologistas é de extrema importância nesse contexto. Esses especialistas possuem um conhecimento profundo sobre doenças de plantas, incluindo a ferrugem asiática, e são treinados para identificar sintomas e características distintas das doenças nas plantas, atuando em uma variedade de ambientes, como laboratórios de pesquisa, instituições acadêmicas, empresas do setor agrícola e agências governamentais.

A presença de um fitopatologista em uma plantação ou região pode ter um custo associado, seja na forma de salários, equipamentos ou recursos de laboratório. No entanto, os benefícios potenciais podem superar amplamente esses custos. Ao detectar a ferrugem asiática ou outras doenças em estágios iniciais, os fitopatologistas capacitam os agricultores a tomar ações proativas para mitigar os impactos adversos.

O uso de ferramentas de detecção de ferrugem asiática por meio de imagens pode ser uma ferramenta valiosa para identificar a doença em seus estágios iniciais, mesmo por pessoas que não são especialistas na área. Isso permite que os fitopatologistas concentrem seus esforços nos casos em que a detecção foi antecipada, otimizando o uso de recursos humanos na lavoura e evitando o desperdício. Dessa forma, é possível garantir um controle mais eficaz da doença e minimizar as perdas na produção.

Esta dissertação apresenta um novo modelo de rede neural convolucional aplicada à segmentação de imagens, tendo como foco a detecção de lesões de ferrugem asiática em folhas de soja, em seu estágio inicial.

Recentemente, a aplicação de técnicas avançadas em processamento de imagem e aprendizado de máquina tem impulsionado significativamente a pesquisa na detecção da ferrugem asiática em folhas de soja. Entre as abordagens emergentes, encontram-se aquelas que capitalizam o poder das redes neurais já estabelecidas (TETILA et al., 2020), como VGG-16, ResNet-50, Xception e Inception-v3, para realizar a classificação das folhas, que foram segmentadas utilizando superpixels, que é uma técnica de clusterização que envolve a subdivisão da imagem em regiões coesas e semanticamente relevantes. Essas arquiteturas de redes neurais convolucionais, que foram originalmente desenvolvidas para tarefas de classificação de imagem, estão sendo adaptadas para a tarefa específica de identificação da ferrugem asiática.

Outro trabalho, aborda uma ideia semelhante utilizando superpixels (SILVA et al., 2022). Essa abordagem é vantajosa, pois ela agrupa os pixels visualmente semelhantes, reduzindo a complexidade da imagem e tornando-a mais gerenciável para análises subsequentes. A segmentação por superpixels pode ser um passo preliminar útil antes de aplicar algoritmos de detecção específicos, permitindo uma abordagem mais focada na região de interesse, no caso, as lesões de ferrugem asiática.

A estratégia empregando redes neurais já preexistentes demonstra eficácia, contudo, convém ressaltar que algumas dessas redes exibem dimensões consideráveis em termos de complexidade do modelo, o que resulta em elevada demanda por recursos computacionais, bem como na utilização de um número substancial de parâmetros ajustáveis. Essa confluência de fatores pode, em determinadas circunstâncias, inviabilizar a implementação destas redes neurais em sistemas computacionais de capacidade limitada e ambientes de treinamento restritos. É importante observar que a viabilidade da adoção dessas redes neurais em tal contexto também está sujeita à escala do conjunto de dados de treinamento, o que, por sua vez, exerce influência sobre a capacidade de processamento requerida e, consequentemente, sobre a exequibilidade da abordagem.

Com o objetivo de otimizar a eficiência e a viabilidade computacional do modelo de rede neural, bem como mitigar a sobrecarga associada à gestão de um grande número de parâmetros treináveis, uma estratégia foi concebida. Nesse âmbito, propõe-se um modelo baseado na incorporação de três níveis distintos em termos de profundidade. Cada um desses níveis é projetado de modo a operar em uma faixa específica de profundidade, com vistas a extrair características em diferentes escalas de resolução. Tal abordagem é fundamentada no pressuposto de que informações discriminativas podem ser capturadas de forma mais eficaz quando a análise é realizada em múltiplas escalas, englobando desde detalhes finos até contextos mais amplos. A singularidade desse modelo reside na subsequente etapa de integração desses níveis. Ao término do processo de extração de características, as saídas individuais provenientes dos três níveis são unificadas, com o propósito combinar as informações relevantes extraídas das diferentes resoluções, combinando-as em uma única representação abstrata. Esta última camada de achatamento (*flatten*), é um estágio determinante para a consecução da tarefa de segmentação da imagem.

A estratégia delineada, portanto, é marcada pela meticulosa seleção e combinação das características extraídas em múltiplas resoluções. Isso permite otimizar a utilização de recursos computacionais, minimizando a demanda por parâmetros treináveis, ao passo que preserva a capacidade do modelo em abarcar informações abrangentes e detalhadas,

essenciais para a precisão da segmentação de imagens. Como tal, essa abordagem representa um avanço significativo em direção à síntese entre eficiência computacional e desempenho preciso na tarefa de segmentação de imagens de folhas de soja afetadas pela ferrugem asiática.

Durante o desenvolvimento do experimento, a rede neural ASRDNet, proposta nessa dissertação, foi treinada com um conjunto de dados composto por 63 imagens de folhas de soja afetadas pela ferrugem asiática, com e sem aumento de dados. Cada treinamento foi realizado utilizando hiperparâmetros escolhidos aleatoriamente, entre um intervalo pré-determinado. Após o treinamento, foi realizado um teste para avaliar o F1-score da rede em relação ao conjunto de dados de teste. Os 10 melhores resultados do F1-score foram selecionados para posterior comparação com os resultados das redes U-Net (Seção 3.2.1), DeepLabNet (Seção 3.2.2) e SegNet (Seção 3.2.3), que foram submetidas às mesmas condições de teste da ASRDNet. Em seguida, foi realizado Welch's t-test entre os resultados da rede ASRDNet e as demais redes (U-Net, SegNet e DeepLabNet). Com base no resultado do teste, foi possível comprovar que o F1-score da rede ASRDNet é superior ou igual ao das técnicas comparadas. Além disso, observou-se que a quantidade de parâmetros treináveis e o tamanho do modelo da rede ASRDNet são menores em comparação às outras técnicas.

Nesse contexto, o desenvolvimento deste modelo inovador de rede neural convolucional, focado na aplicação da segmentação de imagens, e com ênfase na identificação das lesões de ferrugem asiática presentes nas folhas de soja, tem culminado em um avanço de considerável relevância. O referido modelo demonstrou não apenas uma notável ampliação no F1-score em comparação a outras arquiteturas concorrentes, mas também se destaca por sua notável eficiência computacional e leveza, caracterizado por um tamanho de modelo reduzido e uma quantidade significativamente menor de parâmetros treináveis.

Como um todo, essa abordagem exibe um potencial intrínseco para impactar positivamente a produção agrícola, oferecendo soluções tecnológicas que alinham eficácia e eficiência na busca por mitigar os desafios impostos por doenças prejudiciais como a ferrugem asiática. Uma das possíveis soluções para lidar com a detecção precoce da ferrugem asiática nas plantações de soja é a criação de um aplicativo para smartphones. Esse aplicativo faria uso da rede neural ASRDNet, treinada especificamente para identificar e prever a presença de folhas de soja afetadas pela ferrugem asiática em tempo real, diretamente na lavoura. Através da análise das imagens das folhas capturadas pelo smartphone, o aplicativo seria capaz de distinguir entre folhas saudáveis e aquelas afetadas pela doença.

As contribuições desse trabalho são:

- Uma nova arquitetura multiresolução leve para segmentação de imagens, que foi aplicada no problema de segmentação de ferrugem asiática; e
- Um novo dataset de ferrugem asiática.

O presente trabalho, está dividido em **Fundamentação Teórica** (Capítulo 2), que define alguns termos necessários para o entendimento do texto, **Trabalhos Relacionados** (Capítulo 3), que apresenta a arquitetura de algumas redes de segmentação que foram utilizadas no trabalho, **ASRDNet** (Capítulo 4), que apresenta a arquitetura e as premissas utilizadas para o treinamento da rede neural, **Resultados** (Capítulo 5), apresenta o resultado final da pesquisa, o dataset criado, bem como os experimentos e implementações realizadas e ao final as **Conclusões e Trabalhos Futuros** (Capítulo 6) que apresenta o parecer final sobre o trabalho e trabalhos futuros que podem ser derivados desta dissertação.

2 Fundamentação Teórica

Neste capítulo serão mostrados conceitos e algumas técnicas existentes utilizadas na elaboração do trabalho.

2.1 Segmentação de Imagens

A segmentação de imagens é um processo fundamental no campo da visão computacional que visa dividir uma imagem digital em diferentes regiões ou objetos. O objetivo principal é simplificar e/ou transformar a representação da imagem, facilitando sua análise e compreensão.

Em outras palavras, a segmentação de imagens é uma abordagem que permite agrupar ou separar informações detalhadas e de alta precisão sobre a composição de uma imagem.

Ao atribuir rótulos semânticos aos pixeis, a segmentação de imagens permite entender a estrutura da cena retratada na imagem, identificando objetos e regiões de interesse com base em suas características visuais e semânticas. Isso é fundamental para diversas aplicações, desde a condução autônoma de veículos até o monitoramento de tráfego, análise de imagens médicas, detecção de objetos em tempo real e muito mais.

2.2 Modelos supervisionados

Os modelos de treinamento supervisionado são algoritmos de aprendizado de máquina que utilizam um conjunto de dados de treinamento rotulados. Cada exemplo desse conjunto é composto por um conjunto de características (variáveis de entrada) e um rótulo conhecido (variável de saída). Esses modelos são projetados para aprender os padrões presentes nos dados de treinamento, de modo que possam fazer previsões ou tomar decisões com base nessas informações.

Durante o treinamento supervisionado, o objetivo dos modelos é aprender uma função

que mapeie as características de entrada para os rótulos correspondentes. Essa função aprendida é então utilizada para realizar previsões ou tomar decisões para novos dados de entrada, que não foram previamente vistos durante o treinamento.

Os modelos de treinamento supervisionado são amplamente aplicados em diversas áreas, como reconhecimento de padrões, processamento de linguagem natural, visão computacional, entre outras. Eles desempenham um papel fundamental no campo do aprendizado de máquina, permitindo que os sistemas automatizados realizem tarefas de classificação, regressão e outras formas de análise preditiva.

Alguns dos modelos supervisionados mais comuns serão apresentados a seguir.

Regressão Linear: É um modelo estatístico utilizado para realizar previsões de valores numéricos contínuos com base em um conjunto de variáveis de entrada. Essas variáveis são conhecidas como variáveis independentes ou preditoras, e a variável que se pretende prever é chamada de variável dependente ou resposta (BISHOP, 2006, p. 137).

No caso da regressão linear, o objetivo é encontrar uma relação linear entre as variáveis independentes e a variável dependente. Essa relação é representada por uma equação linear que descreve como as variáveis independentes influenciam o valor da variável dependente. A regressão linear busca estimar os coeficientes dessa equação para melhor ajustar os dados de treinamento.

A regressão linear poderia ser empregada para estimar o preço de uma casa com base em características como área, número de quartos, entre outras variáveis relevantes. A ideia é encontrar uma equação linear que relacione essas características com o preço das casas observadas nos dados de treinamento. Uma vez que o modelo tenha sido treinado, ele pode ser utilizado para fazer previsões de preços para novas casas com base em suas características.

É importante ressaltar que a regressão linear pressupõe uma relação linear entre as variáveis e pode ser limitada em casos onde essa suposição não é atendida. Existem também diferentes variantes da regressão linear, como a regressão linear múltipla, que permite incluir várias variáveis independentes na equação de regressão.

Regressão Logística: É uma técnica estatística amplamente utilizada para realizar a classificação binária, onde o objetivo é separar amostras em duas classes distintas. É uma abordagem popular para lidar com problemas de classificação em que as variáveis de entrada (características) são utilizadas para determinar a classe de uma determinada amostra (BISHOP, 2006, p. 205).

No contexto mencionado, a regressão logística pode ser aplicada para classificar e-mails como spam ou não spam com base em características específicas de cada mensagem. Essas características podem incluir informações como palavras-chave, presença de determinados padrões, características do remetente, entre outros fatores relevantes. O modelo de regressão logística busca aprender uma função que estabeleça uma relação entre essas características e a probabilidade de um e-mail ser classificado como spam.

Ao treinar o modelo de regressão logística, é utilizado um conjunto de e-mails previamente rotulados como spam ou não spam. Com base nessas informações, o modelo ajusta os coeficientes da função logística para estimar as probabilidades das diferentes classes para cada e-mail de acordo com suas características. Essas probabilidades podem ser convertidas em uma classificação final, onde um valor de probabilidade acima de um determinado limiar é atribuído à classe de spam, e abaixo do limiar é atribuído à classe de não spam.

É importante destacar que a regressão logística utiliza a função logística (ou sigmoide) para modelar a relação entre as características e as probabilidades de classificação. Essa função garante que as probabilidades fiquem no intervalo entre 0 e 1, o que facilita a interpretação e a aplicação do modelo em tarefas de classificação.

Árvores de Decisão: São modelos de aprendizado de máquina que têm a capacidade de dividir o espaço de características em regiões distintas com base em regras de decisão. Essas estruturas de árvore são amplamente utilizadas em problemas de classificação e regressão (BISHOP, 2006, p. 663).

No contexto mencionado, as árvores de decisão podem ser empregadas para prever se um cliente irá cancelar um serviço com base em seus atributos. Esses atributos podem incluir características relevantes do cliente, como idade, histórico de uso do serviço, satisfação anterior, entre outros fatores que possam influenciar a decisão de cancelamento.

A construção de uma árvore de decisão envolve o processo de dividir o conjunto de dados com base em características específicas e regras de decisão. Em cada nó da árvore, uma pergunta é feita em relação a uma determinada característica e, com base na resposta, o caminho a ser seguido é determinado até que uma decisão final seja alcançada. As divisões sucessivas e as perguntas realizadas em cada nó são guiadas por algoritmos de aprendizado de máquina, como o algoritmo ID3, C4.5 ou CART.

Uma vez que a árvore de decisão tenha sido construída, é possível utilizá-la para fazer previsões para novos clientes, percorrendo a árvore de acordo com as características

fornecidas para determinar a probabilidade de cancelamento do serviço.

As árvores de decisão têm a vantagem de serem facilmente interpretáveis, pois as regras de decisão são visíveis nos ramos e nós da árvore. Além disso, elas podem lidar com uma variedade de tipos de dados, incluindo variáveis categóricas e numéricas.

Máquinas de Vetores de Suporte (SVM): São modelos de aprendizado de máquina capazes de separar amostras em diferentes classes, procurando maximizar a margem de separação entre elas. Esses modelos são amplamente utilizados em problemas de classificação, onde o objetivo é atribuir rótulos a amostras com base em suas características (BISHOP, 2006, p. 325).

No contexto mencionado, as SVMs são frequentemente empregadas para a classificação de imagens em categorias como "gato" ou "cachorro" com base em suas características visuais. As características visuais podem incluir texturas, formas, cores, entre outros elementos relevantes presentes nas imagens. O modelo de SVM busca encontrar um hiperplano que melhor separe as características das duas classes, maximizando a margem entre elas.

A ideia por trás das SVMs é encontrar um limite de decisão ótimo que maximize a distância entre as amostras de diferentes classes, permitindo uma classificação mais precisa. Além disso, as SVMs podem lidar com dados não linearmente separáveis, utilizando truques matemáticos, como o uso de funções de kernel, para mapear as características originais para um espaço de maior dimensionalidade onde a separação linear é possível.

Uma vez treinado, o modelo de SVM pode ser usado para classificar novas imagens, atribuindo rótulos de acordo com sua posição em relação ao hiperplano de decisão aprendido durante o treinamento.

É importante ressaltar que as SVMs têm sido aplicadas em uma variedade de problemas de classificação, não se limitando apenas à classificação de imagens. Elas têm sido utilizadas em áreas como bioinformática, reconhecimento de voz, análise de sentimentos, entre outros.

Redes Neurais: São modelos de aprendizado de máquina inspirados no funcionamento do cérebro humano. Elas são compostas por camadas de neurônios interconectados que processam informações de entrada e geram saídas com base em pesos sinápticos ajustáveis. Essas redes são capazes de aprender a partir de exemplos e realizar tarefas complexas de classificação e regressão, entre outras (BISHOP, 2006, p. 225).

No exemplo mencionado, uma rede neural pode ser treinada para reconhecer dígitos

escritos à mão em uma imagem, o que é extremamente útil em sistemas de reconhecimento de caracteres. Durante o treinamento, a rede neural é alimentada com um conjunto de imagens contendo dígitos escritos à mão, juntamente com suas correspondentes classes (dígitos de 0 a 9). A rede neural ajusta automaticamente os pesos sinápticos entre os neurônios em cada camada, de modo a aprender a associar as características visuais dos dígitos com as classes corretas. Uma vez que a rede neural tenha sido treinada, ela pode ser aplicada a novas imagens contendo dígitos escritos à mão para fazer previsões de classe. A rede neural analisa as características da imagem através das camadas de neurônios interconectados e, com base nos padrões aprendidos durante o treinamento, atribui a cada imagem um rótulo correspondente ao dígito identificado.

As redes neurais artificiais são altamente flexíveis e podem ser adaptadas para diversas tarefas, desde reconhecimento de objetos e reconhecimento de fala até tradução automática e análise de sentimentos. Esses modelos têm sido impulsionados pelos avanços na capacidade computacional e pelos desenvolvimentos em algoritmos de treinamento, como o algoritmo de retropropagação de erro.

2.2.1 Modelos de CNN existentes

As Redes Neurais Convolucionais (CNN) são um tipo de modelo de aprendizado de máquina que foi desenvolvido especificamente para processamento de imagens. Esses modelos são compostos por várias camadas interconectadas, que são responsáveis por realizar convoluções e operações de agrupamento (*pooling*) nas imagens de entrada. A principal ideia por trás das CNNs é que cada camada aprende a extrair características cada vez mais complexas, permitindo que o modelo reconheça objetos, texturas e padrões em imagens (BISHOP, 2006, p. 267).

As camadas convolucionais utilizam filtros para extrair informações relevantes da imagem, como bordas, texturas e padrões. Esses filtros são aplicados na imagem de entrada, gerando uma nova imagem com as características extraídas. Em seguida, as camadas de agrupamento reduzem a resolução espacial da imagem, agrupando pixeis em regiões maiores e reduzindo o tamanho da imagem. Esse processo é repetido várias vezes, com o objetivo de extrair características cada vez mais complexas da imagem.

As CNNs são capazes de lidar com imagens de tamanhos e proporções diferentes, graças ao processo de convolução que é aplicado em toda a imagem. Além disso, elas são capazes de aprender de forma automática as características relevantes da imagem, sem a necessidade de intervenção humana. Essas características podem ser usadas para

realizar diferentes tarefas, como reconhecimento de objetos, classificação de imagens e segmentação de imagens.

Estas redes têm sido amplamente utilizadas em diversas áreas, como reconhecimento facial, diagnóstico médico, análise de imagens geológicas e detecção de objetos em vídeos. A eficácia desses modelos tem sido demonstrada em vários estudos e competições de aprendizado de máquina, o que mostra o potencial dessas técnicas para solucionar problemas complexos.

Pré-processamento: Antes de iniciar o treinamento de um modelo de aprendizado, costuma ser necessária a preparação adequada dos dados de treinamento. Esse processo envolve uma série de etapas cuidadosas para garantir a qualidade e a consistência dos dados, visando obter resultados precisos e confiáveis.

Uma etapa inicial nesse processo é o redimensionamento das imagens para um tamanho consistente. Isso é necessário porque as imagens podem ser capturadas em diferentes resoluções e proporções, o que pode afetar a capacidade do modelo de aprender padrões relevantes. Portanto, ao redimensionar as imagens, é possível ajustá-las para uma resolução fixa, permitindo que o modelo processe sempre imagens com o mesmo tamanho de entrada para a rede.

Além disso, é importante normalizar os valores dos pixeis das imagens. A normalização é realizada para garantir que os dados estejam em uma escala adequada, o que facilita o processo de treinamento do modelo. Normalmente, isso é feito dividindo os valores dos pixeis pelo valor máximo possível, como 255 para imagens de 8 bits. Essa etapa é crucial, pois evita que valores discrepantes dominem o treinamento e influenciem negativamente o desempenho do modelo.

Caso seja necessário aumentar a diversidade do conjunto de treinamento, é comum aplicar transformações adicionais, como o aumento de dados (*data augmentation*). Essa técnica consiste em gerar novas imagens a partir das existentes, aplicando transformações como rotações, espelhamentos, ampliações e deslocamentos. Isso cria variações artificiais nos dados de treinamento, aumentando sua quantidade e diversidade.

Construção da arquitetura da rede: Arquiteturas baseadas em CNNs são amplamente utilizadas em tarefas de segmentação de imagens. Essas arquiteturas são tipicamente compostas por camadas de convolução, camadas de *pooling* e camadas de *upsampling*, que desempenham papéis fundamentais no aprendizado de características relevantes em diferentes níveis de abstração nas imagens.

A primeira etapa da arquitetura da rede é a convolução. Essa operação consiste na aplicação de um conjunto de filtros convolucionais a uma imagem de entrada. Cada filtro convolucional é responsável por detectar características específicas, como bordas, texturas ou formas, através da convolução do filtro com a imagem. A camada de convolução gera um mapa de características, que é uma representação intermediária da imagem, destacando informações relevantes para a tarefa de segmentação.

Após a convolução, a arquitetura da rede emprega camadas de *pooling*, também conhecidas como camadas de subamostragem (*downsampling*). Essas camadas têm como objetivo reduzir a dimensionalidade espacial do mapa de características, diminuindo a resolução da imagem. O *pooling* é geralmente realizado por meio da operação de *max pooling*, onde cada região do mapa de características é dividida em pequenos blocos e o valor máximo de cada bloco é selecionado e colocado na saída da camada, descartando informações menos relevantes. Esse processo permite a redução do número de parâmetros e, ao mesmo tempo, mantém as principais características aprendidas até o momento.

Camadas de *upsampling*, também conhecidas como camadas de sobre-amostragem, têm a função de aumentar a resolução espacial do mapa de características, permitindo a reconstrução da segmentação detalhada da imagem. Existem diferentes abordagens para realizar o *upsampling*. Uma delas é a utilização de camadas deconvolucionais, que aplicam filtros inversos para aumentar a resolução. Outra abordagem comum é o uso de camadas de *upsampling* bilinear, que interpola linearmente os valores dos pixeis para aumentar a resolução.

Além das camadas de convolução, *pooling* e *upsampling*, é comum a utilização de funções de ativação, como a função Rectified Linear Unit (ReLU). A função ReLU aplica uma transformação não-linear aos valores dos pixeis, definida como $f(x) = max(0, x)$. Essa função é aplicada elemento a elemento, substituindo qualquer valor negativo por zero.

Ao permitir que neurônios ativem-se apenas quando a entrada é positiva, a função ReLU ajuda a lidar com problemas de desvanecimento do gradiente (*vanishing gradient*) comuns em redes neurais profundas. A ativação não linear introduzida pela função ReLU evita que os gradientes se tornem muito pequenos à medida que passam pelas camadas, facilitando o treinamento eficaz de redes neurais mais profundas, permitindo que o modelo aprenda a representar relações complexas entre os dados.

No geral, a arquitetura da rede neural é projetada para aprender características em diferentes níveis de abstração, começando com características de baixo nível, como bor-

das e texturas, nas camadas iniciais, e avançando para características mais abstratas e semânticas, como objetos e regiões, nas camadas posteriores. Essa estrutura permite que o modelo capture informações contextuais relevantes para a segmentação de imagens, melhorando a precisão e a qualidade dos resultados obtidos.

2.3 Modelos de rede convolucional aplicadas à segmentação de imagens

Uma Rede Neural de Segmentação Semântica (SSNN) é uma arquitetura específica projetada para a tarefa de segmentação de imagens, onde o objetivo é atribuir uma etiqueta de classe a cada pixel da imagem, identificando as diferentes regiões semânticas presentes na imagem.

A CNN é composta por camadas convolucionais para extração de características globais da imagem e camadas totalmente conectadas para a classificação final da rede neural de segmentação de imagens, como a Fully Convolutional Networks (FCN) (LONG et al., 2015).

Além da FCN, outras arquiteturas comuns incluem a U-Net (Seção 3.2.1), a DeepLabNet (Seção 3.2.2), SegNet (Seção 3.2.3), entre outras. Cada uma dessas arquiteturas possui suas próprias características e modificações em relação à FCN, visando melhorar o desempenho ou lidar com desafios específicos da tarefa de segmentação de imagens.

Essas diferentes arquiteturas podem variar na forma como realizam o mapeamento de características, a incorporação de informações contextuais, o uso de conexões residuais, a aplicação de técnicas de atenção, entre outros aspectos. Algumas arquiteturas também podem combinar técnicas de detecção de objetos com a segmentação de imagens, permitindo identificar objetos individuais enquanto segmenta a imagem, como abordado a seguir.

U-Net: A arquitetura U-Net é conhecida por sua forma característica de "U", composta por uma rede *encoder* e uma rede *decoder*. O *encoder* captura as características da imagem em diferentes níveis de abstração, enquanto o *decoder* realiza a reconstrução da segmentação em alta resolução. A U-Net utiliza conexões residuais, que permitem o fluxo direto de informações entre as camadas do *encoder* e do *decoder*, auxiliando na preservação de detalhes.

SegNet: A arquitetura SegNet também possui um *encoder* e um *decoder*, mas difere

da FCN em como lida com a resolução espacial. Ao realizar o *pooling* durante a fase de *downsampling*, a SegNet salva os índices dos pixeis máximos, permitindo o *upsampling* preciso durante a fase de reconstrução. Isso ajuda a manter os detalhes espaciais importantes para a segmentação de imagens.

DeepLabNet: A arquitetura DeepLabNet utiliza um conceito chamado *dilated convolutions* (convoluções dilatadas) para capturar informações contextuais em diferentes escalas. Essa abordagem permite que a rede neural receba um campo receptivo maior sem reduzir a resolução espacial, evitando perda de detalhes. A DeepLabNet também utiliza Atrous Spatial Pyramid Pooling (ASPP), que combina convoluções dilatadas em paralelo em diferentes taxas de dilatação para capturar informações em diferentes escalas.

Portanto, embora a FCN seja uma utilizada para a segmentação de imagens, é importante destacar que existem outras arquiteturas de redes neurais específicas para essa tarefa. A escolha da arquitetura depende das necessidades específicas do problema e das características das imagens ou dados em questão.

3 Trabalhos Relacionados

Neste capítulo, abordaremos dois artigos pertinentes à segmentação de imagens de folhas de soja para detecção da ferrugem asiática, além de discorrer sobre os modelos de redes neurais empregados como pontos de comparação nos experimentos da técnica desenvolvida.

3.1 Detecção e classificação de ferrugem asiática em folhas de soja

Nesta seção, iremos explorar uma seleção de pesquisas recentes que têm se dedicado à detecção da ferrugem asiática em folhas de soja. Vamos analisar como essas pesquisas abordam a detecção da ferrugem asiática, as técnicas e algoritmos utilizados.

3.1.1 An automatic phytopathometry system for chlorosis and necrosis severity evaluation of asian soybean rust infection

A técnica desenvolvida no artigo (SILVA et al., 2022) visa automatizar a avaliação da gravidade da infecção por ferrugem asiática da soja (Asian Soybean Rust (ASR)) a partir de imagens de folhas de soja. O artigo apresenta uma nova abordagem (Figura 2) para avaliar a gravidade da ASR em imagens de folhas de soja no espectro visível. Esta abordagem possui três aspectos principais: primeiro, utiliza intervalos de cores CIELab para segmentar áreas de folhas saudáveis, cloróticas e necróticas; segundo, calcula três índices intermediários de avaliação de infecção (Clorose, Necrose e Limiar de Dano); terceiro, propõe um procedimento de seleção de vizinhança para incluir a clorose ao redor da necrose em um novo índice de avaliação (*Rustindex*).

Figura 2: Diagrama de fluxo da abordagem proposta para quantificar a infecção por ASR.

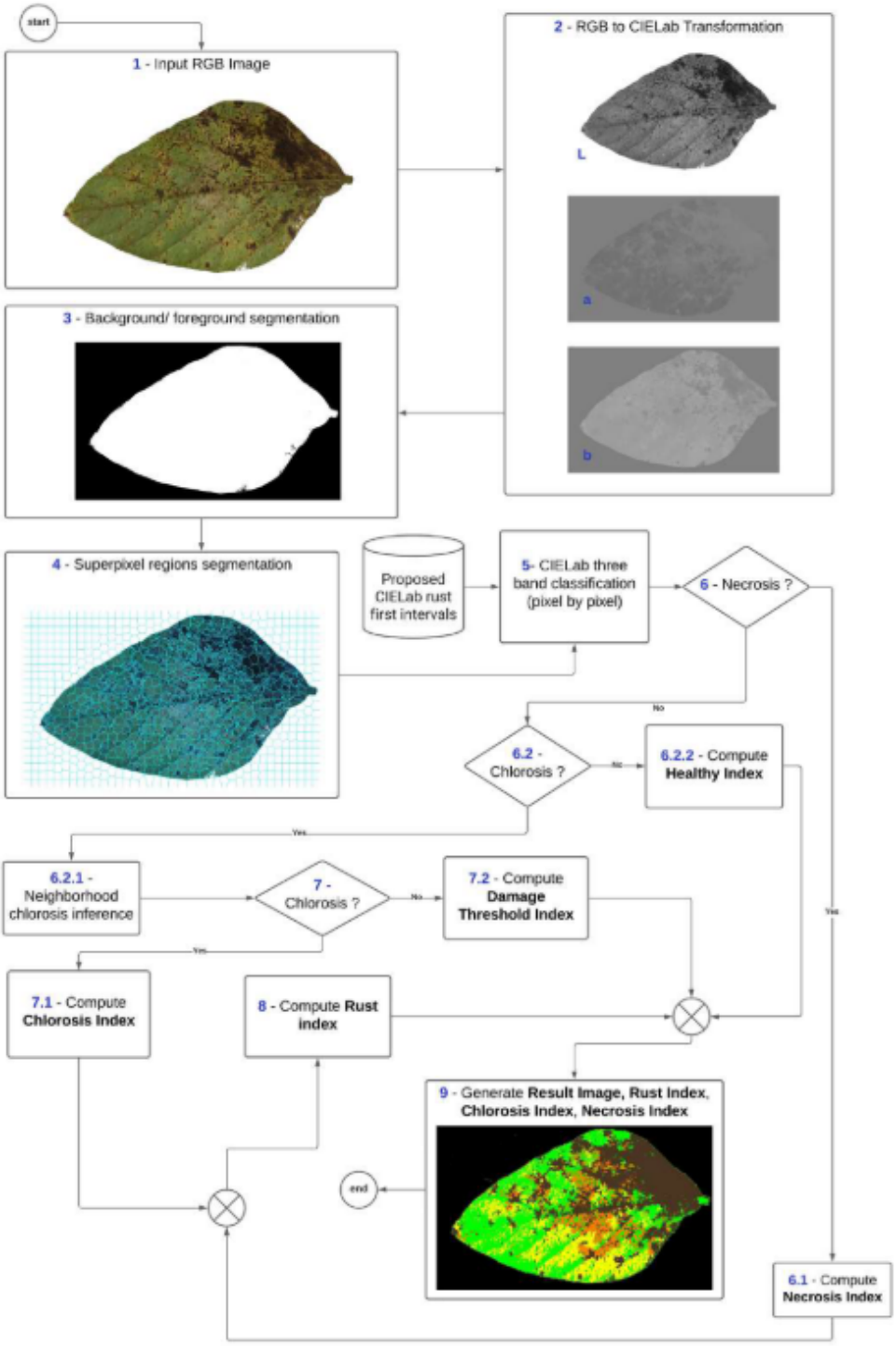


Imagem extraída de (SILVA et al., 2022, p. 3).

Os passos do processo proposto podem ser detalhados abaixo:

Aquisição de Imagens: O processo começa com a aquisição de imagens de folhas de soja infectadas com ASR. Essas imagens são obtidas de diferentes fontes, incluindo experimentos de campo, laboratório e bases de dados externas.

Conversão para o Espaço de Cores CIELab: As imagens de folhas de soja são convertidas do espaço de cores RGB para o espaço de cores CIELab. O espaço de cores CIELab é escolhido devido à sua adequação ao processamento de cores naturais de objetos, solo e plantas. Embora o espaço de cores CIELab tenha canais relacionados ao croma, ele ainda pode oferecer benefícios em relação ao processamento de cores naturais em comparação com o espaço de cores RGB.

Segmentação da Área da Folha: As imagens são processadas para separar a área da folha do plano de fundo. Isso é feito por meio da aplicação de limiares nos canais de cores CIELab. Os valores dos limiares são definidos para separar efetivamente a área da folha do fundo.

Segmentação por Superpixels: A imagem da folha segmentada é então dividida em regiões maiores e mais visualmente coerentes chamadas de superpixels. Isso é alcançado usando um algoritmo de superpixels que agrupa os pixels em regiões que compartilham características visuais semelhantes.

Categorização da Infecção: Cada superpixel é categorizado como saudável, cloró-tico ou necrótico com base em intervalos de cores pré-definidos no espaço CIELab. Esses intervalos são determinados empiricamente e após discussões com especialistas na área. Superpixels que se encaixam nos intervalos de cor específicos são classificados em uma das categorias.

Cálculo dos Índices de Avaliação: Os superpixels classificados são usados para calcular três índices intermediários de avaliação da infecção: Clorose, Necrose e Limiar de Dano. O índice de Clorose leva em consideração a vizinhança de pixels cloróticos para determinar a gravidade da infecção. O índice de Necrose considera a presença de pixels necróticos. O Limiar de Dano é calculado a partir de pixels em uma categoria específica.

Criação do Índice de Ferrugem (*Rustindex*): A partir dos índices intermediários, é calculado o *Rustindex*, que é um índice abrangente que leva em consideração os diferentes tipos de sintomas (clorose e necrose) e seus arredores.

Resultados e Saída: Os índices de avaliação são usados para gerar uma imagem final que destaca as áreas afetadas pela infecção por ASR e sua gravidade. Isso fornece uma representação visual da infecção que pode ser usada para análise e tomada de decisões.

3.1.2 Automatic Recognition of Soybean Leaf Diseases Using UAV Images and Deep Convolutional Neural Networks

A abordagem proposta por Tetila et al. (2020) adotou o método de superpixels Simple Linear Iterative Clustering (SLIC) para segmentar as imagens das folhas de soja (Figura 3) capturadas por Veículos Aéreos Não Tripulados (VANTs), permitindo uma análise mais detalhada das áreas afetadas pelas doenças e foi escolhido porque é mais rápido com complexidade linear e possui mais eficiência de memória do que métodos baseados em redes neurais. Para realizar a classificação da severidade da doença, utilizou-se quatro modelos de rede neural, cada um com diferentes conjuntos de parâmetros, a fim de identificar qual abordagem oferece a maior precisão.

Figura 3: Proposta de um sistema de visão computacional para identificação de doenças em folhas de soja com imagens de VANT. (a) Aquisição de imagens. (b) Segmentação SLIC. (c) Conjunto de dados de imagens. (d) Classificação.

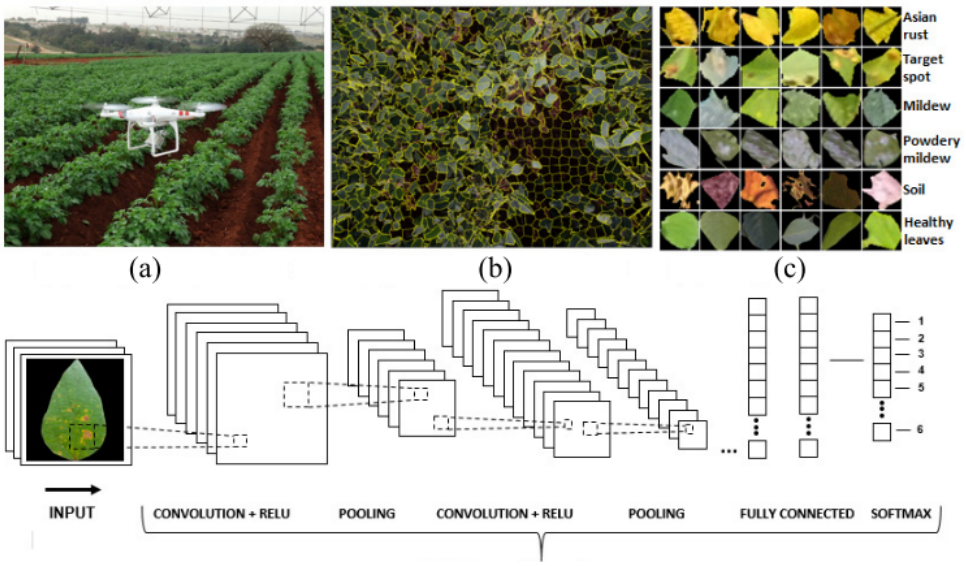


Imagem extraída de (TETILA et al., 2020, p. 2).

O processo envolve dois passos principais: segmentação e classificação.

O primeiro passo é segmentar as imagens das folhas de soja usando a técnica de superpixels SLIC. O algoritmo SLIC agrupa pixels em regiões semelhantes com base em suas características de cor e posição. Os autores optaram pelo SLIC porque é rápido, eficiente em termos de memória e gera regiões que aderem bem às bordas da imagem.

Captura de Imagens: Imagens das plantas de soja foram capturadas por VANTs com uma câmera de alta resolução.

Segmentação SLIC: O algoritmo SLIC foi aplicado às imagens capturadas, criando regiões superpixel.

Análise Visual: Cada segmento de folha foi analisado visualmente por um especialista e rotulado em uma das 7 classes: ferrugem asiática, mancha alvo, míldio, oídio, solo exposto, solo palha ou folha saudável. Isso cria um conjunto de dados rotulado para treinamento e teste.

O segundo passo envolve a classificação das imagens segmentadas usando modelos de aprendizado profundo conhecidos como CNN. Essas redes são altamente eficazes para a tarefa de reconhecimento de padrões em imagens.

Pré-processamento: As imagens de superpixels foram pré-processadas para um tamanho uniforme, usando técnicas de redimensionamento.

Treinamento do Modelo: Diferentes modelos de CNN, incluindo Inception-v3, VGG-19, ResNet-50 e Xception, foram treinados usando o conjunto de dados de imagens de superpixels rotulados. Cada modelo aprende a identificar características visuais associadas às diferentes classes de doenças, saudáveis e regiões de não interesse.

Aumento de Dados: Técnica de aumento de dados foi aplicada para aumentar a quantidade de dados de treinamento. Ela aplica operações como rotação, redimensionamento e espelhamento horizontal para criar variações das imagens.

Avaliação do Modelo: Os modelos treinados foram avaliados usando métricas como precisão, tempo de treinamento e erro de aprendizado. Essas métricas ajudam a identificar qual modelo é mais eficaz para a tarefa de classificação de doenças de soja.

Seleção de Parâmetros: Diferentes estratégias de treinamento, incluindo ajuste fino (*fine-tuning*) e transferência de aprendizado (*transfer learning*), foram testadas para determinar qual abordagem produz os melhores resultados.

3.1.3 Detecting soybean leaf disease from synthetic image using multi-feature fusion faster R-CNN

A técnica proposta por Zhang et al. (2021) visa abordar a questão da detecção precisa de doenças nas folhas de soja, propondo um método que envolve a geração de imagens sintéticas de doenças nas folhas de soja e o treinamento de um modelo Multi-Feature Fusion Faster R-CNN (MF^3 R-CNN) com base nesse conjunto de dados sintético. O MF^3 R-CNN é um modelo aprimorado do Faster R-CNN (REN et al., 2015) padrão e

é projetado para detectar doenças nas folhas de soja em cenários complexos. O modelo utiliza uma rede de extração de características com base na ResNet-50 (HE et al., 2016) e introduz uma *skip-connection* em sua estrutura. Camadas distintas na rede de extração de características se conectam de maneira a integrar eficazmente múltiplas características.

Figura 4: Mecanismo de funcionamento do MF³ R-CNN

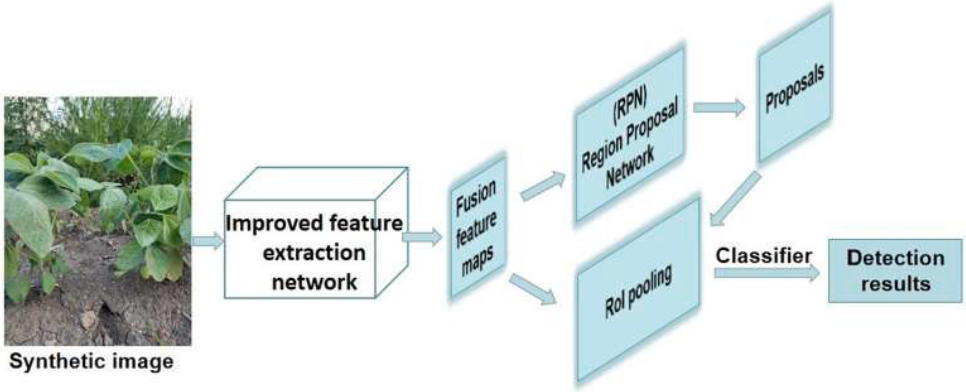


Imagem extraída de (ZHANG et al., 2021, p. 5).

A escolha do ResNet-50 como rede de extração de características se deve ao fato de que o ResNet-50 apresenta um desempenho forte na representação de características. A fim de implementar a "fusão de múltiplas características"no MF³ R-CNN, primeiro, foram introduzidas as *skip-connections* na estrutura do ResNet e, em seguida, os mapas de características gerados pela fusão são enviados para a rede de proposição de região RPN e a camada de agrupamento de região de interesse (*RoI pooling*). O mecanismo de funcionamento do MF³ R-CNN é apresentado na Figura 4.

Cinco métodos de *skip-connections* foram explorados neste trabalho, denominados de *Early skip*, *Halfway skip 1*, *Halfway skip 2*, *Late skip* e *Fusion skip*. A diferença entre as *skip-connections* criadas é a localização da camada conectada à camada de extração de características definida no detector de referência.

3.2 Redes neurais para segmentação

Nesta seção, vamos discutir as redes neurais para segmentação que serão utilizadas neste estudo com o propósito de comparar seus desempenhos. Será fundamental avaliar e comparar essas redes para compreender quais delas se destacam em tarefas de segmentação de imagens.

3.2.1 U-Net

A rede neural U-Net consiste em uma rede *encoder-decoder* com *skip connections*, projetada para realizar tarefas de segmentação de imagens. A rede é composta por camadas convolucionais, camadas de pooling, camadas de convolução transposta e camadas de ativação Rectified Linear Unit (ReLU).

Figura 5: Arquitetura da Rede neural U-Net

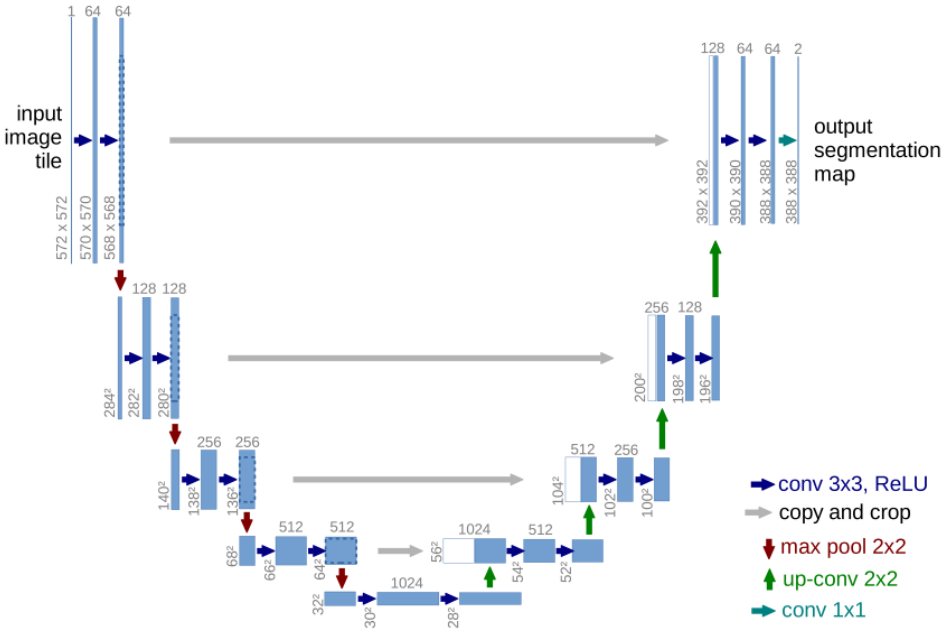


Imagem extraída de (RONNEBERGER et al., 2015, p. 2).

A primeira parte da rede, chamada de *encoder*, consiste em várias camadas convolucionais e de *pooling* que reduzem a resolução da imagem de entrada e extraem características de níveis mais altos. Em seguida, a rede passa por uma parte chamada *decoder*, que é responsável por aumentar a resolução da imagem e produzir uma máscara de segmentação.

A arquitetura da U-Net (Figura 5) é caracterizada pelas *skip-connections*, que permitem que as informações de níveis mais baixos da imagem de entrada sejam incorporadas ao processo de segmentação. Essas conexões são realizadas através de caminhos laterais que conectam as camadas do *encoder* diretamente às camadas correspondentes do *decoder*.

A U-Net se destaca por sua capacidade de segmentar objetos pequenos e detalhados em imagens médicas, como células e estruturas neurais. Ela também é capaz de lidar com imagens de tamanhos e formatos diferentes.

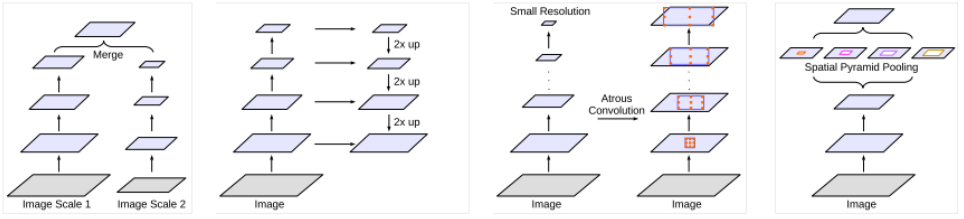
Essa arquitetura foi originalmente proposta por Ronneberger et al. (2015) para seg-

mentação de imagens biomédicas, mas desde então tem sido utilizada amplamente em diversas aplicações, incluindo segmentação de imagens de satélite e de imagens aéreas (ULMAS et al., 2020).

3.2.2 DeepLabNet

DeepLabNet é uma rede neural convolucional de segmentação semântica, desenvolvida inicialmente por uma equipe do Google Research em 2014 (CHEN; PAPANDREOU; KOKKINOS et al., 2014).

Figura 6: Arquitetura da Rede neural DeepLabNet v3



(a) Image Pyramid (b) Encoder-Decoder (c) Deeper w. Atrous Convolution (d) Spatial Pyramid Pooling

Imagem extraída de (CHEN; PAPANDREOU; SCHROFF et al., 2017, p. 2).

A arquitetura do DeepLabNet (Figura 6) é baseada em uma série de convoluções profundas, com um conjunto de camadas convolucionais intercaladas com camadas de agrupamento, que são projetadas para reduzir a dimensionalidade dos recursos da imagem. A arquitetura original foi baseada na rede neural totalmente conectada (FCN) proposta por Long et al. (2014), mas com algumas modificações que melhoraram significativamente a precisão da segmentação de imagens.

Uma das principais contribuições da DeepLabNet foi a incorporação de um mecanismo de atenção espacial que permitiu que a rede se concentrasse em áreas importantes da imagem, em vez de analisar toda a imagem de uma vez. Isto foi possível adicionando uma camada de convolução *atrous* (ou convolução com espaço livre), que permitiu a captura de informações contextuais em uma gama ampla de escalas, sem aumentar a carga computacional (CHEN; PAPANDREOU; KOKKINOS et al., 2014).

A rede também fez adição de um modelo de escala múltipla para capturar informações em escalas diferentes. Isso envolveu a geração de várias resoluções de imagem de entrada e a aplicação da rede neural a cada resolução. Os recursos resultantes de cada resolução foram então agregados em uma única saída de segmentação de imagens (CHEN; PAPANDREOU; KOKKINOS et al., 2016) e incorporando um ASPP aprimorado, normalização

de lote e uma maneira melhor de codificar o contexto multiescala para melhorar ainda mais o desempenho (CHEN; PAPANDREOU; SCHROFF et al., 2017).

Esta rede neural foi avaliada em vários conjuntos de dados de *benchmark*, incluindo o conjunto de dados de segmentação de imagens PASCAL VOC (EVERINGHAM et al., 2010) e o conjunto de dados de segmentação de imagens COCO (LIN, T. Y. et al., 2014). Em ambos os conjuntos de dados, a DeepLabNet superou os melhores resultados anteriores, alcançando uma precisão de segmentação de imagens significativamente maior.

A DeepLabNet também foi utilizada em aplicações, incluindo a segmentação de imagens médicas (RONNEBERGER et al., 2015), detecção de objetos em tempo real (CAI et al., 2018) e análise de imagens de satélite (MARMANIS et al., 2016). Essas aplicações demonstram a versatilidade e eficácia do DeepLabNet em uma variedade de domínios.

3.2.3 SegNet

A rede neural SegNet é uma rede convolucional profunda utilizada para a tarefa de segmentação de imagens. Foi desenvolvida por Badrinarayanan et al. (2017) e é conhecida por ser altamente eficiente e eficaz em sua aplicação.

Figura 7: Arquitetura da Rede neural SegNet

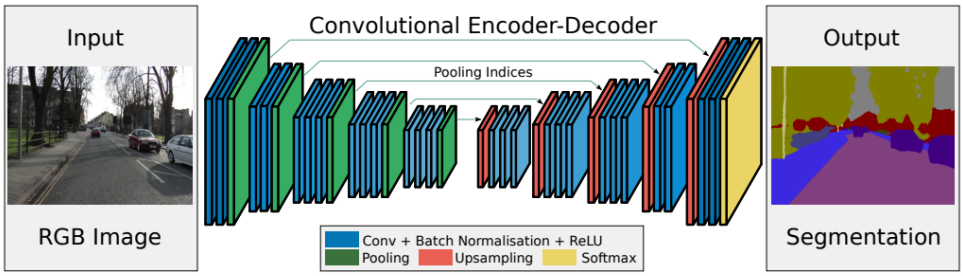


Imagem extraída de (BADRINARAYANAN et al., 2017, p. 4).

A arquitetura da SegNet (Figura 7) é baseada em um modelo de codificador-decodificador, no qual o codificador extrai características da imagem de entrada e o decodificador as usa para gerar a saída da segmentação de imagens. A rede utiliza a técnica de *max-pooling* invertido para preservar as informações espaciais das características durante o processo de decodificação.

O codificador da SegNet é composto por várias camadas de convolução e *pooling*, onde as convoluções identificam padrões e características nas imagens, enquanto as camadas de *pooling* reduzem progressivamente as dimensões espaciais das características, aumentando

sua abstração. No entanto, ao contrário de outras redes que simplesmente descartam as informações durante o processo de *pooling*, a SegNet utiliza índices de *max-pooling* armazenados durante o processo de codificador.

Já o decodificador, é uma parte crucial para a obtenção de segmentações precisas. Ele utiliza os índices armazenados para realizar operações de "*unpooling*" reverso, aumentando gradualmente as dimensões espaciais das características. Isso permite uma reconstrução detalhada da segmentação, mantendo a correspondência precisa com os pixeis de entrada.

A arquitetura SegNet também emprega camadas de convolução 1x1 para reduzir a dimensionalidade antes da reconstrução, o que ajuda a controlar o número de parâmetros e a eficiência computacional da rede.

Uma das principais vantagens da SegNet é sua eficiência em termos de memória e processamento, o que a torna adequada para aplicativos em dispositivos móveis e sistemas embarcados. A SegNet também pode ser adaptada para resoluções diferentes de imagem, tornando-a versátil para várias tarefas de segmentação.

4 ASRDNet - Detecção de Ferrugem Asiática em Imagens de Folhas Próximas

Neste capítulo, é apresentado um modelo de rede neural convolucional especializado em segmentação de imagens. Esse modelo é treinado de maneira supervisionada usando imagens (RGB) de folhas de soja, as quais são acompanhadas por máscaras de anotação elaboradas por um fitopatologista especializado.

4.1 Premissas para dados de treinamento

A rede neural ASRDNet foi desenvolvida com base em algumas premissas essenciais no que diz respeito ao seu conjunto de dados.

Primeiramente, as imagens utilizadas para o treinamento do modelo consistem em fotografias de folhas de soja, cada uma delas com três canais de cores (RGB). Essa escolha permitiu a extração das características relacionadas à coloração das lesões presentes nas folhas. Em seguida, todas as imagens foram redimensionadas para o tamanho uniforme de 224x224 pixeis. Essa padronização foi adotada com o propósito de garantir que todas as imagens de entrada apresentassem o mesmo tamanho durante o processo de treinamento da rede neural. Dessa forma, evitou-se sobrecarregar a rede com imagens excessivamente grandes e pesadas, o que poderia comprometer o desempenho e a eficiência do modelo.

Para a avaliação das lesões nas folhas de soja, foram empregadas máscaras que destacam as áreas de interesse nas imagens. Nesse contexto, as máscaras atribuem o valor 1 (um) às regiões que correspondem a lesões e o valor 0 (zero) às áreas que não são de interesse, como partes da folha saudável, o fundo da imagem, entre outras. Essa abordagem permitiu que o modelo identificasse de maneira precisa e sistemática as áreas afetadas pelas lesões.

Essas premissas fundamentais foram consideradas durante o desenvolvimento da ASRDNet,

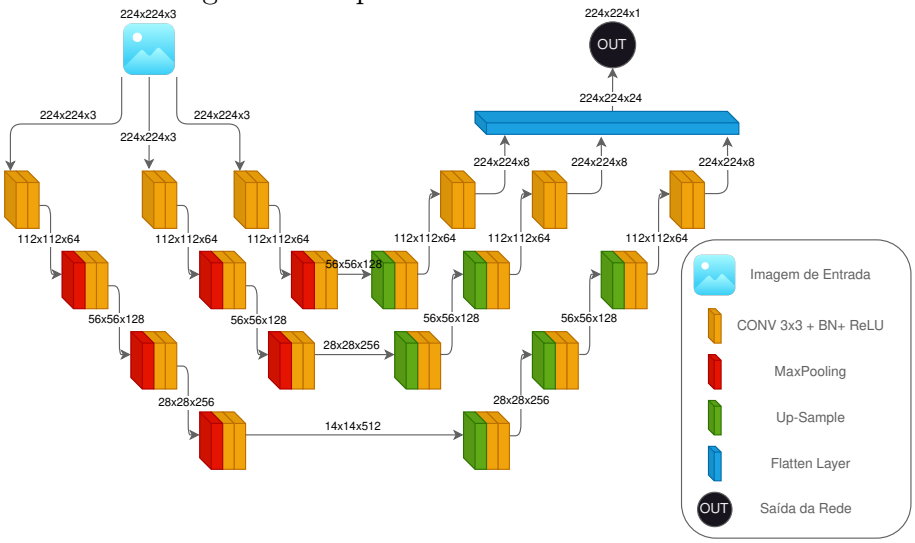
visando criar um modelo de rede neural eficaz e preciso na identificação e avaliação das lesões nas folhas de soja.

4.2 Arquitetura da Rede ASRDNet

A rede neural ASRDNet surgiu com a finalidade de segmentar imagens em cores RGB e detectar a presença da ferrugem asiática em folhas de soja. Durante o processo de desenvolvimento, tornou-se evidente a necessidade de criar uma rede capaz de extrair características das lesões de forma abrangente, independentemente de seus tamanhos variados e formatos diversos, identificados durante a investigação.

Para enfrentar esse desafio, concebeu-se uma rede com a capacidade de extrair características em resoluções diferentes a partir de uma única imagem de entrada. Isso possibilita que os pesos da rede sejam ajustados para identificar lesões de dimensões diversas.

Figura 8: Arquitetura da Rede ASRDNet



As informações do tamanho da imagem e o número de camadas estão informados no formato (largura x altura x nº de canais).

Fonte: De autoria própria.

A inspiração para o desenvolvimento dessa rede originou-se na ideia de uma abordagem de multiresolução semelhante à U-Net (Seção 3.2.1), embora sem a utilização das *skip-connections*. A solução adotada, conforme mostrado na Figura 8, envolve a criação de três "caudas" descendentes na rede (*encoder*), cada uma contendo uma quantidade variável de camadas e recebendo como entrada uma imagem RGB (3 canais). A quantidade de "caudas" escolhida foi guiada pela necessidade de extrair as lesões em várias resoluções, ao

mesmo tempo em que se evitava um aumento excessivo no tamanho do modelo, mantendo também um F1-score alto. A descida, segue a arquitetura típica de uma rede neural convolucional, consistindo na aplicação de duas convoluções 3x3 (com *padding* igual a 1), seguido da normalização em lote (*batch normalization*), da ReLU e ao final de uma operação de *max pooling* de tamanho 2. Em cada uma destas etapas, o número de canais de características é dobrado.

No processo de reconstrução da imagem segmentada, realizado na parte ascendente da rede (*decoder*), cada camada utiliza as características abstraídas em diferentes escalas para preservar lesões que, de outra forma, poderiam ser omitidas por outras partes da rede. Para isto, é realizado a operação de *up-convolution*, com tamanho 2, duplicando a resolução espacial de um mapa de características, seguidas de duas convoluções 3x3 (com *padding* igual a 1), da normalização em lote e da ReLU.

Ao final da subida de cada uma das "caudas" da rede, as saídas produzidas, a partir de convoluções 1x1, são enfim concatenadas, mantendo a mesma quantidade de canais, evitando assim a perda de informação, para posteriormente serem convertidas em um único canal de saída, com uma operação de convolução 1x1, em uma camada chamada de *flatten-layer*. Essa técnica permite que a rede neural combine informações de diferentes níveis de resolução espacial, fornecendo informações contextuais importantes para a tarefa de segmentação, dando a rede a capacidade de lidar com cenários binários (duas classes).

4.3 Discussão

A implementação do modelo de rede neural proposto nesta dissertação inicialmente foi projetada com um total de quatro "caudas", porém como a quantidade de parâmetros treináveis e o tamanho dos parâmetros do modelo não se diferenciavam e até ultrapassavam outras redes neurais, foram realizados vários experimentos, entre as duas redes neurais ASRDNet com 3 e 4 "caudas", para analisar o impacto da retirada desta cauda nos resultados.

Ao longo das experimentações, observou-se que a rede neural com quatro "caudas" alcançou uma pontuação média de 0.8803, conforme avaliado pela métrica F1-score. Por sua vez, a rede com apenas três "caudas" registrou uma pontuação média de 0.8785 usando a mesma métrica. Para verificar se esses resultados são estatisticamente equivalentes, empregou-se o *Welch's t-test*, com um nível de significância (α) de 0.05, utilizando 10 resultados de F1-score como amostra, não rejeitando a hipótese nula de que são esta-

tisticamente iguais. Com a modificação na arquitetura da rede, realizando uma "poda", conseguimos reduzir tanto a complexidade quanto o tamanho da rede. Por exemplo, o número de parâmetros treináveis foi reduzido de 37.3 milhões para 9.1 milhões, e o tamanho total estimado dos parâmetros do modelo diminuiu de 149.388 megabytes para 36.357 megabytes.

Essas mudanças não apenas simplificaram a rede, tornando-a mais eficiente, como também economizaram recursos computacionais significativos. Portanto, a otimização da arquitetura da rede desempenha um papel crucial na obtenção de resultados de alta qualidade, mantendo a viabilidade computacional do modelo.

5 Resultados

Neste texto, iremos examinar detalhadamente o desempenho da rede ASRDNet em comparação com as redes neurais U-Net, SegNet e DeepLabNet, com foco na métrica F1-score. Adicionalmente, investigaremos o efeito positivo do aumento de dados na melhoria do F1-score da rede ASRDNet. Além disso, destacaremos as vantagens computacionais proporcionadas pelo uso da rede ASRDNet.

5.1 Materiais e Métodos

Nesta seção, serão descritos os instrumentos, as configurações de experimentos, as métricas de avaliação e outros aspectos cruciais que permitem compreender como a pesquisa foi conduzida e como as conclusões foram alcançadas.

5.1.1 Dataset e Preparação dos Dados

Para este estudo, utilizou-se um conjunto de dados contendo 63 imagens (Figura 9(a)) RGB de folhas de soja infectadas com a doença da ferrugem asiática, gentilmente disponibilizadas pela Embrapa. É importante destacar que, ao contrário dos trabalhos anteriores discutidos na Seção 3.1, que utilizavam imagens de folha inteira ou de imagens aéreas, nossas imagens são capturadas em proximidade das folhas e em alta resolução. A fim de realizar a anotação dos dados, três fitopatologistas especializados em detecção da ferrugem asiática em folhas de soja foram consultados para identificar e delimitar as áreas das imagens que continham a doença (Figura 9(c)). A imagem anotada por cada técnico foi binarizada e, em seguida, processadas juntas para gerar uma segunda imagem (Figura 9(b)) que representa os pontos de interesse com duas ou mais marcações como "ponto de interesse" e as áreas sem doença com duas ou mais marcações como "não ponto de interesse". Dessa forma, o resultado final é uma imagem binária onde as áreas de interesse, ou seja, as partes da folha que contêm a doença da ferrugem asiática, são destacadas

em relação às áreas saudáveis.

As imagens geradas para este dataset (conforme detalhado no Apêndice A) compreendem um total de 63 imagens de folhas afetadas pela doença da ferrugem asiática. Adicionalmente, para cada imagem, foram produzidas 63 máscaras binárias que destacam as regiões de "ponto de interesse"e as áreas consideradas como "não ponto de interesse". Além disso, foram criadas 63 máscaras probabilísticas que incorporam as avaliações e votos dos fitopatologistas envolvidos no estudo.

Figura 9: Imagem original, máscara binária e a máscara média.

(a) Imagem utilizada como entrada na rede neural.

(b) Máscara das regiões de interesse com 2 ou mais votos.

(c) Máscara das regiões votadas por todos os analistas (Preto: 0 votos, Cinza Escuro: 1 voto, Cinza Claro: 2 votos e Branco: 3 votos).

Fonte: De autoria própria.

Esse processo de anotação é crucial para a criação de um modelo de aprendizado de máquina para detecção automática da doença em folhas de soja. A utilização de anotações precisas e confiáveis pode aumentar significativamente a eficácia e a eficiência dos modelos de aprendizado de máquina na tarefa de detecção da doença. Além disso, a obtenção de um conjunto de dados anotado é fundamental para a validação do modelo e para a comparação com outros modelos em estudos futuros.

A preparação do conjunto de dados para a entrada na rede neural passou por uma série de etapas cuidadosas para garantir a precisão dos resultados durante as fases de validação, treinamento e teste. Primeiramente, todas as imagens foram redimensionadas para 224x224 pixeis, a fim de adequar ao tamanho de entrada da rede neural. Em seguida, foi utilizada uma semente com valor 42 para inicializar o gerador de números aleatórios, permitindo que tanto os dados gerados durante a divisão do conjunto de dados em validação, treinamento e teste quanto os dados gerados durante a etapa de aumento de dados

possam ser reproduzidos com precisão posteriormente. Além disso, o conjunto de dados foi dividido para manter a seguinte proporção entre as imagens: 20% para validação, 20% para teste e 60% para treinamento.

Durante a fase de treinamento, as redes foram treinadas do zero e cada rede neural foi submetida ao conjunto de dados original e também ao conjunto de dados com aumento de dados, a fim de verificar uma possível melhoria no aprendizado das lesões de soja pela rede neural. Para isso, quando a opção de aumento de dados estava ativa, os dados foram expandidos em até 10 vezes o tamanho dos dados originais, ou seja, cada imagem gerava 10 novas imagens. Essas novas imagens foram geradas seguindo um conjunto de parâmetros específicos de transformação, com o objetivo de diversificar o conjunto de dados e aumentar a robustez da rede neural diante de possíveis variações nas imagens de entrada. Os parâmetros de transformação utilizados incluem as seguintes técnicas:

Flip: Essa transformação vira a imagem horizontalmente ou verticalmente com uma probabilidade de 0.5.

Affine: Essa transformação realiza uma transformação afim na imagem, que inclui uma escala, translação e rotação, aplicadas com uma probabilidade de 1.0. Os parâmetros de escala variam de 0.5 a 2.0, enquanto os de translação variam de -50% a 50%. A rotação pode variar entre -180 e 180 graus. A interpolação usada é cúbica.

GaussianBlur: Essa transformação aplica um desfoque gaussiano na imagem com uma probabilidade de 0.125.

MotionBlur: Essa transformação aplica um desfoque de movimento na imagem com uma probabilidade de 0.125.

OpticalDistortion: Essa transformação aplica distorções ópticas na imagem com uma probabilidade de 0.125. Os parâmetros de distorção podem variar até 2 e os de deslocamento podem variar até 50%. As duas distorções aplicadas são a *Barrel Distortion*, onde as linhas retas da imagem parecem curvar-se para fora a partir do centro, criando uma aparência de barril e a *Pincushion Distortion*, que as linhas retas parecem curvar-se para dentro em direção ao centro da imagem, resultando em uma aparência de almofada ou travesseiro.

RandomBrightnessContrast: Essa transformação aplica um aumento aleatório de brilho e contraste na imagem com uma probabilidade de 0.125.

5.1.2 Implementação

Nesta seção serão descritos o ambiente de desenvolvimento utilizado, as ferramentas computacionais e a descrição da implementação da rede neural ASRDNet.

5.1.2.1 Ambiente de Desenvolvimento

O ambiente utilizado para rodar todos os experimentos foi o Google Colab, com as seguintes configurações:

Ambiente de Desenvolvimento Google Colab (https://colab.research.google.com)

Processador 2 Processadores Intel(R) Xeon(R) CPU @ 2.20GHz

Memória RAM 12 GB

Placa de Vídeo Tesla T4

Memória da Placa de Vídeo 16 GB

Sistema Operacional Linux 5.10.147+

Linguagem de Programação Python 3.9.16

5.1.2.2 Ferramentas computacionais utilizadas

Nesta seção serão apresentadas algumas ferramentas que foram utilizadas para facilitar a implementação da rede neural, visualização dos resultados e comparação dos experimentos realizados.

5.1.2.2.1 W&B

Weights and Biases (W&B) é uma ferramenta de monitoramento de experimentos de aprendizado de máquina. Ela permite aos usuários acompanhar o desempenho de modelos de aprendizado de máquina em tempo real, realizar análises detalhadas e compartilhar resultados com outros membros da equipe. A plataforma também oferece recursos de colaboração e integração com outras ferramentas de aprendizado de máquina (WANDB, 2023).

A utilização de ferramentas como o W&B tem se tornado cada vez mais importante no desenvolvimento de modelos de aprendizado de máquina, já que a complexidade dos

modelos e dos conjuntos de dados utilizados para treiná-los torna difícil realizar análises detalhadas sem uma ferramenta adequada. Além disso, a colaboração em equipe é uma parte importante do processo de desenvolvimento de modelos de aprendizado de máquina, e o W&B ajuda a facilitar essa colaboração.

A seguir, são descritas as principais partes da arquitetura do W&B:

Biblioteca de registro: A biblioteca de registro é uma parte central do W&B que permite aos usuários registrar métricas, gráficos e artefatos (como modelos treinados ou conjuntos de dados) durante o treinamento de seus modelos. Os dados registrados são enviados para o servidor do W&B em tempo real e podem ser visualizados e comparados usando a interface da web do W&B.

Servidor de armazenamento: O servidor de armazenamento do W&B é responsável por armazenar os dados registrados pelos usuários. Ele também fornece recursos para gerenciar e organizar experimentos, como a capacidade de pesquisar experimentos anteriores e criar *dashboards* personalizados para visualizar os resultados.

Interface da web: A interface da web do W&B permite aos usuários visualizar e comparar os resultados de seus experimentos em uma única interface unificada. Ele fornece recursos para visualizar métricas, gráficos, modelos e outros artefatos registrados durante o treinamento, além de permitir que os usuários colaborem e compartilhem experimentos com outros membros da equipe.

APIs: O W&B fornece APIs para várias linguagens de programação, incluindo Python, R e JavaScript, que permitem aos usuários acessar os recursos do W&B em seus próprios códigos e scripts. As APIs permitem que os usuários registrem métricas e artefatos, pesquisem experimentos anteriores e criem *dashboards* personalizados em suas próprias ferramentas e fluxos de trabalho.

Varredura de hiperparâmetros: O W&B é uma ferramenta eficaz para a varredura de parâmetros, permitindo que os cientistas de dados conduzam experimentos em larga escala de maneira organizada. Com ele, os usuários podem definir vários hiperparâmetros, rastreando o desempenho do modelo para cada combinação. Além disso, oferece uma interface intuitiva para análise e comparação de resultados, facilitando a identificação das melhores configurações.

A arquitetura do W&B é composta por várias partes interconectadas que trabalham juntas para fornecer uma plataforma completa para gerenciamento e visualização de experimentos de aprendizado de máquina. A biblioteca de registro permite que os usuários

registrem dados durante o treinamento de seus modelos, enquanto o servidor de armazenamento e a interface da web fornecem recursos para gerenciar e visualizar os resultados. As APIs permitem que os usuários integrem o W&B em seus próprios fluxos de trabalho e ferramentas personalizadas.

5.1.2.2.2 Torch

PyTorch é uma biblioteca de tensores que suporta cálculo diferencial em GPU e CPU, o que a torna eficiente em termos computacionais para o treinamento de modelos de aprendizado profundo em grandes conjuntos de dados. É altamente flexível e permite que os usuários criem e modifiquem modelos de aprendizado profundo com facilidade (PYTORCH, 2023).

Esta biblioteca tem como principal objetivo oferecer uma experiência de programação de alto nível que seja simples e flexível para pesquisadores e desenvolvedores em aprendizado de máquina. A biblioteca é baseada em um modelo de execução dinâmico que permite aos usuários definir e modificar grafos computacionais no momento da execução. Isso oferece maior flexibilidade do que outros *frameworks* de aprendizado profundo, que usam um modelo de execução estática.

Além disto, é amplamente utilizado em pesquisas de aprendizado profundo em áreas como visão computacional, processamento de linguagem natural e robótica. A biblioteca tem uma grande comunidade de usuários, incluindo pesquisadores acadêmicos e empresas de tecnologia, e é frequentemente usada em competições de aprendizado de máquina.

O PyTorch oferece uma série de ferramentas e bibliotecas adicionais que facilitam o desenvolvimento de modelos de aprendizado profundo. Alguns exemplos incluem TorchScript, que permite que os usuários exportem modelos PyTorch para outras plataformas, e o Torch Vision, que fornece uma série de conjuntos de dados e modelos pré-treinados para visão computacional.

5.1.2.2.3 Torch Vision

A biblioteca Torch Vision é uma extensão da biblioteca PyTorch, focada em fornecer ferramentas e funcionalidades para o processamento de imagens e tarefas de visão computacional. Ela oferece uma ampla gama de recursos, como pré-processamento de dados, transformações de imagem, conjuntos de dados populares e modelos de redes neurais pré-treinados (PYTORCHVISION, 2023).

Uma das suas principais finalidades é facilitar o desenvolvimento de algoritmos de visão computacional, fornecendo componentes essenciais para tarefas comuns, como classificação de imagens, detecção de objetos, segmentação de imagens e geração de imagens.

Também tem como característica, a disponibilidade de conjuntos de dados populares utilizados em visão computacional, como o ImageNet (DENG et al., 2009), CIFAR (KRIZHEVSKY, 2009), COCO (LIN, T. et al., 2014) e muitos outros. Esses conjuntos de dados são fornecidos em um formato padronizado e podem ser facilmente carregados e utilizados para treinar e avaliar modelos de aprendizado de máquina.

Além dos conjuntos de dados, também oferece um conjunto abrangente de transformações de imagem, permitindo que os usuários realizem pré-processamento flexível e personalizado dos dados de entrada. Essas transformações incluem redimensionamento, recorte, rotação e muitas outras, contribuindo para a preparação adequada dos dados antes do treinamento dos modelos.

Outro aspecto importante é a disponibilidade de modelos de redes neurais pré-treinados. Esses modelos são treinados em grandes conjuntos de dados e pré-configurados para realizar tarefas específicas, como classificação de imagens ou detecção de objetos. Eles podem ser facilmente carregados e utilizados como ponto de partida para fins de transferência de aprendizado, acelerando o processo de desenvolvimento e melhorando o desempenho dos modelos em novos conjuntos de dados.

A biblioteca também oferece ferramentas para avaliação de modelos, métricas de desempenho, visualização de resultados e outros recursos úteis para a análise e interpretação dos resultados dos experimentos em visão computacional.

5.1.2.2.4 PyTorch Lightning

PyTorch Lightning é uma biblioteca Python que fornece uma estrutura de trabalho para acelerar o desenvolvimento de modelos de aprendizado de máquina baseados em PyTorch. A biblioteca é projetada para simplificar tarefas comuns de treinamento e acelerar a iteração do modelo, permitindo que os usuários se concentrem em projetar modelos e experimentos (PYTORCHLIGHTNING, 2023).

A biblioteca PyTorch Lightning foi criada com o objetivo de facilitar o desenvolvimento de modelos de aprendizado de máquina. Ela fornece uma interface simples para tarefas de treinamento, validação e teste de modelos, além de simplificar o processo de configuração e execução de experimentos. Isso permite que os usuários se concentrem em

projetar modelos de alta qualidade e iterar rapidamente em seus experimentos.

A estrutura do PyTorch Lightning é organizada em módulos que fornecem funcionalidades específicas para o desenvolvimento de modelos de aprendizado de máquina. A seguir, são descritos os principais módulos e suas funcionalidades:

pytorch_lightning.core: contém as classes LightningModule e DataModule, que são a base do desenvolvimento de modelos com PyTorch Lightning. A classe LightningModule encapsula o modelo de aprendizado de máquina e define as operações de treinamento, validação e teste, enquanto a classe DataModule encapsula a lógica de carregamento e pré-processamento dos dados utilizados no treinamento.

pytorch_lightning.trainer: contém a classe *Trainer*, que é responsável por executar o treinamento do modelo e gerenciar as configurações de treinamento, como o número de épocas, o tamanho do *batch* e o otimizador utilizado. Também fornece recursos para monitorar o progresso do treinamento, como o cálculo de métricas e o registro de logs.

pytorch_lightning.callbacks: contém várias classes que implementam funções de callback durante o treinamento, como a interrupção do treinamento se uma métrica atingir um determinado limite, a salvaguarda de *checkpoints* do modelo durante o treinamento e o registro de métricas em plataformas externas.

pytorch_lightning.utilities: contém uma série de utilitários para facilitar o desenvolvimento com PyTorch Lightning, como a classe *seed_everything*, que permite definir a semente para reprodução de resultados e a função *import_module_from_file*, que permite carregar módulos Python de arquivos externos.

A estrutura do PyTorch Lightning fornece uma abordagem modular e organizada para o desenvolvimento de modelos de aprendizado de máquina com PyTorch. Cada módulo fornece funcionalidades específicas para tarefas comuns de treinamento e gerenciamento de modelos, permitindo que os usuários se concentrem em projetar modelos de alta qualidade e iterar rapidamente em seus experimentos.

5.1.2.2.5 OpenCV

O Open Source Computer Vision Library (OpenCV) é uma biblioteca de visão computacional de código aberto, desenvolvida originalmente pela Intel e atualmente mantida pela comunidade de desenvolvedores. Ela é amplamente utilizada em diversas áreas, como robótica, automação, processamento de imagem, reconhecimento de padrões, entre outras(OPENCV, 2023).

O OpenCV oferece uma ampla gama de funções e algoritmos para processamento de imagens e visão computacional, como operações aritméticas, filtragem, detecção de bordas, reconhecimento de objetos, entre outros. Além disso, ele suporta várias linguagens de programação, como C++, Python, Java e MATLAB.

Uma das principais vantagens do OpenCV é sua facilidade de uso e implementação, permitindo que até mesmo usuários com pouco conhecimento em programação possam desenvolver aplicativos de visão computacional. Além disso, a biblioteca é altamente otimizada e escalável, oferecendo suporte a múltiplas plataformas e dispositivos.

Em geral, o OpenCV é uma ferramenta essencial para a área de visão computacional, oferecendo uma ampla variedade de recursos e funcionalidades para processamento de imagem e reconhecimento de padrões.

5.1.2.2.6 Albumentation

A biblioteca Albumentations (ALBUMENTATION, 2023) é uma ferramenta poderosa e versátil desenvolvida em Python, projetada para otimizar o pré-processamento de imagens em tarefas de aprendizado profundo, como visão computacional e processamento de imagens. Esta biblioteca destaca-se por sua capacidade de aplicar uma ampla gama de transformações de imagem de maneira rápida e eficiente.

O objetivo fundamental da biblioteca Albumentations é simplificar o pré-processamento de imagens, tornando-o mais eficaz e flexível. Ela oferece uma extensa lista de transformações de imagem, incluindo rotação, corte, ajuste de brilho e contraste, entre outras. Essas transformações podem ser facilmente combinadas para criar pipelines de pré-processamento complexos.

Para atingir tempos de processamento rápidos, a biblioteca utiliza técnicas de programação eficiente e integra-se diretamente com bibliotecas populares de visão computacional, como o OpenCV. Essa eficiência é essencial para o treinamento de modelos de aprendizado profundo em grandes conjuntos de dados.

Também é altamente compatível com várias estruturas de aprendizado profundo, incluindo TensorFlow, PyTorch, Keras e outras. Isso facilita a integração da biblioteca em fluxos de trabalho existentes de cientistas de dados e engenheiros de aprendizado de máquina.

Além das transformações predefinidas, a biblioteca permite a criação de transformações personalizadas, aumentando ainda mais sua adaptabilidade a diferentes cenários de

pré-processamento de imagem.

A aplicação de transformações desta biblioteca durante o pré-processamento pode melhorar a robustez e a capacidade de generalização dos modelos de aprendizado profundo. Isso torna os modelos mais resistentes a variações nos dados de entrada e, portanto, mais confiáveis em aplicações do mundo real.

5.1.2.3 Implementação da ASRD

A implementação da ASRDNet foi realizada em Python, deste modo, iremos detalhar o código fonte da arquitetura da rede.

Classe DoubleConv

A Classe DoubleConv (Apêndice B.1), define um bloco de convolução dupla, com duas camadas convolucionais consecutivas.

O construtor **"__init__"** recebe os seguintes parâmetros:

in_channels: o número de canais de entrada.

out_channels: o número de canais de saída.

mid_channels (opcional): o número de canais na camada intermediária (se não fornecido, será igual a **"out_channels"**).

groups (opcional): o número de grupos de convolução (usado para convolução agrupada).

O método **"forward"** aplica duas camadas convolucionais seguidas, com ativações **ReLU** e normalização **BatchNorm**, à entrada **"x"**.

Classes Down e Up

As Classes Down e Up (Apêndice B.2), definem blocos de decodificação (Down) e codificação (Up) que são usados em uma arquitetura U-Net, comumente usada em tarefas de segmentação de imagens.

A classe **"Down"** recebe **"in_channels"** e **"out_channels"** e realiza uma operação de *downsampling* (*max-pooling*) seguida por uma convolução dupla usando **"DoubleConv"**.

A classe **"Up"** recebe **"in_channels"** e **"out_channels"** e realiza uma operação de *upsampling* seguida por uma convolução dupla usando **"DoubleConv"**.

Classe Out

A Classe Out (Apêndice B.3), representa uma camada de saída (output) da rede neural.

O construtor **"__init__"** recebe os seguintes parâmetros:

in_channels: o número de canais de entrada.

out_channels: o número de canais de saída.

groups (opcional): o número de grupos de convolução (usado para convolução agrupada).

O método **"forward"** aplica uma camada convolucional com *kernel* de tamanho 1, à entrada **"x"**.

Classe ASRDNet

A Classe ASRDNet (Apêndice B.4), implementa o modelo de rede neural proposto.

As camadas de codificação (**"Down"**) são definidas para extrair características relevantes da imagem. Elas incluem camadas convolucionais duplas (**"DoubleConv"**) seguidas por camadas de *downsampling* (**"Down"**). Essas camadas permitem que a rede aprenda características em diferentes escalas.

As camadas de decodificação (**"Up"**) são definidas para realizar a reconstrução da imagem segmentada a partir das características extraídas pelas camadas de codificação. Elas incluem camadas de *upsampling* (**"Up"**) e convoluções duplas (**"DoubleConv"**).

As camadas de saída (**"Out"**) são definidas para gerar as previsões finais da rede. A camada de fusão final combina as saídas das camadas de decodificação.

O método **"forward"** implementa o fluxo de informações através da rede. Ele realiza a codificação, decodificação e fusão das características para gerar as previsões finais.

As previsões finais dependem do número de classes especificado durante a inicialização da rede. Se houver mais de duas classes, é realizada uma operação de *argmax* para obter as previsões. Caso contrário, é aplicada uma ativação *sigmoid* e arredondamento para prever uma máscara binária.

5.1.3 Metodologia de treinamento

Nesta seção serão apresentados os métodos para o treinamento da rede, bem como os passos que foram realizados para a obtenção dos resultados do trabalho.

5.1.3.1 Criação dos Batches

A necessidade de dividir um conjunto de dados em lotes (*batches*) durante o treinamento de modelos de aprendizado profundo é uma prática fundamental que desempenha um papel importante no sucesso do treinamento e na eficiência computacional do processo. Um dos motivos mais óbvios para dividir um conjunto de dados em lotes é a restrição de memória. Conjuntos de dados de treinamento podem ser massivos, não cabendo na memória da GPU. Ao dividir os dados em lotes menores, podemos processar uma parte de cada vez, minimizando os requisitos de memória. A divisão em lotes permite a paralelização dos cálculos durante o treinamento. Os modernos aceleradores de hardware, como GPUs e TPUs, são altamente paralelos e podem realizar operações em vários exemplos de dados simultaneamente. O treinamento em lotes cria uma forma de regularização estocástica, introduzindo ruído nas atualizações dos gradientes a cada lote. Isso ajuda a evitar o superajuste (*overfitting*) e a melhorar a capacidade de generalização do modelo.

Para a escolha do tamanho do lote utilizado no treinamento dos modelos, foi utilizado uma distribuição uniforme, fazendo com que o tamanho do lote seja escolhido de forma aleatória, mas uniformemente, com valores entre 2 e 20. Isso adiciona um elemento de aleatoriedade ao processo, que pode ser benéfico para evitar mínimos locais no espaço de otimização.

A biblioteca W&B foi utilizada otimizar os hiperparâmetros, incluindo o tamanho do lote, durante o treinamento de modelos de aprendizado profundo. O uso de distribuições como a *"int_ uniform"* para o tamanho do lote permite uma busca sistemática de hiperparâmetros com o W&B, registrando métricas de desempenho para diferentes tamanhos de lote e, assim, ajudando na seleção do melhor tamanho do lote para uma tarefa específica.

5.1.3.2 Ajuste do modelo pela descida do gradiente

O treinamento de modelos de aprendizado de máquina é um processo fundamental que visa ajustar os parâmetros do modelo para que ele possa fazer previsões precisas em dados de entrada não vistos. Três conceitos centrais desse processo são a função de perda (*loss*), a retropropagação (*backward*) e a descida de gradiente (*gradient descent*). Esta

seção discutirá detalhadamente a função de perda, a retropropagação e como a descida de gradiente é utilizada para otimizar o ajuste do modelo durante o treinamento.

5.1.3.2.1 Função de Perda

A função de perda (*loss*) é um componente importante no treinamento de modelos de aprendizado de máquina. Ela mede quão bem as previsões do modelo se ajustam aos rótulos ou valores alvo dos dados de treinamento. O objetivo é minimizar essa função para que o modelo faça previsões mais precisas. Uma função de perda é, em essência, uma função matemática que recebe como entrada as previsões do modelo e os valores reais (rótulos) dos exemplos de treinamento. Esta função avalia o desempenho do modelo atribuindo um valor numérico que quantifica o quão próximas ou distantes são as previsões do modelo em relação aos valores reais. Um valor baixo da função de perda indica que o modelo está fazendo previsões precisas, enquanto um valor alto indica previsões imprecisas. O objetivo principal da função de perda é criar um critério para treinar o modelo de forma que a função de perda seja minimizada. Durante o treinamento, os parâmetros do modelo são ajustados iterativamente para reduzir o valor da função de perda. Existem várias funções de perda disponíveis, e a escolha da função de perda depende da natureza da tarefa. Algumas funções de perda comuns incluem:

Erro Quadrático Médio (Mean Squared Error - MSE): Usado em tarefas de regressão para minimizar a diferença quadrática média entre as previsões e os valores reais.

Entropia Cruzada (Cross-Entropy): Usada em tarefas de classificação para medir a divergência entre as distribuições de probabilidade das previsões e dos rótulos reais.

Erro Absoluto Médio (Mean Absolute Error - MAE): Outra métrica usada em tarefas de regressão que mede a diferença absoluta média entre as previsões e os valores reais.

Neste trabalho, foi utilizada a função de perda chamada ***Binary Cross-Entropy Loss with Logits***, que é particularmente adequada para problemas onde é necessário determinar a probabilidade de um exemplo pertencer a uma das duas classes. A função faz uso de *logits*, que são as saídas brutas do modelo antes da aplicação de qualquer função de ativação, como *sigmoid* ou *softmax*. Os *logits* podem assumir qualquer valor real e representam a medida de confiança do modelo nas previsões. Para um classificador binário, geralmente há um único valor de *logits* para cada exemplo de entrada, indicando a probabilidade estimada de pertencer à classe positiva. Antes de calcular

a perda, a função *sigmoid* é aplicada aos *logits*. A função *sigmoid* mapeia os valores reais para o intervalo de [0,1], interpretados como probabilidades. A função sigmoid é definida por $\sigma(x) = \frac{1}{1+e^{-x}}$, onde x são os *logits*. Já a *Binary Cross-Entropy Loss with Logits*(*BCEWithLogitsLoss*), é calculada como uma forma de entropia cruzada entre as probabilidades previstas pela função sigmoid e os rótulos reais do *dataset*. A fórmula da perda é definida por $\text{BCEWLLoss}(y, x) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(\sigma(x_i)) + (1 - y_i) \cdot \log(1 - \sigma(x_i))]$, onde N é o número de exemplos no dataset de treinamento, y_i é o rótulo real para o exemplo i, x_i é o valor do *logits* para o exemplo i e $\sigma(x_i)$ é a probabilidade prevista pela função *sigmoid* para o exemplo i. Esta função de perda mede a divergência entre as probabilidades previstas e os rótulos reais. Quando a previsão está correta (ou seja, quando $y_i = 1$ e $\sigma(x_i)$ está próxima de 0), a perda é próxima de 0. Quando a previsão está errada, a perda aumenta. A perda é calculada para todos os exemplos no *dataset* e, em seguida, é realizada a média.

5.1.3.2.2 Retropropagação e Descida do Gradiente

A retropropagação (*backpropagation*) é um algoritmo utilizado na otimização de modelos de aprendizado de máquina, especialmente em redes neurais profundas (SOUZA SILVA, 2022). Ele é usado para calcular gradientes, que representam a taxa de variação da função de perda em relação aos parâmetros do modelo, envolvendo a aplicação da regra da cadeia para propagar os gradientes dos neurônios da camada de saída de volta para as camadas de entrada, permitindo assim o ajuste dos pesos e vieses em cada camada da rede.

A descida de gradiente (*Gradient Descent*) é um método de otimização utilizado para ajustar os parâmetros de modelos de aprendizado de máquina durante o processo de treinamento. Seu principal propósito é encontrar os valores ideais dos parâmetros do modelo que minimizam uma função de perda pré-definida. A Descida de Gradiente opera de maneira iterativa, seguindo uma sequência de etapas bem definidas:

Inicialização: Nesta etapa, os parâmetros do modelo são definidos com valores iniciais. Esses valores podem ser escolhidos aleatoriamente ou com base em conhecimento prévio.

Forward Pass e Cálculo da Loss: Durante o treinamento, os dados de treinamento são fornecidos ao modelo, que realiza uma "passagem direta" (*forward pass*) para gerar previsões. A função de perda é então calculada, avaliando a discrepância entre as previsões do modelo e os valores reais dos dados de treinamento.

Backward Pass e Cálculo de Gradiente: Nesta fase, os gradientes da função de perda em relação a cada parâmetro do modelo são calculados. Isso é realizado usando o processo de retropropagação (*backpropagation*), que envolve a aplicação da regra da cadeia para propagar os gradientes da camada de saída até a camada de entrada.

Atualização dos Parâmetros: Com os gradientes em mãos, os parâmetros do modelo são atualizados em direção ao mínimo da função de perda. Isso é alcançado multiplicando os gradientes pelo que é chamado de taxa de aprendizado (*learning rate*). A taxa de aprendizado controla o tamanho dos passos que o algoritmo dá na direção dos valores ideais dos parâmetros. É importante ajustar essa taxa adequadamente, pois uma taxa muito alta pode causar oscilações e divergência, enquanto uma taxa muito baixa pode tornar o treinamento excessivamente lento.

Iteração: Exceto pela **Inicialização**, os passos anteriores são repetidos iterativamente. Em cada iteração, um novo conjunto de dados de treinamento é apresentado ao modelo, os gradientes são calculados, os parâmetros são atualizados e a função de perda é avaliada novamente. Esse processo é repetido até que um critério de parada seja atingido, como a convergência da função de perda ou um número máximo de iterações.

5.1.3.2.3 Otimizador

O sucesso do treinamento de modelos de aprendizado de máquina está relacionado à capacidade de otimizar os parâmetros do modelo para minimizar uma função de perda. Os otimizadores desempenham um papel fundamental neste processo, permitindo que os modelos aprendam representações significativas dos dados e façam previsões precisas. Essa otimização ocorre de maneira iterativa ao longo do treinamento do modelo. Durante esse processo, os otimizadores calculam os gradientes da função de perda em relação aos parâmetros do modelo e, em seguida, ajustam esses parâmetros com o objetivo de minimizar o valor da função de perda.

Uma característica significativa dos otimizadores é a capacidade de controlar a taxa de aprendizado (*learning rate*), que determina o tamanho dos passos dados durante a atualização dos parâmetros. A escolha adequada da taxa de aprendizado é de vital importância, uma vez que influencia diretamente o desempenho do treinamento. Uma taxa de aprendizado bem ajustada permite que o processo de otimização seja mais estável e atinja a convergência de maneira eficiente.

Neste trabalho, foram utilizados três otimizadores como hiper-parâmetros durante os treinamentos das redes neurais, Stochastic Gradient Descent (SGD) (TENG et al.,

2022), Adaptive Moment Estimation (Adam) (KINGMA et al., 2017) e Root Mean Square Propagation (RMSprop) (TIELEMAN et al., 2012).

O otimizador SGD atualiza os parâmetros movendo-se na direção oposta ao gradiente instantâneo da função de perda. Ele é eficaz quando o conjunto de dados é grande e é comumente usado em aprendizado profundo. No entanto, a taxa de aprendizado fixa pode ser um desafio em alguns casos. No caso do otimizador Adam, ele mantém estimativas adaptativas dos momentos de primeira e segunda ordem dos gradientes para ajustar a taxa de aprendizado. Isso o torna adequado para uma ampla gama de problemas e é frequentemente o otimizador padrão recomendado. Por fim, o RMSprop adapta a taxa de aprendizado individualmente para cada parâmetro, o que o torna útil em problemas não estacionários. Ele normalmente requer menos ajustes manuais da taxa de aprendizado em comparação com o SGD.

5.1.3.3 Varredura de hiperparâmetros

A otimização de hiperparâmetros é uma das etapas no desenvolvimento de modelos de aprendizado de máquina, uma vez que envolve a busca pelas configurações ideais dos hiperparâmetros do modelo, como taxa de aprendizado, número de camadas em uma rede neural, tamanho do lote, entre outros. A varredura de hiperparâmetros é uma abordagem sistemática para encontrar as melhores configurações dentro de um espaço de busca definido.

Neste trabalho, foi utilizado o W&B para gerenciar os intervalos de hiperparâmetros a serem testados pela rede neural. O método de otimização escolhido é o método bayesiano. Esse método é uma abordagem probabilística que modela a relação entre hiperparâmetros e métricas de desempenho. Ele busca encontrar os melhores hiperparâmetros explorando essa relação de forma eficiente. Também foi inserida a configuração de *early stopping*, que inclui um critério de término precoce, que é do tipo *hyperband* e requer um número mínimo de iterações para avaliar o desempenho de cada conjunto de hiperparâmetros, neste caso, foi escolhido o mínimo de 20 iterações (*epochs*). Esse critério ajuda a economizar tempo de computação, encerrando as avaliações precocemente se um conjunto de hiperparâmetros não estiver mostrando bom desempenho. A taxa de aprendizado é definida com uma distribuição uniforme no intervalo entre 1e-7 e 1e-1, o que significa que a varredura explorará diferentes valores dentro desse intervalo.

5.1.4 Métricas de comparação

A avaliação de modelos de segmentação de imagens é importante na determinação da qualidade e do desempenho desses modelos. Uma métrica robusta é necessária para medir a capacidade do modelo de identificar regiões de interesse em imagens, o que é essencial em aplicações como visão computacional, reconhecimento de objetos e análise de imagens médicas.

A métrica F1 (*F1-score*), apresentada por (HAND, 2012), é utilizada para medir o desempenho e combina duas medidas fundamentais, a precisão e o *recall*. A precisão (*precision* mede a proporção de verdadeiros positivos (TP) em relação ao total de classificações positivas feitas pelo modelo. Em outras palavras, é a capacidade do modelo de classificar corretamente os objetos de interesse, enquanto que o *recall* mede a proporção de verdadeiros positivos (TP) em relação ao total de objetos de interesse presentes na imagem. Ele reflete a capacidade do modelo de identificar corretamente os objetos relevantes. A métrica F1 é calculada pela média harmônica dessas duas medidas, $F1 = \frac{2 \cdot (\text{Precisão} \cdot \text{Recall})}{\text{Precisão} + \text{Recall}}$.

Esta métrica demanda tanto a identificação precisa dos objetos de interesse (alta precisão) quanto a habilidade de detectar todos os objetos relevantes na imagem (alto *recall*). Além disso, em tarefas de segmentação, a presença de falsos positivos (segmentações incorretas) e falsos negativos (objetos não detectados) pode ser prejudicial. A métrica F1 encoraja a minimização de ambos, assegurando que o modelo não apenas segmente corretamente, mas também evite deixar de detectar objetos de interesse.

5.2 Experimentos

Neste trabalho, foi criado o modelo de rede neural de segmentação chamado de ASRDNet (Seção 4.2), baseado na arquitetura da rede U-Net (Seção 3.2.1).

Nos experimentos, foram treinados um total de quatro modelos de redes neurais (U-Net (Seção 3.2.1), SegNet (Seção 3.2.3), DeepLabNet (Seção 3.2.2) e ASRDNet (Seção 4.2)), onde cada rede teve o treinamento realizado com o dataset orginal e o dataset aumentado (Seção 5.1.1) em 10 vezes do seu tamanho.

Nas análises conduzidas, foram avaliados dois aspectos importantes: a quantidade de parâmetros treináveis em cada rede neural e o tamanho desses parâmetros do modelo. A quantidade de parâmetros treináveis é um medida da complexidade e adaptação da rede neural, onde uma rede com grandes números de parâmetros treináveis pode ter

uma capacidade maior de aprender com conjuntos de dados, mas também é mais suscetível ao *overfitting*, o que pode afetar negativamente o desempenho em dados não vistos. Já o tamanho dos parâmetros do modelo, pode indicar em qual cenário a rede neural pode ser implementada, pois dependendo do tamanho do modelo, cenários com recursos computacionais limitados, como dispositivos móveis ou sistemas embarcados, podem ser impraticáveis, devido às restrições de memória e processamento. Deste modo, comparamos estas métricas com as redes neurais analisadas e obtivemos o resultado que pode ser visto na Tabela 1. Foi possível verificar que a rede ASRDNet obteve os menores resultados nas duas métricas, indicando ser a mais leve dentre as quatro redes neurais analisadas.

Rede Neural	**Parâmetros Treináveis**	**Tamanho dos Parâmetros**
ASRDNet	**9.1 M**	**36.357 MB**
U-Net	17.3 M	69.070 MB
SegNet	18.8 M	75.276 MB
DeepLabNet	39.6 M	158.535 MB

Tabela 1: Informação da quantidade de parâmetros treináveis e tamanho dos parâmetros do modelo das redes ASRDNet, U-Net, SegNet e DeepLabNet

Para cada uma das redes neurais consideradas neste estudo, realizamos um conjunto mínimo de 10 execuções independentes. Durante essas execuções, empregamos o framework de otimização W&B para sintonizar automaticamente os hiperparâmetros do modelo em cada iteração. O objetivo principal era permitir que o W&B explorasse diferentes configurações de hiperparâmetros a fim de otimizar o desempenho da rede em termos de F1-score, uma métrica crítica para a avaliação do desempenho em tarefas de classificação.

Após a conclusão dessas múltiplas execuções, realizamos uma seleção criteriosa dos 10 resultados mais destacados em termos de F1-score obtidos para cada rede neural. Essa seleção foi realizada com o intuito de compreender de forma abrangente como cada rede se comportou sob diferentes configurações de hiperparâmetros e condições de treinamento.

É fundamental ressaltar que, com o propósito de conduzir uma análise precisa e justa

da performance das redes selecionadas, garantimos que os subconjuntos de treinamento, validação e teste permaneceram idênticos em todas as execuções dos experimentos. Isso assegurou a consistência dos dados utilizados para avaliar o desempenho das redes, permitindo uma comparação direta e confiável entre os cenários de treinamento com e sem aumento de dados.

Para o *dataset* sem aumento de dados, obtivemos os resultados que podem ser observados na Tabela 2, onde a coluna "Melhor F1-score" corresponde ao maior resultado de F1-score das 10 amostras selecionadas, a coluna "Média F1-score" corresponde ao valor médio das 10 amostras selecionadas, a coluna "STD F1-score" corresponde ao desvio padrão das 10 amostras selecionadas e por fim, a coluna "P-value" é o resultado da métrica do *Welch's t-test*, entre as redes analisadas e a rede ASRDNet.

Rede Neural	**Melhor F1-score**	**Média F1-score**	**STD F1-score**	**P-value**
ASRDNet	**0.9077**	0.8925	0.0056	
U-Net	0.8993	0.8953	0.0030	0.1895
SegNet	0.8022	0.4987	0.2624	0.0010
DeepLabNet	0.8960	0.8864	0.0039	0.0132

Tabela 2: Apresentação dos melhores resultados, médias e desvios padrão obtidos a partir do treinamento das redes ASRDNet, U-Net, SegNet e DeepLabNet, sem aumento de dados

No caso do dataset com aumento de dados, os resultados podem ser observados na Tabela 3.

Rede Neural	**Melhor F1-score**	**Média F1-score**	**STD F1-score**	**P-value**
ASRDNet	**0.9251**	0.9125	0.0082	
U-Net	0.9192	0.9103	0.0045	0.4665
SegNet	0.8747	0.8579	0.0089	0.0001
DeepLabNet	0.9135	0.9109	0.0013	0.5499

Tabela 3: Apresentação dos melhores resultados, médias e desvios padrão obtidos a partir do treinamento das redes ASRDNet, U-Net, SegNet e DeepLabNet, com aumento de dados de 10 vezes

Em ambas Tabelas 2 e 3, os resultados foram comparados entre a rede neural ASRDNet e as demais redes (U-Net, SegNet e DeepLabNet), uma a uma. Para isto, foram utilizados

os 10 melhores resultados de F1-score de cada rede, para verificar se a amostragem dos resultados dos F1-score obtidos são estatisticamente diferentes.

Para verificar se os resultados entre as redes são estatisticamente equivalentes, empregou-se o *Welch's t-test*, com um nível de significância (α) de 0.05, utilizando os 10 melhores resultados de F1-score como amostra, chegando a conclusão de que, nos experimentos com o *dataset* sem aumento de dados, pode se considerar que o resultado entre a rede ASRDNet e as redes SegNet e DeepLabNet, são diferentes, rejeitando a hipótese nula de que são estatisticamente iguais. Para a rede U-Net, a hipótese nula não foi rejeitada, não tendo evidências sólidas para afirmar que elas são diferentes com base nos dados disponíveis e no nível de significância escolhido para o teste. Já para o *dataset* com aumento de dados, a hipótese nula das amostras serem estatisticamente iguais foi rejeitada somente para a rede SegNet. Para as redes U-Net e DeepLabNet, a hipótese não foi rejeitada. Estas informações podem ser observadas na Tabela 4, onde a coluna "Aumento de Dados" define se o dataset utilizado, possui ou não aumento de dados, a coluna "P-value" é o resultado da métrica do *Welch's t-test*, entre as redes analisadas e a rede ASRDNet e a coluna "Hipótese Nula" que indica se a hipótese nula foi ou não rejeitada.

Rede Neural	**Aumento de dados**	**P-value**	**Hipótese Nula**
U-Net	Não	0.1895	Não Rejeitada
SegNet		0.0010	Rejeitada
DeepLabNet		0.0132	Rejeitada
U-Net	Sim	0.4665	Não Rejeitada
SegNet		0.0001	Rejeitada
DeepLabNet		0.5499	Não Rejeitada

Tabela 4: Apresentação da análise do Welch's t-test, entre a rede ASRDNet e as redes U-Net, SegNet e DeepLabNet, para a hipótese nula de que as amostras são estatisticamente iguais.

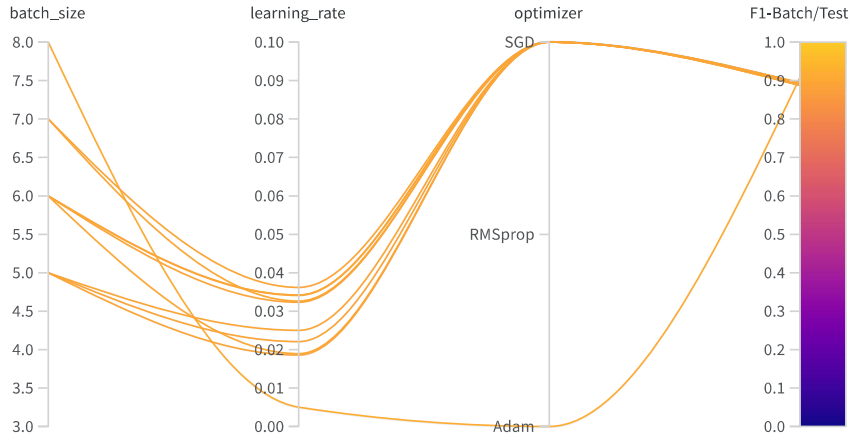
Os treinamentos das redes neurais foram realizados com uma variedade de hiperparâmetros. Esses hiperparâmetros foram otimizados e monitorados usando a ferramenta W&B, garantindo assim a eficácia e a precisão dos resultados.

Foram conduzidos no mínimo 30 treinamentos independentes, cada um com diferentes configurações de hiperparâmetros, para cada uma das redes avaliadas. Posteriormente, a partir desses treinamentos, foram selecionadas as 10 melhores execuções de cada rede

com base nos seus respectivos F1-scores. Essas execuções representam amostras altamente representativas e significativas, sendo empregadas para uma análise comparativa entre os resultados obtidos.

Para a rede neural ASRDNet, os hiperparâmetros utilizados durante os treinamentos foram cuidadosamente escolhidos. No caso do *dataset* sem aumento de dados, esses hiperparâmetros estão representados no gráfico da Figura 10.

Figura 10: Resultado do F1-score das 10 melhores execuções com a rede ASRDNet, sem aumento de dados.



Fonte: De autoria própria.

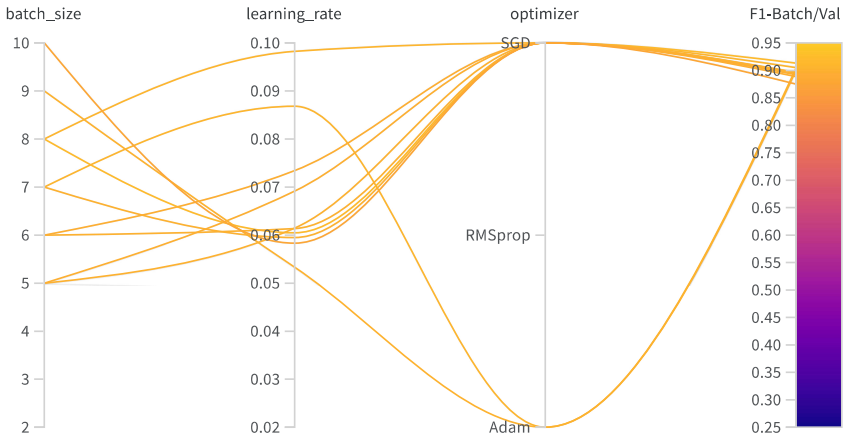
Para criar os dados da Figura 10, foram utilizados os dados contidos na Tabela 5, que foram extraídos durante as várias execuções da rede neural ASRDNet, com a alteração dos hiperparâmetros, a fim de maximizar o valor do F1-score.

Tamanho do Lote	Taxa de Aprendizado	Otimizador	F1-score ↓
8	0.0050	Adam	0.9077
6	0.0341	SGD	0.8942
6	0.0189	SGD	0.8937
7	0.0326	SGD	0.8932
7	0.0361	SGD	0.8924
5	0.0250	SGD	0.8906
6	0.0342	SGD	0.8903
6	0.0323	SGD	0.8878
5	0.0221	SGD	0.8878
5	0.0185	SGD	0.8869

Tabela 5: Valores dos hiperparâmetros da rede ASRDNet para os 10 resultados com melhor F1-score, para o dataset sem aumento de dados

Por outro lado, quando utilizamos o *dataset* com aumento de dados, os hiperparâmetros correspondentes estão ilustrados no gráfico da Figura 11.

Figura 11: Resultado do F1-score das 10 melhores execuções com a rede ASRDNet, com aumento de dados.



Fonte: De autoria própria.

Neste caso, para criar os dados da Figura 11, foram utilizados os dados contidos na Tabela 6, utilizando a mesma metodologia anterior, porém agora pra os dados do *dataset* com aumento de dados.

Tamanho do Lote	Taxa de Aprendizado	Otimizador	F1-score ↓
8	0.0604	SGD	0.9251
7	0.0868	Adam	0.9167
8	0.0982	SGD	0.9160
6	0.0734	SGD	0.9160
7	0.0594	SGD	0.9157
5	0.0614	SGD	0.9138
5	0.0691	SGD	0.9135
6	0.0613	SGD	0.9114
10	0.0583	SGD	0.9041
9	0.0533	Adam	0.8928

Tabela 6: Valores dos hiperparâmetros da rede ASRDNet para os 10 resultados com melhor F1-score, para o dataset com aumento de dados

Com o objetivo de aprofundar nossa análise dos resultados obtidos com a rede ASRDNet, optamos por realizar uma seleção, concentrando-nos na execução que apresentou o mais elevado valor de F1-score. Isso nos permite extrair informações valiosas sobre a configuração de hiperparâmetros que levou ao melhor desempenho desta rede.

Através desse processo de seleção, obtivemos um conjunto específico de hiperparâmetros que se destacaram em termos de desempenho, fornecendo *insights* valiosos sobre as configurações mais eficazes para a ASRDNet.

Os valores dos hiperparâmetros identificados para essa execução estão apresentados na Tabela 7, proporcionando uma visão abrangente das escolhas específicas que levaram ao desempenho superior desta rede.

Aumento de dados	Tamanho do Lote	Taxa de Aprendizado	Otimizador	F1-score
Não	8	0.0050	Adam	0.9077
Sim	8	0.0604	SGD	0.9251

Tabela 7: Apresentação dos hiperparâmetros da rede neural ASRDNet, com o melhor F1-score

Portanto, ao concentrar nossa atenção na configuração de hiperparâmetros de maior

destaque, somos capazes de extrair informações cruciais para analisar melhor o desempenho da ASRDNet.

5.3 Discussão

Como evidenciado na Seção 5.2 desta pesquisa, a rede neural ASRDNet se destacou em termos de desempenho, avaliado por meio da métrica F1-score, quando comparada às redes neurais U-Net, SegNet e DeepLabNet. Esse destaque se manteve consistente, independentemente do *dataset* utilizado, ou seja, tanto no cenário de treinamento com aumento de dados quanto no cenário sem aumento de dados. Os valores de F1-score obtidos pela ASRDNet foram surpreendentes, alcançando aproximadamente 0.9077 para o *dataset* sem aumento de dados e 0.9251 para o *dataset* com aumento de dados.

Ao observarmos o conjunto das dez melhores execuções em cada cenário, notamos que os resultados obtidos pela ASRDNet foram não apenas notáveis, mas também estatisticamente iguais ou superiores em relação às outras redes analisadas. Para evidenciar essas informações, a Tabela 4 apresentada destaca que, no cenário de *dataset* sem aumento de dados, somente o F1-score da rede U-Net foi considerado estatisticamente igual ao da ASRDNet, enquanto as pontuações das redes SegNet e DeepLabNet foram significativamente inferiores, evidenciando a superioridade da ASRDNet. No caso do *dataset* com aumento de dados, os F1-score das redes U-Net e DeepLabNet foram considerados estatisticamente iguais ao da ASRDNet, sendo a SegNet a única rede com um F1-score inferior.

Além do desempenho demonstrado pela ASRDNet, é importante ressaltar que essa rede também apresentou vantagens em termos de eficiência. A quantidade de parâmetros treináveis e o tamanho total do modelo da ASRDNet foram notavelmente inferiores em comparação com as demais redes, conforme destacado na Tabela 1. Mesmo quando comparada com redes que obtiveram um F1-score estatisticamente igual, a ASRDNet se destaca por ser uma opção mais leve e eficiente em termos de recursos computacionais, alcançando uma economia de recursos de aproximadamente 47% em relação à rede U-Net, 51% para a rede SegNet e de 76% para a rede DeepLabNet.

Os resultados obtidos na avaliação do desempenho da rede ASRDNet revelam uma descoberta importante relacionada à utilização de técnicas de aumento de dados no treinamento de modelos de aprendizado profundo. O aumento de dados, uma estratégia que envolve a geração de variações dos exemplos de treinamento existentes, mostrou-se

impactante no contexto deste estudo.

Ao comparar o desempenho da ASRDNet nos conjuntos de dados com e sem aumento, identificamos uma melhoria considerável no F1-score. Mais especificamente, observamos que o maior F1-score alcançado no conjunto de dados com aumento de dados foi de 0.9251, enquanto no conjunto de dados sem aumento, o maior F1-score correspondente foi de 0.9077.

Este aumento no F1-score representa uma elevação de aproximadamente 1.92% em relação à configuração sem aumento de dados. Esse ganho é uma evidência concreta de que a inclusão de exemplos de treinamento adicionais, gerados por meio de técnicas de aumento de dados, contribuiu significativamente para a capacidade da rede ASRDNet em segmentar com precisão objetos de interesse em imagens.

Esses resultados destacam a importância e o impacto positivo do aumento de dados como uma prática eficaz no treinamento de modelos de aprendizado profundo, proporcionando melhorias mensuráveis no desempenho da rede. Além disso, demonstram a relevância do uso dessa estratégia em tarefas de segmentação de imagens, onde a precisão na identificação de objetos é fundamental para a qualidade dos resultados.

6 Conclusões e Trabalhos Futuros

Este trabalho revela uma série de descobertas significativas no contexto da pesquisa sobre a rede ASRDNet e sua aplicação em tarefas de segmentação de imagens. Ao longo deste estudo, realizamos uma análise abrangente do desempenho da ASRDNet em comparação com outras redes neurais amplamente utilizadas, como U-Net, SegNet e DeepLabNet. A métrica F1-score foi utilizada como principal indicador de desempenho, e os resultados foram reveladores.

Primeiramente, identificamos que a ASRDNet demonstrou consistentemente um desempenho superior em relação às redes concorrentes, independentemente do conjunto de dados utilizado. Especificamente, a ASRDNet alcançou valores mais altos de F1-score, destacando sua capacidade de segmentar com precisão objetos de interesse em imagens, o que é fundamental em tarefas de segmentação de imagens.

Outra descoberta relevante foi a influência positiva do aumento de dados no desempenho da ASRDNet. Ao comparar o F1-score entre o conjunto de dados sem aumento e o conjunto de dados com aumento de dados, observamos um aumento de aproximadamente 1.92% na métrica quando o aumento de dados foi aplicado. Isso destaca a importância do uso de estratégias de aumento de dados para melhorar a capacidade de generalização e desempenho da rede em cenários práticos.

Além disso, observamos que a ASRDNet oferece uma vantagem significativa em termos de economia computacional. Com uma quantidade menor de parâmetros treináveis e um tamanho de modelo mais compacto em comparação com as redes concorrentes, a ASRDNet se destaca como uma escolha eficiente para aplicações que demandam recursos computacionais limitados.

Uma das limitações do trabalho é que os resultados do estudo foram comparados apenas com algumas arquiteturas de redes neurais selecionadas. Uma extensão valiosa seria incluir uma análise comparativa mais abrangente com outras redes e abordagens de detecção de doenças em folhas de plantas. Isso permitiria avaliar ainda melhor o

desempenho relativo da rede ASRDNet em relação a uma variedade de modelos e métodos disponíveis.

Durante a fase de implementação da rede ASRDNet, surgiu a hipótese de parametrizar o número de "caudas" da rede, permitindo assim o treinamento com dados em múltiplas resoluções. Esse ajuste possibilitaria uma análise mais detalhada para determinar qual quantidade de "caudas" se adapta de forma ótima ao problema em questão.

Como perspectiva de trabalhos futuros, planejamos incorporar essa modificação no modelo da rede, aprofundando nossa investigação e explorando ainda mais suas capacidades. Uma das direções promissoras que pretendemos explorar é a integração do mecanismo de atenção na rede neural convolucional desenvolvida. O mecanismo de atenção permitirá que a rede se concentre em características específicas das imagens de folhas de soja afetadas pela ferrugem asiática, destacando regiões de interesse que possam indicar a presença da doença. Além disso, consideramos incluir a aplicação de autosupervisão, inicializando o *encoder* com um problema de classificação interno.

Outra possibilidade que pretendemos investigar é a utilização de um espaço de cor onde crominância e luminância sejam separados como entrada da rede. Especificamente, estamos interessados em explorar o espaço de cor LAB, onde a luminância é separada das componentes de crominância. Acreditamos que essa abordagem pode fornecer à rede uma representação mais discriminativa das características das folhas de soja afetadas pela ferrugem asiática.

Além disso, como um trabalho futuro, planejamos considerar a criação de um aplicativo para smartphones, que faz uso da rede neural para realizar a predição da imagem de folha de soja na própria lavoura, em tempo real, identificando folhas saudáveis de folhas com a ferrugem asiática. Essa ferramenta teria o potencial de auxiliar os agricultores na detecção precoce da doença e na implementação de medidas de controle de forma mais eficiente e direcionada.

Portanto este trabalho fornece uma visão abrangente do desempenho, impacto do aumento de dados e eficiência computacional da rede ASRDNet em tarefas de segmentação de imagens. As descobertas destacam a relevância da ASRDNet como uma ferramenta valiosa para aplicações práticas que exigem alta precisão na segmentação de objetos em imagens, ao mesmo tempo em que ressalta a importância do aumento de dados como uma estratégia eficaz para aprimorar o desempenho de modelos de aprendizado profundo.

REFERÊNCIAS

ALBUMENTATION. **Albumentations: fast and flexible image augmentations**. Disponível em: <https://albumentations.ai/>. Acesso em: 15 set. 2023.

BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. en. **IEEE Trans. Pattern Anal. Mach. Intell.**, Institute of Electrical e Electronics Engineers (IEEE), v. 39, n. 12, p. 2481–2495, dez. 2017.

BISHOP, Christopher M. **Pattern Recognition and Machine Learning**. [S. l.]: Springer, 2006. ISBN 978-0-387-31073-2.

CAI, Zhaowei; VASCONCELOS, Nuno. Cascade R-CNN: Delving into high quality object detection. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Salt Lake City, UT: IEEE, jun. 2018.

CHEN, Liang-Chieh; PAPANDREOU, George; KOKKINOS, Iasonas et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, PP, jun. 2016. DOI: 10.1109/TPAMI.2017.2699184.

CHEN, Liang-Chieh; PAPANDREOU, George; KOKKINOS, Iasonas et al. Semantic image segmentation with deep convolutional nets and fully connected CRFs, dez. 2014. arXiv: 1412.7062 [cs.CV].

CHEN, Liang-Chieh; PAPANDREOU, George; SCHROFF, Florian et al. Rethinking Atrous Convolution for Semantic Image Segmentation. **Computer Vision and Pattern Recognition**, 2017. arXiv: 1706.05587 [cs.CV].

CONAB. **Soja em números (safra 2020/21)**. [S. l.: s. n.], 2022. https://www.embrapa.br/web/portal/soja/cultivos/soja1/dados-economicos. Acessado em: 24/06/2022.

DENG, Jia et al. ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. Miami, FL: IEEE, jun. 2009.

EMBRAPA. **Ferrugem asiática da soja: manejo e prevenção**. [S. l.]: Embrapa Soja, 2023.

EVERINGHAM, Mark et al. The pascal visual object classes (VOC) challenge. en. **Int. J. Comput. Vis.**, Springer Science e Business Media LLC, v. 88, n. 2, p. 303–338, jun. 2010.

GODOY, Cláudia Vieira et al. **Eficiência de fungicidas para o controle da ferrugem-asiática da soja, Phakopsora pachyrhizi, na safra 2022/2023: resultados sumarizados dos ensaios cooperativos**. [S. l.]: Embrapa Soja, 2023.

HAND, David J. Assessing the performance of classification methods. **International Statistical Review**, Wiley Online Library, v. 80, n. 3, p. 400–414, 2012.

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

KINGMA, Diederik P.; BA, Jimmy. **Adam: A Method for Stochastic Optimization**. [S. l.: s. n.], 2017. arXiv: 1412.6980 [cs.LG].

KRIZHEVSKY, Alex. Learning Multiple Layers of Features from Tiny Images, p. 32–33, 2009. Disponível em: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.

LIN, T Y et al. Microsoft COCO: Common Objects in Context. European Conference on Computer Vision, p. 740–755, 2014.

LIN, Tsung-Yi et al. Microsoft COCO: Common Objects in Context. **CoRR**, abs/1405.0312, 2014. arXiv: 1405.0312. Disponível em: <http://arxiv.org/abs/1405.0312>.

LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. Fully convolutional networks for semantic segmentation, nov. 2014. arXiv: 1411.4038 [cs.CV].

_____._____. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, USA: IEEE, jun. 2015.

MARMANIS, D; DATCU, M; STILLA, U. Semantic Segmentation of Very High-Resolution Satellite Imagery Using Deep Learning and Overhead Geometrical Constraints. **International Journal of Remote Sensing**, v. 37, n. 19, p. 4624–4646, 2016.

OPENCV. **About - OpenCV**. Disponível em: <https://opencv.org/about/>. Acesso em: 27 mar. 2023.

PUJARI, Jagadeesh Devidas et al. Quantitative detection of soybean rust using image processing techniques. **Journal of Crop Protection**, v. 5, n. 1, 2016. eprint: http://jcp.modares.ac.ir/article-3-8488-en.pdf. Disponível em: <http://jcp.modares.ac.ir/article-3-8488-en.html>.

PYTORCH. **PyTorch documentation - PyTorch 2.0 documentation**. Disponível em: <https://pytorch.org/docs/stable/index.html/>. Acesso em: 25 jun. 2023.

PYTORCHLIGHTNING. **Welcome to PyTorch Lightning - PyTorch Lightning 2.0.0 documentation**. Disponível em: <https://lightning.ai/docs/pytorch/stable/>. Acesso em: 24 mar. 2023.

PYTORCHVISION. **torchvision - Torchvision 0.15 documentation**. Disponível em: <https://pytorch.org/vision/>. Acesso em: 25 jun. 2023.

REN, Shaoqing et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: ADVANCES in Neural Information Processing Systems. [S. l.: s. n.], 2015.

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. In: LECTURE Notes in Computer Science. Cham: Springer International Publishing, 2015. (Lecture notes in computer science). p. 234–241.

SILVA, Diego Alves da et al. An automatic phytopathometry system for chlorosis and necrosis severity evaluation of asian soybean rust infection. **Computers and Electronics in Agriculture**, v. 192, November, 2022. ISSN 01681699. DOI: 10.1016/j.compag.2021.106542.

SOUZA SILVA, José Jorge de. **O ALGORITMO DE RETROPROPAGAÇÃO**. 2022. f. 68. Diss. (Mestrado) – Instituto Federal da Paraíba, CAJAZEIRAS.

TENG, Yunfei et al. **Leader Stochastic Gradient Descent for Distributed Training of Deep Learning Models: Extension**. [S. l.: s. n.], 2022. arXiv: 1905.10395 [cs.LG].

TETILA, Everton Castelão et al. Automatic Recognition of Soybean Leaf Diseases Using UAV Images and Deep Convolutional Neural Networks. **IEEE Geoscience and Remote Sensing Letters**, v. 17, n. 5, p. 903–907, 2020. ISSN 15580571. DOI: 10.1109/LGRS.2019.2932385.

THIAGO, Luiz Roberto Lopes de S; SILVA, José Marques. Soja na Alimentação de Bovinos. **Embrapa Gado de Corte. Circular técnica, 31**, p. 6, 2003. Disponível em: <https://ainfo.cnptia.embrapa.br/digital/bitstream/item/104635/1/Soja-na-alimentacao-de-bovinos.pdf>.

TIELEMAN, Tijmen; HINTON, Geoffrey et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. **COURSERA: Neural networks for machine learning**, v. 4, n. 2, p. 26–31, 2012.

ULMAS, Priit; LIIV, Innar. Segmentation of satellite imagery using U-Net models for land cover classification, mar. 2020. arXiv: 2003.02899 [cs.CV].

WANDB. **Weights & Biases – developer tools for ML**. [S. l.: s. n.]. Disponível em: <https://wandb.ai/site>. Acesso em: 7 mar. 2023.

ZHANG, Keke; WU, Qiufeng; CHEN, Yiping. Detecting soybean leaf disease from synthetic image using multi-feature fusion faster R-CNN. **Computers and Electronics in Agriculture**, Elsevier BV, v. 183, p. 106064, abr. 2021. DOI: 10.1016/j.compag.2021.106064. Disponível em: <https://doi.org/10.1016/j.compag.2021.106064>.

APÊNDICE A – DATASET

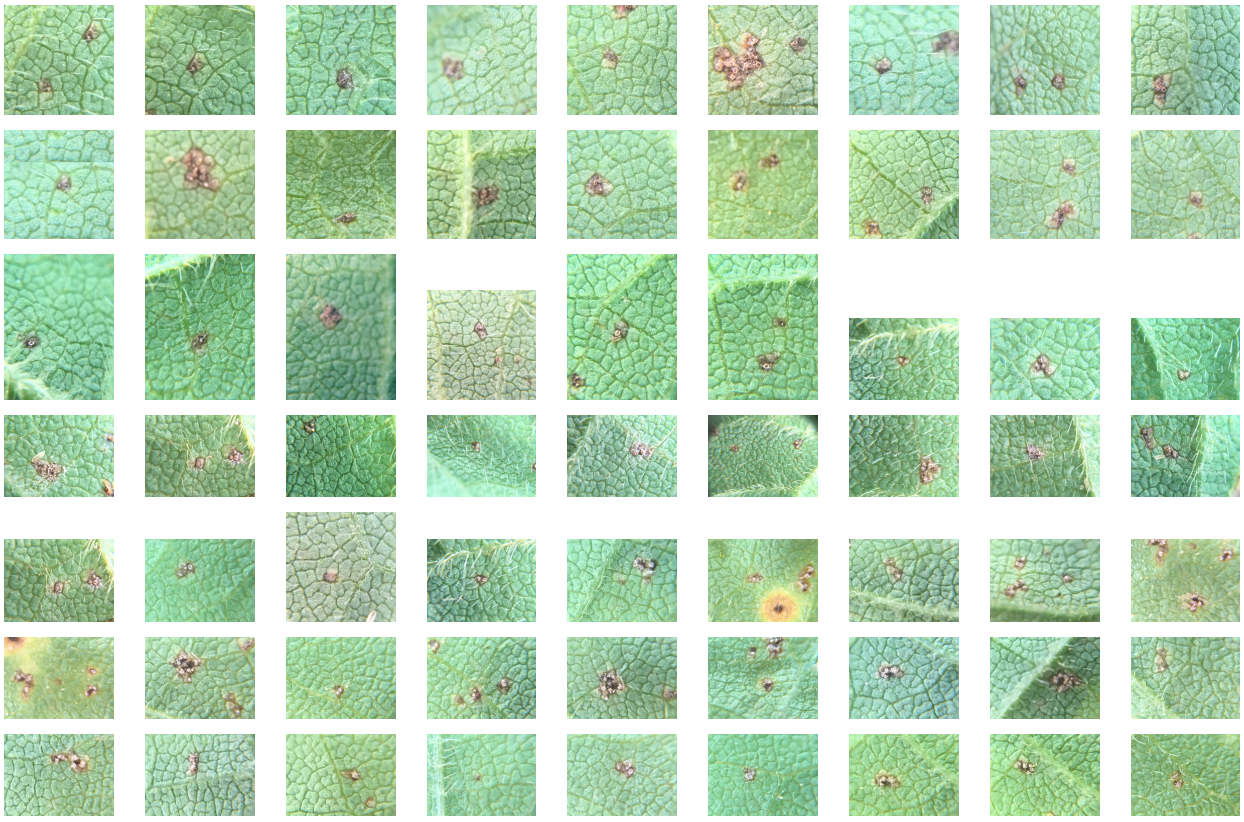
O dataset criado neste trabalho, está hospedado no GitHub, uma plataforma de controle de versão para arquivos e códigos-fonte.

A.1 Imagens Originais

Conjunto de 63 imagens de folhas de soja, infectadas com a doença da ferrugem asiática.

Link para acesso: https://github.com/Prograf-UFF/asrdnet_dataset.git

Pasta: original_images



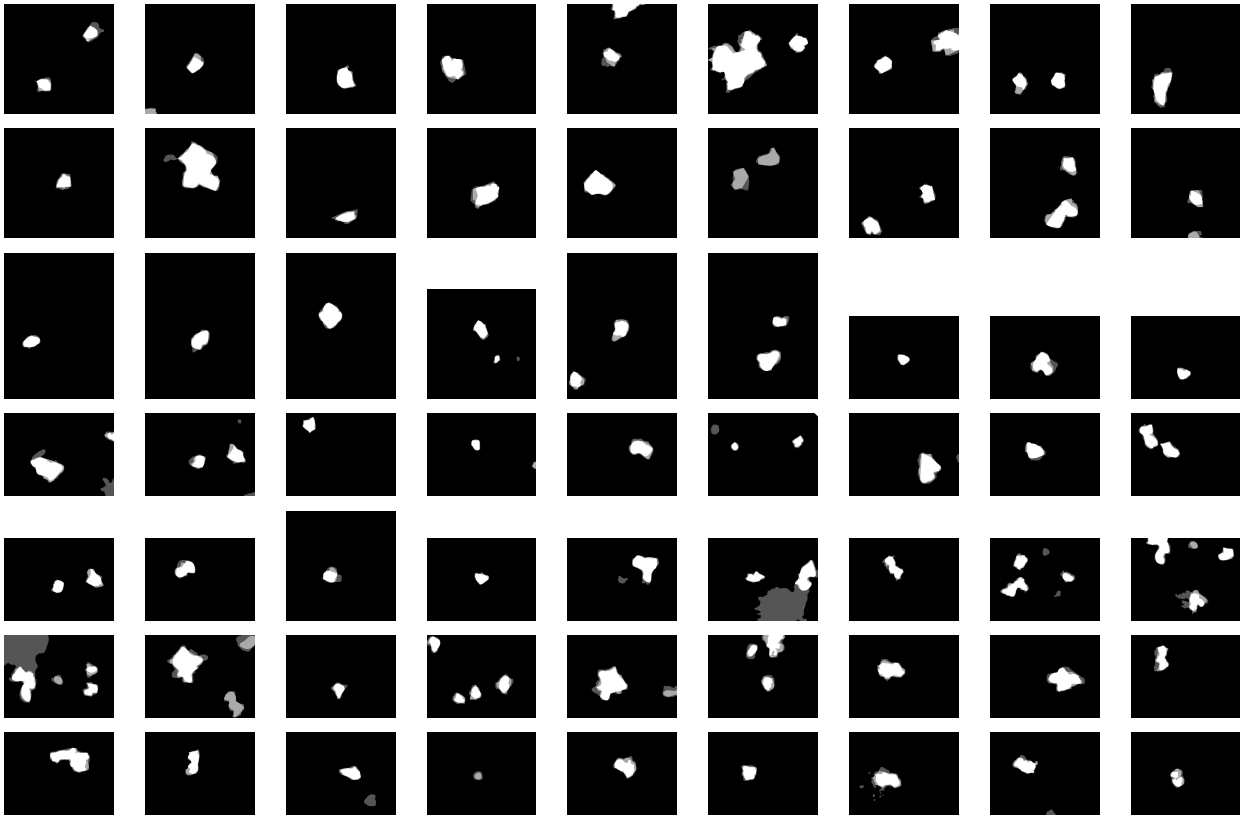
A.2 Máscaras Probabilísticas

Conjunto de 63 máscaras probabilísticas das regiões votadas por todos os analistas.

Preto: 0 votos, Cinza Escuro: 1 voto, Cinza Claro: 2 votos e Branco: 3 votos.

Link para acesso: https://github.com/Prograf-UFF/asrdnet_dataset.git

Pasta: probabilistic_mask

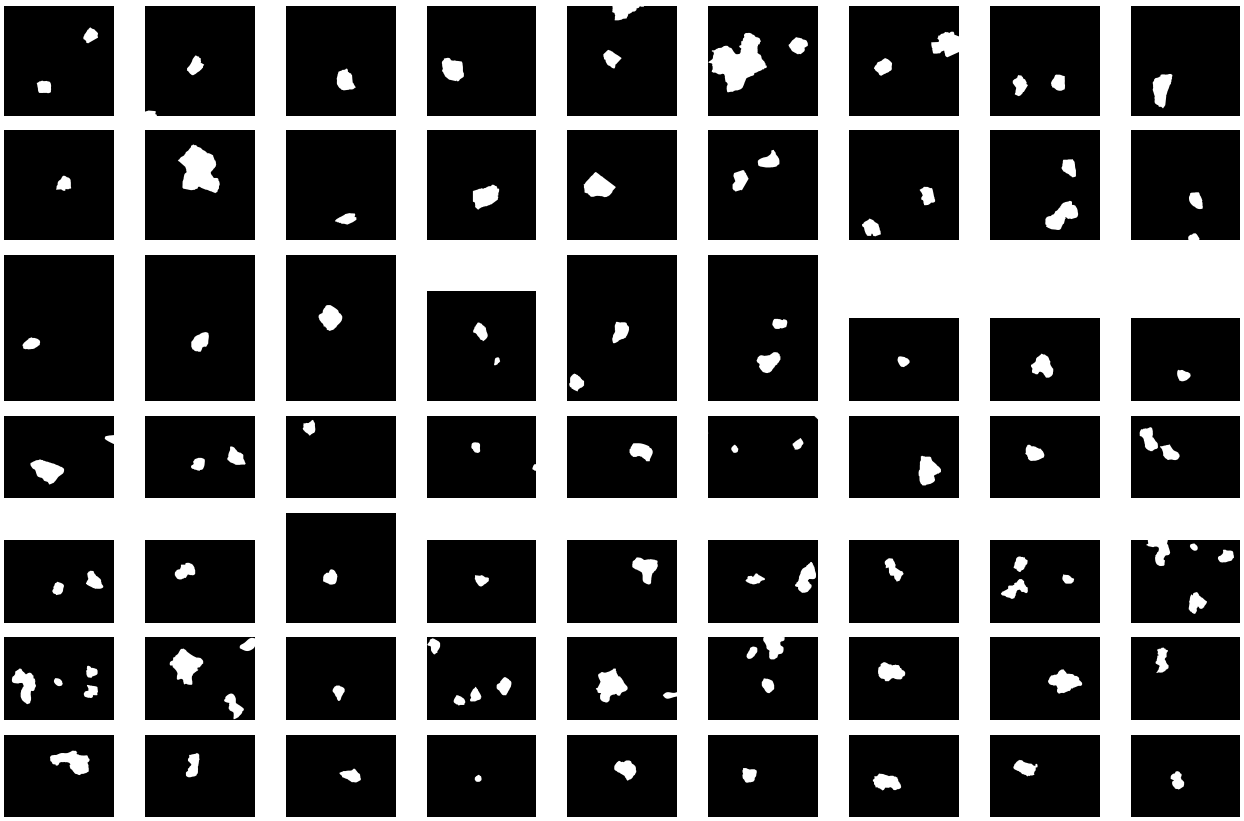


A.3 Máscaras Binárias

Conjunto de 63 máscaras binárias, que representam os pontos de interesse com duas ou mais marcações como "ponto de interesse" e as áreas sem doença com duas ou mais marcações como "não ponto de interesse".

Link para acesso: https://github.com/Prograf-UFF/asrdnet_dataset.git

Pasta: binary_mask



APÊNDICE B – DATASET

Código fonte utilizado na implementação do modelo da rede neural.

B.1 Classe DoubleConv

```
class DoubleConv(torch.nn.Module):

    def __init__(self, in_channels: int, out_channels: int, mid_channels
    : Optional[int] = None, groups: int = 1) -> None:
        super(DoubleConv, self).__init__()
        if not mid_channels:
            mid_channels = out_channels
        self.conv1 = torch.nn.Sequential(
            torch.nn.Conv2d(in_channels, mid_channels, kernel_size=3,
    padding=1, groups=groups),
            torch.nn.BatchNorm2d(mid_channels),
            torch.nn.ReLU(inplace=True)
        )
        self.conv2 = torch.nn.Sequential(
            torch.nn.Conv2d(mid_channels, out_channels, kernel_size=3,
    padding=1, groups=groups),
            torch.nn.BatchNorm2d(out_channels),
            torch.nn.ReLU(inplace=True)
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.conv1(x)
        return self.conv2(x)
```

B.2 Classes Down e Up

```
class Down(torch.nn.Module):

    def __init__(self, in_channels: int, out_channels: int) -> None:
        super(Down, self).__init__()
        self.max_pool = torch.nn.MaxPool2d(2)
        self.double_conv = DoubleConv(in_channels, out_channels)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.max_pool(x)
        return self.double_conv(x)


class Up(torch.nn.Module):

    def __init__(self, in_channels: int, out_channels: int, groups: int
    = 1) -> None:
        super(Up, self).__init__()
        self.up_sample = torch.nn.Upsample(scale_factor=2, mode='
    bilinear', align_corners=True)
        self.double_conv = DoubleConv(in_channels, out_channels,
    in_channels // 2, groups)

    def forward(self, x1: torch.Tensor) -> torch.Tensor:
        x1 = self.double_conv(x1)
        x1 = self.up_sample(x1)
        return x1
```

B.3 Classe Out

```
class Out(torch.nn.Module):

    def __init__(self, in_channels: int, out_channels: int, groups: int
    = 1) -> None:
        super(Out, self).__init__()
        self.conv = torch.nn.Conv2d(in_channels, out_channels,
    kernel_size=1, groups=groups)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        return self.conv(x)
```

B.4 Classe ASDRNet

```
class ASDRNet(torch.nn.Module):

    def __init__(self, **kwargs: Any) -> None:
        super(ASDRNet, self).__init__(**kwargs)
        # Set layers.

        # Down1
        self.inc1 = DoubleConv(3, 64)
        self.down11 = Down(64, 128)
        self.down12 = Down(128, 256)
        self.down13 = Down(256, 512)

        # Down2
        self.inc2 = DoubleConv(3, 64)
        self.down21 = Down(64, 128)
        self.down22 = Down(128, 256)

        # Down3
        self.inc3 = DoubleConv(3, 64)
        self.down31 = Down(64, 128)

        # UP1
        self.up11 = Up(512, 256)
        self.up12 = Up(256, 128)
        self.up13 = Up(128, 64)
        self.out1c = Out(64, 8)

        # UP2
        self.up21 = Up(256, 128)
        self.up22 = Up(128, 64)
        self.out2c = Out(64, 8)

        # UP3
        self.up31 = Up(128, 64)
        self.out3c = Out(64, 8)

        #Mix
        self.out = Out(24, 1)


    def forward(self, input: Image, *, result: Result = Result.
```

```
    PREDICTION) -> Union[Prediction, Tuple[Prediction, Features]]:
42      # Feature extraction.
43
44      # Down1
45      y21 = self.inc1(input)
46      y22 = self.down11(y21)
47      y23 = self.down12(y22)
48      features2 = self.down13(y23)
49
50      # Down2
51      y31 = self.inc2(input)
52      y32 = self.down21(y31)
53      features3 = self.down22(y32)
54
55      # Down3
56      y41 = self.inc3(input)
57      features4 = self.down31(y41)
58
59      # Segmentation.
60
61      # UP1
62      y2 = self.up11(features2)
63      y2 = self.up12(y2)
64      y2 = self.up13(y2)
65      y2 = self.out1c(y2)
66
67      # UP2
68      y3 = self.up21(features3)
69      y3 = self.up22(y3)
70      y3 = self.out2c(y3)
71
72      # UP3
73      y4 = self.up31(features4)
74      y4 = self.out3c(y4)
75
76      mix_out = torch.cat((y2, y3, y4), dim=1)
77      logits = self.out(mix_out)
78
79      if self.num_classes > 2:
80          preds = torch.argmax(logits, dim=1)
81      else:
82          logits = logits.squeeze(1)
83          preds = torch.sigmoid(logits).round().to(dtype=torch.long)
```

```
84          # Return prediction and features.
85          return preds, logits
```