UNIVERSIDADE FEDERAL DE SANTA CATARINA
CTC - CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

Felipe Alfredo Nack

CONTAGEM DE ANÉIS E IDENTIFICAÇÃO DE TORAS DE MADEIRA UTILIZANDO REDES NEURAIS PROFUNDAS

Florianópolis
2024

Felipe Alfredo Nack

CONTAGEM DE ANÉIS E IDENTIFICAÇÃO DE TORAS DE MADEIRA UTILIZANDO REDES NEURAIS PROFUNDAS

Dissertação submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do título de mestre em Engenharia de Automação e Sistemas.
Orientador: Prof. Marcelo Ricardo Stemmer, Dr.
Coorientador: Prof. Maurício Edgar Stivanello, Dr.

Florianópolis
2024

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Nack, Felipe Alfredo
CONTAGEM DE ANÉIS E IDENTIFICAÇÃO DE TORAS DE MADEIRA UTILIZANDO REDES NEURAIS PROFUNDAS / Felipe Alfredo Nack ; orientador, Marcelo Ricardo Stemmer, coorientador, Maurício Edgar Stivanello, 2024.
128 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2024.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2. Visão computacional. 3. Contagem de anéis em toras. 4. Segmentação de toras. 5. Transformers. I. Stemmer, Marcelo Ricardo. II. Stivanello, Maurício Edgar. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. IV. Título.

Felipe Alfredo Nack

CONTAGEM DE ANÉIS E IDENTIFICAÇÃO DE TORAS DE MADEIRA UTILIZANDO REDES NEURAIS PROFUNDAS

O presente trabalho em nível de mestrado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof.(a) Jomi Fred Hübner, Dr.
Universidade Federal de Santa Catarina

Prof.(a) Mário Lúcio Roloff, Dr.
Universidade Federal de Santa Catarina

Certificamos que esta é a **versão original e final** do trabalho de conclusão que foi julgado adequado para obtenção do título de mestre em Engenharia de Automação e Sistemas.

Documento assinado digitalmente
JULIO ELIAS NORMEY RICO
Data: 07/05/2024 10:59:52-0300
CPF: ***.840.859-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Julio Elias Normey-Rico, Dr.
Coordenador do Programa

Documento assinado digitalmente
Marcelo Ricardo Stemmer
Data: 06/05/2024 18:37:16-0300
CPF: ***.093.279-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador

Florianópolis, 2024.

Este trabalho é dedicado à minha família.

AGRADECIMENTOS

Ao meu pai Jean e minha mãe Evanite pelo constante incentivo, apoio e compromisso. A minha namorada, Adrielle, por todo amor, companheirismo, cumplicidade e por estar sempre ao meu lado quando precisei. Aos meus avós, Hilda e Adalberto, por estarem sempre presentes.

Agradeço ao meu orientador Prof. Dr. Marcelo Ricardo Stemmer e ao meu coorietandor Prof. Dr. Maurício Edgar Stivanello, pela orientação, conselhos, ideias e apoio durante todo o processo. Ao coordenador do programa, Prof. Dr. Julio Elias Normey-Rico por todo o trabalho prestado ao departamento. Ao Sr. Enio Snoeijer e a Sra. Lívia Scheffer Santos pelo comprometimento e rápido suporte quando necessário. A todos os profesores do PPGEAS pelas lições e conhecimentos valiosos compartilhados comigo. A UFSC, pela oportunidade de fazer parte de um programa de qualidade. Agradeço, por fim, ao CNPq pelo auxílio financeiro durante esta etapa.

”All we have to decide is what to do with the time that is given to us.”
John Ronald Reuel Tolkien

RESUMO

O Brasil atua extensamente no fornecimento de matéria prima para a indústria madeireira. Recentemente, esse fornecimento chegou ao valor de 160 milhões de metros cúbicos de madeira em toras produzidos apenas na silvicultura. Apesar do grande fornecimento, todo esse valor se concentra em um grupo mínimo e seleto de espécies de madeira. Essa concentração se deve, principalmente, à escassez de informações referentes a outras espécies de madeira. Neste sentido, o presente trabalho se propõe a desenvolver algoritmos de visão computacional, baseados em redes neurais profundas, que sejam capazes de identificar a posição da tora de madeira em uma imagem e estimar a quantidade de anéis de crescimento encontrados. Para realizar esta tarefa, o presente trabalho também envolve a adaptação de uma base de dados existente de identificação de toras e a criação de uma base de dados para contagem de anéis. A identificação de toras é feita através de redes neurais profundas baseadas em CNNs e Transformers para segmentação semântica. Os melhores resultados foram obtidos pela rede Swin que atingiu métricas mIoU e mAcc acima de 90 pontos em todos os testes. A contagem de anéis é feita pela família de redes neurais profundas ResNet, adaptadas para realizar a tarefa de regressão. Os melhores resultados foram apresentados por uma variação da ResNet34, com MSE de 195.29, MAE de 12.00 e R2-Score de 0.60. Todos os tempos de inferência, seja para segmentação semântica ou para contagem de anéis, ficaram abaixo da marca de 10 segundos. Dados os objetivos do projeto e os resultados obtidos, a conclusão do trabalho é que o emprego destas redes na tarefa de extrair características de toras de madeira é promissora e deve ser pesquisada mais a fundo.

Palavras-chave: Contagem de anéis em toras. Segmentação de toras. Visão computacional. Transformers. CNNs.

ABSTRACT

Brazil plays a significant role in supplying raw materials to the timber industry. Recently, this supply reached a value of 160 million cubic meters of timber logs produced solely in forestry. Despite the substantial supply, this value is concentrated in a minimal and select group of wood species. This concentration is primarily due to the scarcity of information regarding other wood species. In this context, the present work aims to develop computer vision algorithms based on deep neural networks capable of identifying the position of a timber log in an image and estimating the number of growth rings present. To accomplish this task, the present work also involves adapting an existing database for log identification and creating a new database for ring counting. Log identification is achieved through deep neural networks based on CNNs and Transformers for semantic segmentation. The best results were obtained with the Swin network, achieving mIoU and mAcc metrics above 90 points in all tests. Ring counting is performed by the family of deep neural networks ResNet, adapted for regression tasks. The best results were presented by a variation of ResNet34, with an MSE of 195.29, MAE of 12.00, and R2-Score of 0.60. All inference times, whether for semantic segmentation or ring counting, remained below the 10-second mark. Given the project's objectives and the results obtained, the conclusion of the work is that the use of these networks in extracting features from timber logs is promising and should be further researched.

Keywords: Timber growth rings counting. Timber segmentation. Computer vision. Transformers. CNNs.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O Brasil é um dos principais países detentores de recursos naturais abundantes, sendo o único que possui extensa área de florestas tropicais. Por isso atua de forma expressiva no fornecimento de matéria prima para a indústria madeireira (REUWSAAT *et al.*, 2017) (SNIF, 2022). Segundo dados do IBGE (IBGE, 2022), em 2022 cerca de 160 milhões de metros cúbicos de madeira em toras foram produzidos na silvicultura brasileira. No entanto, a produção madeireira se concentra apenas em uma quantidade mínima e seleta de espécies (RIBEIRO, 2008), mesmo com a grande biodiversidade encontrada em território nacional. Essa limitação do setor madeireiro se deve principalmente à escassez de informações na literatura envolvendo outras espécies de madeira, sejam essas informações referentes às suas propriedades mecanicas ou outras características.

Nesse sentido, fica claro que levantar informações referentes à toras de madeiras é um desafio relevante para a indústria nacional. As informações levantas podem ser através equipamentos mais refinados, como equipamentos de tomografia computacional, ou através de técnicas mais simples, como a medida da largura dos anéis de crescimento de árvores. Como visto em (NIKLAS; SPATZ, 2010), a largura dos anéis de crescimento está diretamente relacionada com propriedades mecânicas da madeira. Outras características interessantes podem ser a área da seção reta da tora de madeira, estimativa de volume da tora, quantidade de anéis de crescimento, estimativa de idade da tora ou mesmo a contagem de toras empilhadas.

O desafio, nesse sentido, se caracteriza especificamente como desenvolver algum método que possa ser utilizado para acelerar a aquisição de dados sobre toras de madeiras. Este desafio, no entanto, já foi proposto em outras pesquisas. Frequentemente as pesquisas buscam utilizar técnicas de visão computacional para esta tarefa. Desta forma, o presente trabalho se propõe a avaliar algoritmos que possam, de alguma forma, extrair características de imagens de toras de madeira. Trabalhos prévios, como (NACK, 2021) utilizam técnicas de visão computacional clássica para estimar o volume de toras de madeira.

Durante a revisão sistemática da literatura, dois desafios envolvendo características de toras de madeira foram evidenciados. Existe a necessidade de localizar a posição da tora dentro de uma imagem e existe a necessidade de se obter uma estimativa da quantidade de anéis de crescimento presentes nesta tora. Seguindo estas informações obtidas na revisão, o presente trabalho se propõe a utilizar técnicas modernas de segmentação semântica para identificar a posição de uma tora de madeira em uma imagem, similar a abordagem clássica encontrada em (NACK, 2021) e na abordagem baseada em CNNs vista em (DECELLE; JALILIAN, 2020), isto é, avaliar técnicas modernas encontradas em redes neurais recentes. Adicionalmente, o

presente trabalho também se propõe a treinar redes neurais profundas para estimar a quantidade de anéis de crescimento em toras de madeira.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Este trabalho tem como objetivo desenvolver métodos para a localização e contagem de anéis de crescimento em imagens digitais de toras de madeira.

1.1.2 Objetivos Específicos

Através do objetivo geral, foram definidos os seguintes objetivos específicos:

- Encontrar e adaptar uma base de dados com fotos da seção reta de toras de madeira para a tarefa de segmentação semântica;
- Selecionar diferentes modelos de segmentação semântica encontrados na literatura, tanto baseados em CNNs quanto baseados em Transformers;
- Treinar os modelos encontrados na base de dados adaptada;
- Realizar análises comparativas dos modelos tanto no âmbito de precisão quanto no âmbito de velocidade e consumo de memória;
- Adaptar a base de dados para a contagem de anéis de crescimento;
- Encontrar redes neurais profundas que sejam amplamente utilizadas em aplicações gerais;
- Adaptar as redes neurais profundas para a tarefa de regressão;
- Treinar as redes neurais profundas na base de dados voltada para contagem de anéis;
- Realizar análises comparativas dos modelos de contagem de anéis nos âmbitos de erro, estabilidade, velocidade de processamento e consumo de memória.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são discutidos os principais conceitos teóricos necessários para o desenvolvimento deste sistema de extração de características de toras de madeira. A metodologia de revisão sistemática da literatura proposta por (KITCHENHAM, 2004) foi utilizado como guia para realizar esta revisão de literatura. As três principais bases científicas utilizadas nas pesquisas são: Google Scholar, IEEE e Elsevier.

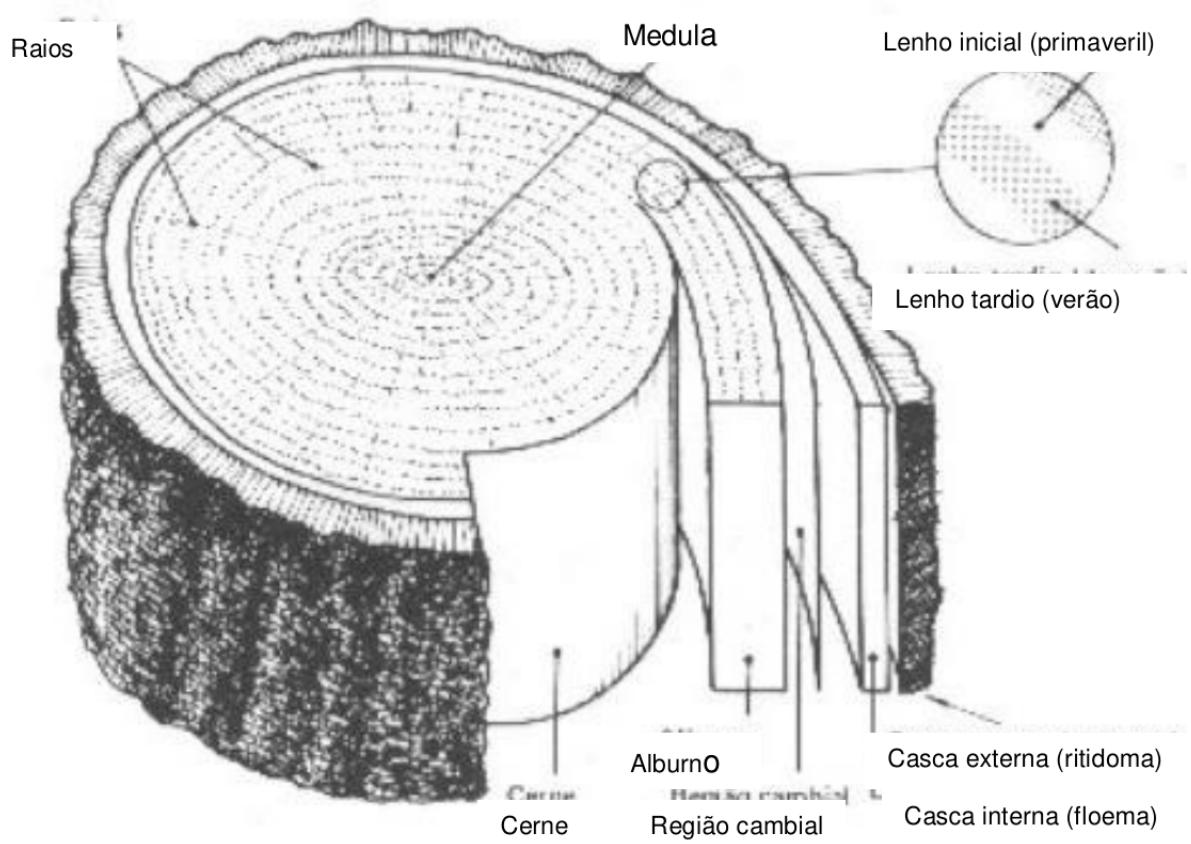
2.1 MADEIRA

A madeira é uma estrutura biológica complexa, um composto de muitos produtos químicos e tipos de células que atuam em conjunto para atender às necessidades de uma planta viva. Ao tentar compreender a madeira no contexto da tecnologia da madeira, muitas vezes negligenciamos o fato fundamental e básico de que a madeira evoluiu ao longo de milhões de anos para servir três funções principais nas plantas: condução de água das raízes para as folhas, suporte mecânico de o corpo da planta e armazenamento de produtos bioquímicos. Não existe nenhuma propriedade da madeira – física, mecânica, química, biológica ou tecnológica – que não seja fundamentalmente derivada do fato de a madeira ser formada para satisfazer as necessidades da árvore viva. Para realizar qualquer uma dessas funções, a madeira deve ter células projetadas e interligadas de maneira suficiente para suprir as necessidades. Estas três funções influenciaram a evolução de aproximadamente 150.000 espécies diferentes de plantas lenhosas, cada uma com propriedades, usos e capacidades únicas, tanto em contextos vegetais como humanos ((US), 2021).

Uma árvore viva e em crescimento tem dois domínios principais: o broto e as raízes. As raízes são as estruturas subterrâneas responsáveis pela absorção de água e nutrientes minerais, ancoragem mecânica da parte aérea e armazenamento de produtos bioquímicos. Se alguém cortar uma árvore e olhar para o troco, diversas observações grosseiras poderão ser feitas. O tronco é composto por diversos materiais presentes em faixas concêntricas. De fora para dentro da árvore estão a casca externa, a casca interna, o câmbio vascular, o alburno, o cerne e a medula ((US), 2021).

Observando uma seção transversal (Figura 1) do tronco percebem-se as seguintes partes: casca, lenho, medula, e raios medulares (SZÜCS *et al.*, 2015). A casca protege a árvore contra agentes externos e é dividida em duas partes: camada externa (camada cortical), composta de células mortas e camadas internas, formadas por tecidos vivos moles úmidos. O lenho é a parte resistente do tronco, apresenta as seguintes partes: alburno e cerne. O alburno é formado de madeira jovem, mais permeável, menos denso, e mais sujeito ao ataque de fungos apodrecedores e insetos e com menor resistência mecânica, enquanto que o cerne é formado das modificações do alburno, onde ocorre a madeira mais densa mais resistente que a do alburno. A

Figura 1 – Visão macroscópica de uma seção transversal de um tronco de árvore.



Fonte: (SZÜCS *et al.*, 2015)

medula é parte central que resulta do crescimento vertical, onde ocorre madeira de menor resistência. Os raios medulares ligam as diferentes camadas entre si e também transportam e armazenam a seiva. Entre a casca e o lenho existe uma camada delgada, visível com o auxílio de lentes, aparentemente fluida, denominada câmbio. Ela é a parte viva da árvore. Todo o aumento de diâmetro da árvore vem dela, por adição de novas camadas e não do desenvolvimento das mais antigas (SZÜCS *et al.*, 2015).

2.1.1 Planos de corte

Embora a madeira possa ser cortada em qualquer direção para exame, a organização e a inter-relação entre os sistemas axial e radial dão origem a três perspectivas principais a partir das quais podem ser visualizadas para obter o máximo de informações. Essas três perspectivas são o plano transversal da seção, o plano radial da seção e o plano tangencial da seção ((US), 2021).

O plano transversal da seção é a face que fica exposta quando uma árvore é cortada. O plano transversal da seção fornece informações sobre feições que variam tanto na direção medula-casca (chamada de direção radial) quanto aquelas que variam

na direção circunferencial (chamada de direção tangencial). Não fornece informações sobre variações para cima e para baixo no tronco ((US), 2021).

O plano radial da seção segue na direção medula-casca (Figura 2) e é paralelo ao sistema axial, portanto fornece informações sobre mudanças longitudinais no caule e da medula à casca ao longo do sistema radial. Para descrevê-lo geometricamente, ele é paralelo ao raio de um cilindro e se estende para cima e para baixo ao longo do comprimento do cilindro. No sentido prático, é a face ou plano que fica exposto quando uma tora é dividida exatamente da medula à casca. Ele não fornece nenhuma informação sobre recursos que variam na direção tangencial ((US), 2021).

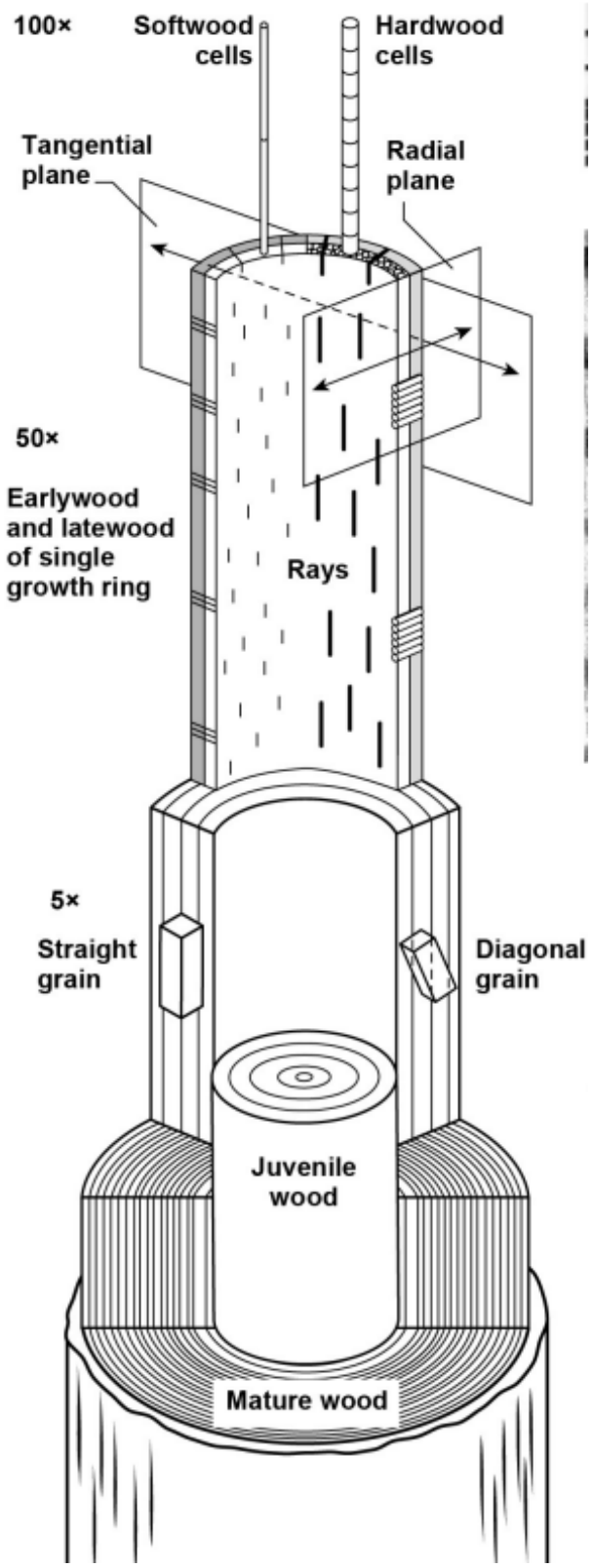
O plano tangencial forma um ângulo reto com o plano radial (Figura 2). Geometricamente, é paralelo a qualquer linha tangente que tocaria o cilindro e se estende ao longo do comprimento do cilindro. Uma maneira pela qual o plano tangencial ficaria exposto seria se a casca fosse descascada de um tronco; a face exposta é o plano tangencial. O plano tangencial da seção não fornece nenhuma informação sobre feições que variam na direção radial, mas fornece informações sobre as dimensões tangenciais das feições. Todos os três planos de seção são importantes para a observação adequada da madeira, e somente olhando para cada um deles é possível obter uma compreensão holística e precisa da estrutura tridimensional da madeira. Os três planos de seção são determinados pela estrutura da madeira e pela forma como as células da madeira estão dispostas ((US), 2021).

2.1.2 Anéis de crescimento

A existência dos anéis de crescimento foi observada há muito tempo, sendo que os primeiros relatos foram feitos na Grécia antiga. Já no século XVI, Leonardo da Vinci reconheceu a relação entre os anéis de crescimento e o clima em árvores de Pinus da região de Toscana (Itália). Ele relatou que “os anéis de crescimento permitem estimar o número de anos e, em função de sua espessura, indicar os anos mais e os menos secos”. Durante muito tempo, era quase consenso entre os pesquisadores que somente as espécies folhosas de clima temperado formam anéis de crescimento anuais. As árvores nas regiões tropicais e subtropicais não apresentariam sazonalidade da atividade cambial, pelas condições climáticas serem consideradas praticamente constantes durante o ano. Portanto, não formariam anéis anuais de crescimento (BOTOSSO; MATTOS, 2002).

A madeira é produzida pelo câmbio vascular, uma camada de divisões celulares de cada vez, mas sabemos por experiência geral que em muitas madeiras grandes grupos de células são produzidos mais ou menos juntos no tempo, e esses grupos agem juntos para servir a árvore. Essas coleções de células produzidas juntas durante um intervalo de tempo discreto são conhecidas como incrementos de crescimento ou anéis de crescimento. As células formadas no início do incremento de crescimento

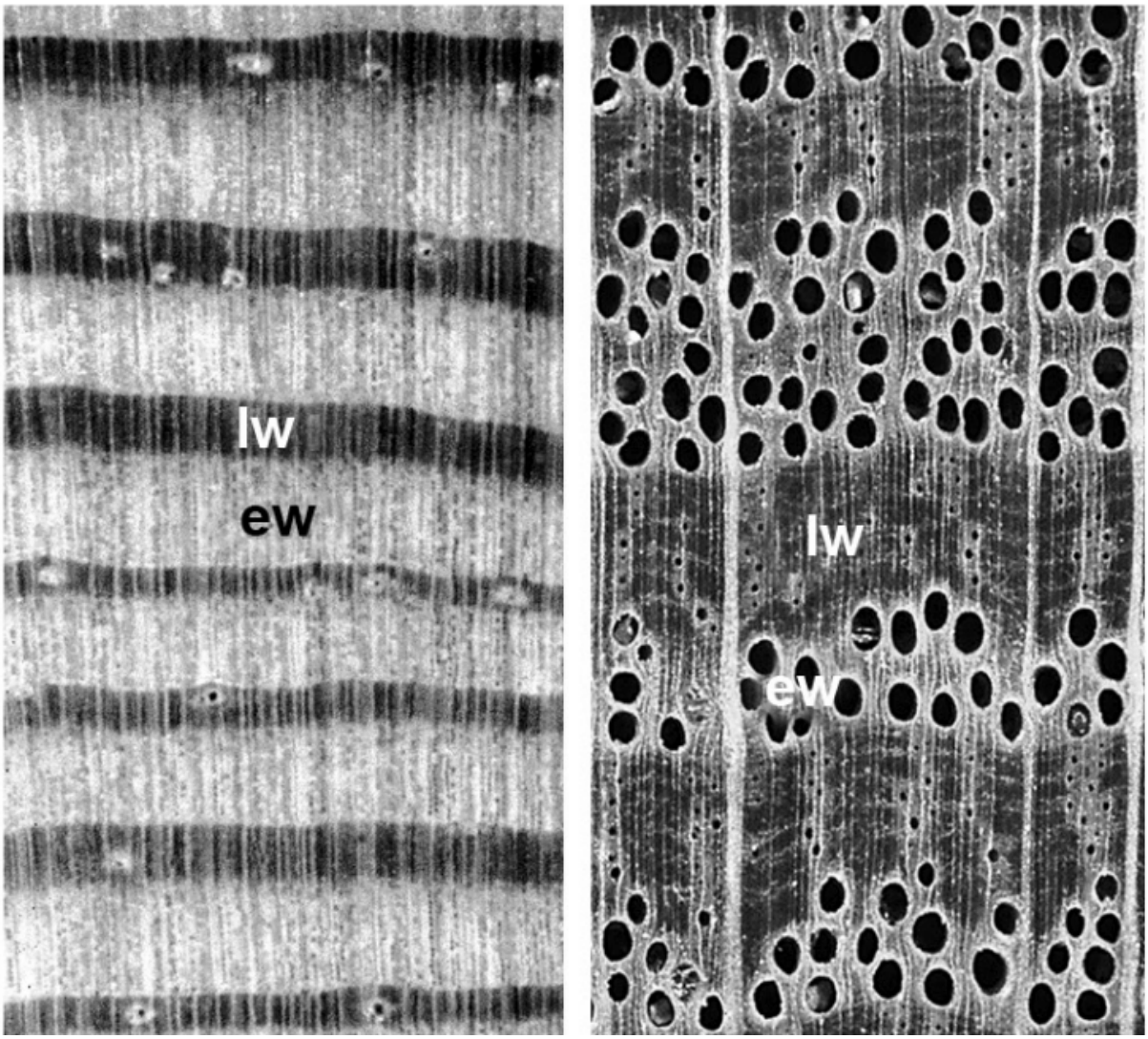
Figura 2 – Planos de corte de um tronco de árvore.



Fonte: ((US), 2021)

são chamadas de células do lenho inicial, e as células formadas na última porção do incremento de crescimento são chamadas de células do lenho tardio (Figura 3). Nas regiões temperadas do mundo e em qualquer outro lugar com sazonalidade distinta e regular, as árvores formam a sua madeira em incrementos anuais de crescimento; isto é, toda a madeira produzida numa estação de crescimento é organizada em conjunto numa entidade funcional e reconhecível que muitas fontes chamam de anéis anuais. Tal terminologia reflete esta tendência temperada, pelo que um termo preferido é incremento de crescimento, ou anel de crescimento ((US), 2021).

Figura 3 – Vistas manuais dos anéis de crescimento, ambos compostos de lenho inicial (ew) e lenho tardio (lw).



Fonte: ((US), 2021)

O lenho inicial corresponde ao crescimento da árvore no início do período vegetativo, normalmente a primavera, quando as plantas despertam do período de dormência e reassumem suas atividades fisiológicas com todo o vigor. Com a aproximação

do fim do período vegetativo, normalmente o outono, as células diminuem paulatinamente a sua atividade fisiológica. Em consequência, suas paredes celulares tornam-se gradualmente mais espessas e suas cavidades menores. Isso dá ao lenho tardio uma tonalidade mais escura que permite distingui-lo do inicial ou primaveril (BOTOSSO; MATTOS, 2002).

Além das características próprias de cada espécie, é fácil compreender que árvores de regiões onde as estações do ano são bem definidas apresentam anéis de crescimento nítidos. Ao contrário, as que crescem em locais de condições climáticas mais constantes têm normalmente anéis de crescimento indistintos ou pouco evidentes. Em muitas árvores tropicais e subtropicais, os anéis correspondem aos períodos de chuva e de seca, inundações, queda das folhas e/ou simplesmente dormência, podendo ocorrer dois ou mais ciclos em um ano. Os anéis de crescimento não são, portanto, necessariamente anuais. Também, é comum encontrar-se anéis de crescimento descontínuos, que não formam um círculo completo em torno da medula. Há, ainda, os chamados falsos anéis de crescimento que ocorrem quando se forma mais de um anel por período vegetativo. Essas situações fora do padrão normal dificultam a determinação exata da idade de uma árvore (BOTOSSO; MATTOS, 2002).

O tamanho das árvores é uma função do período de tempo em que elas se desenvolveram. Sobre este período de tempo, o tamanho do indivíduo será o resultado das interações da capacidade genética inerente do crescimento e do ambiente no qual está habitando. Anualmente o período durante o qual os fatores climáticos, tais como temperatura, umidade do ar, duração e intensidade de luz, e outros fatores como a fertilidade do solo, se modificam e tornam-se elementos decisivos no crescimento das árvores. Alterações favoráveis nas condições ambientais podem produzir períodos de crescimento estacionário/sazonal nas plantas. Muitas árvores em sua estrutura anatômica da madeira e em respostas fenológicas, no início e fim da estação de crescimento, adquirem características estruturais bem definidas e facilmente observáveis (IMAÑA ENCINAS; SILVA, G. F. d.; PINTO, 2005).

2.1.3 Idade das árvores

A idade de qualquer organismo vivo é o período de vida que ele tem, considerando desde a sua origem ou surgimento até um ponto determinado no tempo. A idade de uma floresta ou povoamento florestal torna-se um conceito vago, pois nem todas as árvores que as compõe iniciam o seu crescimento ao mesmo tempo. Nesse sentido, emprega-se a idade média das árvores como maneira de aproximação. Porém para as práticas de manejo florestal, se faz necessário que as florestas nativas e os reflorestamentos possam ser caracterizados por uma idade definida (IMAÑA ENCINAS; SILVA, G. F. d.; PINTO, 2005). Na mensuração florestal a idade de uma árvore é uma variável muito importante, especialmente na estimativa da produção florestal.

Fundamentalmente é utilizada nas avaliações do crescimento e da produtividade de um sítio e nos ordenamentos florestais. A idade é também utilizada como ferramenta para práticas silviculturais, na determinação do crescimento presente e futuro da floresta e nas decisões dos planos de manejo (IMAÑA ENCINAS; SILVA, G. F. d.; PINTO, 2005). Ao estimar a idade das arvores é possível:

- avaliar o incremento em termos de diâmetro, área basal, volume e altura de uma espécie em um determinado local, permitindo comparar a capacidade produtiva de diferentes locais;
- estimar o crescimento em altura das árvores dominantes nos povoamentos, para que sejam construídas curvas de índice de sítio de modo a se determinar a capacidade produtiva dos locais onde estes povoamentos estão implantados;
- definir parâmetros a serem utilizados nas práticas de manejo florestal, servindo principalmente como base comparativa entre povoamentos e decidindo metas na exploração da floresta.

Sendo assim, o conhecimento da idade das árvores e das informações que podem ser inferidas do estudo dos seus anéis de crescimento são de suma importância para a otimização do uso da floresta (BOTOSSO; MATTOS, 2002). A idade de uma árvore pode ser determinada com diferentes métodos, dependendo da precisão desejada.

2.2 VISÃO COMPUTACIONAL

Nesta seção são apresentados alguns conceitos básicos referentes a visão computacional. De forma abrangente, os assuntos à seguir são tratados: o que é a visão computacional, o que são imagens digitais, técnicas para processar imagens, conceitos básicos de classificação de imagens, conceitos básicos para regressão em imagens e conceitos básicos para segmentação semântica em imagens.

2.2.1 O que é a visão computacional?

Uma maneira de perceber o mundo é capturar e interpretar padrões de luz ambiente refletida da cena. Isso é o que nossos olhos e cérebro fazem, nos dando o sentido da visão. Nossa própria experiência é que a visão é um sensor muito eficaz para a maioria das coisas que fazemos, incluindo reconhecimento, navegação, desvio de obstáculos e manipulação. Não estamos sozinhos nisso, e quase todas as espécies animais usam os olhos — na verdade, a evolução inventou o olho muitas vezes. A Figura 4 mostra parte da diversidade de olhos encontrados na natureza (CORKE, 2023).

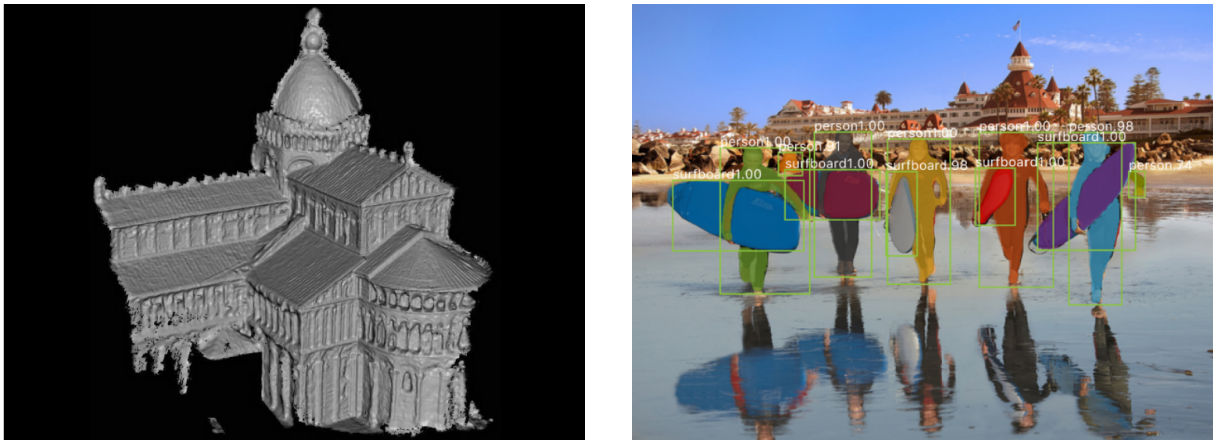
(a) mosca assassina, *Holcocephala fusca*; (b) aranha saltadora, *Phidippus putnami*; (c) vieira, onde cada pequena bola azul é um olho; (d) olho humano.

Figura 4 – Olhos em diversas espécies animais. Fonte: Adaptado de (CORKE, 2023)

É interessante notar que até mesmo animais muito simples, como as abelhas, por exemplo, com cérebros compostos por apenas 10^6 neurônios (em comparação com nossos 10^{11}), são capazes de realizar tarefas complexas e críticas para a vida, como encontrar alimentos e levá-los de volta à colmeia usando apenas a visão. Para animais mais complexos como nós, os benefícios da visão superam o custo biológico muito alto de possuir um olho: o próprio olho complexo, músculos para movê-lo, pálpebras e dutos lacrimais para protegê-lo e um córtex visual grande (um terço do nosso cérebro) para processar os dados que ele produz (CORKE, 2023).

Naturalmente, conforme as tecnologias avançam, pesquisadores em visão computacional têm desenvolvido, em paralelo, técnicas matemáticas para recuperar a forma tridimensional e aparência de objetos em imagens. Aqui, o progresso nas últimas duas décadas tem sido rápido. Atualmente possuimos técnicas confiáveis para calcular com precisão um modelo 3D de um ambiente (Figura 5a) a partir de milhares de fotografias parcialmente sobrepostas. Com um conjunto grande o suficiente de vistas de um objeto específico ou fachada, podemos criar modelos de superfície 3D densos e precisos usando correspondência estéreo. Até mesmo podemos, com sucesso moderado, delinear a maior parte das pessoas e objetos em uma fotografia (Figura 5b) (SZELISKI, 2022). Todos estes avanços tecnológicos no campo de visão computacional buscam, de forma geral, se aproximar da capacidade humana de interpretar o mundo através de imagens.

Neste sentido é possível descrever a visão computacional como um campo interdisciplinar, que utiliza-se tanto de técnicas encontradas na inteligência artificial quanto na ciência da computação, concentrando-se em permitir que máquinas possam adquirir, processar, avaliar e entender tanto videos quanto imagens. Através das técnicas existentes no campo da visão computacional, uma máquina pode identificar objetos, reconhecer faces, estimar sentimentos, reconhecer movimentos, rastrear movimentos, desvendar padrões, buscar por falhas, reconstruir ambientes tridimensionais dentre



(a) Reconstrução 3D de um edifício. (b) Segmentação de instâncias.

Figura 5 – Exemplos de sistemas que envolvem visão computacional. Fonte: Adaptado de (SZELISKI, 2022)

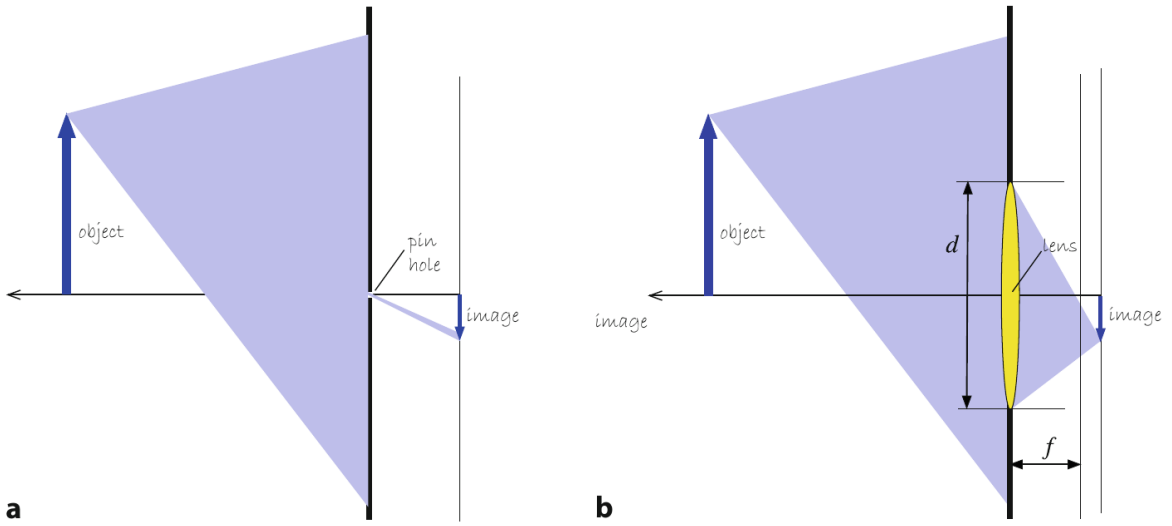
outros. Claro que as aplicações da visão computacional são vastas, tangindo diversos campos como a medicina moderna, a segurança do trabalho, a automação, veículos autonomos e gerenciamento de acesso.

2.2.2 Imagens digitais

Dentro do campo da visão computacional, as imagens digitais são o principal objeto de estudo. As imagens digitais, em conjunto com algoritmos e técnicas da visão computacional, permitem que sistemas computacionais entendam, analisem e extraiam informações relevantes do meio. Estas imagens são adquiridas, em geral, por câmeras digitais.

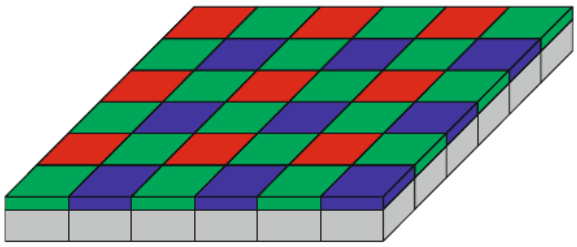
Para entender o princípio de aquisição de uma imagem digital podemos olhar para um ancestral rudimentar da câmera digital: a câmera *pinhole*. Como visto em (FORSYTH; PONCE, 2011), é possível reproduzir o experimento da câmera *pinhole* utilizando uma caixa com um pequeno furo e uma vela. De forma análoga, um pequeno buraco na parede de um quarto escuro irá projetar uma imagem invertida do mundo exterior, na parede oposta do quarto escuro, como podemos ver na Figura 6a. Essa imagem invertida, no entanto, será bastante fracada dado que apenas uma pequena parcela da luz externa conseguirá passar pelo pequeno orifício da câmera *pinhole*. Uma forma de solucionar este problema é incluir na câmera uma lente, de forma que mais luz seja captada (Figura 6b) (CORKE, 2023). Note que a Figura 14 sugere claramente que o modelo com lentes é capaz de absorver mais luz, permitindo que a imagem final tenha maior intensidade.

No modelo com lentes, o local onde a luz adquirida pela lente é focada é chamado de plano de imagem. O plano de imagem gera uma imagem nítida do ambiente e, por este motivo, o plano de imagem é o local onde são colocados os sensores óticos das câmeras digitais. No sensor de uma câmera digital está presente o filtro de Bayer



(a) (b)

Figura 6 – Habilidades para captar luz do **a**) modelo pinhole e **b**) modelo com lentes. Fonte: Adaptado de (CORKE, 2023)



G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb

(a) (b) (c)

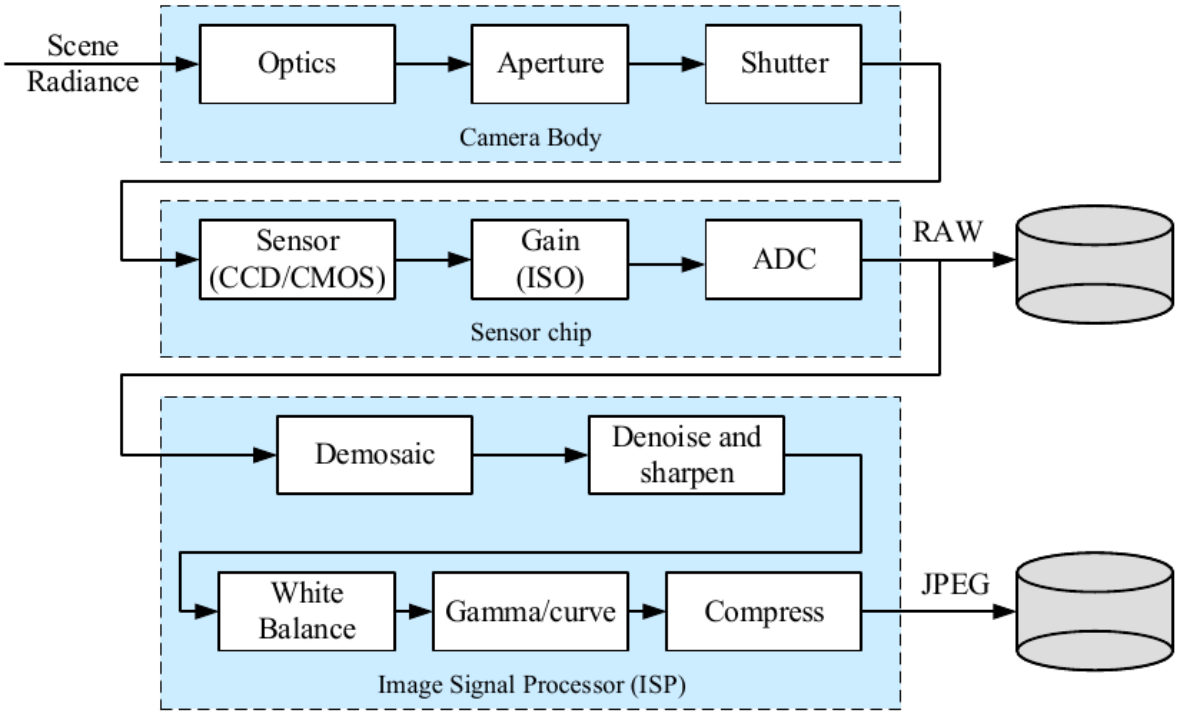
Figura 7 – Filtro Bayer. **a**) os blocos cinza representam a matriz de sensores fotossensíveis de silício; **b**) matriz de filtros para cor vermelha, verde e azul; **c**) esquema de interpolação para cálculo do valor de cada *pixel*. Fonte: Adaptado de (CORKE, 2023) e (SZELISKI, 2022)

(Figura 7a). A filtragem de Bayer utiliza uma matriz de filtros vermelhos, verdes e azuis, que permitem apenas a passagem de seus respectivos comprimentos de onda, que chegam então em sensores fotossensíveis de silício. Os sensores de silício captam então a intensidade da luz incidente. Os resultados destas medições são então combinados (Figuras 7b e 7c) para dar origens aos *pixels* de uma imagem digital (CORKE, 2023) (SZELISKI, 2022).

Nesse sentido, a Figura 8 expressa uma versão simplificada das etapas de processamento que acontecem em câmeras digitais modernas. Tipicamente duas principais tecnologias são utilizadas nos sensores das câmeras para captar a luz: *charge-coupled device* (CCD) e *complementary metal oxide on silicon* (CMOS). Após a luz ser captada pelo sensor, o sinal passa por um circuito de ganho analógico e então é direcionado para um circuito de conversão analógico-digital. A saída do conversor

analógico-digital é chamada de imagem em formato bruto. Após alguns passos de pós-processamento no processador de sinal de imagem - como balanceamento de branco e ajuste de gamma - a imagem digital está finalizada. Neste ponto a imagem digital é interpretada meramente como uma matriz tridimensional (largura, altura e canais de cor (Cinza ou RGB)) (SZELISKI, 2022).

Figura 8 – Etapas no sensoreamento de uma imagem, indicando várias fontes de ruído e alguns passos de pós-processamento.



Fonte: (SZELISKI, 2022).

2.2.3 Processamento de imagens

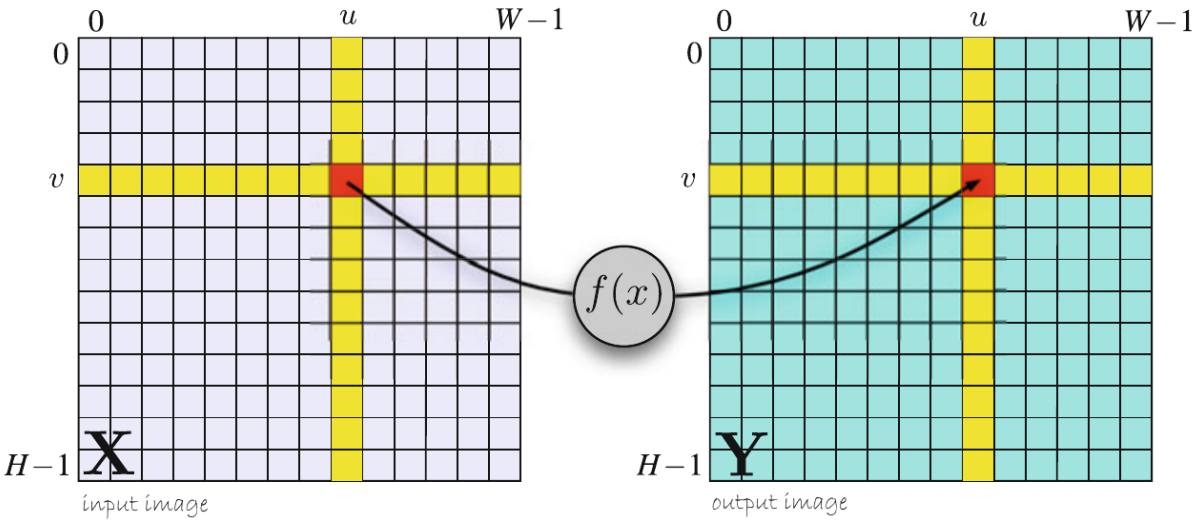
O processamento de imagem é um processo computacional que transforma uma ou mais imagens de entrada em uma imagem de saída. Frequentemente, é usado para aprimorar uma imagem para visualização ou interpretação humana, como, por exemplo, para melhorar o contraste, brilho ou mesmo normalizar imagens(CORKE, 2023). Dentre as operações de processamento de imagens para visão computacional podemos citar: operações monádicas, operações diádicas e operações espaciais.

A operação monádica de processamento de imagem pode ser vista esquematicamente na Figura 9. O resultado **Y** é uma imagem de mesmo tamanho **W***x***H** que a sua equivalente de entrada **X**, e cada *pixel* da saída é uma função do *pixel* correspondente da imagem de entrada. Alguns exemplos de operações monádicas são ajuste de brilho

e contraste, inversão de cores, binarização da imagem e limiarização (CORKE, 2023). Essa operação pode ser expressa matematicamente de acordo com a Eq. (1), onde $Y_{u,v}$ é o *pixel* de saída na posição (u,v) e $X_{u,v}$ é seu equivalente na imagem original.

$$Y_{u,v} = f(X_{u,v}), \forall (u,v) \in X \tag{1}$$

Figura 9 – Esquema de uma operação monádica.



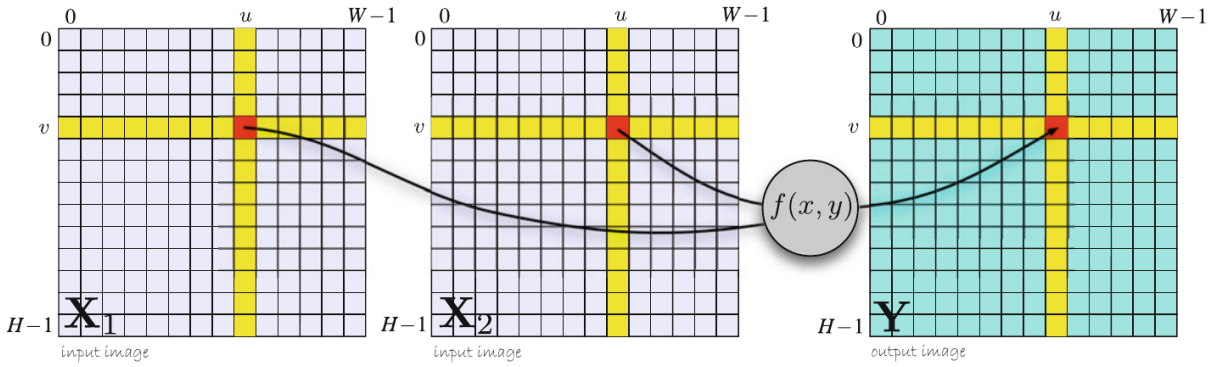
Fonte: (CORKE, 2023).

Na Figura 10 está apresentada uma representação esquemática da operação diádica. Nesta operação, dois conjuntos de imagens de entrada resultam em uma única imagem de saída, e todas as três imagens possuem o mesmo tamanho. Cada *pixel* da imagem de saída é uma função dos *pixels* correspondentes nas duas imagens de entrada. Alguns exemplos de operações diádicas são: adição e subtração de imagens, mistura de imagens e operações lógicas (AND, OR, XOR) entre imagens (CORKE, 2023). A operação diádica pode ser expressa matematicamente de acordo com a Eq. (2), onde $Y_{u,v}$ é o *pixel* de saída na posição (u,v) e $X_{1\,u,v}$ e $X_{2u,v}$ são seus equivalentes na imagem original.

$$Y_{u,v} = f(X_{1\,u,v}, X_{2u,v}), \forall (u,v) \in X_1, X_2 \tag{2}$$

A operação espacial é mostrada esquematicamente na Figura 11. Nesta operação, cada *pixel* na saída é uma função de todos os *pixels* em uma região circundante ao *pixel* correspondente na imagem de entrada. A operação espacial é expressa matematicamente pela Eq. (3), nde $Y_{u,v}$ é o *pixel* de saída na posição (u,v) e $\omega_{u,v}$ é uma região conhecida como janela ou *kernel* de tamanho *wxw* centrada na posição (u,v). w geralmente é ímpar e geralmente é descrito como $w = 2h + 1$, onde h é a metade

Figura 10 – Esquema de uma operação diádica.

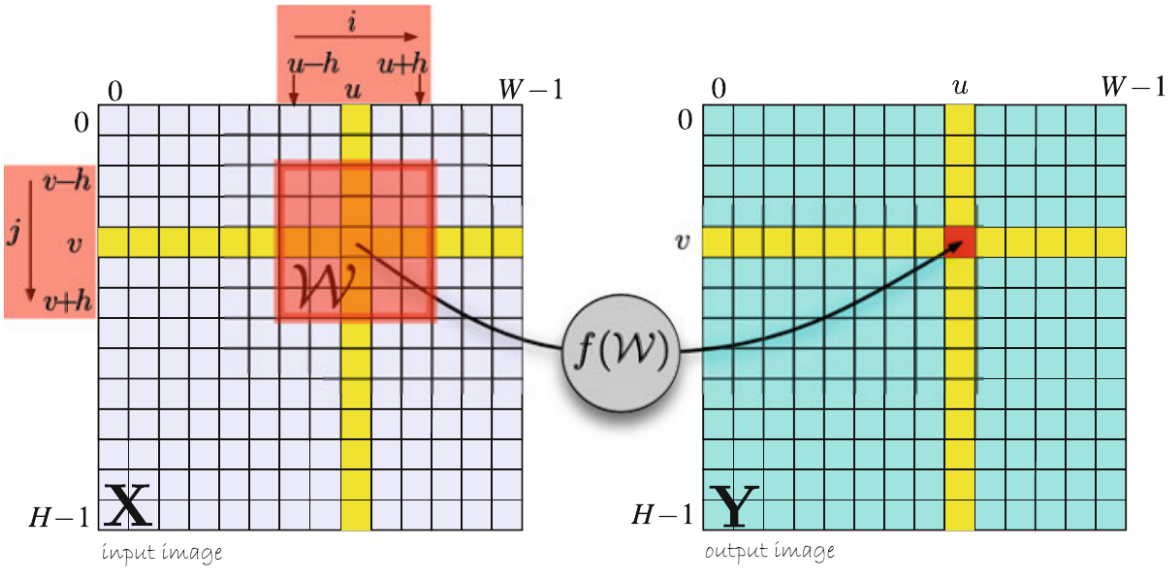


Fonte: (CORKE, 2023).

par de w. A função $f(.)$ aplicada na região pode ser tanto linear quanto não linear (CORKE, 2023) (FORSYTH; PONCE, 2011) (DAVIES, 2018). Duas das operações mais conhecidas e tema deste trabalho são a correlação e a convolução.

$$Y_{u,v} = f(\omega_{u,v}), \forall (u,v) \in X \tag{3}$$

Figura 11 – Esquema de uma operação espacial.



Fonte: (CORKE, 2023).

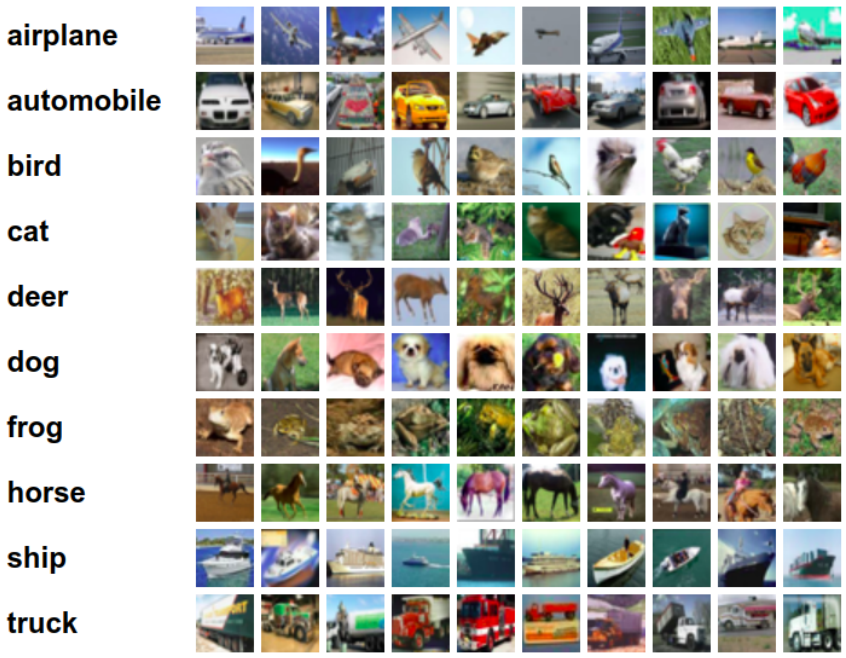
Enquanto as operações monádicas, diádicas e espaciais são fundamentais no processamento básico de imagens, há também outras tarefas de mais alto nível

cruciais na visão computacional. A classificação, regressão e segmentação semântica em imagens são focos avançados nesse campo.

2.2.3.1 Classificação e Regressão

A classificação de imagens é uma espécie de habilidade biologicamente primária do sistema de percepção visual humano. Tem sido uma tarefa ativa e desempenha um papel crucial no campo da visão computacional, que visa classificar automaticamente imagens em classes pré-definidas (FENG *et al.*, 2019). Até então, a maioria dos sistemas de visão computacional dependiam de características manualmente elaboradas para descrever uma imagem de maneira específica. Em seguida, classificadores passíveis de aprendizado, como *random forests* ou a árvore de decisão, eram aplicados para extrair características e chegar a uma decisão final (SINGH, A.; SINGH, P., 2020). Tradicionalmente, estes modelos de classificação atingem bom desempenho apenas em conjuntos de dados pequenos, como o CIFAR-10 (Figura 12) (KRIZHEVSKY; HINTON *et al.*, 2009) e o MNIST (Figura 13) (LECUN *et al.*, 1998). Avanços razoáveis no desenvolvimento da classificação de imagens ocorreu quando o conjunto de dados de imagens em grande escala, ImageNet, foi criado por (DENG *et al.*, 2009). A criação da ImageNet aconteceu concomitantemente ao emergente sucesso das redes de aprendizado profundo no campo da visão computacional (FENG *et al.*, 2019).

Figura 12 – Exemplo da base de dados CIFAR-10. A base conta com 10 classes diferentes e cerca de 6.000 imagens para cada classe.



Fonte: (KRIZHEVSKY; HINTON *et al.*, 2009).

Figura 13 – Exemplo da base de dados MNIST.

Fonte: (LECUN *et al.*, 1998).

No entando, é comum que em alguns problemas não seja desejável categorizar as imagens em classes fixas. Nesse sentido, enquanto a classificação de imagens busca pela categorização de imagens com base em seus conteúdos visuais discretos, a regressão em imagens expande essa abordagem ao buscar estimativas numéricas ou funções contínuas para representar relações quantitativas dentro das imagens. Essa transição reflete uma mudança de paradigma, passando da identificação de categorias específicas para a estimativa de valores contínuos ou características espaciais mais detalhadas, ampliando as possibilidades de análise e compreensão dos dados visuais em contextos diversos na visão computacional. Isto é, ao invés de rotular uma imagem em classes predeterminadas, a regressão busca prever e modelar características numéricas, como a localização precisa de objetos, o tamanho de regiões específicas ou até mesmo valores contínuos associados a propriedades dentro da imagem. Essa abordagem permite uma análise mais detalhada e específica das informações presentes na imagem, extrapolando para além das classificações discretas para incluir informações quantitativas que são cruciais em muitos contextos de análise visual.

Alguns dos desafios amplamente encontrados na literatura envolvendo a regressão em imagens são a estimativa de idade (SHEN *et al.*, 2018) e a estimativa de densidade de multidões (SAQIB *et al.*, 2019). Ainda, é importante ressaltar que ape-

sar da diferença fundamental entre classificação e regressão para imagens, é comum encontrar arquiteturas de redes neurais que façam simultaneamente estas operações, combinando-as em saídas mais complexas e ricas, como a detecção de objetos executada por (REDMON *et al.*, 2016).

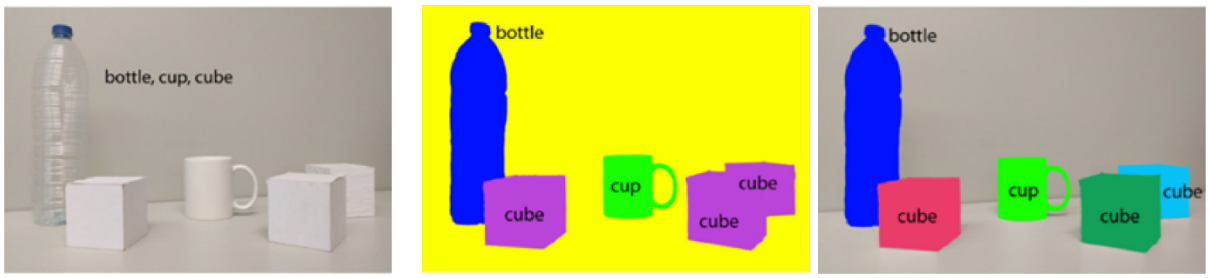
2.2.3.2 Segmentação

A segmentação em imagens é uma tarefa desafiadora em sistemas de visão computacional. Muitos métodos foram desenvolvidos para lidar com esse problema, abrangendo desde veículos autônomos, interação humano-computador, robótica, pesquisa médica, agricultura e assim por diante. Muitos desses métodos foram construídos usando o paradigma de aprendizado profundo, que tem mostrado um desempenho proeminente (LATEEF; RUICHEK, 2019). A segmentação de imagem é considerada uma classificação em nível de pixel, que tem como objetivo dividir uma imagem em regiões significativas, classificando cada pixel em uma entidade específica (FENG *et al.*, 2019).

Na segmentação tradicional de imagens, a ideia de fusão e divisão de regiões locais não supervisionadas tem sido extensivamente explorada com base em agrupamento (OHLANDER; PRICE; REDDY, 1978), otimização de critérios globais (FELZENSZWALB; HUTTENLOCHER, 2004) ou interação com o usuário (YANG *et al.*, 2010). As emergentes tecnologias de aprendizado profundo promoveram classificações supervisionadas em larga escala, avançando da classificação de objetos no nível da imagem para a localização de objetos em caixas delimitadoras, e ainda mais para a segmentação de objetos em nível de pixel. Nesse sentido, a segmentação de imagem atual (Figura 14a) é orientada a objetos e pode ser dividida em dois ramos principais: (i) a segmentação semântica, que atribui a cada pixel em uma imagem uma classe semântica de objeto, como mostrado na Figura 14b; e (ii) a segmentação de instância, que prevê rótulos diferentes para diferentes instâncias de objetos como uma melhoria adicional à segmentação semântica, conforme mostrado na Figura 14c (FENG *et al.*, 2019).

De forma similar ao problema de classificação, a segmentação em imagens também conta com algumas bases de dados que são amplamente utilizadas como padrão de comparação. Conforme indica (FENG *et al.*, 2019), os desafios propostos pelas bases de dados Pascal VOC (Figura 15) (EVERINGHAM *et al.*, 2015) e MS-COCO (Figura 16) (LIN, T.-Y. *et al.*, 2015) são os mais relevantes e amplamente utilizados na tarefa de segmentação. Adicionalmente, as bases Cityscapes (Figura 17) (CORDTS *et al.*, 2016) e MVD (Figura 18) (NEUHOLD *et al.*, 2017) fornecem imagens de ruas com diversos objetos devidamente rotulados. Estas bases são utilizadas em conjunto com as métricas que serão discutidas posteriormente para avaliar o desempenho de diversos algoritmos, principalmente modelos de redes neurais profundas.

Figura 14 – Segmentação em imagens. **a**) imagem original **b**) segmentação semântica e **c**) segmentação de instâncias.



(a) (b) (c)

Fonte: (FENG *et al.*, 2019)

Figura 15 – Exemplo da base de dados Pascal VOC. **a**) imagem original e **b**) anotação.

(a) (b)

Fonte: (EVERINGHAM *et al.*, 2015)

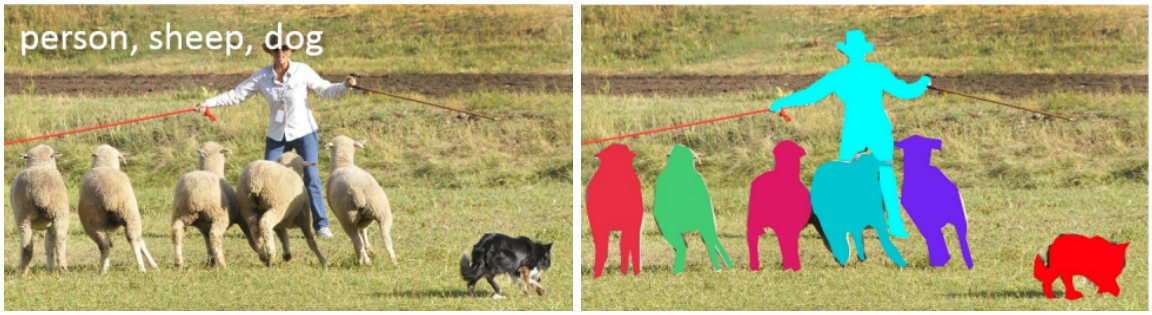
2.3 REDES NEURAIS ARTIFICIAIS

A área das Redes Neurais Artificiais foi, originalmente, inspirada pelo desafio de modelar redes neurais biológicas. Eventualmente, tornou-se uma área que tem como objetivo atingir bons resultados em tarefas de aprendizado de máquina. Nesse sentido, podemos iniciar a discussão com uma descrição muito breve e de alto nível do sistema biológico que grande parte da área foi inspirada (LI, 2023b).

2.3.1 Perceptron

De modo geral, a unidade computacional básica do cérebro é o neurônio. Aproximadamente 86 bilhões de neurônios podem ser encontrados no sistema nervoso humano e eles estão conectados por aproximadamente 10^{14} - 10^{15} sinapses. O diagrama da Figura 19 mostra um desenho esquemático de um neurônio biológico (Figura 19a) e um modelo matemático comum (Figura 19b). Cada neurônio recebe sinais de entrada de seus dendritos e produz sinais de saída ao longo de seu axônio. O axônio

Figura 16 – Exemplo da base de dados MS-COCO. **a**) imagem original e **b**) anotação.



(a) (b)

Fonte: (LIN, T.-Y. *et al.*, 2015)

Figura 17 – Exemplo da base de dados Cityscapes. **a**) imagem original e **b**) anotação.

(a) (b)

Fonte: (CORDTS *et al.*, 2016)

se ramifica e se conecta através de sinapses aos dendritos de outros neurônios. No modelo computacional de um neurônio, os sinais que viajam ao longo dos axônios (e.g.: x0), interagem de forma multiplicativa (e.g.: w0x0) com os dendritos de outro neurônio com base na força sináptica naquela sinapse (e.g.: w0). A ideia é que as forças sinápticas, os pesos w, sejam aprendíveis e controlem a força de influência, e sua direção: excitatória - peso positivo - ou inibitória - peso negativo - de um neurônio sobre outro. No modelo básico, os dendritos transportam o sinal para o corpo celular onde todos são somados. Se a soma final estiver acima de um certo limite, o neurônio pode disparar, enviando um impulso ao longo de seu axônio. No modelo computacional, assumimos que os tempos precisos dos impulsos não importam, e que apenas a frequência do disparo comunica informações. Com base nessa interpretação, modelamos a taxa de disparo do neurônio com uma função de ativação f, que representa a frequência dos impulsos ao longo do axônio (LI, 2023b). Damos o nome de perceptron para o diagrama matemático de um neurônio apresentado na Figura 19b.

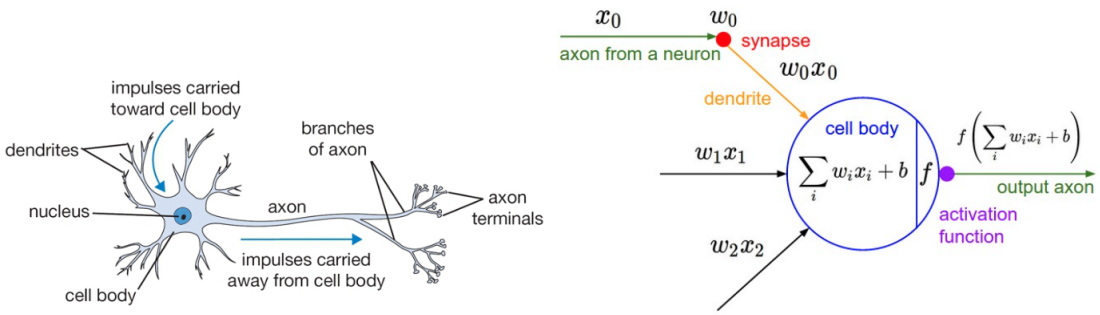
É importante ressaltar que o diagrama apresentado na Figura 19a é bastante rudimentar e não leva em consideração os diferentes tipos de neurônios e suas respectivas nuâncias. No entanto, ele é suficiente para entendermos a origem do modelo

Figura 18 – Exemplo da base de dados MVD. **a**) imagem original e **b**) anotação.

(a) (b)

Fonte: (NEUHOLD *et al.*, 2017)

Figura 19 – Neurônios. **a**) neurônio biológico e **b**) modelo matemático de um neurônio artificial.



(a) (b)

Fonte: (LI, 2023b)

matemático de um neurônio artificial.

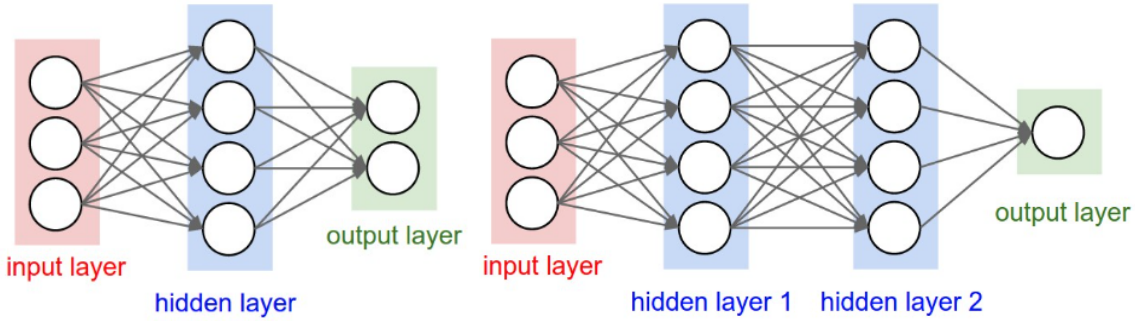
2.3.1.1 Perceptron Multicamadas

Os perceptrons multi camadas são Redes Neurais Artificiais modeladas como coleções de neurônios artificiais conectados em um grafo acíclico. Em outras palavras, as saídas de alguns neurônios podem se tornar entradas de outros neurônios. Ciclos não são permitidos, pois isso implicaria em um loop infinito na passagem direta de uma rede. Em vez de amontoados amorfos de neurônios conectados, os modelos de Redes Neurais Artificiais frequentemente são organizados em camadas distintas de neurônios. Para redes neurais regulares, o tipo de camada mais comum é a camada totalmente conectada, na qual os neurônios entre duas camadas adjacentes são conectados completamente em pares, mas os neurônios dentro de uma única camada não compartilham conexões (LI, 2023b). Este tipo de rede pode ser encontrada na literatura

tanto como camada totalmente conectada quanto como perceptron multi camadas.

Nos diagramas da Figura 20 então apresentados dois exemplos de perceptrons multi camadas. Estas diferentes formas de montar uma rede neural também são frequentemente chamadas de arquiteturas. Na Figura 20a é apresentada uma arquitetura que conta com apenas uma única camada oculta (*hidden layer*). Por convenção, ao nomear uma rede neural, tipicamente não contamos a camada de entrada, então a rede da Figura 20a é chamada de "Perceptron Multi Camadas com 2 camadas". De forma similar, a rede da Figura 20b é chamada de "Perceptron Multi Camadas com 3 camadas". Além da nomenclatura, também nos referimos a redes por seus respectivos tamanhos, seja este calculado através da contagem total de neurônio ou da contagem total de parâmetros que podem ser aprendidos. Como visto anteriormente, os parâmetros que podem ser aprendidos em um perceptron são os pesos de suas conexões **w** e o termo de viés (*bias*) **b**. Desta forma, os tamanhos das redes das Figuras 20a e 20b são, respectivamente, 6 neurônios - 26 parâmetros e 9 neurônios - 41 parâmetros.

Figura 20 – Perceptron Multi Camadas **a**) com uma camada oculta e **b**) com duas camadas ocultas.



(a) (b)

Fonte: (LI, 2023b)

Em arquiteturas de redes neurais modernas é comum encontrar a camada totalmente conectada ou perceptron multi camadas posicionado no final da arquitetura, conforme veremos nas próximas seções. Este posicionamento estratégico permite que o bloco de neurônios totalmente conectados exerça uma função de classificador e/ou regressor. Em outras palavras, para tarefas de classificação, por exemplo, redes modernas utilizam outros mecanismos - como a convolução ou os blocos de transformers - para codificar e reduzir a dimensão de informações complexas de forma que um perceptron multi camadas possa executar a tarefa de classificação destas informações comprimidas.

2.3.2 Redes Neurais Convolucionais

As redes neurais tradicionais, como visto anteriormente, recebem como entrada um único vetor que é transformado através de uma série de camadas ocultas. Cada camada oculta é composta de um conjunto de neurônios, onde cada neurônio está completamente conectado a todos os neurônios da camada anterior. Os neurônios de uma única camada trabalham de forma totalmente independente e não compartilham nenhuma conexão. Essas redes não podem ser facilmente escalonadas para imagens complexas, por exemplo, uma imagem de tamanho 32x32x3, deve ter um neurônio totalmente conectado em uma primeira camada oculta com 32x32x3 = 3072 pesos, isso significa que imagens maiores aumentarão para um número excessivo de pesos que serão difíceis de manejar, tornando sua conectividade um desperdício (LEMOS *et al.*, 2021).

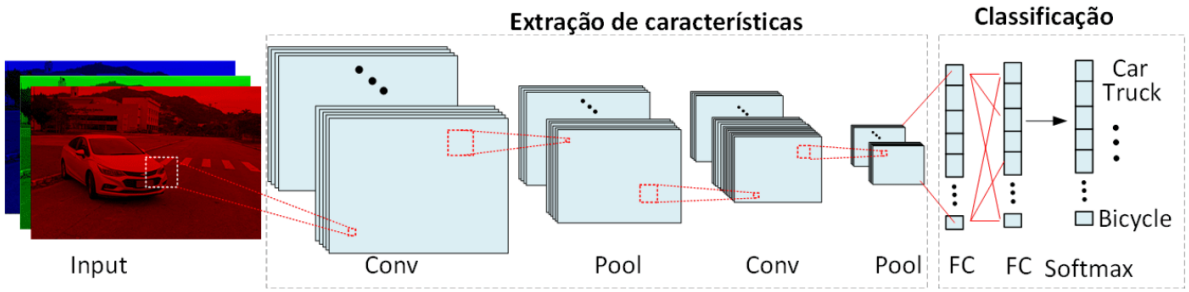
Convolutional Neural Networks (CNNs) ou ConvNets são muito similares às Redes Neurais vistas anteriormente. Elas são compostas por neurônios que possuem pesos (**w**) e vieses (**b**) aprendíveis. Cada neurônio recebe alguns inputs, realiza um produto escalar e opcionalmente segue com uma função não linear. As arquiteturas de ConvNet pressupõem, em geral, que as entradas são imagens, o que nos permite codificar certas propriedades na arquitetura. Isso torna a função de propagação mais eficiente para implementar e reduz drasticamente a quantidade de parâmetros na rede (LI, 2023a).

Nesse sentido, as CNNs são um tipo de modelo de aprendizado profundo para processar dados que têm um padrão de grade, como imagens. Ela é inspirada na organização do córtex visual animal (HUBEL; WIESEL, 1968) e projetada para aprender automaticamente e de forma adaptativa hierarquias espaciais de características, desde padrões de baixo nível até padrões de alto nível (YAMASHITA *et al.*, 2018). Este aprendizado de características de baixo e alto nível é atingido através dos blocos de construção de uma CNN, que são tipicamente três: a camada convolucional, a camada de pooling e a camada totalmente conectada. O diagrama simplificado da Figura 21 indica que as camadas de convolução (conv) e pooling (pool) são responsáveis pela extração de características (baixo e alto nível) do input (imagens), enquanto as camadas totalmente conectadas (FC) são responsáveis por, neste caso, executar o processo de classificação.

2.3.2.1 Camada Convolucional

A camada convolucional (conv) é o principal elemento de uma CNN, sendo responsável por extrair as características de baixo e alto nível da entrada através de uma combinação de operações lineares (convoluções ou correlações cruzadas) e não-lineares (funções de ativação). As operações de convolução e correlação cruzada

Figura 21 – Representação simplificada de uma CNN.



Fonte: (MAJIN ERAZO *et al.*, 2021).

são, na verdade, algumas das possíveis operações espaciais discutidas nas seções anteriores.

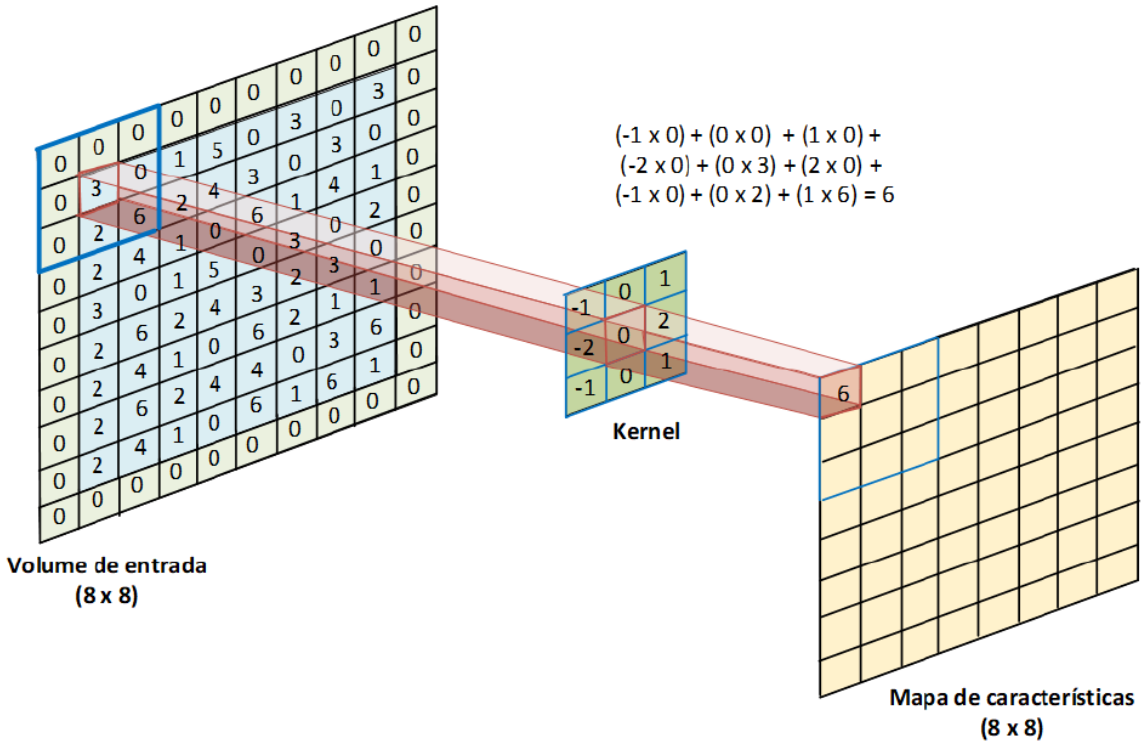
O principal elemento de uma camada convolucional é o filtro ou *kernel*, que é um pequeno elemento matricial - tipicamente de lado h=3 ou h=5 - cuja profundidade é igual ao da entrada. Isto é, se a entrada de uma camada convolucional é uma imagem de tamanho 100x100x3 (RGB), então um típico filtro para se aplicar tem tamanho 3x3x3 ou 5x5x3. Durante a propagação direta (*forward pass*), deslizamos (mais precisamente, convoluímos) cada filtro através da largura e altura do volume de entrada e calculamos produtos escalares entre as entradas do filtro e a entrada em qualquer posição. À medida que deslizamos o filtro sobre a largura e altura do volume de entrada, produziremos um mapa de ativação bidimensional que fornece as respostas desse filtro em cada posição espacial. De maneira intuitiva, a rede aprenderá filtros que se ativam quando detectam algum tipo de característica visual, como uma borda de determinada orientação ou uma mancha de alguma cor na primeira camada, ou eventualmente padrões inteiros, como favos de mel ou padrões circulares em camadas mais altas da rede. Neste ponto, teremos um conjunto completo de filtros em cada camada de convolução (por exemplo, 12 filtros), e cada um deles produzirá um mapa de ativação bidimensional separado. A quantidade de filtros de uma camada convolucional determina a profundidade da camada seguinte, isto é, uma camada convolucional com 12 filtros gera uma saída cuja profundidade é 12 unidades (LI, 2023a). No treinamento de uma CNN, apenas os filtros são aprendidos automaticamente. No entanto, existem outros elementos que chamamos de hiperparâmetros, ou parâmetros configuráveis de uma rede. No caso das CNNs, os hiperparâmetros atrelados a uma camada convolucional são: tamanho do filtro, o número de filtros, o *stride* e o *padding*:

- *Stride*: define o passo com que o filtro é deslizado. Quando o stride é 1, os filtros são movidos um pixel de cada vez. Quando o stride é 2 os filtros saltam 2 pixels por vez enquanto deslizam. Isso produzirá volumes de saída menores espacialmente (LEMOS *et al.*, 2021).

- *Padding*: é uma técnica que consiste em adicionar linhas e colunas a cada lado do volume de entrada, para ajustar o centro de um núcleo no elemento mais externo e manter o mesmo plano de dimensão através da operação de convolução (Figura 22) (LEMOS *et al.*, 2021). Os valores utilizados para preencher estas colunas e linhas novas podem ser zero, o valor real adjacente ou ainda um espelhamento dos valores do volume original.

A Figura 22 representa de forma simplificada a operação de convolução. Note que as colunas e linhas verdes no volume de entrada são, na verdade, adicionadas artificialmente (*padding*) para que, após o processo de convolução, o volume de saída tenha o mesmo tamanho do volume de entrada. O filtro ou *kernel* (verde escuro) é deslizado por todo o volume de entrada (*stride* = 1) para gerar o mapa de características (saída). A convolução é uma operação linear, então sua saída (mapa de características) é passada por uma função de ativação para incluir não-linearidade ao sistema.

Figura 22 – Convolução com *padding*.



Fonte: (LEMOS *et al.*, 2021) (WANGENHEIM, 2019).

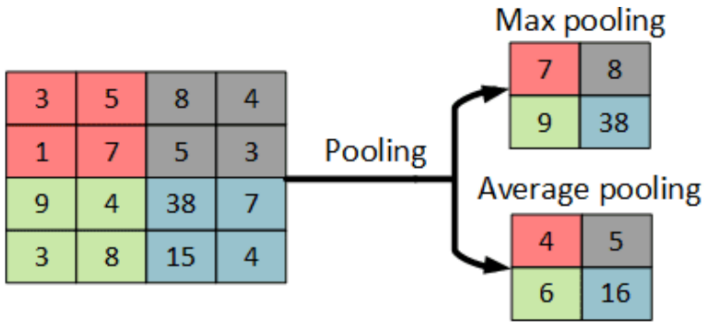
2.3.2.2 Camada de Pooling

A camada de *pooling* é outro elemento muito presente em CNNs. É comum inserir periodicamente uma camada de *pooling* entre camadas de convolução sucessivas

em uma arquitetura de CNN. Sua função é reduzir progressivamente o tamanho espacial da representação para diminuir a quantidade de parâmetros e cálculos na rede, também controlando o overfitting. A camada de Pooling opera independentemente em cada fatia de profundidade da entrada e a redimensiona espacialmente (LI, 2023a). Em outras palavras, a camada de *pooling* reduz a largura e altura de um volume de entrada, mantendo a profundidade igual.

Existem duas principais operações de *pooling*: *max pooling* e *average pooling*. Ambas as operações dividem a entrada em um conjunto de retângulos - tipicamente de tamanho 2×2 - e então computam a operação de *pooling*. Conforme indica a Figura 23, a operação *max pooling* seleciona o valor mais alto do retângulo, enquanto a operação *average pooling* seleciona o valor médio do retângulo.

Figura 23 – Exemplos de *max pooling* e *average pooling* com *stride* = 2 e filtro de tamanho 2×2 em uma entrada 4×4. O mapa de características resultante tem tamanho 2×2.



Fonte: (MAJIN ERAZO *et al.*, 2021).

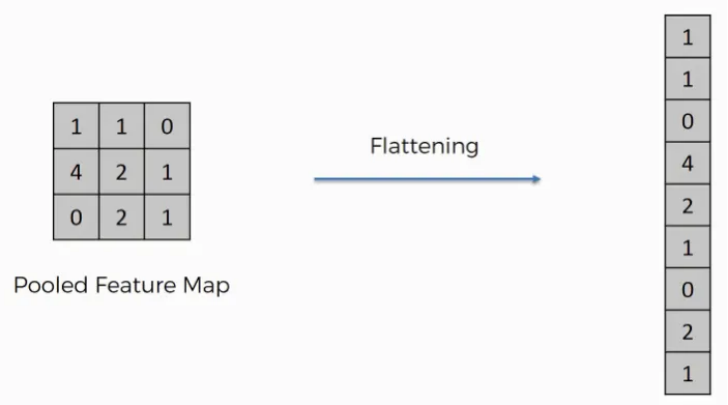
2.3.2.3 Camada totalmente conectada

A camada totalmente conectada de uma CNN se encontra, em geral, no final da rede. Essa camada tem como principal objetivo analisar os mapas de características criados anteriormente pelas camadas de convolução e *pooling* utilizando um perceptron multi camadas. Para conectar estes mapas de características com a camada totalmente conectada, uma operação chamada *flattening* precisa ser aplicada no volume de entrada (Figura 24).

2.3.2.4 U-Net

A U-Net é uma arquitetura de CNN que foi proposta por (RONNEBERGER; FISCHER; BROX, 2015). Esta arquitetura foi proposta exclusivamente para a tarefa de segmentação semântica em imagens e para isto não utiliza camadas totalmente conectadas. Como visto na Figura 25, esta arquitetura utiliza camadas de convolução, *max*

Figura 24 – Operação de *flattening*. O volume de entrada (tridimensional) é achatado em um vetor unidimensional. Este novo vetor é utilizado como entrada para um perceptron multi camadas.



Fonte: (ALI, 2022).

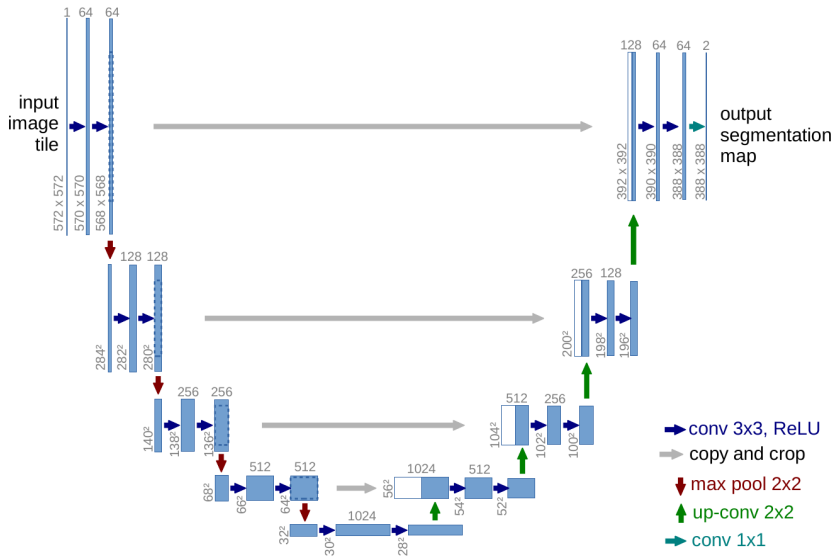
pooling, convolução transposta (*up-conv*) e cópia/corte de mapas de características. Nesse sentido, a U-Net é chamada na literatura de "rede completamente convolucional"(FCN - *Fully Convolutional Network*) ou ainda arquitetura EncoderDecoder (WU *et al.*, 2019), inspirada no trabalho realizado por (LONG; SHELHAMER; DARRELL, 2015).

Em contraste a FCN proposta por (LONG; SHELHAMER; DARRELL, 2015), a arquitetura U-Net apresenta maior quantia de canais de características no processo de *upsampling*, permitindo que a rede consiga propagar informações para camadas de maior resolução. Por conta deste fator, como podemos verificar na Figura 25, o diagrama desta arquitetura se assemelha com a letra "u", sendo esta a origem do seu nome. Também, para prever os pixels na região da borda da imagem, o contexto ausente é extrapolado refletindo a imagem de entrada (*padding com espelhamento*). Essa estratégia de mosaico é importante para aplicar a rede a imagens grandes, já que, caso contrário, a resolução seria limitada pela memória da GPU (RONNEBERGER; FISCHER; BROX, 2015). No artigo, os autores indicam que a U-Net apresenta bom desempenho em bases de dados com presença de *data augmentation*, que é o caso deste trabalho.

2.3.2.5 FastFCN

Diversas abordagens para segmentação semântica utilizam convoluções dilatadas para extrair mapas de características de alta resolução, o que traz uma complexidade computacional pesada e alto custo de memória computacional. Para substituir as convoluções dilatadas, que consomem tempo e memória, os autores da rede FastFCN (WU *et al.*, 2019) propõe um novo módulo de *upsampling* chamado de *Joint*

Figura 25 – Exemplo de arquitetura da U-Net. Cada bloco azul corresponde a um mapa de características. O número de canais (profundidade do mapa) está denotado acima dele. Largura (x) e altura (y) são mostradas no canto inferior esquerdo do bloco. Blocos brancos representam mapas de características copiados. Cada flecha representa uma operação diferente.

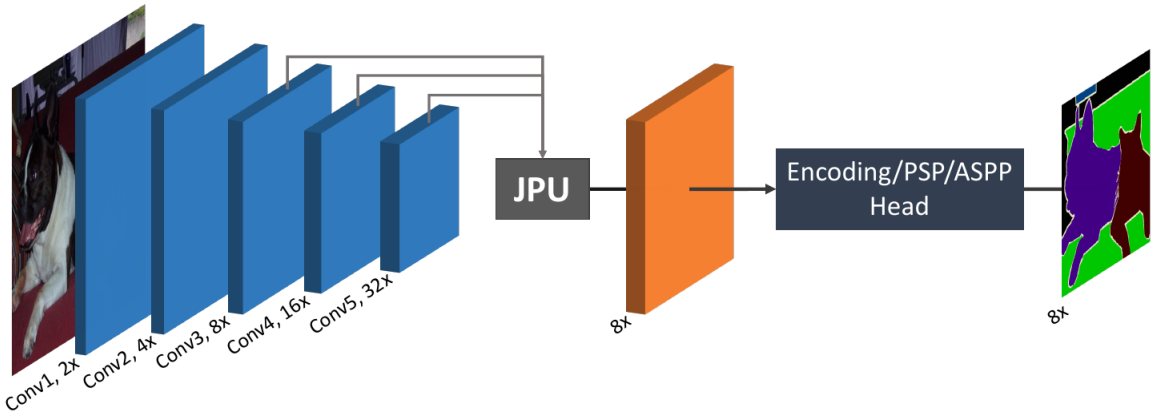


Fonte: (RONNEBERGER; FISCHER; BROX, 2015).

Pyramid Upsampling (JPU) responsável por extrair mapas de características de alta resolução. Utilizando o JPU, essa rede é capaz de reduzir a complexidade computacional em cerca de três vezes sem perda de desempenho, quando comparada com outros trabalhos similares como (LONG; SHELHAMER; DARRELL, 2015). Ao substituir convoluções dilatadas pelo módulo JPU proposto, os autores apresentam resultados estado-da-arte (no momento de sua publicação) no conjunto de dados Pascal Context (mIoU de 53,13%) (MOTTAGHI *et al.*, 2014) e no conjunto de dados ADE20K (pontuação final de 0,5584) (ZHOU *et al.*, 2017) (ZHOU *et al.*, 2019) com tempo de inferência até três vezes menor em comparação com a FCN original.

De modo geral, a FastFCN mantém o mesmo *backbone* da FCN original. Conforme indica a Figura 26, após o *backbone* (elementos convolucionais em azul), os autores incluem o elemento JPU (Figura 27) que utiliza os resultados das últimas 3 convoluções para criar uma mapa de caractrísticas de alta resolução. No elemento JPU, cada uma das três entradas é processada por um elemento de convolução, então passa pelo processo de *upsampling* e é concatenada com as outras. Nesse ponto, temos um volume y_c que é utilizado como entrada para quatro convoluções independentes (S-CONV) (HOWARD *et al.*, 2017) (CHOLLET, 2017) com taxas de dilatação diferentes (D = 1, 2, 4 e 8). Os resultados das convoluções dilatadas são concatenados e esse novo volume é utilizado como entrada para uma última convolução que gera o

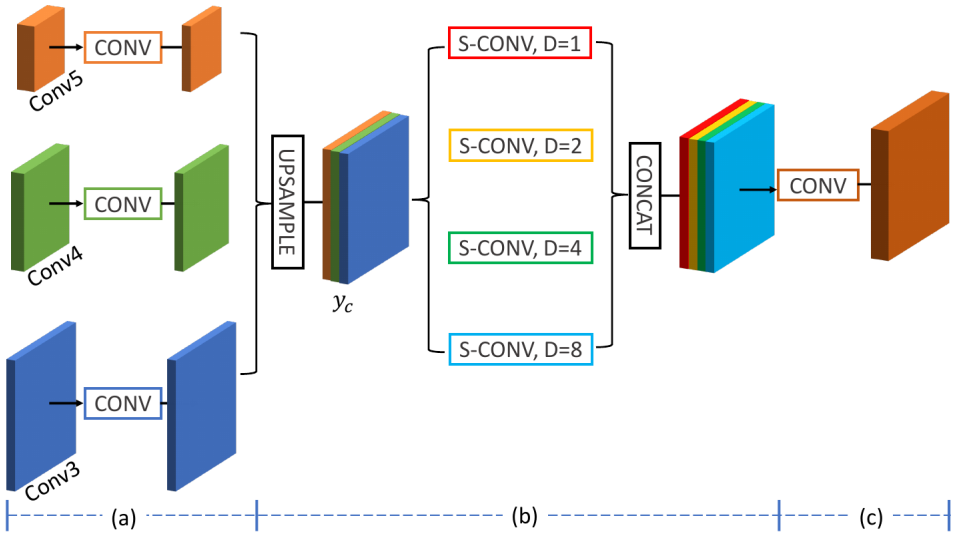
Figura 26 – Visão superficial da rede FastFCN.



Fonte: (WU *et al.*, 2019).

mapa de características final com alta resolução. Este mapa então deve ser entregue a um módulo especializado para produzir os resultados finais de segmentação semântica, como o PSP (ZHAO *et al.*, 2017), ASPP (CHEN *et al.*, 2017) ou ainda *Context Encoding* (ZHANG *et al.*, 2018).

Figura 27 – *Joint Pyramid Upsampling* (JPU).



Fonte: (WU *et al.*, 2019).

2.3.3 Transformers

O Transformer (LIN, T. *et al.*, 2022a) é um modelo proeminente de aprendizado profundo que foi amplamente adotado em vários campos, como processamento

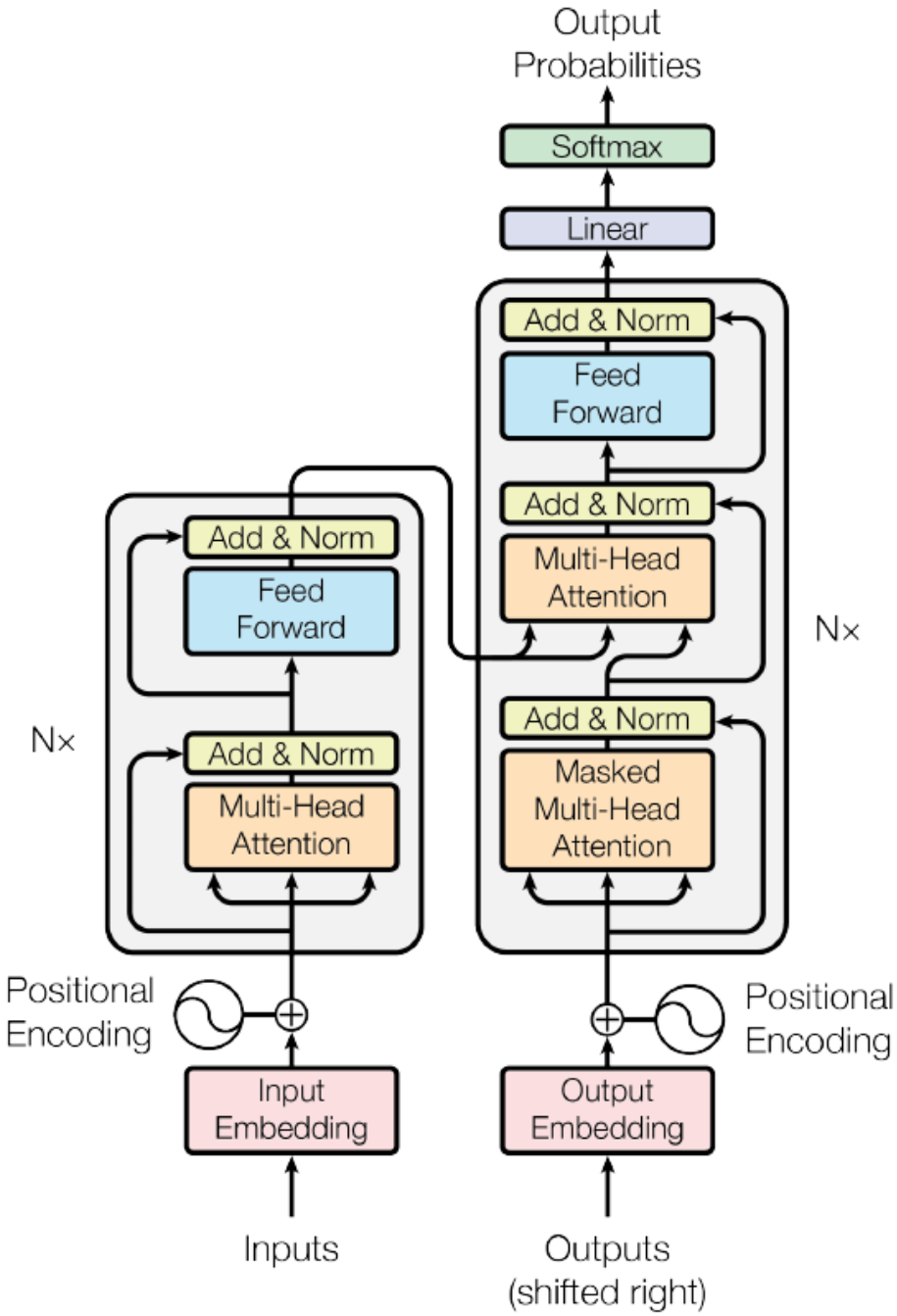
de linguagem natural (NLP), visão computacional (CV) e processamento de voz. No campo da visão computacional, trabalhos que exemplificam o uso dos Transformers são (PARMAR *et al.*, 2018) na geração de imagens, (CARION *et al.*, 2020) na detecção de objetos, (DOSOVITSKIY *et al.*, 2021) no reconhecimento de imagens (LIN, T. *et al.*, 2022b) e na segmentação semântica em imagens (XIE *et al.*, 2021) (CHU *et al.*, 2021) (LIU, Z. *et al.*, 2021). Os resultados promissores do Transformer no campo do processamento de linguagem natural fez que com diversas pesquisas propusessem variações desta arquitetura de diversas formas, como:

- **Eficiência do Modelo.** Um desafio importante na aplicação do Transformer é sua ineficiência no processamento de sequências longas, principalmente devido à complexidade computacional ($O(n^2)$) e de memória do módulo de autoatenção. Os métodos de melhoria incluem atenção leve (por exemplo, variantes de atenção esparsa) e métodos de dividir e conquistar (por exemplo, mecanismos recorrentes e hierárquicos) (LIN, T. *et al.*, 2022b).
- **Generalização do Modelo.** Como o transformer é uma arquitetura flexível e faz poucas suposições sobre o viés estrutural dos dados de entrada, é difícil treinar em dados em pequena escala. Neste sentido, o Transformer é frequentemente chamado de *data hungry* ou faminto por dados. Os métodos de melhoria incluem a introdução de viés estrutural ou regularização, pré-treinamento em dados não rotulados em grande escala, entre outros (LIN, T. *et al.*, 2022b).
- **Adaptação da Arquitetura.** A adaptação do modelo visa ajustar o Transformer para tarefas e aplicações específicas (LIN, T. *et al.*, 2022b), como visão computacional, processamento de voz e outros.

O modelo padrão dos Transformers (*vanilla* Transformer) apresentado na Figura 28 é um modelo criado para o desafio de sequências-para-sequências (*sequence-to-sequence*) descrito em detalhes por (SUTSKEVER; VINYALS; LE, 2014). Para solucionar este problema, (LIN, T. *et al.*, 2022a) criaram um modelo baseado na ideia de codificar (*encode*) e decodificar (*decode*) informações. Nesse sentido, o Transformer padrão é na verdade uma sequência de N blocos idênticos. Cada bloco de *encoder* é composto principalmente por um elemento chamado de *multi-head self-attention* e uma rede *feed-forward* - por exemplo o MLP - (LIN, T. *et al.*, 2022b). Os blocos *decoder*, por outro lado, inserem adicionalmente módulos de atenção cruzada (*cross-attention*) entre os módulos de *multi-head self-attention* e as redes *feed-forward* (FFNs). Além disso, os módulos de *self-attention* no *decoder* são adaptados para evitar que cada posição atenda a posições subsequentes (LIN, T. *et al.*, 2022b) (LIN, T. *et al.*, 2022a).

Naturalmente, esta arquitetura do Transformer padrão (Figura 28) foi proposta para a tarefa de processamento natural de linguagem e não está apta para realizar o processamento de imagens. Neste sentido, a 2.3.3.1 trás informações de como o

Figura 28 – Arquitetura padrão do Transformer (*vanilla* Transformer) para processamento de linguagem natural.



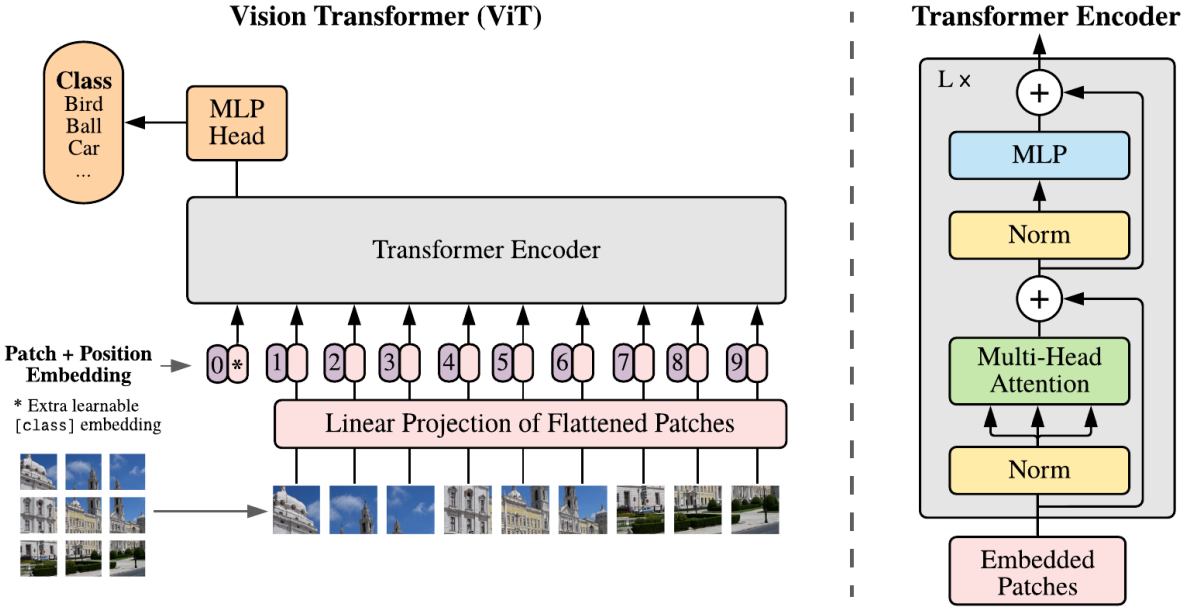
Fonte: (LIN, T. *et al.*, 2022a).

Transformer pode ser adaptado para tarefas no campo da visão computacional, abordando tanto o pré-processamento da imagens quanto os elementos do Transformer e algumas arquiteturas para a segmentação semântica de imagens.

2.3.3.1 *Vision Transformers*

Inspirados pelos sucessos de escalabilidade do Transformer em processamento de linguagem natural, (DOSOVITSKIY *et al.*, 2021) aplicaram um Transformer padrão diretamente a imagens, com o menor número possível de modificações (Figura 29). Para fazer isso, as imagens são divididas em *patches*, passam por um processo de projeção linear e então recebem incorporações posicionais (*positional embedding*). Os *patches* da imagem são tratados da mesma forma que os *tokens* (palavras) em uma aplicação de processamento de linguagem natural. Estes *patches* tratados são então enviados para o bloco Transformer *Encoder*, então a saída deste processo é enviada para uma camada totalmente conectada (MLP) realizar a classificação.

Figura 29 – Visão geral da arquitetura do *Vision* Transformer.



Fonte: (DOSOVITSKIY *et al.*, 2021).

Os *Vision* Transformers (ViTs) são muito flexíveis nas tarefas de visão computacional (DOSOVITSKIY *et al.*, 2021). De certo modo, os ViTs funcionam como um codificador-decodificador (*encoder-decoder*). O *encoder* processa a sequência de entrada passo a passo (ou elemento por elemento) e captura informações relevantes em cada etapa. Pode ser uma rede neural recorrente (RNN), uma rede neural convolucional (CNN) ou neste caso uma variação de Transformer. O *decoder* por outro lado recebe a representação intermediária gerada pelo *encoder* e a utiliza como contexto para gerar a sequência de saída desejada. Assim como o *encoder*, pode ser uma RNN, CNN ou um módulo baseado em Transformers. Neste sentido, podemos identificar os elementos principais do Transformer na visão computacional como *patch embedding*,

Tabela 1 – Modelos de ViTs disponíveis pelos autores.

Modelo	Camadas (*Encoders*)	Projeção *D*	Tamanho do MLP	MSA *Heads*	Parâmetros
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Fonte: (DOSOVITSKIY *et al.*, 2021).

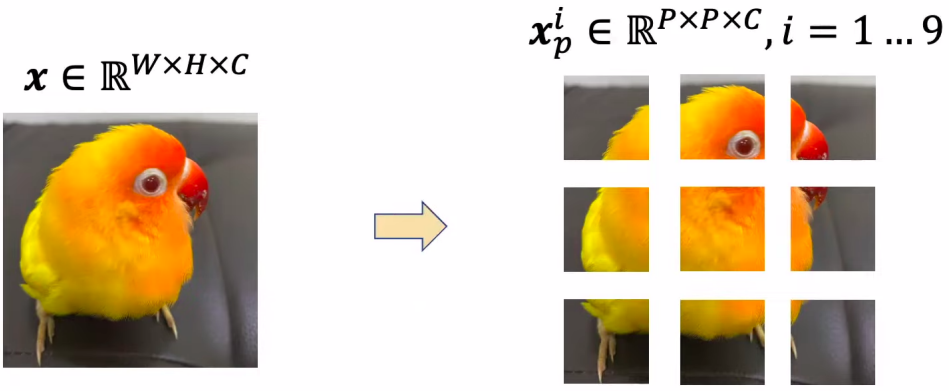
positional encoding e módulo de anteção. O resto da arquitetura é, em geral, específica para cada aplicação. Os Transformers, neste trabalho, são utilizados para a tarefa de segmentação semântica, então três arquiteturas específicas serão apresentadas na Seção 2.3.3.7, Seção 2.3.3.5 e Seção 2.3.3.6.

O ViT de (DOSOVITSKIY *et al.*, 2021) é apresentado em diversos tamanhos, assim como diversos outros modelos encontrados na literatura (LIU, Z. *et al.*, 2021) (CHU *et al.*, 2021) (XIE *et al.*, 2021). A Tabela 1 apresenta algumas opções disponíveis.

2.3.3.2 *Patch Embedding*

O Transformer padrão (Figura 28) recebe como entrada uma sequência unidimensional de *tokens*. Para manusear imagens, que são tipicamente elementos bidimensionais (2D) ou tridimencionais (3D), a camada de *patch embedding* realiza algumas transformações. Inicialmente a imagem $x \in \mathbb{R}^{W\times H\times C}$ é transformada em uma sequência de imagens menores (*patches*) $x_p^i \in \mathbb{R}^{P\times P\times C}, i = 1, ..., 9$ conforme indica a Figura 30 (ATIENZA, 2022) (DOSOVITSKIY *et al.*, 2021). Neste caso, (W,H) é o par largura e altura da imagem original, C é a quantidade de canais de cor ($C = 1$ para escala de cinza e $C = 3$ para escala RGB) e P é a largura de um *patch*.

Figura 30 – Esquema de divisão da imagem original (esquerda) em *patches* (direita).



Fonte: (ATIENZA, 2022).

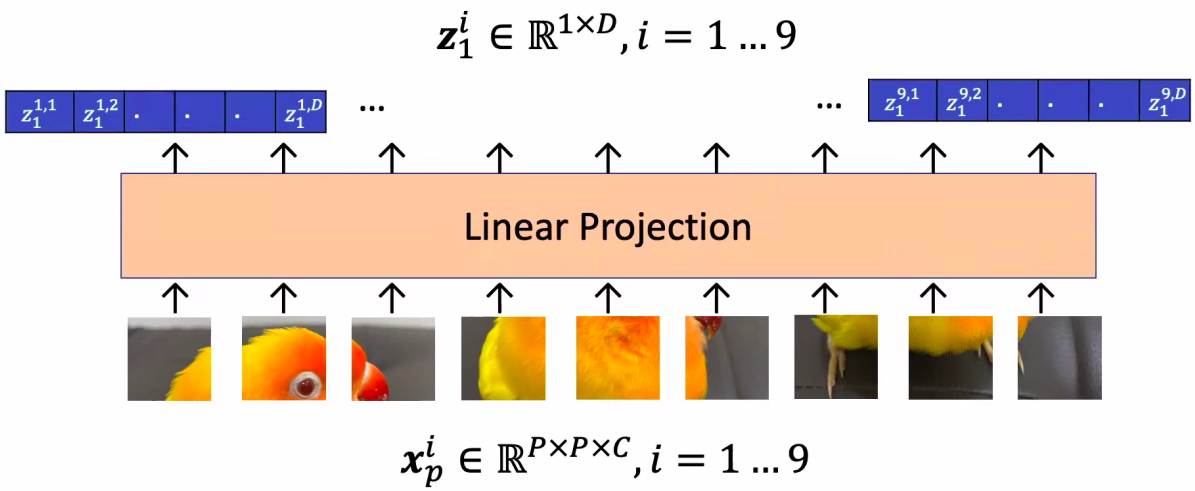
Os *patches* são então organizados em ordem, do topo superior esquerdo para o canto inferior direito, de forma sequêncial como na Figura 31. Cada um dos *patches*

é achatado em um vetor unidimensional (Equação (4)) e então multiplicado por uma matriz $W \in \mathbb{R}^{P^2C \times D}$ para ser linearmente projetado em um vetor $z_1^i \in \mathbb{R}^{1 \times D}$, onde D é o tamanho do vetor no qual os *patches* são projetados, seguindo a Equação (5). Esta operação pode ser feita através de um MLP sem *bias* ou através de uma camada convolucional com D filtros W_D, onde cada filtro tem tamanho $P \times P$ e o valor de *stride* é P. É importante ressaltar que os parâmetros da projeção linear são parâmetros aprendíveis.

$$x_p^i \in \mathbb{R}^{P \times P \times C} \rightarrow x_p^i \in \mathbb{R}^{1 \times P^2 C} \tag{4}$$

$$x_p^i \in \mathbb{R}^{1 \times P^2 C}.W \in \mathbb{R}^{P^2 C \times D} \rightarrow z_1^i \in \mathbb{R}^{1 \times D} \tag{5}$$

Figura 31 – Operação de achatamento e projeção linear dos *patches*.



Fonte: (ATIENZA, 2022).

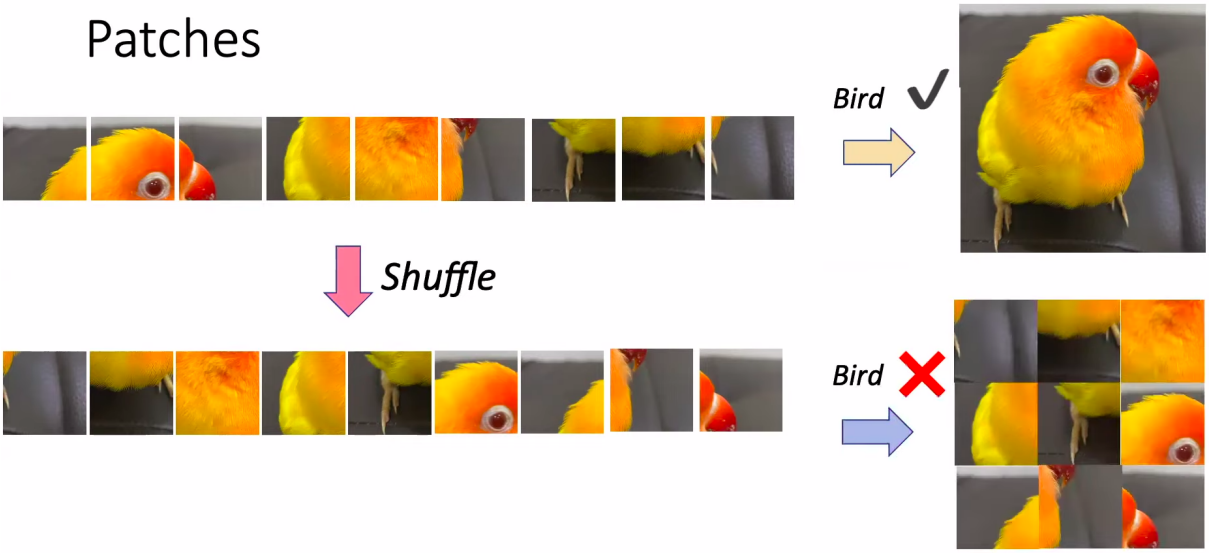
2.3.3.3 *Positional Encoding*

Como visto em (ATIENZA, 2022) e (KAINZ, 2020), as redes CNNs possuem operações equivariantes (convolução) e operações invariantes (*pooling*) que são muito importantes nas tarefas de classificação, detecção de objetos e segmentação semântica em imagens. Essas propriedades permitem que a rede seja razoavelmente resistente a pequenas modificações de posição, escala e rotação dos objetos nas imagens. Em outras palavras, uma rede com ambas estas propriedades é sensível a posição relativa dos *pixels* em uma imagem.

No entanto, os blocos padrão de Transformer não possuem operações equivariantes (ATIENZA, 2022). A operação *positional encoding* é inserida no modelo justamente para somar esta propriedade. Apesar de existirem alguns algoritmos de

positional encoding, (DOSOVITSKIY *et al.*, 2021) concluem em seu trabalho que não houveram ganhos ao usar abordagens mais complexas. Neste sentido, a abordagem escolhida pelos autores é aquela onde cada projeção linear de um *patch* $z_1^i \in \mathbb{R}^{1\times D}$ é somada a um vetor $p^i \in \mathbb{R}^{1\times D}$, onde os elementos deste vetor são aprendíveis durante o treinamento. O objetivo dos vetores p^i é inserir a propriedade de equivariância na rede, melhorando seu desempenho em tarefas de visão computacional onde a ordem dos *pixels* ou *patches* é extremamente importante (Figura 32). Note que, como indica a Figura 32, no caso de baixo (*patches* misturados) não é desejável que a rede classifique a imagem como pássaro.

Figura 32 – Resultado do ViT sem o uso de *positional encoding*. Este resultado se deve ao fato de que, sem o *positional enconding*, a ordem dos *patches* é irrelevante para a rede. Em ambos os casos a rede irá classificar a imagem como "pássaro".



Fonte: (ATIENZA, 2022).

Note que supor que a ordem dos *pixels* é importante para as tarefas de visão computacional é um viés indutivo. Por sinal, no ViT de (DOSOVITSKIY *et al.*, 2021), este é o único viés indutivo, isto é, direciona o modelo a generalizar de uma forma específica.

2.3.3.4 Atenção

Nos *Vision* Transformers (ViTs), o mecanismo de atenção desempenha um papel fundamental na captura de relações espaciais entre os *patches* de uma imagem. Ele permite que o modelo entenda as interações entre diferentes partes da imagem, considerando a dependência entre os *patches*, e é crucial para a capacidade do ViT de processar efetivamente informações visuais.

Uma função de atenção pode ser descrita como uma mapeamento de uma consulta (query) Q e um conjunto de pares chave-valor K e V para uma saída, onde a consulta, chaves (keys), valores e saída são todos vetores. A saída é calculada como a soma ponderada dos valores, onde o peso atribuído a cada valor é calculado por uma função de compatibilidade entre a consulta e a chave correspondente (LIN, T. *et al.*, 2022a). A formulação geral do mecanismo de atenção, ou *self-attention*, está apresentada na Equação (6). Podemos interpretar as variáveis da seguinte forma:

- *Query* (Q): Matriz de características que estamos interessados;
- *Key* (K): Matriz de características que podem ser relevantes para as características que estamos interessados;
- *Value* (V): Matriz de características originais que serão escaladas pelas probabilidades geradas pela função *softmax*;
- d_k: Fator de normalização. Este fator pode ser igual à dimensão do vetor z, que neste caso é D, ou no caso de (DOSOVITSKIY *et al.*, 2021) D/k, onde k é a quantidade de operações de *self-attention* acontecendo em paralelo;
- *softmax*: Uma função de ativação utilizada para transformar o produto escalar das matrizes Q e K em probabilidades.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{6}$$

Para calcular as matrizes Q, K e V, considere os exemplos anteriores da Seção 2.3.3.2 e Seção 2.3.3.3, onde $N = \frac{HW}{P^2}$ é o número total de *patches* em um modelo de ViT. No ViT de (DOSOVITSKIY *et al.*, 2021), após o processo de *positional encoding*, temos z^{N+1} vetores, sendo que o vetor z^0 é um *token* especial para o caso de classificação e os vetores z^1,z^2,...,z^N representam os *patches*. Assim, as matrizes Q, K e V são calculadas utilizando as equações abaixo.

$$Query : Q = [z_1^0, z_1^1, \ldots, z_1^N]W^Q \tag{7}$$

$$Key : K = [z_1^0, z_1^1, \ldots, z_1^N]W^K \tag{8}$$

$$Value : V = [z_1^0, z_1^1, \ldots, z_1^N]W^V \tag{9}$$

As matrizes W^Q, W^K e W^V são matrizes de parâmetros aprendíveis durante o treinamento. A dimensionalidade destas matrizes é $W^{QKV} \in \mathbb{R}^{D\times J}$, onde J é um hiperparâmetro da rede. Em (DOSOVITSKIY *et al.*, 2021), os autores adotam $J = 3D_h$, com $D_h = D/k$ e k é a quantidade de operações de *self-attention* paralelas.

Quando existem operações de *self-attention* acontecendo em paralelo, chamamos esta operação de *multi-head self-attention* (MSA). A MSA é uma extensão da operação *self-attention* (SA) na qual executamos k operações de *self-attention*, chamadas de "*heads*", em paralelo, e projetamos suas saídas concatenadas. Para manter a computação e o número de parâmetros constantes ao alterar k, geralmente define-se $D_h = \frac{D}{k}$. O bloco de MSA é calculado conforme a equação abaixo. Na Equação (10), a matriz de parâmetros aprendíveis que concatena as várias operações de *self-attention* tem dimensionalidade $W^{msa} \in \mathbb{R}^{kD_h \times D}$ e todos os parâmetros são aprendíveis.

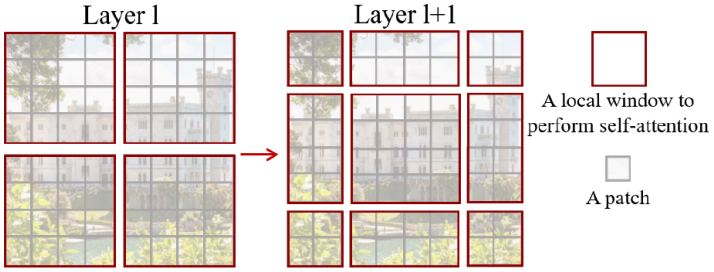
$$MSA(z) = [SA_1(z); SA_2(z); \ldots; SA_k(z)]W^{msa} \tag{10}$$

2.3.3.5 Swin Transformer

O objetivo do Swin Transformer é servir como uma estrutura *backbone* de propósito geral para problemas de visão computacional. Em suma, o Swin Transformer é proposto para ser uma camada de extração de características que serão decodificadas, de acordo com cada problema, para gerar os resultados desejados (LIU, Z. *et al.*, 2021).

A principal novidade que o Swin Transformer apresenta é o uso de janelas de atenção deslocadas (*shifted windows*) entre diferentes blocos de Transformer (Figura 33). As janelas deslocadas conectam as janelas da camada anterior, proporcionando conexões entre elas que aumentam significativamente a acurácia da rede (LIU, Z. *et al.*, 2021).

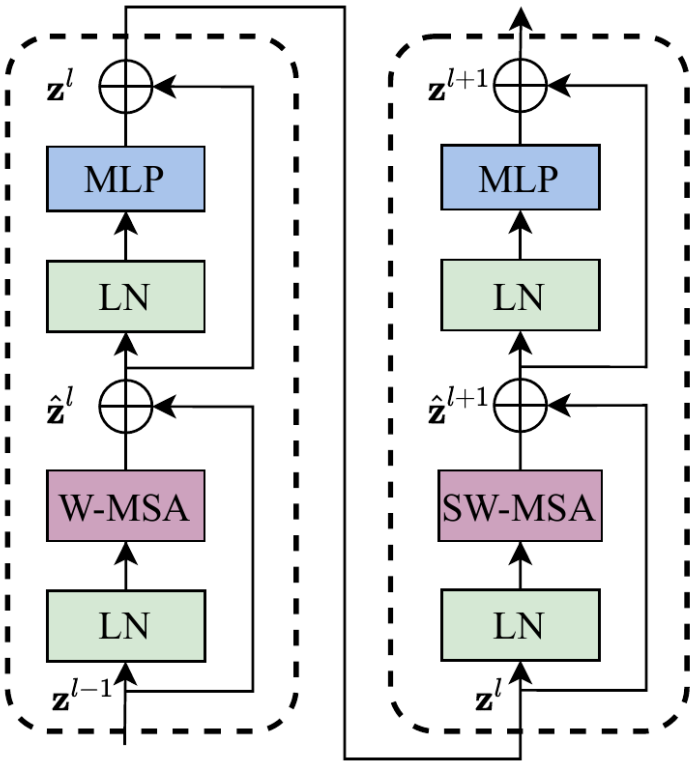
Figura 33 – Uma ilustração da abordagem de janela deslocada para calcular a operação *self-attention* na arquitetura proposta do Swin Transformer. Na camada *l* (esquerda), um esquema regular de particionamento de janelas é adotado, e a *self-attention* é calculada dentro de cada janela. Na próxima camada *l + 1* (direita), o particionamento da janela é deslocado, resultando em novas janelas. O cálculo da *self-attention* nas novas janelas atravessa os limites das janelas anteriores na camada *l*, fornecendo conexões entre elas (LIU, Z. *et al.*, 2021).



Fonte: (LIU, Z. *et al.*, 2021).

A Figura 34 apresenta a principal mudança interna nos blocos de Transformer deste modelo. Note que o bloco de MSA é substituídos por dois outros: W-MSA (*windowed multi-head self-attention*)) e SW-MSA (*shifted windowing multi-head self-attention*). A operação de *self-attention* é computada apenas nas janelas locais, conforme Figura 33. Esta abordagem reduz a complexidade computacional do Transformer.

Figura 34 – Dois blocos do Swin Transformer conectados em sequência.



Fonte: (LIU, Z. *et al.*, 2021).

Para os dois blocos de Transformer da Figura 34, os vetores de características z^l e z^{l+1} são calculados de acordo com as Equações (11),(12),(13) e (14). Nestas equações, W-MSA e SW-MSA significam, respectivamente, as operações de *multi-head self-attention* com janela fixa e janela deslocada. A sigla LN representa uma camada de normalização.

$$\hat{z}^{l} = W - MSA(LN(z^{l-1})) + z^{l-1} \tag{11}$$

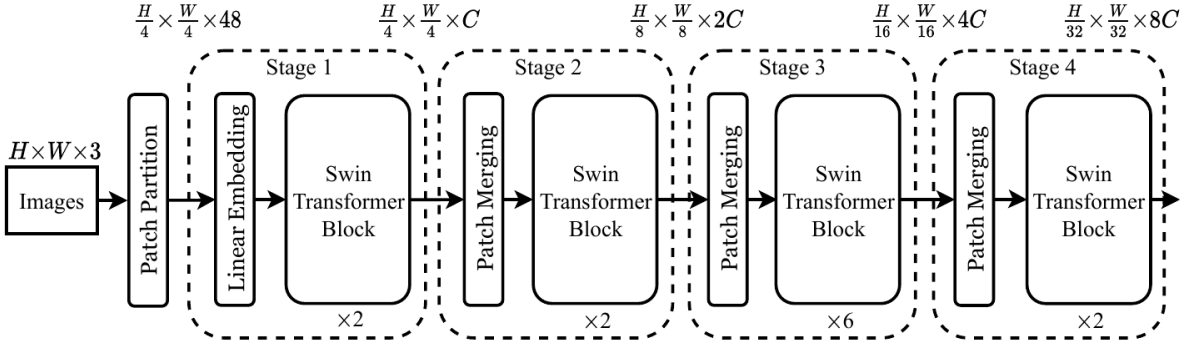
$$z^{l} = MLP(LN(\hat{z}^{l})) + \hat{z}^{l} \tag{12}$$

$$\hat{z}^{l+1} = SW - MSA(LN(z^{l})) + z^{l} \tag{13}$$

$$z^{l+1} = MLP(LN(\widehat{z}^{l+1})) + \widehat{z}^{l+!} \tag{14}$$

Similar ao ViT, o Swin Transformer também possuí diversos tamanhos. A Figura 35 apresenta a arquitetura do Swin-T, que é a menor disponível. As outras possíveis arquiteturas são Swin-S (pequeno), Swin-B (grande) e Swin-L (muito grande). No momento de sua publicação, o Swin Transformer atingiu resultados estado-da-arte nas bases de dados COCO (LIN, T.-Y. *et al.*, 2015) e ADE20K (ZHOU *et al.*, 2017) (ZHOU *et al.*, 2019).

Figura 35 – Arquitetura do Swin-T. Um dos modelos propostos pelos autores.



Fonte: (LIU, Z. *et al.*, 2021).

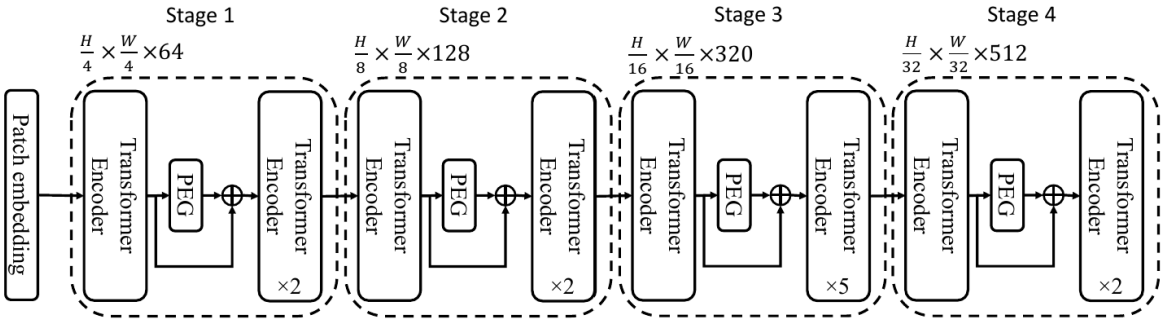
2.3.3.6 Twins

Em (CHU *et al.*, 2021) os autores revisitam o *design* da atenção espacial em Transformers de visão computacional (ViTs). A primeira descoberta é que a atenção global subamostrada em PVT (WANG, Wenhai *et al.*, 2021) é altamente eficaz e, com *positional embeddings* aplicáveis (CHU *et al.*, 2023), seu desempenho pode ser equivalente ou até melhor do que transformers de visão de ponta (por exemplo, Swin). Desta forma, os autores propõe uma primeira arquitetura, denominada Twins-PCPVT. No entanto, como mencionado anteriormente, ViTs sofrem muito com a complexidade da operação de *self-attention*. Para isto, a arquitetura Twins também é proposta utilizando um mecanismo chamado de *spatially separable self-attention* (SSSA). O mecanismo SSSA pode ser decomposto em outros dois, chamados de *locally-grouped self-attention* (LSA) e *global sub-sampled attention* (GSA). Esta outra alternativa é chamada de Twins-SVT.

Neste trabalho a arquitetura Twins-PCPVT é utilizada (Figura 36). Neste modelo, os autores utilizam a codificação posicional condicional (CPE), proposta no CPVT (CHU *et al.*, 2023), para substituir a codificação posicional absoluta no PVT. O gerador de codificação posicional (PEG), que gera a CPE, é colocado após o primeiro bloco de

encoder de cada estágio. Aqui é utilizada a forma mais simples de PEG, isto é, uma convolução 2D. O Twins-PCPVT herda as vantagens tanto do PVT (WANG, Wenhai *et al.*, 2021) quanto do CPVT (CHU *et al.*, 2023). Com estas vantagens, o Twins-PCPVT tem a possibilidade de atingir resultados parecidos com o Swin (2.3.3.5) fornecendo tempos de inferência muito menores.

Figura 36 – Arquitetura do Twins-PCPVT-S. Um dos modelos propostos pelos autores.



Fonte: (CHU *et al.*, 2021).

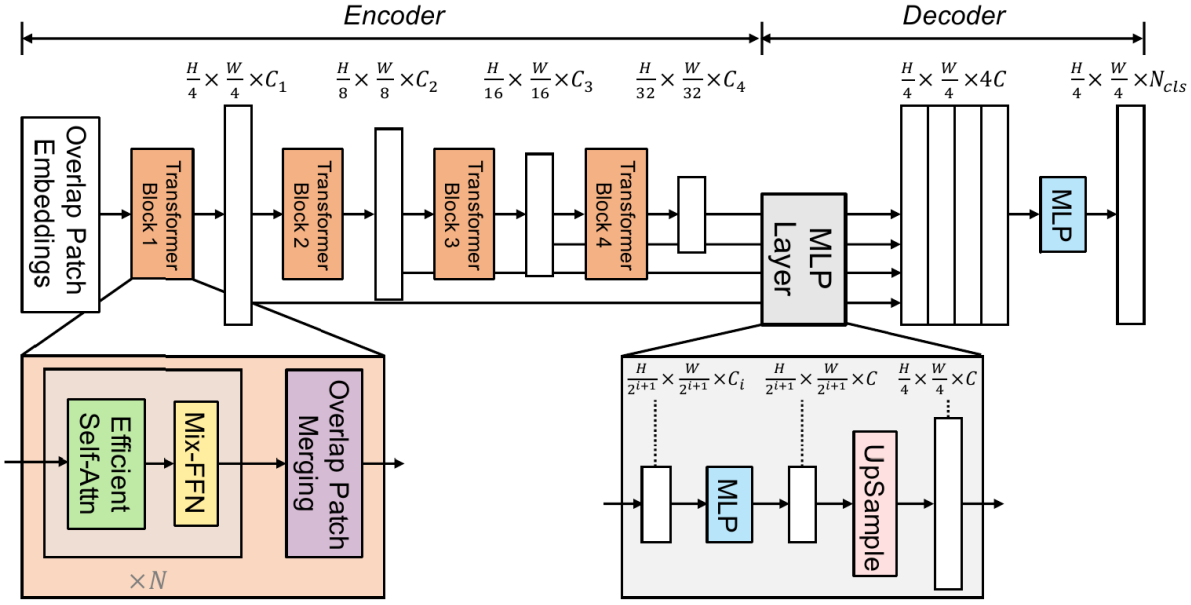
2.3.3.7 SegFormer

O SegFormer é um modelo de segmentação semântica simples, eficiente e poderoso que une Transformers com *decoders* de perceptron multicamadas (MLP) leves. O SegFormer apresenta duas características interessantes: 1) Ele compreende um *encoder* Transformer estruturado de forma hierárquica que produz características de múltiplas escalas. Este modelo não precisa de *positional encoding*, evitando assim a interpolação que leva a uma diminuição de desempenho quando a resolução de teste difere do treinamento. 2) O SegFormer evita *decoders* complexos. O *decoder* MLP proposto agrega informações de diferentes camadas, combinando assim atenção local e atenção global para criar as representações (XIE *et al.*, 2021). Veja a visão geral do SegFormer na Figura 37.

A Figura 37 indica que o processo de *patch embedding* agora é chamado de *overlap patch embedding*, e isto acontece porque no SegFormer existe uma sobreposição parcial entre os *patches*, ou seja, alguns *patches* compartilham *pixels*. O tamanho do *patch* no SegFormer também é menor do que no ViT de (DOSOVITSKIY *et al.*, 2021), com tamanho 4×4. Esta mudança, segundo (XIE *et al.*, 2021), influencia muito na tarefa de segmentação semântico, visto que ela precisa de mais informações por ser uma tarefa de predição densa.

O mecanismo de atenção utilizado no SegFormer é chamado de *Efficient Self-Attention*. Como a atenção é uma operação com complexidade computacional $O(n^2)$, este fator torna a inferência muito custosa em imagens de alta resolução. Neste sentido,

Figura 37 – Visão geral do modelo SegFormer.



Fonte: (XIE *et al.*, 2021).

a estratégia adotada pelo SegFormer é reduzir a dimensão do vetor K utilizando o método proposto por (WANG, Wenhai *et al.*, 2021) . Essa operação reduz a complexidade computacional da operação de *self-attention* para $O(\frac{n^2}{R})$, onde R é um hiperparâmetro.

Outra mudança em relação ao ViT é o bloco chamado de Mix-FFN. Os autores (XIE *et al.*, 2021) argumentam que, na verdade, a operação *positional enconding* não é necessária para a tarefa de segmentação semântica. Nesse sentido, a entrada do bloco de Transformer do SegFormer não utiliza esta operação e, ao invés, aplica os conceitos vistos em (ISLAM; JIA; BRUCE, 2020). Para superar o uso do *positional encoding*, o bloco Mix-FFN opera sobre a saída do bloco de *self-attention* seguindo a Equação (15)

$$x_{out} = MLP(GELU(Conv_{3\times3}(MLP(x_{in})))) + x_{in} \tag{15}$$

Onde x_{in} é o vetor de características oriundo do bloco de *self-attention* e GELU é uma função de ativação proposta por (HENDRYCKS; GIMPEL, 2023). Nos experimentos de (XIE *et al.*, 2021), é evidente que nesta aplicação a operação $Conv_{3\times3}$ é suficiente para fornecer informações de posição para o Transformer.

Por fim, o *decoder* utilizado no SegFormer é apresentado como um simples MLP, que é constituído por quatro principais passos. Inicialmente, seguindo o exposto por (XIE *et al.*, 2021), características F_i ($i = 1,\ldots,4$) de vários níveis oriundas dos *encoders* atravessam um bloco de MLP para unificar a dimensão.

$$\widehat{F}_i = MLP(C_i, C)(F_i), \forall i \tag{16}$$

Então, esses vetores de características são superdimensionados para 1/4 do tamanho original da imagem.

$$\widehat{F}_i = Upsample(\frac{W}{4} \times \frac{h}{4})(\widehat{F}_i), \forall i \tag{17}$$

Na sequência, outro MLP é utilizado para fundir os vetores de características concatenados.

$$F = MLP(4C, C)(Concat(\widehat{F}_i)), \forall i \tag{18}$$

E por fim um último MLP é utilizado para prever a máscara de segmentação semântica M com resolução $\frac{H}{4} \times \frac{W}{4} \times N_{cls}$, onde N_{cls} é o número de classes diferentes no problema de segmentação proposto.

$$M = MLP(C, N_{cls})(F) \tag{19}$$

Na Figura 38 é possível ver o desempenho do SegFormer em relação a outros modelos de segmentação semântica. Note que, na comparação feita pelos autores, o SegFormer atinge resultados estado-da-arte na base de dados ADE20K ((ZHOU *et al.*, 2017) (ZHOU *et al.*, 2019)).

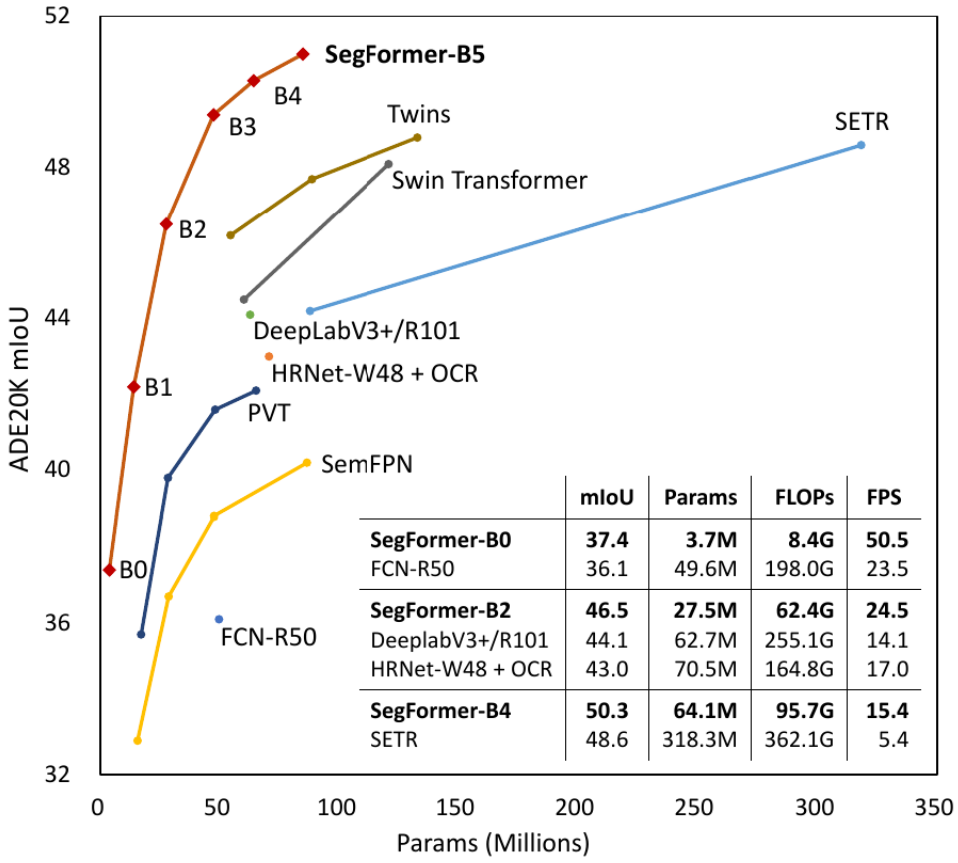
2.3.4 Funções de perda e custo

Existem dois termos que são amplamente encontrados e discutidos no âmbito do aprendizado supervisionado: função de perda e função de custo. A função de custo usada por um algoritmo de aprendizado de máquina frequentemente se decompõe como uma soma sobre exemplos de treinamento de alguma função de perda (GOODFELLOW; BENGIO; COURVILLE, 2016). No entanto, ambos os termos são amplamente encontrados de forma intercambiável na literatura.

Suponha que exista uma base de dados com m pares de treinamento $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), ..., (x^{(m)}, y^{(m)})\}$, onde $x^{(i)}$ é uma entrada do sistema e $y^{(i)}$ é a saída desejada para essa entrada. Se Θ é o conjunto de parâmetros aprendíveis de uma rede neural, então podemos criar uma função $L(x^{(m)}, y^{(m)}, \Theta)$ (função *Loss* ou função perda) que calcula o 'erro' da rede neural seguindo alguma métrica específica para um par entrada-saída. A função de custo, como visto anteriormente, é na verdade uma soma dos resultados da função *Loss* ao longo de toda base de treinamento (Eq. (20)).

$$J(\Theta) = \frac{1}{m} \sum_{m}^{i=1} L(x^{(i)}, y^{(i)}, \Theta) \tag{20}$$

Figura 38 – Resultados apresentados no artigo (XIE *et al.*, 2021) na base de dados ADE20K.



	mIoU	Params	FLOPs	FPS
SegFormer-B0	**37.4**	**3.7M**	**8.4G**	**50.5**
FCN-R50	36.1	49.6M	198.0G	23.5
SegFormer-B2	**46.5**	**27.5M**	**62.4G**	**24.5**
DeeplabV3+/R101	44.1	62.7M	255.1G	14.1
HRNet-W48 + OCR	43.0	70.5M	164.8G	17.0
SegFormer-B4	**50.3**	**64.1M**	**95.7G**	**15.4**
SETR	48.6	318.3M	362.1G	5.4

Fonte: (XIE *et al.*, 2021).

Neste caso, basta definir o formato da função *Loss* para que seja possivel calcular o 'erro' da rede neural em toda a base de treinamento. Naturalmente, o formato dessa função depende de cada aplicação, como veremos à seguir.

2.3.4.1 Erro Quadrático Médio (MSE)

A função de custo *Mean Square Error* (MSE), ou Erro Quadrático Médio, é uma métrica amplamente usada para medir a diferença entre os valores previstos por um modelo e os valores reais dos dados (RUSSELL; NORVIG, 2021) (GOODFELLOW; BENGIO; COURVILLE, 2016) (CHOLLET, 2021). Essa métrica é principalmente aplicada em problemas de regressão. Uma possível implementação da MSE está apresentada à seguir:

$$MSE = \frac{1}{m}\sum_{m}^{i=1}(y^{(i)} - \widehat{y}^{(i)})^2 \tag{21}$$

Onde m é o tamanho da base de dados, y é o valor desejável de saída e $\widehat{y}$ é a

resposta gerada pela rede neural. Note que $\widehat{y}$ pode também ser expresso como uma função de Θ e $x^{(i)}$.

2.3.4.2 Entropia Cruzada (Cross-Entropy)

A função de custo Entropia Cruzada, também chamada de Log *Loss* ou *Cross-Entropy*, é uma métrica utilizada amplamente em problemas de classificação multi-classe. É uma extensão da função de custo *Binary Cross-Entropy* para o cenário em que há mais de duas classes distintas. O prolema de classificação multiclasse é um problema de classificação onde existem K classes mutuamente exclusivas, e cada exemplo de dados é rotulado como pertencente a uma dessas K classes (RUSSELL; NORVIG, 2021) (GOODFELLOW; BENGIO; COURVILLE, 2016) (CHOLLET, 2021) À fórmula para a Cross-Entropy em um problema de classificação multiclasse está apresentada à seguir:

$$Cross - Entropy = -\frac{1}{m}\sum_{m}^{i=1}\sum_{K}^{j=1} y_{ij} log(\widehat{y}_{ij}) \tag{22}$$

Neste caso, temos m exemplos na base de dados e cada um deles pertence exatamente a uma das K classes. y_{ij} é uma variável indicadora que assume valor 1 se o exemplo i pertencer à classe k e assume o valor 0 caso contrário. $\widehat{y}_{ij}$ é a probabilidade prevista pela rede neural de que o exemplo i pertence à classe j (RASCHKA, 2021).

De forma geral, a função *Cross-Entropy* mede a divergência entre as distribuições de probabilidade previstas ($\widehat{y}$) e as distribuições de probabilidade reais (y). Quando a classe correta não é prevista, ou a confiança da predição é baixa, essa métrica penaliza severamente a rede, induzindo que os parâmetros treináveis sejam alterados com maior intensidade, melhorando a capacidade do modelo de prever as probabilidades corretas para cada classe.

2.3.5 Treinamento

O objetivo do processo de treinamento de uma rede neural artificial é ajustar os parâmetros aprendíveis até que os resultados gerados pela rede sejam satisfatórios, e essa satisfação é frequentemente medida pelo valor de *Loss* da função de custo ou por métricas de avaliação da rede (Seção 2.3.6).

Uma vez que a base de dados já esteja pronta e a arquitetura da rede neural já esteja determinada, o passo seguinte para o treinamento é a inicialização dos pesos ou parâmetros aprendíveis. Os pesos e os vieses da rede neural são inicializados aleatoriamente ou usando métodos específicos como aqueles propostos por (HE, K. *et al.*, 2015) (YOSINSKI *et al.*, 2014) e (GLOROT; BENGIO, 2010). Estes pesos são ajustados ou aprendidos durante o processo de treinamento da rede neural. Para re-

des neurais muito profundas é interessante explorar a possibilidade de utilizar pesos que já foram pré-treinados em outra tarefa similar com bases de dados muito maiores. Esta técnica é possível porque muitos filtros utilizados para gerar os mapas de características são intercambeáveis entre diferentes tarefas, e.g.: filtro de detecção de bordas.

O passo seguinte é tipicamente encontrado na literatura como *Forward Propagation*. Nesta etapa os dados de entrada são passados pela rede neural, camada por camada, indo da entrada até a saída da rede. Para camadas com parâmetros aprendíveis, os dados de entrada são multiplicados pelos pesos w, somados aos viéses b e quando necessário aplicados em uma função de ativação. A saída deste processo é a resposta da rede neural $\widehat{y}$.

Os resultados $\widehat{y}$ do processo de *Forward Propagation* são então comparados com os resultados reais y da base de dados. Com ambos os valores ($\widehat{y}$ e y) é possível calcular o valor de perda, conforme discutido na Seção 2.3.4. O resultado da função de custo é utilizado no algoritmo *Backpropagation* ou Retropropagação para atualizar os pesos da rede neural, conforme visto à seguir.

2.3.5.1 Propagação de erros (Backpropagation)

Um dos principais elementos dos algoritmos de aprendizado supervisionado é uma função objetiva definida a ser otimizada durante o processo de aprendizado. Essa função objetiva é frequentemente uma função de perda ou custo que queremos minimizar (RASCHKA *et al.*, 2022). A intuição por trás dessa niminização de custos/perdas é dada pelo algoritmo *gradient descent*.

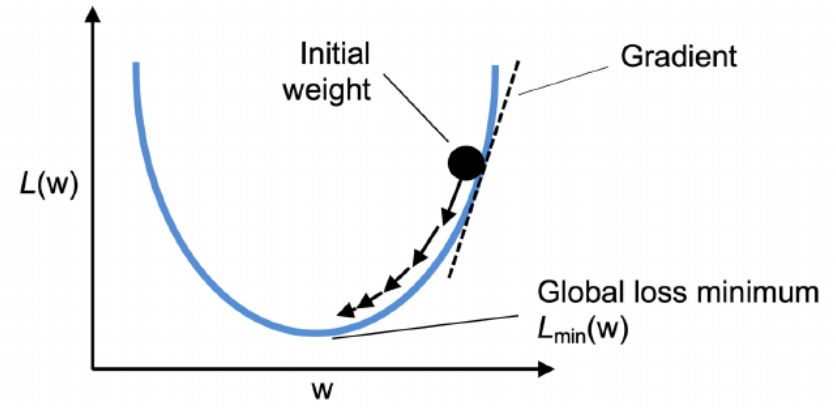
Considere uma rede neural muito simples, que tenha apenas os parâmetros aprendíveis w e b. Por simplicidade, a Figura 39 leva em consideração o parâmetro w, então podemos descrever a ideia principal por trás do *gradient descent* como descer uma colina até atingir um mínimo local ou global de perda (*loss*). Em cada iteração, damos um passo na direção oposta ao gradiente, onde o tamanho do passo é determinado pelo valor da taxa de aprendizado η, assim como pela inclinação do gradiente (RASCHKA *et al.*, 2022).

Utilizando a estratégia do *gradient descent* é possível efetuar pequenos ajustes nos pesos w e viéses b de forma que a função de custo seja minimizada, utilizando o gradiente $\nabla L(w,b)$ como guia, de forma que:

$$w := w + \Delta w \tag{23}$$

$$\Delta w = -\eta \nabla_w L(w,b) \tag{24}$$

Figura 39 – Visão geral do funcionamento do algoritmo *gradient descent*.



Fonte: (RASCHKA *et al.*, 2022).

$$b := b + \Delta b \tag{25}$$

$$\Delta b = -\eta \nabla_b L(w,b) \tag{26}$$

Naturalmente, $\nabla_w L(w,b)$ é a derivada parcial da função de custo em relação aos pesos da rede neural e $\nabla_b L(w,b)$ é a derivada parcial da função de custo em relação aos viéses da rede neural. A derivada parcial depende diretamente da função de perda utilizada. No entanto, este processo decrito é tipicamente referido na literatura como *full batch gradient descent*, elucidando que os pesos e viéses da rede são apenas atualizados após todos os dados serem utilizados.

Para bases de dados muito grandes ou para redes com tempo de inferência muito longo é importante explorar a possibilidade de atualizar os pesos de forma incremental para cada exemplo de treinamento. Esta técnica é encontrada na literatura com o nome de ***stochastic gradient descent*** (SGD) (RASCHKA *et al.*, 2022) (RUSSELL; NORVIG, 2021). Para exemplificar o algoritmo SGD, tome como exemplo a Equação (27), que representa a função de custo MSE para uma base de dados com m exemplos e a saída da rede neural é a função de ativação sigmóide σ sobre os valores da última camada z, isto é, $\sigma(z^{(i)})$.

$$L(w,b) = \frac{1}{m} \sum_{m}^{i=1} (y^{(i)} - \sigma(z^{(i)}))^2 \tag{27}$$

Neste caso, o passo de atualização dos pesos Δw para um peso j no algoritmo *full batch gradient descent* é

$$\Delta w_j = -\frac{2\eta}{m}\sum_{m}^{i=1}(y^{(i)} - \sigma(z^{(i)}))x_j^{(i)} \tag{28}$$

enquanto que para o caso do algoritmo SGD temos

$$\Delta w_j = \eta(y^{(i)} - \sigma(z^{(i)}))x_j^{(i)} \tag{29}$$

onde *i* é a variável que faz iteração sobre a base de dados de tamanho *m*.

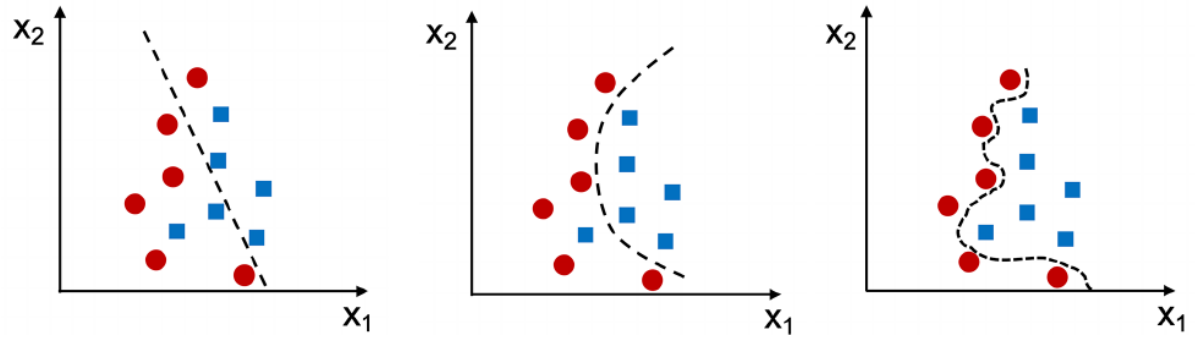
Embora o SGD possa ser considerado uma aproximação do *full batch gradient descent*, ele geralmente alcança a convergência muito mais rapidamente devido às atualizações de pesos mais frequentes. Como cada gradiente é calculado com base em um único exemplo de treinamento, a superfície de erro é mais ruidosa do que no *full batch gradient descent*, o que também pode ter a vantagem de permitir que o SGD escape de mínimos locais rasos com mais facilidade se estivermos trabalhando com funções de perda não lineares. Para obter resultados satisfatórios via SGD, é importante apresentar os dados de treinamento em uma ordem aleatória e embaralhar o conjunto de dados de treinamento a cada época para evitar ciclos (RASCHKA *et al.*, 2022). Além do SGD, outros métodos se propõe a otimizar os pesos e viéses de forma similar (KINGMA; BA, 2017) (HINTON, 2012) e (DUCHI; HAZAN; SINGER, 2011).

2.3.5.2 Sobreajuste (Overfitting)

Dizemos que uma função está sobreajustada ou sofrendo *overfitting* em um conjunto de dados quando ela dá muita atenção para os dados utilizados em seu treinamento mas falha ao realizar generalizações e prever resultados não vistos anteriormente (RUSSELL; NORVIG, 2021). O problema fundamental no aprendizado de máquina é a linha tênue entre otimização e generalização. Otimização se refere ao processo de ajustar um modelo para obter o melhor desempenho possível nos dados de treinamento (o aprendizado no aprendizado de máquina), enquanto generalização se refere a quão bem o modelo treinado se sai em dados que nunca viu antes. O objetivo é, claro, obter uma boa generalização, mas você não controla a generalização; você só pode ajustar o modelo com base nos seus dados de treinamento (CHOLLET, 2021).

Suponha que sua base de dados é divida dois conjuntos: conjunto utilizado para treinar a rede (treinamento) e conjunto utilizado para validar a rede (teste). No início do treinamento, otimização e generalização estão correlacionadas: quanto menor a perda nos dados de treinamento, menor a perda nos dados de teste. Enquanto isso está acontecendo, seu modelo é dito estar subajustado (Figura 40a) (*underfit*): ainda há progresso a ser feito; a rede ainda não modelou todos os padrões relevantes nos dados de treinamento. Mas após um certo número de iterações nos dados de treinamento, a generalização para de melhorar, e as métricas de validação estagnam e começam

Figura 40 – Exemplos de **a**) *underfit*, **b**) bom resultado e **c**) *overfitting*.



(a) Subajuste (*Underfit*). (b) Resultado satisfatório. (c) Sobreajuste (*Overfitting*).

Fonte: (RASCHKA *et al.*, 2022)

a piorar: o modelo está começando a superajustar (Figura 40c) (*overfit*). Em outras palavras, o modelo está começando a aprender padrões específicos dos dados de treinamento, mas que são enganosos ou irrelevantes quando se trata de novos dados (CHOLLET, 2021). Entre os estados de *underfit* e *overfit* temos, em geral, a solução satisfatório (Figura 40b).

A solução óbvia para este problema seria aumentar cada vez mais o tamanho da base de dados, que é um dos grandes desafios do campo do aprendizado supervisionado. No entanto, existem outras técnicas que tem como principal objetivo minimizar os riscos de *overfitting*.

- **Regularização L1 e L2**. Adiciona termos de penalização aos pesos durante o treinamento para limitar sua magnitude. A regularização L1 adiciona a soma dos valores absolutos dos pesos à função de custo. A Regularização L2 adiciona a soma dos quadrados dos pesos à função de custo. Essa penalização na função de custo desencoraja o modelo a aprender pesos muito grandes, reduzindo a complexidade da rede neural.
- **Dropout**. Proposto por (SRIVASTAVA *et al.*, 2014). Durante o processo de treinamento, desliga aleatoriamente um percentual de neurônios da camada, forçando o modelo a aprender representações mais robustas, isto é, não permite que poucos neurônios tenham muita influência na saída. Este efeito impede a co-adaptação entre neurônios, que é a hiper especialização em detectar características do conjunto de treinamento.
- **Divisão da base de dados**. Divide as base de dados em treinamento, validação e teste. Permite avaliar o modelo em diversos dados, inclusive dados nunca vistos no treinamento.
- **Parada antecipada (Early stopping)**. Monitora o desempenho do modelo em um conjunto de validação. Interrompe o treinamento assim que o desem-

penho no conjunto de validação começa a piorar, evitando que o modelo continue a se ajustar demais aos dados de treinamento.

- **Aumento artificial de dados (Data augmentation)**. Geração de mais exemplos de treinamento por meio de transformações nos dados existentes, como rotações, espelhamentos, cortes, filtros de cor, seleção de canais de cor das imagens, zooms e outros. Aumenta a variedade dos dados de treinamento sem coletar mais exemplos, ajudando o modelo a generalizar melhor.

2.3.6 Métricas para avaliação de desempenho

Não basta apenas treinar redes neurais e olhar para a sua perda (*loss*). Diferentes arquiteturas podem produzir valores de perda similares e, no entanto, seus resultados práticos podem ser muito diferentes. Nesse sentido, métricas para avaliação de desempenho são inseridas nas análises de redes neurais, especialmente redes profundas. Nas tarefas de segmentação de imagens, classificação e regressão é comum encontrar métricas como: *Mean Intersection Over Uniont* (MIoU), *Mean Pixel Accuracy* (MAcc), Acurácia, Matriz de Confusão, Precisão, *Recall* e Erro Médio Quadrático (MSE).

2.3.6.1 MIoU

A métrica IoU, ou índice de Jaccard, é uma métrica que quantifica o grau de sobreposição entre duas entidades, neste caso imagens. No caso da segmentação semântica, esta métrica mede a sobreposição entre o rótulo verdadeiro (*Ground Truth* ou GT) e a predição gerada pela rede neural (*Segmentation Mask* ou S). Para a segmentação semântica esta é a principal métrica de avaliação, encontrada em diversos trabalhos envolvendo as bases de dados (ZHOU *et al.*, 2017) (ZHOU *et al.*, 2019) (CORDTS *et al.*, 2016) e (MOTTAGHI *et al.*, 2014).

Para calcular a métrica MIoU três outros valores são necessários: *True Positive* (TP) ou verdadeiro positivo, *False Positive* (FP) ou falso positivo e *False Negative* (FN) ou falso negativo. Estes termos são amplamente encontrados em outras aplicações, mas aqui possuem significados específicos.

Verdadeiro Positivo. Área de intersecção entre o rótulo verdadeiro (GT) e a predição gerada pela rede (S). Do ponto de vista matemático, em imagens, esta é a operação lógica AND entre GT e S:

$$TP = GT \ \& \ S \tag{30}$$

Falso Positivo. Toda a área prevista pela rede neural que não está contemplada no rótulo verdadeiro. FP pode ser calculado como a operação lógica OR entre GT e S, menos GT:

$$FP = (GT||S) - GT \tag{31}$$

Falso Negativo. Número de *pixels* do rótulo verdadeiro que o modelo não foi capaz de predizer. FN é calculado como o operador OR entre GT e S, menos S:

$$FN = (GT||S) - S \tag{32}$$

A Figura 41 apresenta graficamente estes elementos. Note que, partindo destes valores, para calcular a interseção das formas sobre a sua união, basta utilizar a Equação (33). Se houverem K classes, basta calcular a média dos valores de IoU de todas as classes para formar a métrica MIoU.

Figura 41 – Visualização do rótulo verdadeiro (GT), predição da rede (S), verdadeiro positivo (TP), falso positivo (FP) e falso negativo (FN).

Fonte: (KUKIL, 2022).

$$IoU = \frac{TP}{TP + FP + FN} \tag{33}$$

$$MAcc = \frac{1}{K} \sum_{K}^{i=1} \frac{TP_K}{P_K} \tag{34}$$

2.3.6.2 Acurácia de *Pixels* Média (MAcc)

A Acurácia de *Pixels* Média é utilizada geralmente em conjunto com a métrica MIoU para avaliar o desempenho de modelos em segmentação semântica. Essa métrica se concentral no percentual de *pixels* que foram preditos corretamente pelo modelo. Tome TP_K como a quantia de *pixels* da classe K que foram preditos corretamente e P_K como a quantia total de *pixels* da classe K existentes na imagem. A métrica MAcc é dada pela Equação (35).

$$MAcc = \frac{1}{K} \sum_{K}^{i=1} \frac{TP_K}{P_K} \tag{35}$$

2.3.6.3 Acurácia

Em um problema de classificação multiclasses, a acurácia é a relação entre as predições corretas do modelo e a quantidade total de exemplos na base de dados. Tome TP como a quantidade total de predições feitas corretamente pelo modelo e m como a quantidade de exemplos na base de dados, então a métrica Acurácia (Acc) é calculada conforme a Equação (36).

$$Acc = \frac{TP}{m} \tag{36}$$

2.3.6.4 Precisão e *Recall*

Precisão. Em um problema de classificação, a precisão mede a proporção de exemplos positivos identificados corretamente entre todos os exemplos identificados como positivos. Seja TP a quantidade de exemplos identificados corretamente e FP a quantidade de exemplos marcados erroneamente como verdadeiros, o cálculo da precisão se dá pela Equação (37).

$$Preciso = \frac{TP}{TP + FP} \tag{37}$$

Recall. Em um problema de classificação, o *recall* mede a proporção de exemplos positivos identificados corretamente em relação ao total de exemplos positivos. Seja TP a quantidade de exemplos identificados corretamente e FN a quantidade de exemplos não identificados, o cálculo do *recall* se dá pela Equação (38)

$$Recall = \frac{TP}{TP + FN} \tag{38}$$

2.3.6.5 Erro Médio Quadrático (MSE)

O Erro Quadrático Médio (Mean Squared Error - MSE) é uma métrica fundamental e amplamente usada na avaliação de modelos de regressão. Ele calcula a média dos quadrados das diferenças entre as previsões do modelo e os valores reais. Tome m como a quantidade de exemplos na base de dados, y como o valor verdadeiro da base de dados e $\widehat{y}$ como o valor previsto pelo modelos. Segue o cálculo da métrica MSE como:

$$MSE = \frac{1}{m} \sum_{m}^{i=1} (y^{(i)} - \widehat{y}^{(i)})^2 \tag{39}$$

Note que a métrica MSE é a mesma utilizada como função de custo na Seção 2.3.4. O MSE penaliza fortemente grandes erros de previsão devido à sua natureza quadrática. Isso significa que valores muito discrepantes têm um impacto maior na atualização dos pesos. Essa características também sugere que valores discrepantes na base de dados (*outliers*) têm um peso muito maior do que os demais.

3 TRABALHOS CORRELATOS

Este capítulo apresenta detalhes sobre os principais trabalhos encontrados na literatura envolvendo a segmentação, identificação de madeiras em imagens e análises de características envolvendo os anéis de crescimento. Inicialmente é apresentada a metodologia empregada para realizar uma revisão sistemática da literatura, baseada no trabalho de (KITCHENHAM, 2004). Na sequência, uma breve discussão sobre os principais trabalhos selecionados é conduzida.

A metodologia descrita por (KITCHENHAM, 2004) é centrada na busca por responder uma pergunta, e essa resposta é atingida através da avaliação de diversos trabalhos e de algumas atividades, tais como:

- Definição da pergunta científica;
- Planejamento da revisão;
- Seleção de trabalhos iniciais;
- Análise e síntese dos resultados.

Para tanto, a revisão sistemática deste trabalho busca responder a pergunta "Quais informações, em imagens de madeiras, poderiam ser extraídas automaticamente por sistemas de visão computacional e como é possível realizar esta tarefa com aprendizado profundo?". Para mais detalhes da metodologia empregada, considere a Tabela 2.

Tabela 2 – Detalhes da condução da pesquisa.

Locais de busca	**Termos de pesquisa**
IEEE Xplore Science Direct ArXiv Springer Link Google Scholar	"wood analysis¨ "wood segmentation" "deep neural networks for wood analysis" "wood identification" "computer vision for wood analysis" "wood feature extraction from images"
Critérios para inclusão dos resultados	
Trabalhos publicados entre 2005 e 2023; Trabalhos escritos em língua inglesa ou portuguesa; Artigos que abordem a análise de madeira; Artigos que utilizem técnicas de visão computacional;	
Critérios para exclusão dos resultados	
Artigos curtos e com poucos detalhes; Trabalhos que utilizem imagens de drones, satélites ou geradas artificialmente; Trabalhos que não apresentem claramente as métricas utilizadas na avaliação; Artigos cujo abstract não seja claro e direto.	

Para cada um dos locais de busca na Tabela 2, todos os termos de pesquisa foram utilizados. Então, para todos os resultados obtidos, os critérios de inclusão

foram aplicados. Na sequência, foi realizada a leitura do título, resumo e referências de cada artigo. Novos trabalhos também foram selecionados através das referências do primeiro conjunto de artigos. Para estes novos artigos, os critérios de inclusão também foram aplicados. Por fim, os critérios de exclusão foram aplicados em todos os artigos restantes. Todos os artigos deste último conjunto foram cuidadosamente lidos e analisados. Deste último grupo, 17 artigos foram categorizados como trabalhos correlatos. Os 17 artigos estão divididos em duas grandes categorias: identificação de madeira e análise de elementos na madeira.

3.1 IDENTIFICAÇÃO DE MADEIRA

No escopo deste trabalho entende-se como identificação de madeiras tanto a tarefa de encontrar a madeira na imagem quanto a tarefa de classificar diferente espécies de madeira. Ambas as tarefas são importantes no contexto dos estudos envolvendo madeira. Neste sentido, (SILVA, J. L. *et al.*, 2022) e (HWANG; SUGIYAMA, 2021) apresentam revisões atualizadas no contexto da identificação de madeiras utilizando técnicas de visão computacional.

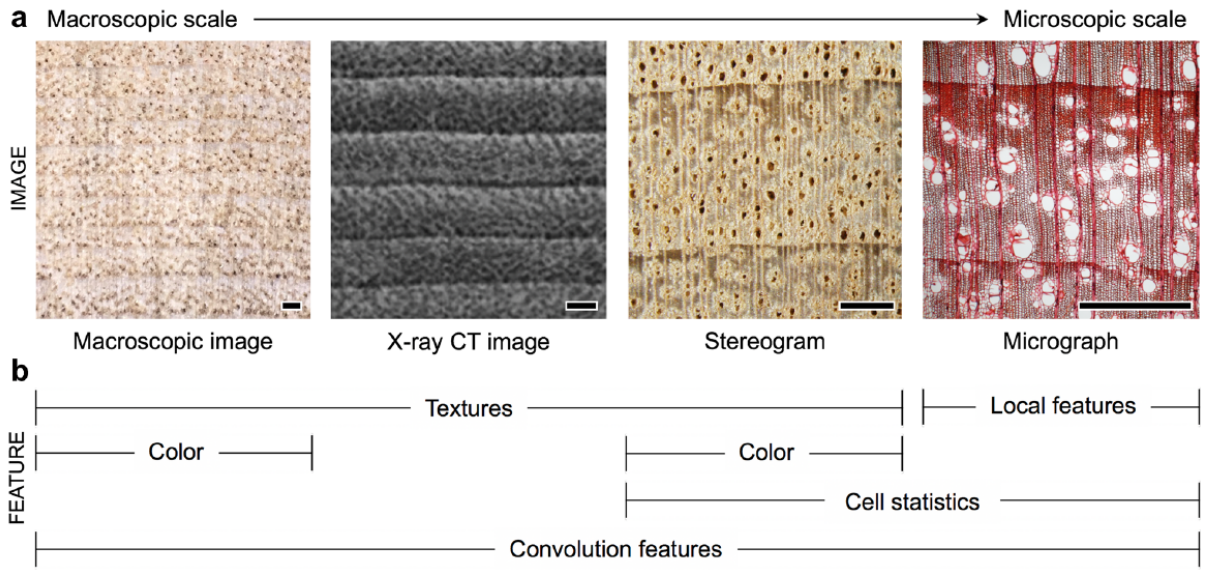
A identificação de madeira é uma ferramenta importante em diversas áreas, desde biologia até patrimônio cultural. Na luta contra o desmatamento ilegal, ela possui uma aplicação ainda mais necessária e impactante. Identificar uma amostra de madeira até o nível de gênero ou espécie é difícil, caro e demorado, mesmo ao utilizar os métodos mais recentes, resultando em uma crescente necessidade de um método prontamente acessível e aplicável em campo para a identificação científica de madeira. Oferecendo resultados rápidos e facilidade de uso, a tecnologia baseada em visão computacional é uma opção economicamente acessível atualmente utilizada para atender à demanda por identificação automatizada de madeira. No entanto, apesar das características promissoras e resultados precisos desse método, ele continua sendo uma área de pesquisa de nicho nas ciências da madeira e é pouco conhecido em outros campos de aplicação, como o patrimônio cultural (SILVA, J. L. *et al.*, 2022).

O trabalho de (SILVA, J. L. *et al.*, 2022) trás, principalmente, uma lista extensa de bases de dados disponíveis para uso em trabalhos envolvendo a identificação de madeira, totalizando 18 bases. Destas, 13 estão amplamente disponíveis sem custos e 1 está disponível apenas para pesquisas sem fins lucrativos. A maior base, chamada de *InsideWood*, conta com mais de 60.000 imagens microscópicas de madeiras de lei (*hardwood*), madeiras macias (*softwood*) e fósseis de madeiras de lei. Além das listas de bases de dados, também há uma lista de trabalhos utilizando redes neurais profundas na tarefa de classificação de imagens. Muitos destes trabalhos, inclusive, utilizam CNNs na tarefa de classificação.

De forma similar, (HWANG; SUGIYAMA, 2021) descrevem em seu trabalho brevemente o funcionamento de sistemas de visão computacional - principalmente

aqueles que utilizam redes neurais - e suas aplicações nas ciências que estudam a madeira. Naturalmente, sistemas baseados em visão computacional inicial com a aquisição de imagens, que é um desafio neste nicho. A maioria dos estudos captura as imagens da seção reta das toras de madeira. Na Figura 42 vemos possíveis tipos de imagens obtidos a partir de madeira e as características de imagem correspondentes extraíveis. Na parte superior da imagem, da esquerda para a direita temos: imagem macroscópica, imagem tomográfica computadorizada por raios X (CT), estereograma e micrografia de cortes transversais da *Cinnamomum camphora*. Na parte inferior, características por tipo de imagem em publicações. As barras de escala representam 1 mm.

Figura 42 – Tipos de imagens adquiridas para a análise de madeiras.



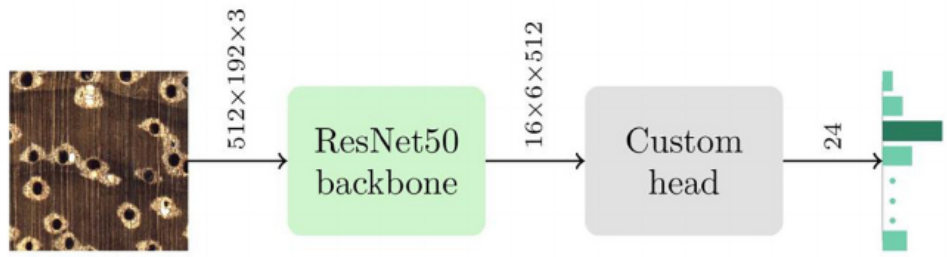
Fonte: (HWANG; SUGIYAMA, 2021).

Uma linha de estudos interessante apresentada por (HWANG; SUGIYAMA, 2021) é a segmentação de células de madeira que, até então, era altamente dependente da qualidade das imagens por conta dos algoritmos clássicos de visão computacional. No entanto, com o advento das redes neurais profundas, especialmente as CNNs, essa tarefa ficou substancialmente mais simples. De forma similar, a classificação de diferentes tipos de madeira com base em imagens macroscópicas também avançou.

Em (RAVINDRAN *et al.*, 2021) vemos um exemplo de trabalho que aplica redes neurais profundas para a classificação da espécie de madeiras através de imagens macroscópicas. A base de dados utilizada neste trabalho é própria e é composta por madeiras de lei encontradas principalmente nas florestas peruanas. O modelo utilizado pelos autores é baseado em CNNs e pode ser visto na Figura 43. O modelo recebe

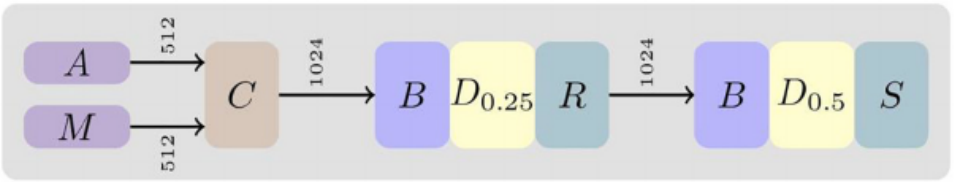
como entrada imagens coloridas de tamanho $512 \times 192 \times 3$ que são enviadas para um *encoder* ResNet50. A saída da ResNet50 é enviada para um classificador customizado (Figura 44). Este classificador customizado inclui a operação de *average pooling* (A), *max pooling* (M), concatenação (C), normalização em lote (B), camada de *dropout* (D) e camadas lineares com ativações ReLU (R) e softmax (S). D_p representa uma camada de *dropout* com o parâmetro de probabilidade de eliminação p. A acurácia desta rede em um teste de validação cruzada foi de 97%.

Figura 43 – Modelo utilizado por (RAVINDRAN *et al.*, 2021).



Fonte: (RAVINDRAN *et al.*, 2021).

Figura 44 – Método de classificação customizado utilizado no modelo da Figura 43.



Fonte: (RAVINDRAN *et al.*, 2021).

De forma similar a classificação de madeiras, outros trabalhos buscam identificar a posição da madeira em imagens complexas. Estas imagens geralmente possuem diversos ruídos como fundos adversos, chuva, neve e sujeira. A identificação da posição pode ser tanto aproximada, como o cálculo do centróide e diâmetro (para imagens da seção reta de toras), ou ainda identificação de posição a nível de *pixel*, como na segmentação semântica.

Seguindo esta linha, o trabalho feito por (GALSGAARD *et al.*, 2015) propõe uma solução para o problema de pilhas de toras de madeira (Figura 45). Neste desafio, deseja-se contar o número de toras, estimar seus diâmetros e estimar seus volumes.

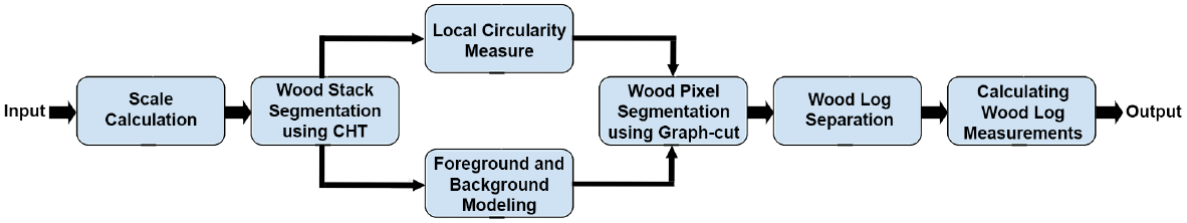
A proposta dos autores é utilizar a otimização de corte de gráfico, apresentada por (BOYKOV; VEKSLER; ZABIH, 2001), para realizar a identificação das toras de madeira. No entanto, um dos grandes desafios desta otimização é a escolha de pe-

Figura 45 – Pilhas de toras de madeira.

Fonte: (GALSGAARD *et al.*, 2015).

sos razoáveis que possam gerar bons resultados. Este problema é contornado em seu trabalho através do uso da transformada circular de Hough (CHT), que permite encontrar a posição aproximada de objetos razoavelmente circulares em uma imagem. Através das informações obtidas pela CHT, os pesos são escolhidos e o algoritmo de otimização de corte de gráfico gera os resultados. A Figura 46 apresenta o diagrama de funcionamento do sistema completo, a Figura 47a apresenta um bom exemplo de resultado e a Figura 47b apresenta um exemplo de resultado ruim.

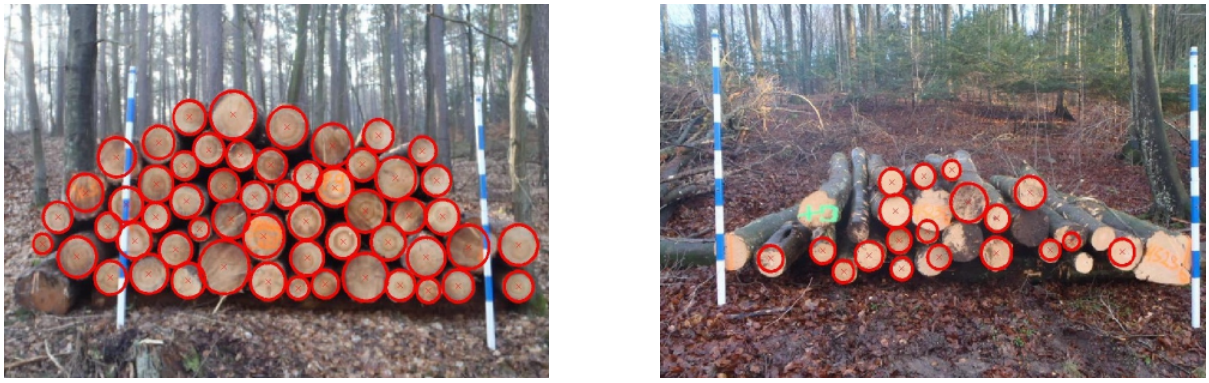
Figura 46 – Diagrama do sistema proposto por (GALSGAARD *et al.*, 2015).



Fonte: (GALSGAARD *et al.*, 2015).

Similarmente, o trabalho desenvolvido por (CHIRYSHEV; KRUGLOV; ATAMANOVA, 2018) tem o mesmo objetivo: identificar, contar e estimar as dimensões de toras

Figura 47 – Resultados do sistema proposto por (GALSGAARD *et al.*, 2015).



(a) Resultado bom. (b) Resultado ruim.

Fonte: (GALSGAARD *et al.*, 2015)

em pilhas de madeira. No entanto, o algoritmo utilizado é diferente. (CHIRYSHEV; KRUGLOV; ATAMANOVA, 2018) propõe o uso de histogramas de gradientes orientados (HOG) em conjunto com a técnica de floresta de decisão aleatória (*random decision forest*). O HOG é um método utilizado para fornecer uma descrição indicativa que caracteriza a forma de um objeto. Para construir este vetor descritor, existem dois passos principais:

1. Destacar as fronteiras de uma imagem digital. Esse processo é baseado no cálculo do gradiente ao longo das direções vertical e horizontal e consiste em obter as derivadas parciais para cada ponto da imagem original $f(x, y)$. Vários operadores diferenciais podem ser utilizados para obter o gradiente, como os operadores Sobel, Prewitt ou Scharr, no entanto, os autores utilizam um simples filtro de diferença entre *pixels* vizinhos na forma [-1,0,1] (CHIRYSHEV; KRUGLOV; ATAMANOVA, 2018).
2. Descrição vetorial de regiões de imagem. Ao descrever uma região de uma imagem com um vetor de características, ela é dividida em várias pequenas seções (células). Os histogramas hi dos gradientes orientados são calculados nessas células. Cada pixel na célula participa da votação ponderada para os canais do histograma com base em seu valor de gradiente (CHIRYSHEV; KRUGLOV; ATAMANOVA, 2018).

Os histogramas são normalizados e combinados em um único vetor descritor. Este vetor descritor é enviado para uma série de árvores de decisão. Cada árvore de decisão executa a classificação de acordo com seus parâmetros aprendidos no treinamento, então cada saída de cada árvore é entendida como um voto. A classe mais votada é interpretada como a resposta. Na Figura 48 podemos ver os resultados deste algoritmo na base de dados HAWKwood (HERBON, 2014).

No entando, existem outras abordagens para a identificação de madeiras que

Figura 48 – Resultados do sistema proposto por (CHIRYSHEV; KRUGLOV; ATAMANOVA, 2018).

(a) (b)

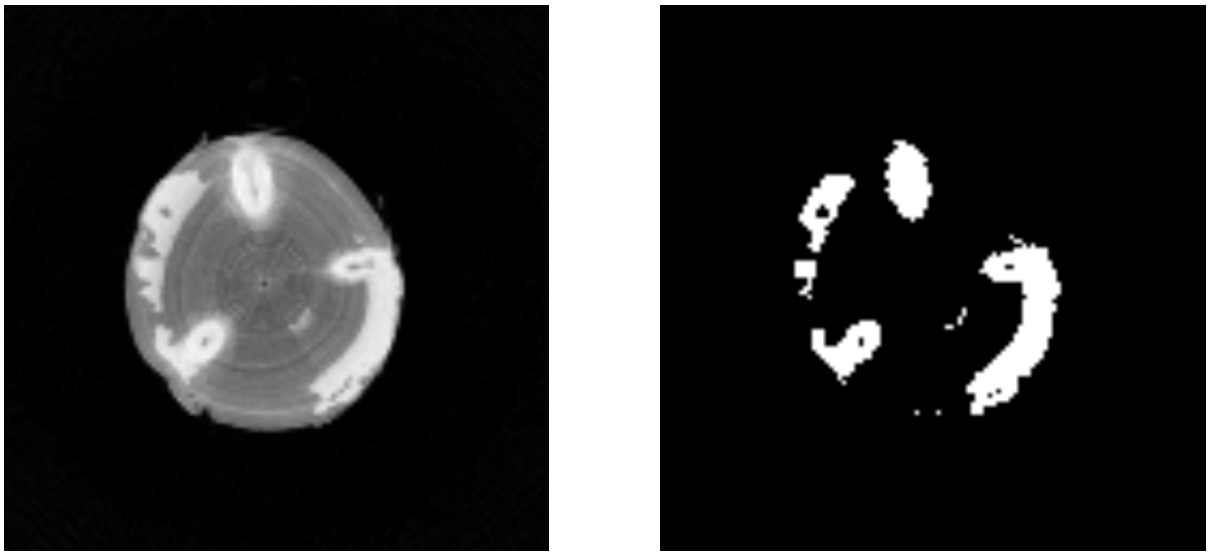
Fonte: (CHIRYSHEV; KRUGLOV; ATAMANOVA, 2018)

não utilizam câmeras convencionais. Em aplicações industriais, é comum escanear objetos em uma esteira transportadora em movimento. As medidas de tomografia computadorizada (TC) em 2D, fatia por fatia, de um objeto em movimento, são chamadas de geometria de escaneamento sequencial. Nesse caso, cada fatia por si só não contém informações suficientes para reconstruir uma imagem tomográfica útil. Portanto, (SPRINGER *et al.*, 2023) propõe o uso de um Filtro de Kalman de dimensão reduzida para acumular informações entre as fatias e permitir reconstruções suficientemente precisas para a avaliação adicional do objeto.

Adicionalmente, também é proposto o uso de uma abordagem de agrupamento não supervisionado conhecida como *Density Peak Advanced* para realizar uma segmentação e identificar anomalias de densidade na estrutura interna dos objetos reconstruídos. Os testes são conduzidos através do escaneamento de toras de madeira no processo industrial de serragem, onde o objetivo é identificar anomalias dentro da tora de madeira para permitir padrões de serragem ótimos. A qualidade da reconstrução e segmentação é avaliada a partir de dados experimentais de medição para vários cenários subamostrados (SPRINGER *et al.*, 2023). A Figura 49a apresenta uma imagem de referência e a Figura 49b é a segmentação das anomalias presentes nesta imagem.

No âmbito da segmentação em imagens, diversos trabalhos utilizam esta técnica para identificar a posição de madeiras em imagens com maior precisão. Em (SCHRAML; UHL, 2014) o objetivo dos autores é segmentar a seção reta de toras de madeira em imagens utilizando um algoritmo de crescimento por similaridade de região. A Figura 50a é um exemplo de entrada para esse sistema, a Figura 50b mostra a inicialização de regiões feita pela análise de possíveis grupos similares, a Figura 50c mostra a seleção das regiões de borda da etapa anterior, a Figura 50d determina o limite de crescimento das regiões através de um ajuste elipsoidal e por fim uma operação

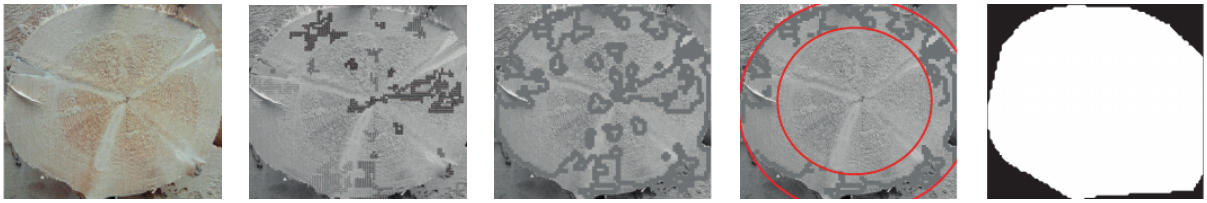
Figura 49 – Segmentação de anomalias em imagens reconstruídas à partir de medições subamostradas de tomografia computadorizada.



(a) Imagem referência. (b) Anomalias segmentadas.

Fonte: (SPRINGER *et al.*, 2023)

Figura 50 – Passos da segmentação por crescimento de região.



(a) Imagem original. (b) Inicialização das regiões. (c) Blocos de borda. (d) Limites do crescimento. (e) Máscara final.

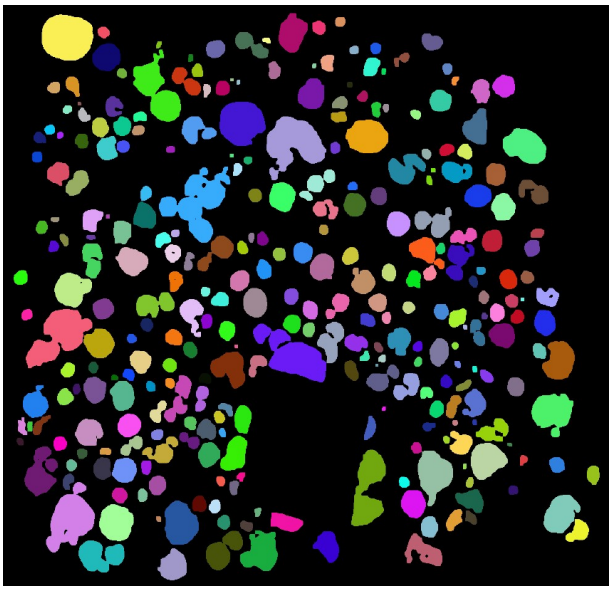
Fonte: (SCHRAML; UHL, 2014)

conhecida como *concave hull* é utilizada para gerar a máscara final de segmentação da Figura 50e.

Ainda seguindo a abordagem de utilizar segmentação semântica, em (SAMDANGDECH; PHIPHOBMONGKOL, 2018) é proposto um sistema para identificar e contar toras de madeiras empilhadas. Este sistema possui 3 grandes etapas: detecção da madeira, segmentação das toras e contagem das toras. Para detectar o local da imagem onde está a pilha de madeira os autores utilizam o método proposto por (LIU, W. *et al.*, 2016), que gera uma *bounding box* na região de interesse, que é utilizada para recortar a imagem original. Esta primeira etapa permite eliminar ruídos de fundo e diminuir a resolução da imagem, atenuando o custo computacional para o processo seguinte. Na sequência, a nova imagem recortada é enviada para uma versão da rede VGG-16 ((SIMONYAN; ZISSERMAN, 2015)) adaptada para realizar segmentação semântica. O objetivo deste passo é segmentar as seções retas das toras de madeira.

Figura 51 – Entrada e saída do sistema proposto por (SAMDANGDECH; PHIPHOBMONGKOL, 2018).

(a) Imagem original.



(b) Saída esperada.

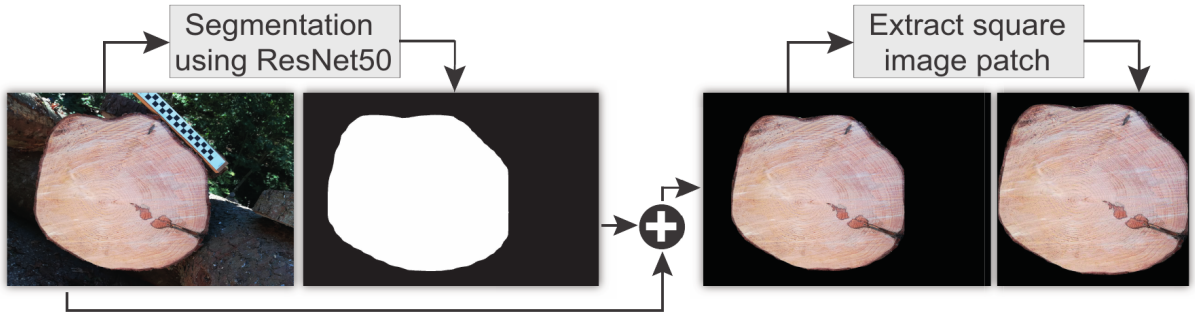
Fonte: (SAMDANGDECH; PHIPHOBMONGKOL, 2018)

Por fim, algoritmos clássicos de visão computacional como operações morfológicas e análise de componentes conexos são aplicados. O resultado esperado por todo este processamento está apresentado na Figura 51b. Note que cada cor é um identificador único para as diferentes seções retas segmentadas.

Em (WIMMER *et al.*, 2021) um sistema de dois estágios baseado em CNNs é proposto. O primeiro estágio é responsável por segmentar a seção reta ou *cross-section* (CS) da tora de madeira na imagem. Esta etapa é realizada com o auxílio do modelo de rede neural Mask R-CNN (HE, K. *et al.*, 2017), desenvolvido especialmente para a tarefa de segmentação, utilizando neste caso como *backbone* a rede ResNet50 e um pós processamento para selecionar uma porção quadrada da imagem (Figura 52). O segundo estágio recebe como entrada o resultado da segmentação para classificar

as toras de madeira. Para tanto, os autores utilizam a rede Squeeze-Net (IANDOLA *et al.*, 2016) em conjunto com a função de perda tripla (WEINBERGER; BLITZER; SAUL, 2005) (SCHROFF; KALENICHENKO; PHILBIN, 2015). Em seu trabalho, os autores concluem que o primeiro estágio (segmentação) impulsiona significativamente o desempenho da rede no segundo estágio (classificação).

Figura 52 – Diagrama do primeiro estágio proposto por (WIMMER *et al.*, 2021).

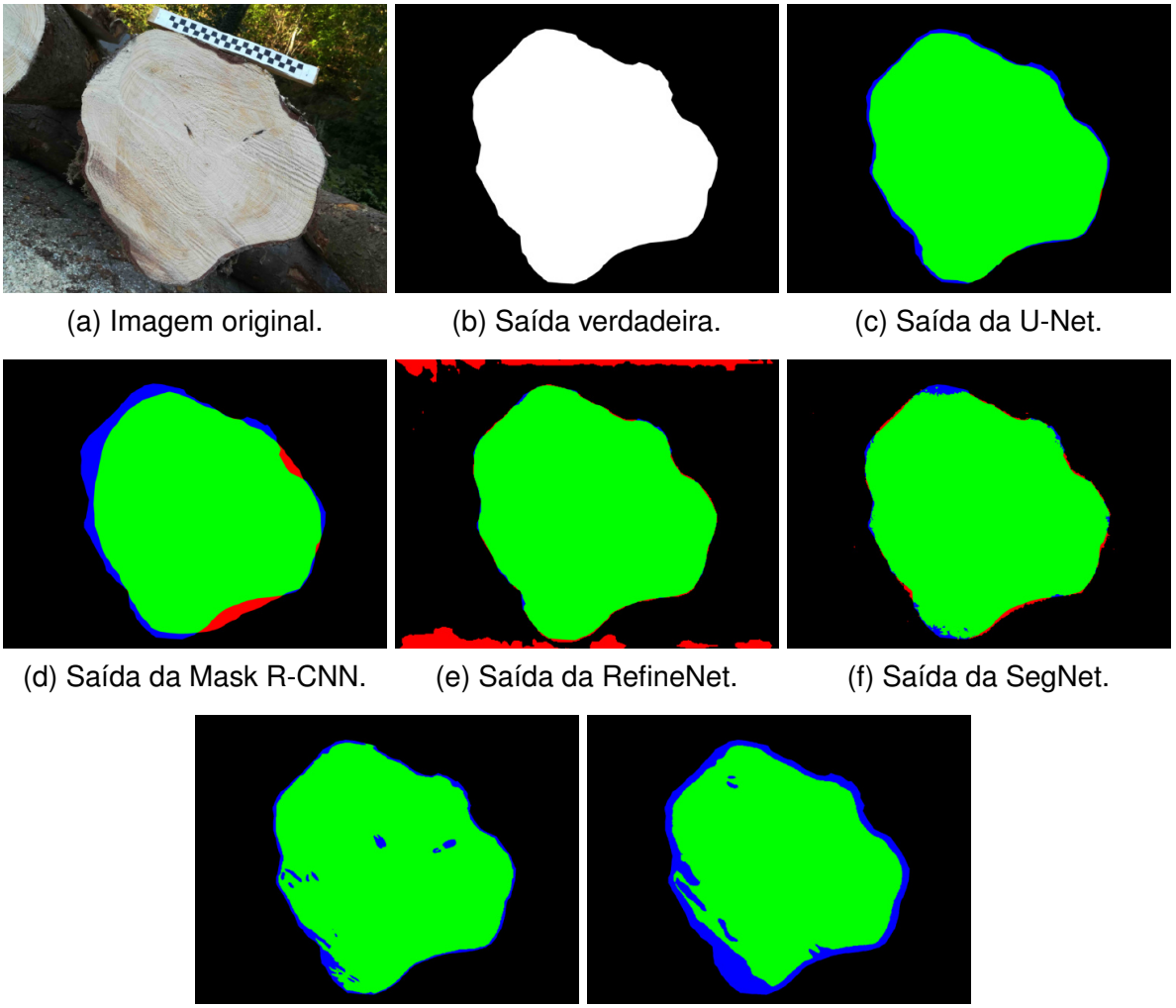


Fonte: (WIMMER *et al.*, 2021).

Em (DECELLE; JALILIAN, 2020), uma extensa comparação entre diferentes técnicas para segmentação da seção reta de toras de madeira é feita. Os autores se propõe a comparar 6 técnicas distintas: U-Net (RONNEBERGER; FISCHER; BROX, 2015), Mask R-CNN (HE, K. *et al.*, 2017), RefineNet (LIN, G. *et al.*, 2017), SegNet (BADRINARAYANAN; KENDALL; CIPOLLA, 2017), K-means (ARTHUR; VASSILVITSKII, 2007) e Snake (CHAN; VESE, 2001). A Figura 53 mostra os resultados obtidos pelos autores. Os *pixels* verdes são verdadeiros positivos (TP), os *pixels* azuis são falsos negativos (FN) e os *pixels* vermelhos são falsos positivos (FP). Note que os autores tratam esta tarefa como uma tarefa de classificação binária. Além disso, é importante ressaltar que a base de dados utilizada pelos autores é de criação própria e também é uma das contribuições do trabalho.

Em (FORTIN *et al.*, 2022) os autores argumentam que o processo de coleta de toras de madeira é desafiador para automatizar, de forma que a maioria das soluções existentes consideram que as toras já estejam em posições conhecidas. Para tanto, os autores se propõe a atacar o problema através da visão computacional. Existem duas principais contribuições para este trabalho. A primeira é a criação de uma base de dados, chamada de TimberSeg 1.0, focada na tarefa de segmentação de instâncias. Alguns exemplos desta base de dados podem ser vistos na Figura 54. A segunda contribuição é a aplicação de 3 redes neurais profundas diferentes para resolver o desafio proposto por esta base de dados criada. As redes neurais utilizadas são: Mask R-CNN (HE, K. *et al.*, 2017), Rotated Mask R-CNN (MA *et al.*, 2018) e Mask2Former (CHENG *et al.*, 2022). Os autores concluem em seu trabalho que, neste caso, para

Figura 53 – Resultados do trabalho feito por (DECELLE; JALILIAN, 2020).



(a) Imagem original. (b) Saída verdadeira. (c) Saída da U-Net.

(d) Saída da Mask R-CNN. (e) Saída da RefineNet. (f) Saída da SegNet.

(g) Saída do K-Means. (h) Saída do Snake.

Fonte: (DECELLE; JALILIAN, 2020)

ambientes extramemente ruidosos a solução baseadas no Transformer (Mask2Former) apresenta melhores resultados.

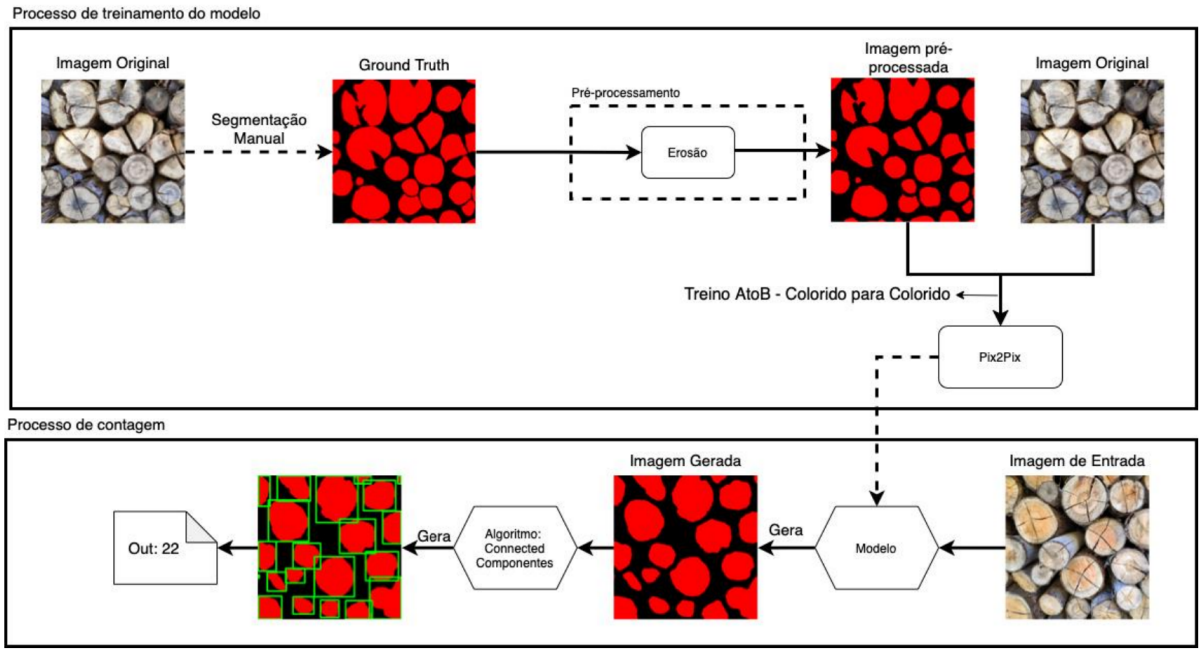
Por fim, em (TIECKER GUSTAVO; MAZZOCHIN, 2021) vemos uma abordagem recente utilizando redes generativas adversárias (GANs). Em seu trabalho, os autores utilizam a rede Pix2Pix (ISOLA *et al.*, 2017) como principal elemento para realizar a segmentação de toras de madeira empilhadas. Após o processo de segmentação, de forma similar ao trabalho apresentado por (SAMDANGDECH; PHIPHOBMONGKOL, 2018), os autores utilizam operações morfológicas e o algoritmo de análise de componentes conexos para contar as toras de madeira. A Figura 55 apresenta de forma superficial o processo de treinamento e inferência propostos pelos autores. Os resultados reportados pelos autores para a contagem de toras atinge a marca de 97% de acurácia.

Figura 54 – Base de dados TimberSeg 1.0.

(a) (b)

Fonte: (FORTIN *et al.*, 2022)

Figura 55 – Treinamento e inferência propostos por (TIECKER GUSTAVO; MAZZOCHIN, 2021).



Fonte: (TIECKER GUSTAVO; MAZZOCHIN, 2021).

3.2 ANÁLISE DE ELEMENTOS NA MADEIRA

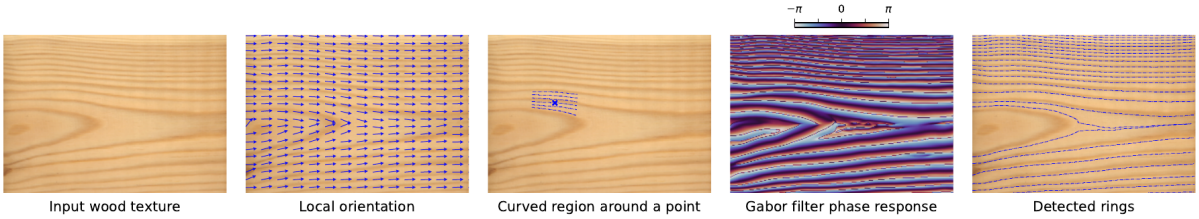
Outros trabalhos correlatos tratam da análise de características nas madeiras. Em (NINDEL *et al.*, 2023), uma das contribuições dos autores é a detecção dos anéis de crescimento em imagens de madeira. A Figura 56 mostra superficialmente as etapas para detectar os anéis de crescimento. As etapas são computadas da seguinte forma:

- A imagem de entrada é suavizada utilizando um filtro Gaussiano;
- As orientações locais são calculadas através do operador de Scharr (SCHARR,

2014);

- As regiões curvadas são calculadas através de um método proposto pelos autores;
- Um complexo filtro de Gabor (GOTTSCHLICH, 2012) é aplicado nas regiões curvadas;
- A resposta em frequência do filtro de Gabor é utilizada em conjunto com com o campo de orientação para determinar a posição dos anéis de crescimento.

Figura 56 – Etapas para a detecção dos anéis de crescimento em madeiras, proposto por (NINDEL *et al.*, 2023).



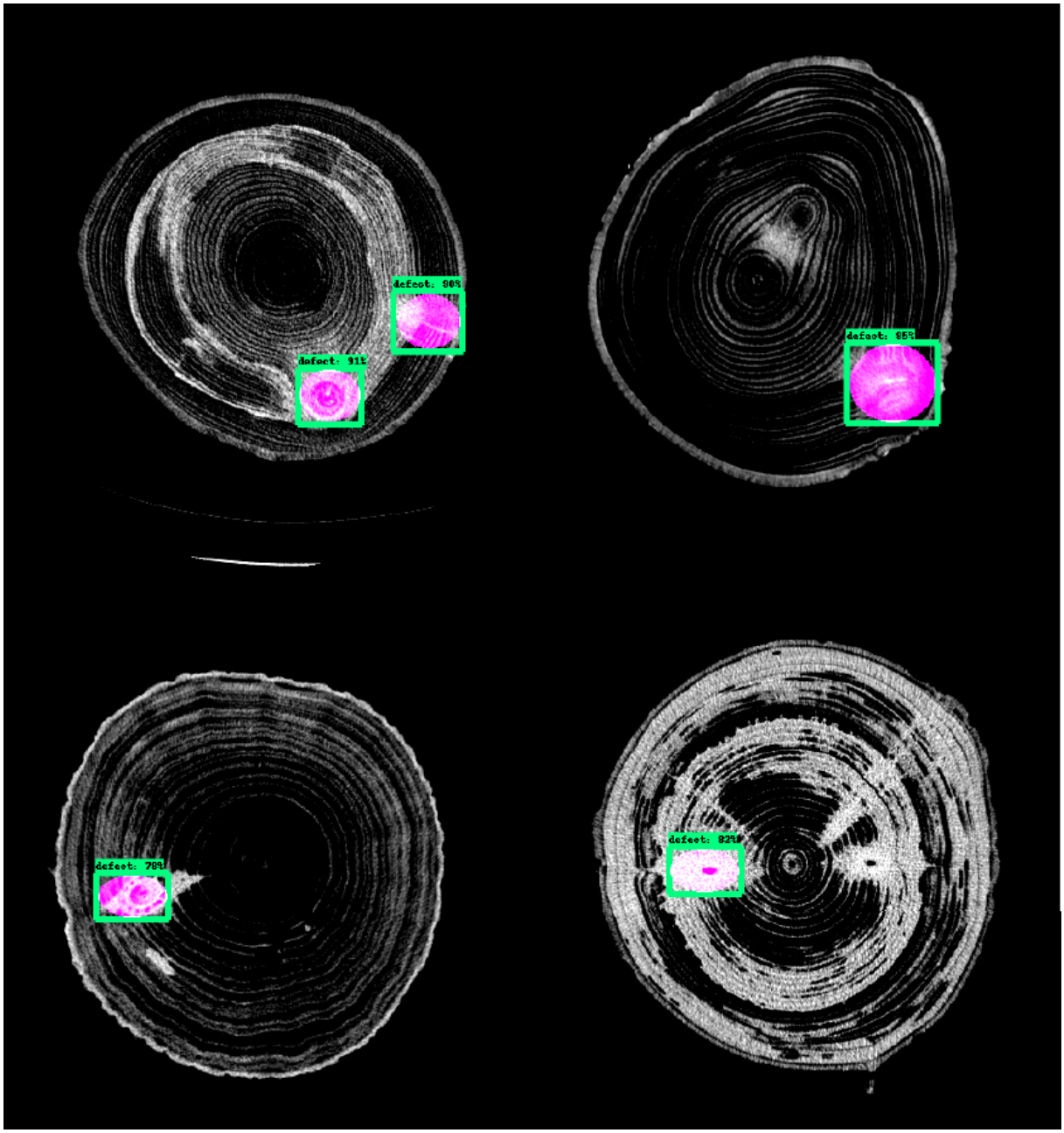
Fonte: (NINDEL *et al.*, 2023).

Em (MOHAMED; RICHARD; PRADALIER, 2020) um trabalho envolvendo a detecção de nós internos na madeira é proposto. Os autores argumentam em seu trabalho que a maioria das soluções para esse desafio utilizam imagens feita através de tomografia computadorizada, que demanda equipamentos de alto valor. A alternativa proposta pelos autores é utilizadas redes neurais profundas baseadas em CNNs para detectar os nós internos com base em imagens exteriores da madeira. A Figura 57 demonstra os resultados obtidos pelos autores. Este sistema pode otimizar substancialmente o corte de madeiras, reduzindo desperdícios na indústria madeireira.

Em (CERDA; HITSCHFELD-KAHLER; MERY, 2007) os autores propõe também um abordagem para realizar a deteção dos anéis de crescimento em toras de madeira. A base do algoritmo proposto é uma variação da transformada geral de Hough. O algoritmo de Canny é utilizado na imagem original (Figura 58a) para detectar suas bordas (Figura 58b). Todos os pontos na imagem de borda que vão de regiões escuras para regiões claras são mantidos (Figura 58c). Então um algoritmo de crescimento por região é aplicado, utilizando como limites o centro e o perímetro da tora. Este algoritmo de crescimento gera todos os polígonos aceitáveis (Figura 58d). Então um acumulador varre a imagem da Figura 58d contabilizando todos os máximos locais. O resultado do algoritmo pode ser visto na Figura 58e.

Em (POLÁČEK *et al.*, 2023) os autores modificam uma versão da Mask R-CNN para executar a tarefa de segmentação de instâncias. Através desta abordagem, os autores são capazes de detectar os anéis de crescimento em imagens da seção

Figura 57 – Deteção de nós internos em toras de madeira utilizando CNNs.

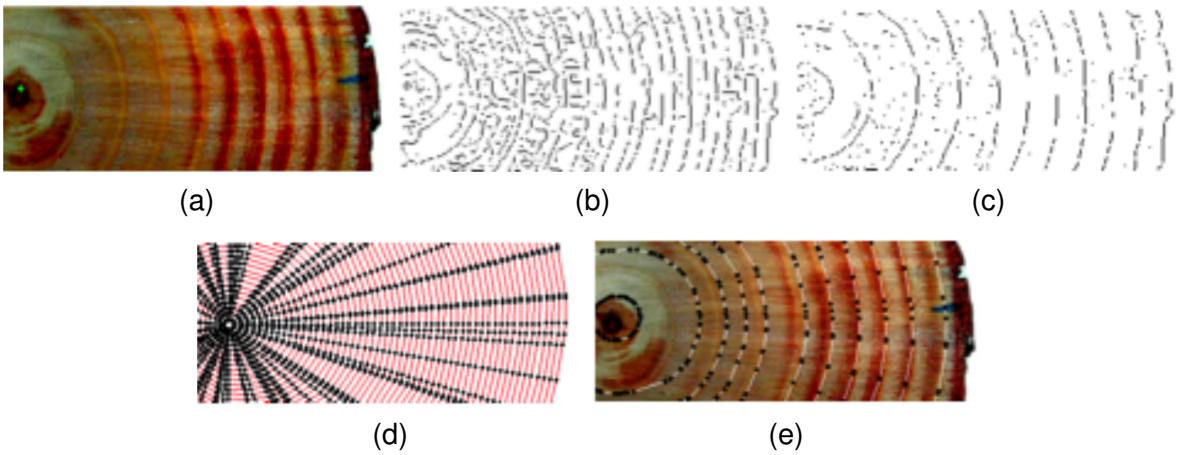


Fonte: (MOHAMED; RICHARD; PRADALIER, 2020).

reta de toras de madeira. A Figura 59 demonstra de forma superficial o processo de segmentação dos anéis de crescimento. A imagem original é dividida em pequenos segmentos e cada um destes segmentos é enviado para a Mask R-CNN 3 vezes: sem rotação, com rotação de 90° e 45°. Todos os resultados são combinados em uma única máscara. Com 186 épocas de treinamento, os autores conseguiram atingir 0.29 na medida mAP e 0.55 na medida mIoU.

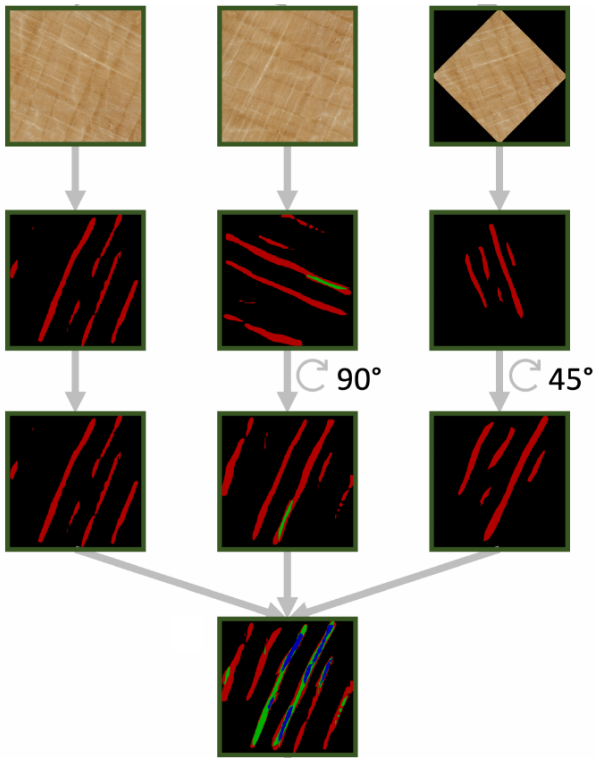
Por fim, o trabalho de (ALZELEY; ALJEDDANI, 2023) tem como objetivo analisar

Figura 58 – Sequência de processamento do algoritmo proposto por (CERDA; HITSCHFELD-KAHLER; MERY, 2007).



Fonte: (CERDA; HITSCHFELD-KAHLER; MERY, 2007)

Figura 59 – Segmentação de instâncias de anéis de crescimento.



Fonte: Adaptado de (POLÁČEK *et al.*, 2023).

dados referentes a anéis de crescimento em árvores através de algoritmos de aprendizado de máquina e teoria dos valores extremos. O estudo envolve dados da quantidade total de anéis, data do primeiro anel e data do último anel da árvore. Em seu trabalho, os autores mostram que estes dados são extremamente úteis para entender o clima da região ao longo dos anos.

4 DESENVOLVIMENTO GERAL

Este capítulo apresenta a metodologia proposta para desenvolver algoritmos capazes de analisar e extrair informações de imagens da seção reta de toras de madeira. É apresentada uma descrição da base de dados utilizada e dos algoritmos desenvolvidos para esta tarefa.

4.1 ANÁLISE DO PROBLEMA E LEVANTAMENTO DE REQUISITOS

A primeira etapa desta metodologia foi definir o conjunto de requisitos para o desenvolvimento dos algoritmos de análise de imagens de toras de madeira. Estes requisitos foram levantados através da revisão sistemática de trabalhos correlatos. A lista dos requisitos funcionais e não funcionais pode ser vista à seguir:

1. Requisitos funcionais:
    a) Dada a imagem da seção reta de uma tora de madeira, o algoritmo deve ser capaz de estimar quais elementos na imagem são madeira e quais elementos não são. Se houver mais de uma tora de madeira na imagem, o algoritmo deve se concentrar naquela que estiver mais ao centro;
    b) Dada a imagem da seção reta de uma tora de madeira que contenha anéis de crescimento, o algoritmo deve ser capaz de estimar a quantidade destes anéis de crescimento. Se houver mais de uma tora de madeira na imagem, o algoritmo deve ser concentrar naquela que estiver mais ao centro.
2. Requisitos não-funcionais:
    a) O algoritmo não deve necessitar de *hardware* especializado como GPUs (*Graphics Processing Units*) ou TPUs (*Tensor Processing Units*) para executar o processamento das imagens;
    b) O algoritmo deve poder ser reproduzido nas plataformas x86 Linux, Windows e MacOS.

Apesar de diversos *hardwares* embarcados, como as NVIDIA Jetson, já possuírem GPUs, é comum que sistemas de visão computacional portáteis ainda sejam implementados em tablets e smartphones. Como os objetivos deste trabalho estão fundamentados na necessidade de se extrair características de toras de madeira com maior facilidade, é importante que ele contemple os requisitos não-funcionais elencados, de forma que possa ser embarcados em *hardwares* como smartphones e tablets.

Após a revisão sistemática e o levantamento de requisitos funcionais e não funcionais, foi possível elaborar a proposta deste trabalho para solucionar os desafios propostos. Nesse sentido, o presente trabalho é dividido em três principais atividades:

aquisição e preparação da base de dados, desenvolvimento do algoritmo de identificação de madeira e desenvolvimento do algoritmo de estimativa de anéis de crescimento.

4.2 MATERIAIS

Esta seção descreve as informações gerais das redes neurais profundas e da base de dados utilizada neste trabalho, bem como todas as preparações necessárias para que ela seja aplicada neste escopo.

4.2.1 Redes neurais profundas

Neste projeto são propostos diversos métodos para a identificação de madeira e contagem de anéis de crescimento baseados no uso de redes neurais profundas. A identificação das toras de madeira em imagens é feita através de redes neurais profundas, sejam elas CNNs ou Transformer, capazes de realizar a segmentação semântica em imagens. Para a tarefa de segmentação semântica, as seguintes redes foram utilizadas: U-Net, FastFCN, SegFormer, Twins e Swin. Adicionalmente, a contagem de anéis é feita através de redes neurais profundas preparadas para a tarefa de regressão. As redes utilizadas nesta etapa foram versões modificadas da ResNet18, ResNet34, ResNet50 e ResNet101. As redes foram treinadas em uma base de dados customizada para esta aplicação e avaliadas de acordo com métricas relevantes em seus domínios.

4.2.2 Base de dados

Diversas bases de dados envolvendo madeiras foram encontradas durante o processo de revisão sistemática, no entanto, apenas algumas possuem imagens da seção reta de toras de madiera. O artigo desenvolvido por (DECELLE; JALILIAN, 2020) apresenta grande similaridade com o presente trabalho, além de ter como uma de suas contribuições a criação de uma nova base de dados.

Nesse sentido, a base de imagens utilizada neste trabalho é parte da base original, criada por (DECELLE; JALILIAN, 2020), que foi obtida ao entrar em contato com os autores. Nesta base, existem imagens de duas espécies de árvores: abeto-de-douglas (*Pseudotsuga menziesii*) e abeto norueguês (*Picea abies*). Existe um total de 2342 fotografias da seção reta de toras de madeira que são divididas em 5 bases menores de acordo com as características de aquisição das imagens: Sbg_TS3, Sbg_TS12, Lumix, Huawei e Ane. A Tabela 3 trás informações sobre cada uma destas divisões da base de dados. Note que as bases possuem diferentes tamanhos de imagens devido ao uso de câmeras diferentes, além de também existirem diferentes quantidades de imagens por base.

Tabela 3 – Informações da base de dados utilizada neste trabalho.

Nome da base	Camera	Número de imagens	Tamanho das imagens	Tipo de madeira
Sbg_TS12	Canon EOS 70D	1504	1368 x 912	Abeto norueguês
Sbg_TS3	Canon EOS 5D Mark II	768	2048 x 1365	Abeto-de-douglas
Lumix	Panasonic DMC-FZ45	37	4320 x 3240	Abeto-de-douglas
Huawei	Huawei PRA-LX1	22	3968 x 2976	Abeto norueguês
Ane	Huawei ANE-LX1	11	4608 x 3456	Abeto-de-douglas

Fonte: O autor.

A Figura 60 apresenta alguns exemplos das imagens que podem ser encontradas nas bases de dados disponíveis. Note que na base de dados original, elaborada por (DECELLE; JALILIAN, 2020), existe um outro conjunto de imagens chamado de *Sawmill* que não está sendo utilizado aqui. Algumas imagens, como a Figura 60a e a Figura 60b, exemplificam bem os desafios encontrados: ruídos no fundo da imagem, presença de outras toras de madeira na mesma imagem e condições adversas de iluminação.

Figura 60 – Parte da base de dados criada por (DECELLE; JALILIAN, 2020).

(a) Sbg_TS12.

(b) Sbg_TS3.

(c) Lumix.

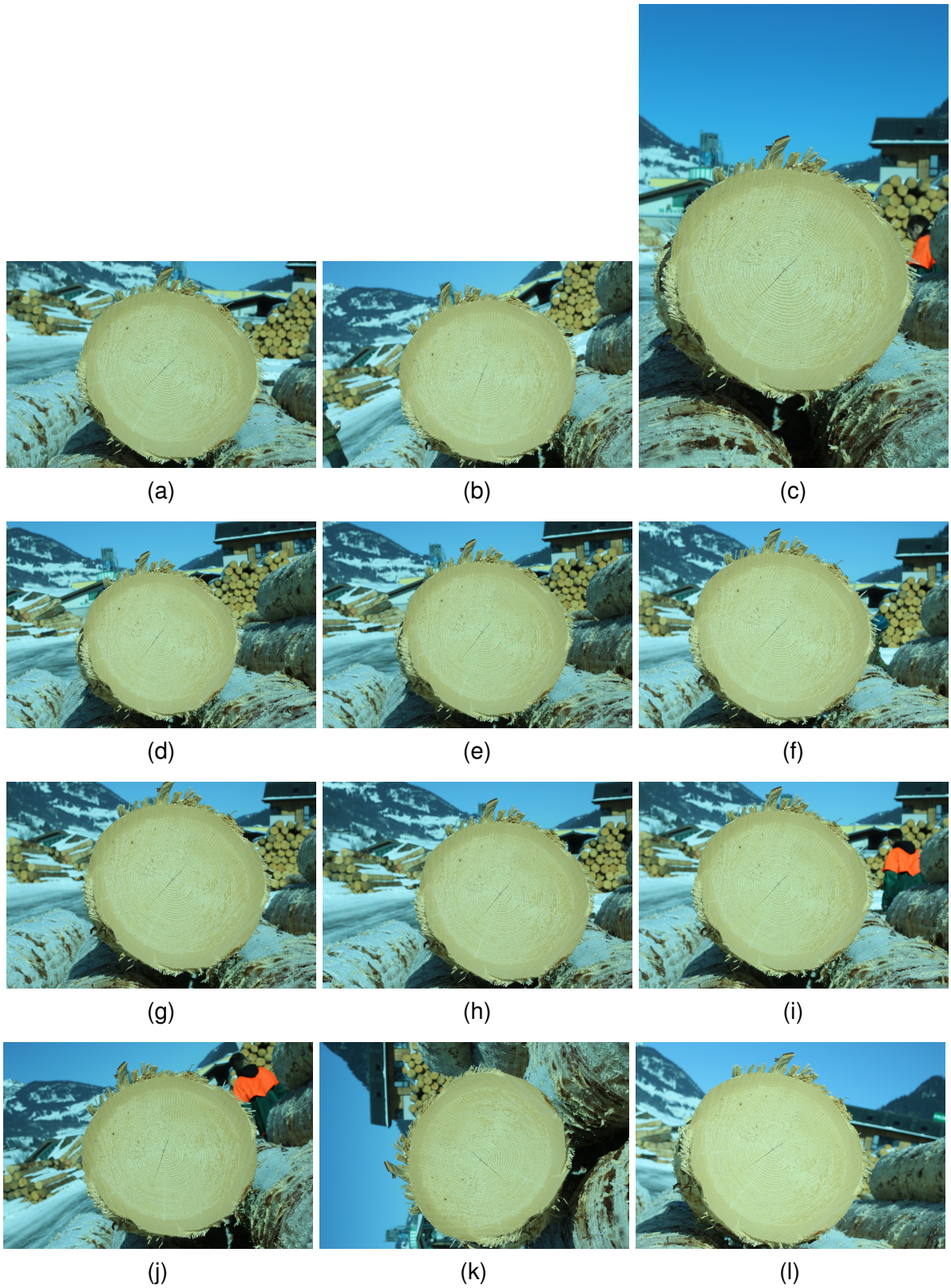
(d) Huawei.

(e) Ane.

Fonte: (DECELLE; JALILIAN, 2020)

As bases de dados Sbg_TS12 e Sbg_TS3 passaram por um intenso processo de aumento de dados (*data augmentation*). Segundo (DECELLE; JALILIAN, 2020), as principais técnicas empregadas para efetuar o aumento de dados foram: mudança de escala, rotação, deslocamento vertical, deslocamento horizontal, zoom e cisalhamento. Tome como referência a Figura 61, que apresenta um dos casos de aumento de dados presente na base de imagens.

Figura 61 – Aumento de dados aplicado na base de dados Sbg_TS3. Neste exemplo, uma imagem gerou 11 novas imagens.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Fonte: (DECELLE; JALILIAN, 2020)

4.2.3 Preparação da base de dados

Para cada uma das bases de dados, cerca de 70% das imagens foram selecionadas para compor um conjunto chamado de conjunto de treinamento, enquanto os 30% remanescentes foram atribuídos a um conjunto chamado de conjunto de validação. Como algumas bases de dados passaram por aumento de dados, esta divisão não foi feita de forma aleatória. A divisão aleatória pode fazer com que imagens aumentadas, oriundas da mesma imagem original, estejam presentes tanto no conjunto de treinamento quanto no conjunto de validação, causando o efeito chamado de vazamento de dados no treinamento (GUTS, 2018). Este efeito faz com que a rede neural profunda tenha seu desempenho superinflado durante o treinamento, impedindo que uma análise objetivo da sua real capacidade de generalização seja feita.

Esta base de dados foi criada para o processo de aprendizado supervisionado, portanto, é composta por pares entrada-saída (Figura 62). No entanto, as imagens desta base não estão padronizadas, possuindo diversas extensões como .jpg, .png e .tif e tamanhos (Tabela 3). Desta forma, como parte do processo de preparação, a base foi padronizada para a extensão .png e resolução 2048 $\times$ 1024. As máscaras de segmentação (Figura 62b), originalmente, são tratadas como máscaras binárias onde o valor 0 representa o fundo e o valor 255 representa a presença da madeira. As máscaras foram padronizadas da seguinte forma: [0,0,0] para o fundo e [1,1,1] para a madeira. Todos os devidos cuidados foram tomados para que as máscaras possuíssem apenas valores [0,0,0] e [1,1,1] após o processo de redução de resolução. Adicionalmente, as imagens foram divididas em pastas seguindo a padronização da base de dados (CORDTS *et al.*, 2016). Adicionalmente um sexto conjunto de dados foi criado, denominado aqui como *All*, contendo todas as imagens de todas as outras bases de dados. O conjunto *All* será tratado como um conjunto de dados normal e será utilizado nos testes e validações da mesma forma que os outros.

Figura 62 – Par (a) entrada e (b) saída da base de dados.

(a) Imagem original. (b) Saída verdadeira.

Fonte: (DECELLE; JALILIAN, 2020)

Naturalmente, a base original foi criada para o processo de segmentação semântica então possui apenas informações das máscaras de segmentação. Uma das contribuições deste trabalho é a criação de um segundo conjunto de saídas, agora contendo também informações da quantidade de anéis em cada imagem. Em suma, cada imagem agora também possui um número inteiro associado a ela, indicando a quantidade de anéis, *latewood* e *earlywood*, contados manualmente.

Para criar esta segunda base de dados, algumas imagens da base de dados originais foram selecionadas aleatoriamente. Esta nova base de dados, contendo informações sobre a quantia de anéis de crescimento, está apresentada na Tabela 4. Para cada uma das imagens nesta base de dados, há uma linha equivalente em um arquivo de texto contendo a quantidade de anéis que foram contados manualmente. Para mais detalhes, utilize a Figura 63 como referência. Como sugere a Figura 63, existem duas variações da base de dados: uma cujas imagens estão em seu formato original e outra onde as figuras já estão com a tora de madeira segmentada. Ambas variações são utilizadas neste trabalho para avaliar impactos na performance das redes neurais profundas utilizadas na tarefa de contagem de anéis de crescimento.

Tabela 4 – Informações da base de dados para contagem de anéis.

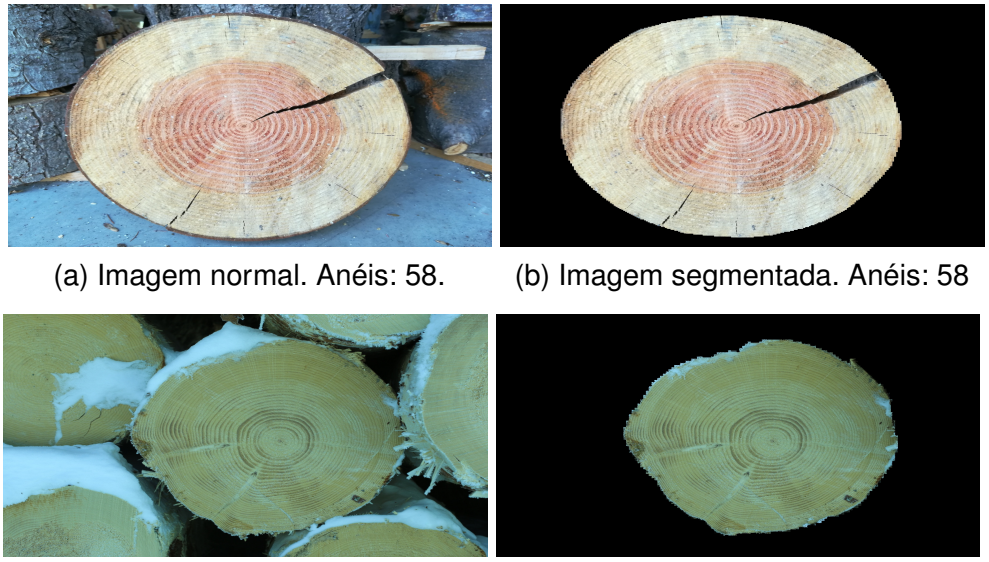
Nome da base	**Número de imagens**
Treino	
Sbg_TS12	147
Sbg_TS3	141
Lumix	22
Huawei	16
Ane	8
Validação	
Sbg_TS12	31
Sbg_TS3	35
Lumix	10
Huawei	5
Ane	3

Fonte: O autor.

4.3 IDENTIFICAÇÃO DE TORAS DE MADEIRA

A identificação das toras de madeira em imagens é um dos requisitos funcionais deste trabalho. Para executar esta tarefa, o presente trabalho aborda o tema através do uso de redes neurais profundas. A técnicas específica executada pelas redes neurais profundas é chamada de segmentação semântica, onde uma classe é atribuída para cada um dos *pixels* presentes na imagem. Na modelagem proposta, existem apenas duas classes: madeira ([1,1,1]) e fundo ([0,0,0]). Inicialmente os modelos de redes

Figura 63 – Exemplos encontados na base de contagem de anéis. Figuras (a) e (c) são imagens originais, figuras (b) e (d) são imagens com a máscara de segmentação aplicada.



(a) Imagem normal. Anéis: 58. (b) Imagem segmentada. Anéis: 58

(c) Imagem normal. Anéis: 41. (d) Imagem segmentada. Anéis: 41.

Fonte: O autor.

neurais profundas foram escolhidos com base na revisão da literatura, então foram implementados na linguagem de programação Python e por fim treinados. Todas as implementações envolvendo a identificação de toras de madeira podem ser encontradas neste repositório do GitHub (NACK; STIVANELLO; STEMMER, 2024).

4.3.1 Escolha dos modelos

Durante o processo de revisão da literatura, o trabalho escrito por (DECELLE; JALILIAN, 2020) foi aquele que melhor se assimilou com a proposta apresentada aqui. Desta forma, as redes neurais profundas para compor o conjunto de testes foram escolhidas segundo os seguintes critérios:

1. Rede neural profunda com o melhor desempenho, segundo as métricas propostas, no trabalho de (DECELLE; JALILIAN, 2020);
2. Rede baseada em CNNs amplamente adotada para a tarefa de segmentação semântica;
3. Rede baseada em Transformer com o melhor desempenho, segundo as métricas propostas;
4. Rede baseada em Transformer com o segundo melhor desempenho, segundo as métricas propostas;
5. Rede baseada em Transformer com a maior velocidade de inferência, medida em millisegundos.

A rede neural profunda com o melhor desempenho encontrada em (DECELLE; JALILIAN, 2020) é a U-Net, a CNN mais amplamente usada nas pesquisas encontradas foi a FastFCN, a melhor rede baseada em Transformer foi o SegFormer, a segunda melhor rede baseada em Transformer foi o Swin e a rede baseada em Transformer com o menor tempo de inferência foi o Twins. Os Transformers foram escolhidos para compor o conjunto de redes neurais por terem apresentado avanços muito promissores no campo da visão computacional nos últimos anos.

4.3.2 Implementação e treinamento

Todas as redes neurais profundas foram implementadas através da linguagem de programação Python. Dentre as diversas opções para implementar redes neurais, a escolhida foi o *framework* OpenMM Segmentation (CONTRIBUTORS, 2020). O OpenMM Segmentation é um *framework* baseado na biblioteca PyTorch 1.6+ e tem como principal objetivo permitir que pesquisadores compartilhem facilmente modelos de aprendizado profundo focados nas tarefas de segmentação semântica e segmentação de instâncias. A versão utilizada do OpenMM Segmentation é a 0.30.0. Os treinamentos foram conduzidos na plataforma *Amazon Web Services* (AWS), utilizando uma máquina virtual EC2 g4dn.2xlarge com as seguintes configurações:

- GPU: 1x Nvidia Tesla T4;
- CPU: 8x vCPU Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz;
- RAM: 32GB;
- Armazenamento: 256GB SSD.

O *framework* OpenMM Segmentation funciona através de arquivos de configurações, onde o usuário deve escolher em detalhes a arquitetura de rede neural profunda desejada, base de dados, transformações que as imagens de entrada sofrem, otimizador, taxa de aprendizado e outras informações. Na sequência estão apresentados os detalhes de cada implementação.

4.3.2.1 U-Net

A U-Net é a rede com o melhor desempenho apresentada no trabalho de (DECELLE; JALILIAN, 2020). Naturalmente, U-Net é um termo geral e existem formas diversas de implementar esta rede, podendo variar a quantidade de camadas de convolução, quantidade de estágios e outros parâmetros. Dentro do arquivo de configuração do *framework* OpenMM Segmentation, a rede foi montada da seguinte forma:

- **in_channels (int)**: Número de canais de imagem de entrada. Escolhido como 3 por serem imagens RGB.

- **base_channels (int)**: Número de canais base de cada estágio. Também conhecido como a quantidade de *kernels* nas primeiras operações de convolução. Valor original mantido: 64.
- **num_stages**: Quantidade de estágios no *encoder*. Valor original mantido: 5.
- **stride**: Passo com o qual a operação de convolução avança na entrada. Valor original mantido: 1.
- **kernel_size**: Tamanho do *kernel*. Valor original mantido: 3×3.
- **act_cfg**: Função de ativação. Função original mantida. ReLU.
- **enc_num_convs**: Quantidade de convoluções por estágio. Valor original mantido: 2.
- **loss_decode**: Função de perda no decodificador. Função original mantida: *Cross-Entropy*
- **optimizer**: Otimizador utilizado para atualizar os pesos da rede. Otimizador escolhido através de testes: SGD com momentum.
- **lr**: Taxa de aprendizado. Valor escolhido através de testes: 0.01.

Inicialmente o modelo foi pré-treinado na base de dados ImageNet (DENG *et al.*, 2009) e então passou por um processo de ajuste fino por todas as 6 bases deste trabalho. Na Tabela 5 estão apresentadas algumas informações do treinamento. Os modelos foram treinados até a marca de 50 épocas. O *checkpoint* escolhido foi aquele que apresentou as maiores métricas mIoU e mAcc.

Tabela 5 – Treinamento da U-Net.

U-Net	**Ane**	**Lumix**	**Huawei**	**Sbg_TS13**	**Sbg_TS12**	**All**
Épocas	11	8	9	17	19	23
Batch	1					
Memória	13GB					

Fonte: O autor.

4.3.2.2 FastFCN

A rede FastFCN foi proposta inicialmente por (WU *et al.*, 2019) e, da mesma forma que a U-Net, pode ser montada de diversas formas. Nesta implementação em específico o *backbone* da rede é formado por uma ResNet50-C (HE, T. *et al.*, 2018)(SZEGEDY *et al.*, 2016) que é uma variação da ResNet50 convencional (HE, K. *et al.*, 2016). Essa variação da ResNet substitui uma convolução com *kernel* 7×7 por três convoluções seguidas cujo *kernel* tem tamanho 3×3.

A implementação da FastFCN neste trabalho também é feita através do *framework* OpenMM Segmentation. Os parâmetros do arquivo de configuração podem ser vistos à seguir:

- **depth**: Profundidade da ResNet, podendo ser 18, 34, 50, 101 e 152. Escolhido o valor original: 50.
- **num_stages**: Número de estágios da ResNet. Escolhido valor original: 4.
- **strides**: Valor do *stride* dos primeiros blocos de cada estágio. Escolhido valor original: (1,2,2,2).
- **neck**: Módulo intermediário entre o *backbone* e o *decoder*. Escolhido o proposto por (WU *et al.*, 2019): JPU.
- **in_channels**: Número de canais de entrada para cada operação de convolução antes do processo de aumento de resolução. Mantido o valor original: (512, 1024, 2048).
- **decoder**: Camada que decodifica a saída do bloco intermediário em uma máscara de segmentação. Bloco escolhido: EncNet (ZHANG *et al.*, 2018).
- **loss_decode**: Função de perda no decodificador. Função original mantida: *Cross-Entropy*
- **optimizer**: Otimizador utilizado para atualizar os pesos da rede. Otimizador escolhido através de testes: SGD com momentum.
- **lr**: Taxa de aprendizado. Valor escolhido através de testes: 0.01.

Da mesma forma que a U-Net, o modelo foi pré-treinado na base de dados ImageNet (DENG *et al.*, 2009) e então passou por um processo de ajuste fino por todas as 6 bases deste trabalho. Na Tabela 6 estão apresentadas algumas informações do treinamento. Os modelos foram treinados até a marca de 50 épocas. O *checkpoint* escolhido foi aquele que apresentou as maiores métricas mIoU e mAcc.

Tabela 6 – Treinamento da FastFCN.

FastFCN	**Ane**	**Lumix**	**Huawei**	**Sbg_TS13**	**Sbg_TS12**	**All**
Épocas	6	9	8	15	16	21
Batch	1					
Memória	10GB					

Fonte: O autor.

4.3.2.3 SegFormer

A rede SegFormer foi proposta por (XIE *et al.*, 2021) e é a rede com o melhor desempenho encontrado dentre aquelas que foram avaliadas em bases de dados

públicas, como o ADE20K (ZHOU *et al.*, 2017) (ZHOU *et al.*, 2019). Dentre as possíveis variações do SegFormer, neste trabalho a configuração implementada é apresentada no trabalho original como SegFormer-MiT-B0 e sua configuração está apresentada à seguir:

- **in_channels (int)**: Número de canais de imagem de entrada. Escolhido como 3 por serem imagens RGB.
- **embed_dims (int)**: Dimensão do espaço (C) para o qual os dados são mapeados durante o processo de incorporação. Valor escolhido: 32.
- **num_stags (int)**: Número de estágios no *backbone*. Mantido o valor original: 4.
- **num_layers**: Quantidade de camadas para cada camada de *encoder* Transformer. Valor padrão mantido: (2, 2, 2, 2).
- **num_heads**: Quantidade de *attention heads* em cada bloco de *encoder*. Valor padrão mantido: (1, 2, 5, 8).
- **patch_sizes**: Tamanho do *patch* para cada operação de *patch overlapping*. Valor padrão mantido: (7, 3, 3, 3).
- **sr_ratios**: Taxa de redução espacial em cada uma das camadas de *encoding*. Valor padrão mantido: (8, 4, 2, 1).
- **mlp_ratio**: Taxa de expansão da camada *feed-forward* (MLP) (LIN, T. *et al.*, 2022a). Mantido o valor padrão: 4.
- **decoder**: Camada que decodifica a saída das camadas de *encoding* do Transformer em uma máscara de segmentação. Bloco escolhido: SegformerHead proposta pelo autores originais (XIE *et al.*, 2021).
- **loss_decode**: Função de perda no decodificador. Função original mantida: *Cross-Entropy*
- **optimizer**: Otimizador utilizado para atualizar os pesos da rede. Otimizador escolhido através de testes: AdamW com betas 0.9 e 0.999.
- **lr**: Taxa de aprendizado. Valor escolhido através de testes: 6×10^{-5}.

Seguindo o exemplo das redes neurais profundas anteriores, o modelo foi pré-treinado na base de dados ImageNet (DENG *et al.*, 2009) e então passou por um processo de ajuste fino por todas as 6 bases deste trabalho. Na Tabela 7 estão apresentadas algumas informações do treinamento. Os modelos foram treinados até a marca de 50 épocas. O *checkpoint* escolhido foi aquele que apresentou as maiores métricas mIoU e mAcc.

Tabela 7 – Treinamento do SegFormer.

SegFormer	**Ane**	**Lumix**	**Huawei**	**Sbg_TS13**	**Sbg_TS12**	**All**
Épocas	35	34	20	9	7	5
Batch	4					
Memória	13GB					

Fonte: O autor.

4.3.2.4 Swin

O Swin Transformer foi originalmente proposto por (LIU, Z. *et al.*, 2021) e atingiu desempenhos surpreendentes em todas as principais tarefas de visão computacional nas principais bases de dados públicas. Este trabalho se concentra em reproduzir a maior versão proposta pelos autores originais, o Swin-L, que é criado através das seguintes configurações no *framework* OpenMM Segmentation:

- **in_channels (int)**: Número de canais de imagem de entrada. Escolhido como 3 por serem imagens RGB.
- **embed_dims (int)**: Dimensão do espaço (C) para o qual os dados são mapeados durante o processo de incorporação. Valor escolhido: 192.
- **patch_size (int)**: Tamanho do *patch*. Valor padrão mantido: 4.
- **window_size**: Tamanho das janelas deslocadas propostas por (LIU, Z. *et al.*, 2021). Valor padrão mantido: 12.
- **mlp_ratio**: Taxa de expansão da camada *feed-forward* (MLP) (LIN, T. *et al.*, 2022a). Mantido o valor padrão: 4.
- **depths**: Profundidade de cada estágio do *encoder* Swin Transformer. Valor padrão mantido: (2,2,18,2).
- **num_heads**: Quantidade de *attention heads* em cada bloco de *encoder*. Valor padrão mantido: (6,12,24,48).
- **act_cfg**: Função de ativação. Função original mantida. GELU.
- **decoder**: Camada que decodifica a saída das camadas de *encoding* do Transformer em uma máscara de segmentação. Bloco original mantido: UPer-Net (XIAO *et al.*, 2018).
- **loss_decode**: Função de perda no decodificador. Função original mantida: *Cross-Entropy*
- **optimizer**: Otimizador utilizado para atualizar os pesos da rede. Otimizador escolhido através de testes: AdamW com betas 0.9 e 0.999.
- **lr**: Taxa de aprendizado. Valor escolhido através de testes: 6×10^{-5}.

Novamente, o modelo foi pré-treinado na base de dados ImageNet (DENG *et al.*, 2009) e então passou por um processo de ajuste fino por todas as 6 bases deste trabalho. Na Tabela 8 estão apresentadas algumas informações do treinamento. Os modelos foram treinados até a marca de 50 épocas. O *checkpoint* escolhido foi aquele que apresentou as maiores métricas mIoU e mAcc.

Tabela 8 – Treinamento do Swin.

Swin	**Ane**	**Lumix**	**Huawei**	**Sbg_TS13**	**Sbg_TS12**	**All**
Épocas	28	27	22	6	4	4
Batch	2					
Memória	13GB					

Fonte: O autor.

4.3.2.5 Twins

A última rede implementada neste trabalho para a tarefa de identificação de toras de madeiras em imagens é chamada de Twins, proposta por (CHU *et al.*, 2021). De forma similar as outras redes, este modelo possuir diversas variações. A versão do Twins implementada neste trabalho é apresentada no artigo original como Twins-PCPVT-B e é o modelo intermediários proposto pelos autores. A configuração deste modelo no *framework* OpenMM Segmentation é a seguinte:

- **in_channels (int)**: Número de canais de imagem de entrada. Escolhido como 3 por serem imagens RGB.
- **embed_dims (int)**: Dimensão do espaço (C) para o qual os dados são mapeados durante o processo de incorporação. Valor original mantido: (64, 128, 320, 512). Esta rede realiza múltiplos processos de mapeamento de dados.
- **patch_size (int)**: Tamanho do *patch*. Valor padrão mantido: (4, 2, 2, 2). Como a rede realiza múltiplos processos de mapeamento de dados, existe mais de um tamanho de *patch*.
- **mlp_ratio**: Taxa de expansão da camada *feed-forward* (MLP) (CHU *et al.*, 2021). Mantido o valor padrão: (8, 8, 4, 4).
- **depth**: Profundidade de cada estágio. Valor escolhido: (3,4,18,3).
- **neck**: Módulo intermediário entre o *backbone* e o *decoder*. Escolhido o proposto por (CHU *et al.*, 2021): FPN (LIN, T.-Y. *et al.*, 2017).
- **decoder**: Camada que decodifica a saída da camada intermediária do Transformer em uma máscara de segmentação. Bloco original mantido: FPN (LIN, T.-Y. *et al.*, 2017).

- **loss_decode**: Função de perda no decodificador. Função original mantida: *Cross-Entropy*
- **optimizer**: Otimizador utilizado para atualizar os pesos da rede. Otimizador escolhido através de testes: AdamW com betas 0.9 e 0.999.
- **lr**: Taxa de aprendizado. Valor escolhido através de testes: 1×10^{-5}.

Por fim, seguindo o exemplo das redes anteriores, o modelo foi pré-treinado na base de dados ImageNet (DENG *et al.*, 2009) e então passou por um processo de ajuste fino por todas as 6 bases deste trabalho. Na Tabela 9 estão apresentadas algumas informações do treinamento. Diferente das outras redes, o modelo Twins precisou ser treinado até a marca de 100 épocas por apresentar dificuldade em generalizar nas bases de dados pequenas. O *checkpoint* escolhido foi aquele que apresentou as maiores métricas mIoU e mAcc.

Tabela 9 – Treinamento do Swin.

Twins	**Ane**	**Lumix**	**Huawei**	**Sbg_TS13**	**Sbg_TS12**	**All**
Épocas	96	47	45	8	7	5
Batch	4					
Memória	13GB					

Fonte: O autor.

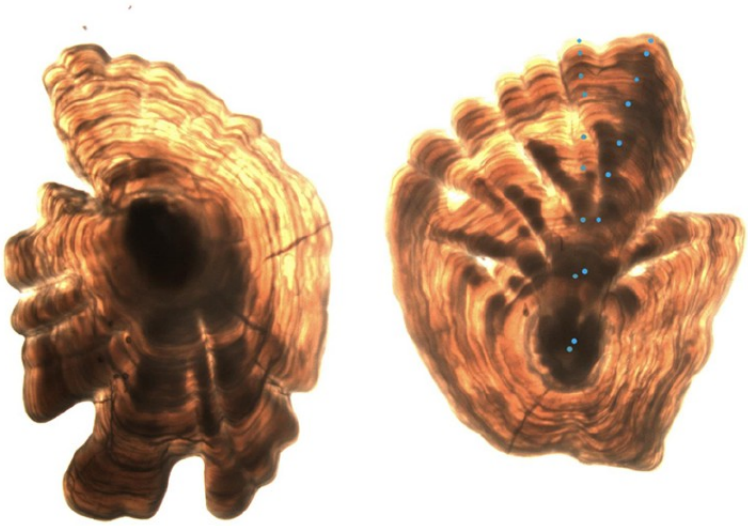
4.4 CONTAGEM DE ANÉIS EM TORAS DE MADEIRA

A estimativa de anéis de crescimento em imagens de toras de madeira é um dos requisitos funcionais deste trabalho. Para executar esta tarefa, este trabalho aborda o tema através do uso de redes neurais profundas. A técnica utilizada neste trabalho para abordar o problema é chamada de regressão e tem como principais objetivo estimar um número real à partir das imagens de entrada. A base de dados utilizada nesta tarefa é aquela criada neste trabalho especificamente para esta tarefa. A base possui duas variações: uma com imagens normais e outra com imagens segmentadas. Ambas foram utilizadas para treinar as redes individualmente a fim de comparar os resultados. Todas as implementações desta seção foram feitas na linguagem de programação Python com auxílio do framework PyTorch.

Apesar de não envolver a análise de madeiras, em (MARTINSEN; HARBITZ; BIANCHI, 2022) é apresentado um trabalho que tem como objetivo estimar a idade de Alabotes da Groelândia (*Reinhardtius hippoglossoides*) através de imagens feitas de seus otólitos (Figura 64), que são concreções de carbonato de cálcio presentes dentro de câmaras no aparelho vestibular do ouvido interno dos vertebrados. Note que, de certa forma, os otólitos se assemelham muito a uma imagem da seção reta de uma tora de madeira, apresentando também anéis de crescimento. Os autores propõe o

uso de da CNN Xception (CHOLLET, 2017), adaptada para utilizar a função de perda MSE. Neste sentido, o desafio é tratado como um problema de regressão. Como a Figura 64 tem certa similaridade com as imagens de seções retas de toras de madeira, a abordagem escolhida é promissora.

Figura 64 – Imagens dos otólitos dos Alabotes da Groelândia.



Fonte: Adaptado de (MARTINSEN; HARBITZ; BIANCHI, 2022).

4.4.1 Escolha dos modelos

Durante o processo de revisão da literatura, diversas redes neurais profundas foram encontradas. No entanto, o grupo de redes ResNet (HE, K. *et al.*, 2016) foi aquele que mais se destacou por ser aplicado em tarefas gerais. Nesse sentido, para a tarefa de contagem de anéis de crescimento, as redes ResNet18, ResNet34, ResNet50 e ResNet101 foram utilizadas. O artigo original utiliza a família de redes ResNet para a tarefa de conhecimento de imagens, então, neste trabalho a implementação destas redes modifica a última camada para a tarefa de regressão. Ao invés de existirem *n* neurônios na saída, como proposto por (HE, K. *et al.*, 2016) para a tarefa de classificar *n* classes, este trabalho utiliza apenas 1.

4.4.2 Implementação e treinamento

As redes ResNet desta seção foram imlpementadas conforme visto em(HE, K. *et al.*, 2016) e seguem a seguinte nomenclatura:

- ResNet18-Norm-CR;

- ResNet18-Seg-CR;
- ResNet34-Norm-CR;
- ResNet34-Seg-SR;
- ResNet50-Norm-CR;
- ResNet50-Seg-SR;
- ResNet101-Norm-SR;
- ResNet101-Seg-SR

Onde ResNetXX indica a profundidade da rede, seguindo a mesma arquitetura de (HE, K. *et al.*, 2016), Norm indica que a rede foi treinada na base de dados com imagens normais, Seg indica que a rede foi treinada na base de dados com imagens segmentadas, SR indica que a última cada da rede não possuí função de ativação e CR indica que a última camada da rede utiliza a função de ativação ReLU. A inclusão e exclusão da ativação na última camada da rede foi determinada experimentalmente de acordo com a curva *Loss* nos treinamentos.

A última camada de todas as redes implementadas foi substituída por um MLP com 512 valores de entrada e apenas 1 valor de saída, para efetuar a regressão. Anterior ao MLP, uma camada de *adaptative average pooling* é aplicada para garantir a dimensão da entrada do MLP, independetemente do tamanho da imagem de entrada. O *adaptative average pooling* realiza o cálculo da média dos valores de sua entrada enquanto mantém sua profundidade, isto é, para uma entrada de tamanho $H \times W \times D$, a saída sempre será $1 \times 1 \times D$.

Em todos os casos, o otimizador escolhido é o SGD. Todas as redes utilizam as função de perda L1. Similarmente, em todos os casos as redes foram treinadas até a marca de 200 épocas. O *checkpoint* escolhido para cada uma delas foi determinado de acordo com o menor valor de perda encontrado ou caso indícios de sobreajuste (2.3.5.2) fossem detectados. Os valores da taxa de aprendizado foram ajustados iterativamente durante o treinamento. As métricas utilizadas para avaliar a performance das redes na tarefa de regressão são: MSE (*Mean Square Error*), MAE (*Mean Absolute Error*) e R2 *Score* ou coeficiente de determinação.

5 RESULTADOS

Este capítulo apresenta os resultados obtidos pelos algoritmos de identificação de toras de madeira e contagem de anéis de crescimento, selecionados e configurados conforme descrição realizada na Seção 4.3.1 e Seção 4.4.1, empregados sobre as bases de dados indicadas na Seção 4.2.2 e Seção 4.2.3. Para a identificação de madeira, os testes foram conduzidos em todas as 6 bases de dados propostas: Ane, Lumix, Huawei, Sbg_TS12, Sbg_TS3 e All. As métricas de avaliação usadas para a identificação de madeiras são mIoU e mAcc. Para a contagem de anéis de crescimento, os testes foram conduzidos em uma base de dados criada especificamente para esta atividade. As métricas utilizadas na avaliação de desempenho desta etapa são MSE, MAE e R2-Score.

5.1 RESULTADOS DOS ALGORITMOS DE IDENTIFICAÇÃO DE MADEIRA

Após o processo de treinamento, os pesos das redes neurais profundas foram escolhidos de acordo com o melhor desempenho apresentado nas fases de validação. A validação foi realizada a cada época do treinamento. A Tabela 10 apresenta os resultados da inferência de todas as redes para cada uma das bases de dados. Em negrito, na Tabela 10, estão destacados todos os melhores resultados para cada uma das bases de dados. Adicionalmente, a Figura 65 apresenta os resultados visuais da segmentação para uma imagem de cada base de dados em comparação com a segmentação esperada (*ground truth*). Na Figura 65 os *pixels* branco representam as predições corretas da classe "madeira", os *pixels* pretos representam as predições corretas da classe "fundo"e os *pixels* vermelhos representam predições incorretas de qualquer uma das duas classes.

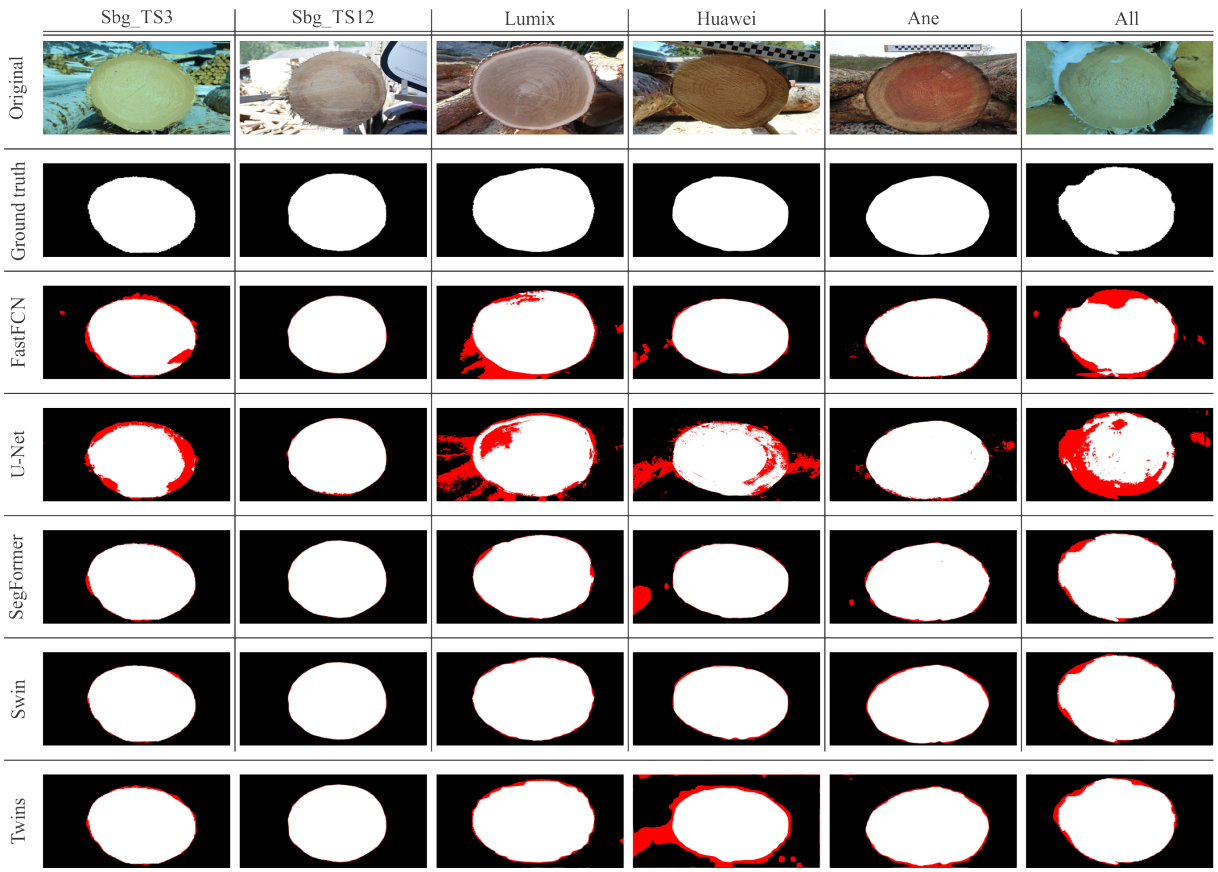
Tabela 10 – Resultados das inferências de todas as redes neurais profundas propostas para a tarefa de segmentação semântica de toras de madeira.

Fonte	Ane		Lumix		Huawei		Sbg_TS3		Sbg_TS12		All	
Métricas	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU	mAcc	mIoU
FastFCN	**97.58**	**95.10**	89.72	78.46	94.20	88.76	83.50	67.79	92.30	87.04	83.23	64.06
U-Net	96.31	93.09	84.04	70.91	83.76	74.00	70.06	49.33	95.57	89.65	73.37	53.50
SegFormer	96.07	92.60	92.78	85.62	96.32	90.87	94.05	86.52	99.12	98.13	95.86	90.71
Swin	97.17	94.83	**95.33**	**90.98**	**96.92**	**93.20**	**96.75**	**93.42**	**99.26**	**98.43**	**97.49**	**93.82**
Twins	94.99	93.12	89.43	81.45	88.94	84.53	89.98	78.99	98.10	95.74	94.21	87.37

Fonte: O autor.

A rede FastFCN, segundo a Tabela 10, apresentou o melhor resultado na base de dados Ane para ambas as métricas. Na base Sbg_TS12, apresentou os piores resultados entre todas as redes. Como pode ser visto na Figura 65, a rede apresenta boa precisão em termos de estimativa de *pixel*, estimativa de formato e estimativa de tamanho, apesar de ser possível verificar erros na borda da tora nas bases Lumix,

Figura 65 – Result of the inference of networks on an image from each set of images base.



Fonte: O autor.

Sbg_TS3 e All, além dos ruídos oriundos de outras madeiras na imagem, como visto no exemplo Huawei. De modo geral, esta arquitetura apresentou apenas grande dificuldade em generalizar a base Sbg_TS3, pelo excesso da presença de múltiplas toras na mesma imagem.

Quanto a rede U-Net, os resultados indicam que, em geral, apresentou a pior performance em comparação com todas as outras redes. Quando treinada na base de dados Ane, a U-Net foi capaz de ultrapassar os resultados obtidos pelo SegFormer por 0.49 pontos na métrica mIoU e também foi capaz de superar o SegFormer e o Twins por 0.24 e 1.32 pontos, respectivamente, na métrica mAcc. Quando treinada na base de dados Sbg_TS12, a U-Net foi capaz de apenas superar a rede FastFCN por 2.61 pontos na métrica mIoU e 3.17 pontos na métrica mAcc. As métricas também indicam que nestas bases de dados a U-Net foi capaz atingir boa precisão a nível de *pixel* e formato da madeira, apesar de apresentar pequenas falhas na predição do fundo. Por outro lado, nas outras bases de dados, verifica-se que tanto a predição a nível de *pixel* quanto a nível de formato foram amplamente impactadas. A U-Net apresenta, em geral, diversas falhas na região interna da tora de madeira (Huawei, Lumix e All) e também gera diversos ruídos laterais como madeira (Huawei, Lumix). Na base de

dados Sbg_TS3, a U-Net apenas apresentou problemas na estimativa de escala da madeira, preservando razoavelmente o formato esperado e gerando ruídos laterais mínimos. No entanto, este erro de subdimensionamento foi suficiente para gerar os piores resultados desta rede em ambas as métricas.

A rede SegFormer atingiu resultados considerados bons e consistentes. Em todos os casos de treinamento, sua acurácia a nível de *pixel* ficou acima da marca de 90%, enquanto sua pontuação na métrica mIoU se manteve sempre acima de 85%. As imagens de inferência na Figura 65 claramente demonstram que esta rede não apresenta maiores problemas com ruídos ou mesmo com o formato do objeto segmentado, isto é, este modelo apenas apresenta leve deformação nas bordas da madeira. De forma geral, as métricas apresentadas pelo SegFormer são bastante promissoras e indicam que esta rede é bastante estável.

A rede Swin apresentou os melhores resultados em termos gerais neste trabalho. Todas as métricas de precisão de *pixel* (mAcc) se mantiveram acima da marca de 90%, enquanto os resultados para a métrica mIoU se mantiveram acima de 90%. Considerando o tamanho das bases de dados, estes resultados se mostraram bastante promissores e evidenciam novamente os avanços das redes baseadas em Transformer. Os resultados de inferência produzidos pelo Swin são, de modo geral, muito parecidos com aqueles produzidos pelo SegFormer.

A rede Twins gerou os piores resultados quando comparada com os outros modelos que utilizam o mecanismo Transformer. Na base de dados Ane, o Twins superou o SegFormer por 0.51 pontos na métrica mIoU, mas perdeu todas as outras comparações, tanto para o SegFormer quanto para o Swin. Quanto a sua performance em comparação com as outras redes baseadas em CNNs, o Twins foi capaz de superar os resultados da U-Net e da FastFCN em ambas as métricas nas bases de dados Sbg_TS12, Sbg_TS3 e All. Na base Ane, perdeu na métrica mAcc para U-Net (1.32 pontos) e FastFCN (2.59 pontos), e na métrica mIoU perdeu para FastFCN (1.98 pontos). Na base Lumix, houve perda para FastFCN na métrica mAcc (0.29 pontos), mas venceu em outras comparações. Já para a base Huawei, houve perda em ambas as comparações com FastFCN e vitória em ambas as comparações com U-Net. Considerando esses resultados, é possível argumentar que Twins supera FastFCN e U-Net na maioria dos casos. Os resultados obtidos com a rede Twins são bastante parecidos com aqueles vistos nas redes SegFormer e Swin, tanto em formato quanto em precisão de *pixel*. A única exceção aqui é o resultado visto na base de dados Huawei, que apresentou grandes problemas relacionados à confusão com o fundo e bordas da imagem.

Em geral, podemos separar os resultados em duas categorias mais amplas: redes baseadas em CNNs (U-Net e FastFCN) e redes baseadas em Transformers (SegFormer, Swin e Twins). Os resultados das métricas deixam claro que arquiteturas

baseadas em Transformers alcançam melhores resultados na tarefa proposta. Essa diferença se deve ao mecanismo de atenção em redes baseadas em Transformers, que permite capturar contextos globais e locais simultaneamente, possibilitando tanto uma boa precisão em nível de pixel quanto uma boa precisão em nível de forma (IoU). Por outro lado, as CNNs têm janelas de atenção limitadas devido ao tamanho de seus kernels e à profundidade da rede, o que não captura eficientemente esses contextos mais complexos. Neste caso específico, não estamos apenas tentando segmentar madeira, mas especificamente o tronco central de madeira na imagem. Os resultados mostram que as CNNs enfrentaram dificuldades principalmente nesses casos.

Entre as redes modernas baseadas em Transformer, a arquitetura Swin parece se sair melhor nesta tarefa devido à sua complexidade, principalmente devido à sua Atenção Espacialmente Variante. Por outro lado, a arquitetura Twins demonstra uma boa eficiência de processamento devido ao seu design inovador, que sugere o processamento de dados por diferentes caminhos. Por fim, SegFormer parece ser um bom equilíbrio, oferecendo velocidade moderada e entregando bons resultados. No entanto, é importante ressaltar que a versão do SegFormer implementada neste trabalho é a menor proposta por (XIE *et al.*, 2021), isto é, possivelmente resultados melhores poderiam ser atingidos se suas variações mais complexas fossem testadas.

Outra informação importante pode ser encontrada ao analisar as Tabelas 5, 6, 7, 8 e 9. Os dados destas tabelas são analisados melhor através da Figura 66, que apresenta a relação entre a quantidade de épocas de treinamento necessárias versus o tamanho da base de dados que a rede neural está sendo treinada. Este gráfico sugere que as redes baseadas em CNNs necessitam de mais épocas à medida que as bases de dados vão aumentando, enquanto as redes baseadas em Transformers apresentam o cenário inverso: quanto maior a base, menos épocas são necessárias.

O tempo médio de inferência de cada uma das redes propostas pode ser visto na Tabela 11. Ao contrário do treinamento, as inferências foram realizadas em um computador pessoal, sem o uso de placas gráficas. Para isso, foi utilizado um processador Ryzen(R) 7-4800H juntamente com 16GB de memória RAM @ 3200MHz.

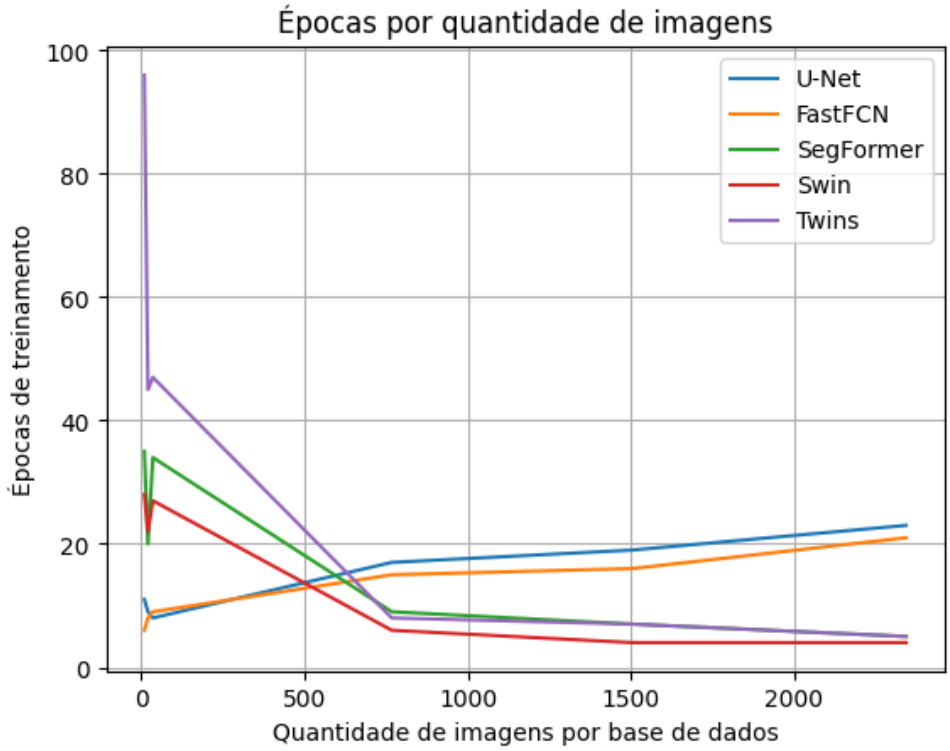
Tabela 11 – Tempo médio de inferência para cada arquitetura em segunds (s). Testes conduzidos utilizando uma CPU Ryzen(R) 7-4800H.

	FastFCN	U-Net	SegFormer	Swin	Twins
Inferência (s)	6.7	9.1	5.1	7.2	2.7

Fonte: O autor.

A avaliação dos resultados da Tabela 11, juntamente com os resultados presentes na Tabela 10, evidencia que a complexidade computacional das redes não implica necessariamente seu desempenho em uma tarefa específica. Neste trabalho, é possível verificar que a rede com os piores resultados de segmentação (U-Net) também

Figura 66 – Relação entre a quantidade de épocas necessárias para treinar a rede neural e a quantidade de imagens na base de dados.



Fonte: O autor.

apresenta os piores resultados relacionados à velocidade. Além disso, é importante destacar que redes que utilizam o mecanismo de Transformer não necessariamente serão lentas, como evidenciado pelo exemplo da rede Twins neste trabalho, alcançando a maior velocidade entre todas para inferência em uma CPU. No entanto, vale ressaltar que a complexidade computacional do mecanismo de atenção é $O(n^2)$, então imagens com alta resolução certamente afetam muito a performance do sistema.

Quando os resultados deste trabalho e os resultados apresentados por (DECELLE; JALILIAN, 2020) são comparados, é possível observar ligeiro aumento das métricas mAcc e mIoU. No entanto, a principal melhoria percebida foi na consistência das redes em diferentes conjuntos de dados. Conforme mostrado na Tabela 10, a arquitetura Swin foi capaz de produzir resultados consistentes em todos os conjuntos de dados, ao contrário da RefineNet em (DECELLE; JALILIAN, 2020). Ainda, deve ser mecionado que os resultados da U-Net apresentados em (DECELLE; JALILIAN, 2020) não puderam ser reproduzidos. Finalmente, os resultados deste artigo indicam que novas redes neurais profundas, especialmente baseadas em Transformer, são preferíveis às mais antigas nesta tarefa específica.

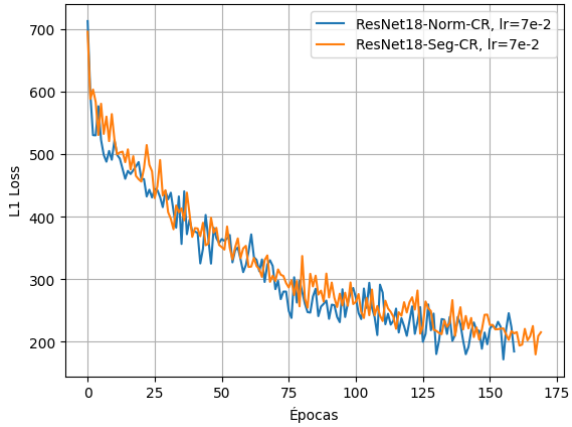
5.2 RESULTADOS DOS ALGORITMOS DE CONTAGEM DE ANÉIS

Esta seção trata dos resultados obtidos pelas redes neurais profundas aplicadas na tarefa de contagem de anéis de crescimento em toras de madeira. Ao total, 8 redes foram treinadas para esta tarefa. Destas, 4 foram treinadas na base de dados contendo imagens normais e 4 foram treinadas na base contendo imagens segmentadas. A Figura 67 apresente as curvas de treinamento de todas estas redes.

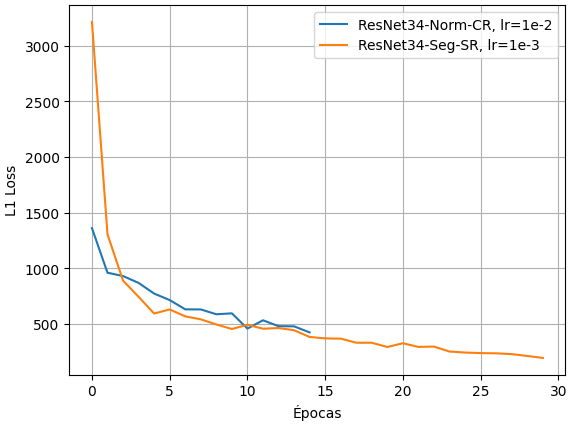
O treinamento das redes ResNet18, apresentado na Figura 67a, não evidencia nenhuma melhora aparente entre usar a base de dados comum ou a base de dados segmentada. Ambas as redes ResNet18 apresentaram treinamentos bastantes instáveis, no entanto, a rede ResNet18-Norm-CR apresentou características de sobreajuste com menos épocas. Ambas as ResNets18 finalizaram o treinamento com valores de perda similares. Por outro lado, o treinamento das redes ResNet34 indicam que há diferenças entre treinar as redes na base normal e na base segmentada. A ResNet34-Norm-CR apresentou sintomas de sobreajuste nas primeiras 15 épocas, enquanto a ResNet34-Seg-SR não apresentou sintomas de sobreajuste mas estabilizou seu valor de perda por volta da época 30. De forma geral, a ResNet34 atingiu valores de perda menores e aparenta ter se beneficiado da base de dados segmentada. A ResNet50, de forma similar, apresentou diferenças entre as bases utilizadas. A ResNet50-Norm-CR apresentou treinamento mais lento, valores de perda maiores e características de sobreajuste por volta da época 50, enquanto sua versão ResNet50-Seg-SR apresentou valores menores de perda, curva de treinamento mais acentuada e estabilizou seu valor de perda por volta da época 70 sem apresentar características de sobreajuste. Estes dados sugerem que o uso da base de dados segmentada melhorou seu desempenho na tarefa. Por fim, a ResNet101 apresentou situações similares, onde a ResNet101-Seg-SR atingiu valores de erda ligeiramente menores e estabilizou por volta da época 60. Sua equivalente, treinada na base de dados comum (ResNet101-Norm-SR) também estabilizou o valor de perda por volta da época 60. Ambas ResNet101 não apresentaram sintomas de sobreajuste e seus tempos de treinamento foram similares. Por conta da pequena diferença no valor de perda final, os treinamentos das ResNets101 sugerem leve benefício em utilizar a base segmentada.

Dando sequência à análise, a Tabela 12 mostra todos os resultados obtidos para as métricas MSE, MAE e R2-Score. Os resultados indicam claramente que as redes ResNet18 e ResNet50 tiveram suas medidas de performance severamente impactadas quando expostas a base de treinamento previamente segmentada. A rede ResNet101, por outro lado, apresentou significative melhora em todas as métricas utilizadas. A rede ResNet34, por outro lado, apresentou melhora em 3 avaliações e piora nas 3 avaliações restantes. Destacado em negrito, na Tabela 12, estão indicadas as redes que obtiveram os melhores resultados em cada métrica, separadas por testes no conjunto de treinamento e no conjunto de validação. De forma geral, a rede ResNet34-

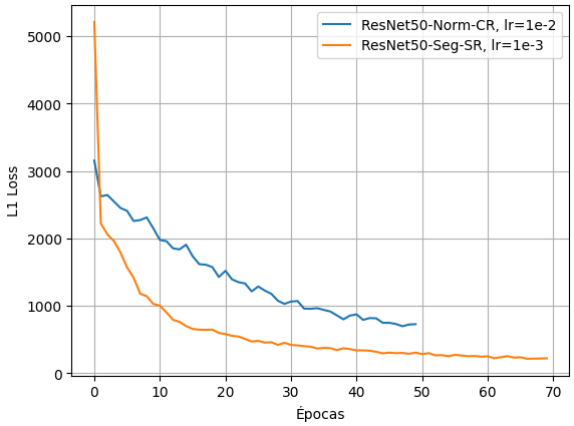
Figura 67 – Perda L1 das redes ao longo das épocas de treinamento. As figuras (a), (b), (c) e (d) mostram o comparativo entre treinar as redes na base de dados normal e na base de dados segmentada.



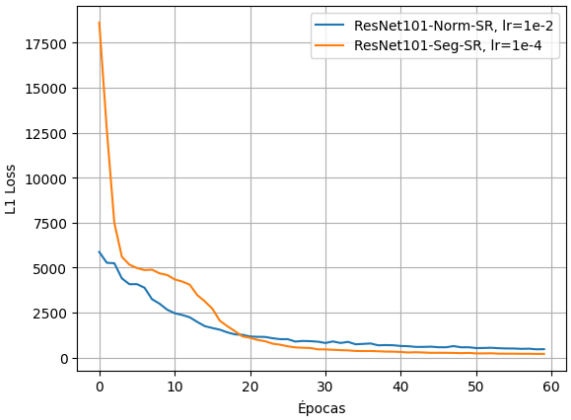
(a) ResNet18.



(b) ResNet34



(c) ResNet50.



(d) ResNet101.

Fonte: O autor.

Seg-SR apresentou a maioria das vitórias nas comparações de performance.

O MSE é uma métrica amplamente utilizada em tarefas de regressão para medir a média dos quadrados das diferenças entre os valores previstos e os valores reais. Essa métrica quantifica a magnitude média dos erros ou desvios das previsões em relação aos valores verdadeiros. Erros maiores têm um impacto mais significativo no MSE do que erros menores. Isso significa que a métrica é sensível a pontos fora da curva (*outliers*) ou instâncias com desvios elevados. O MAE é outra métrica comum utilizada em tarefas de regressão. O MAE calcula a média das diferenças absolutas entre os valores previstos e os valores reais. Ao contrário do MSE, o MAE não utiliza o quadrado das diferenças, o que significa que não dá peso extra a erros maiores. Nesse sentido, o MAE é mais robusto a pontos fora da curva. O Coeficiente de Determinação, ou R2-Score, é uma métrica utilizada em tarefas de regressão que quantifica a proporção da variância na variável dependente (o alvo) que é previsível a partir das variáveis

independentes (valores verdadeiros) em um modelo de regressão. o R2-Score compara o desempenho do modelo com a previsão média. Como exemplo, um R2-Score de 0 indica que o modelo não melhora a previsão em relação à média, enquanto um R2-Score de 1 indica uma previsão perfeita, isto é, o modelo compreende as variações encontradas e R2-Scores negativos indicam que o modelo prediz os resultados pior do que a hipótese nula, também chamada de linha horizontal.

Tabela 12 – Resultados das inferências de todas as redes neurais profundas propostas para a tarefa de contagem de anéis de crescimento.

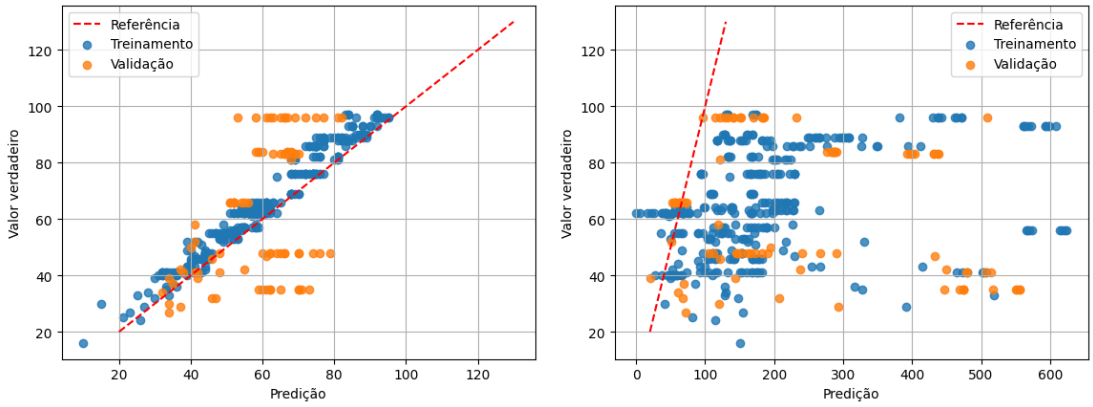
	Métricas					
	MSE		MAE		R2-Score	
Redes Neurais	Train	Val	Train	Val	Train	Val
ResNet18-Norm-CR	36.14	434.12	4.96	18.24	0.90	0.21
ResNet18-Seg-CR	33557.04	62986.29	130.82	189.14	-95.56	-114.04
ResNet34-Norm-CR	**29.15**	337.47	4.01	15.35	**0.91**	0.32
ResNet34-Seg-SR	306.75	**195.29**	15.70	**12.00**	0.11	**0.60**
ResNet50-Norm-CR	29.70	449.79	**3.68**	17.45	0.91	0.18
ResNet50-Seg-SR	815.32	1098.46	27.70	29.42	-1.35	-1.01
ResNet101-Norm-SR	1839.19	2575.01	36.02	43.37	-4.29	-3.70
ResNet101-Seg-SR	752.42	776.58	23.35	22.58	-1.17	-0.42

Fonte: O autor.

A Figura 68 ilustra os resultados de ambas as redes ResNet18. O eixo horizontal apresenta os valores previstos, enquanto o eixo vertical apresenta os valores verdadeiros. A linha tracejada em vermelho é a linha de referência, onde valores previstos e verdadeiros são idênticos. A Figura 68a, quando analisada em conjunto com as métricas da Tabela 12, indica que a rede ResNet18-Norm-CR foi capaz de desempenhar bem no conjunto de treino, mas não foi capaz de generalizar suficiente a tarefa de forma que pudesse atingir os mesmos resultados no conjunto de validação. No outro extremo, temos os resultados da ResNet18-Seg-CR apresentados na Figura 68b que demonstram péssimo desempenho em ambos os conjuntos de treinamento e validação. De forma geral, a aplicação da base de dados segmentada se mostrou muito prejudicial para esta arquitetura em específico, gerando diversos pontos fora da curva e variação nos resultados obtidos.

A Figura 69 apresenta os resultados obtidos pelas redes ResNet34. De modo geral, as ResNet34 apresentaram os melhores resultados, tanto do ponto de vista das métricas da Tabela 12 quanto das predições avaliadas. A Figura 69a, quando analisada em conjunto com os resultados da Tabela 12, indica que a rede treinada na base normal resultou em melhor predições no conjunto de treinamento e piores predições no conjunto de validação, quando comparada com sua equivalente ResNet34-Seg-SR. A ResNet34-Seg-SR, conforme indicam a Figura 69b e a Tabela 12, desempenhou melhor no conjunto de validação e obteve, de forma geral, predições homogêneas tanto

Figura 68 – Comparação entre os resultados obtidos pelas redes ResNet18.

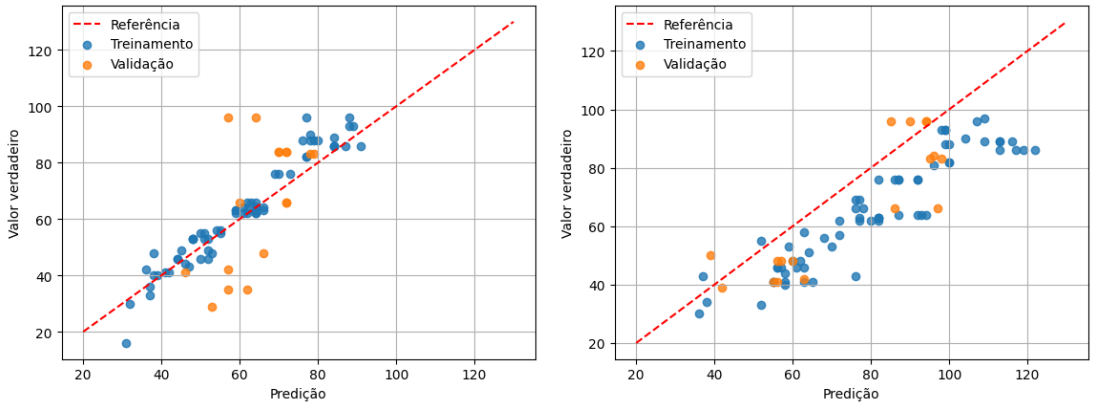


(a) ResNet18-Norm-CR. (b) ResNet18-Seg-CR.

Fonte: O autor.

no conjunto de treinamento quanto no conjunto de validação. Este comportamento homogêneo mostra que a rede foi capaz de generalizar melhor a tarefa proposta.

Figura 69 – Comparação entre os resultados obtidos pelas redes ResNet34.



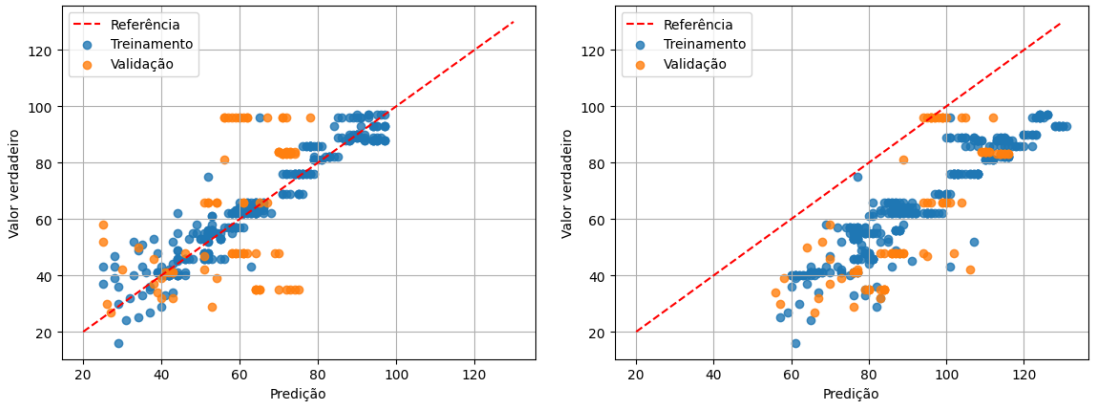
(a) ResNet34-Norm-CR. (b) ResNet34-Seg-SR.

Fonte: O autor.

De forma similar, a Figura 70 apresenta os resultados obtidos pelas redes ResNet50. Nesta comparação é notável que o uso da base de dados segmentada não contribuiu com a melhora do desempenho da rede, como esperado. A Figura 70b e a Tabela 12 indicam que a rede ResNet50-Seg-SR obteve piora significativa nos resultados das métricas e predições. A ResNet50-Norm-CR, por outro lado, demonstra resultados parecidos com a rede ResNet34-Norm-CR, apesar de apresentar maior variação nas predições.

A Figura 71 apresenta os resultados obtidos pelas redes ResNet101. A Figura 71a e Tabela 12 indicam presença acentuada de pontos fora da curva na predição da rede ResNet101-Norm-SR, o que explica o alto valor da métrica MSE. Por outro

Figura 70 – Comparação entre os resultados obtidos pelas redes ResNet50.

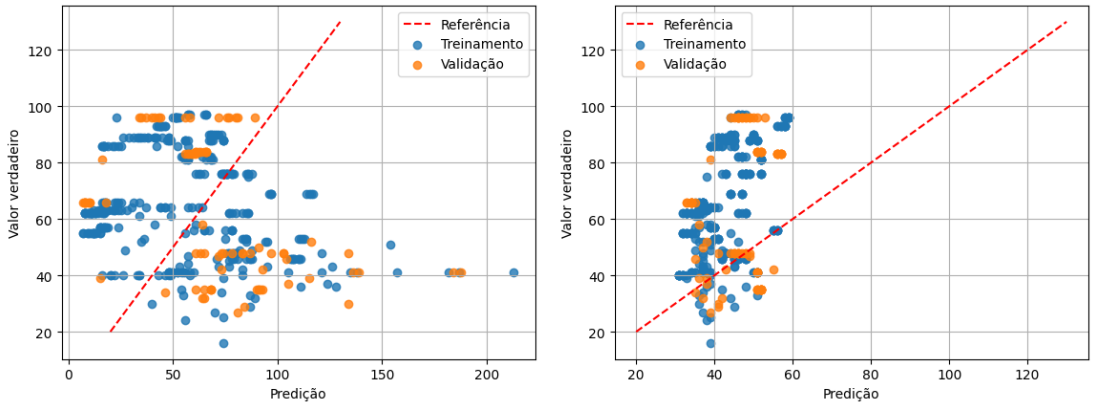


(a) ResNet50-Norm-CR. (b) ResNet50-Seg-SR.

Fonte: O autor.

lado, os resultados obtidos no conjunto de treino e no conjunto de validação são parecidos, indicando generalização por parte da rede. Por fim, ao utilizar a base segmentada em conjunto com a rede ResNet101-Seg-SR, as métricas indicam significativa melhora. De modo geral, os dados de predição se distribuem com menos variação, mantendo consistência entre os conjuntos de treinamento e validação.

Figura 71 – Comparação entre os resultados obtidos pelas redes ResNet101.



(a) ResNet101-Norm-SR. (b) ResNet101-Seg-SR.

Fonte: O autor.

Desta forma, a análise envolvendo o uso de uma base de dados previamente processada aparenta ser positiva. Os treinamentos se mostraram, de forma geral, mais rápidos e estáveis. Ainda, a rede com a melhor performance neste trabalho, a ResNet34-Seg-SR, foi treinada na base segmentada e gerou resultados que aqui são entendidos como promissores por sua consistência no conjuntos de treinamento e validação e melhor performance de acordo com as métricas avaliadas. Apesar da ResNet34-Seg-SR não seguir a linha de referência, os dados se encontram distribuídas

em seu entorno, indicando que possíveis ajustes finos futuros possam melhorar ainda mais a precisão do modelo.

Finalmente, a Tabela 13 mostra tanto o tempo de inferência quanto o uso de memória RAM para cada uma das redes. Os testes foram conduzidos em um computador pessoal, sem o uso de placas gráficas. Para isso, foi utilizado um processador Ryzen(R) 7-4800H juntamente com 32GB de memória RAM @ 3200MHz.

Tabela 13 – Tempo de inferência médio e uso de memória RAM para cada uma das redes propostas. Testes conduzidos utilizando uma CPU Ryzen(R) 7-4800H.

Rede Neural	**Inferência (s)**	**RAM (GB)**
ResNet18-Norm-CR	0.45 s	3 GB
ResNet18-Seg-CR	0.45 s	3 GB
ResNet34-Norm-CR	0.79 s	4 GB
ResNet34-Seg-SR	0.79 s	4 GB
ResNet50-Norm-CR	2.1 s	8 GB
ResNet50-Seg-SR	2.1 s	8 GB
ResNet101-Norm-SR	2.3 s	12 GB
ResNet101-Seg-SR	2.3 s	12 GB

Fonte: O autor.

Os resultados mostram que todas as redes estão aptas a serem utilizadas sem *hardwares* especializados como GPUs e TPUs. Além disto, os tempos de inferência são razoáveis quando comparados com aqueles vistos na tarefa de segmentação semântica. De modo geral, os algoritmos atendem aos requisitos funcionais e não funcionais deste trabalho.

5.2.1 Discussão geral

As redes neurais profundas utilizadas na tarefa de identificação de toras de madeira apresentaram resultados promissores. Em destaque, a rede Swin foi capaz de segmentar a tora central da imagem, matendo as métricas avaliadas sempre acima da marca de 90 pontos. De forma similar, as redes ResNet foram capazes de produzir resultados relevantes na contagem de anéis de crescimento. Em destaque, a rede ResNet34-Seg-SR apresentou métricas melhores que a maioria de suas concorrentes, além de demonstrar boa capacidade de generalizar os conjuntos de treinamento e validação.

Quanto a inferência, os considerar os maiores tempo de ambos os casos somados, a espera total fica ligeiramente acima da marca de 10s. Ao considerar os tempos de inferência para as melhores redes (Swin e ResNet34-Seg-SR), a espera total é de aproximadamente 8 segundos, permitindo que um possível usuário do sistema seja capaz de escanear rapidamente pilhas de toras de madeira para extrair suas informações utilizando.

Ainda, como nenhuma biblioteca ou linguagem de programação utilizada é específica de qualquer sistema operacional, os resultados podem ser facilmente reproduzidos em diversos computadores, smartphones ou tablets que tenham memória RAM mínima de 13GB para a segmentação, utilizando a rede Swin, e 4GB para a regressão, utilizando a rede ResNet34-Seg-SR. Em termos gerais, a pesquisa apresentou resultados interessantes e permite que trabalhos futuros sejam desenvolvidos partindo de tudo que foi apresentado neste trabalho.

6 CONCLUSÃO

Este trabalho teve como principal objetivo avaliar algoritmos de aprendizado profundo na tarefa de análise de imagens da seção reta de toras de madeira. Neste contexto, três contribuições principais foram realizadas: padronização e criação de bases de dados, avaliação de algoritmos de segmentação semântica e avaliação de algoritmos de regressão. As bases de dados padronizadas e criadas são derivadas da base criada por (DECELLE; JALILIAN, 2020). A segmentação semântica é realizada através do emprego de redes neurais profundas adaptadas para esta tarefa. A contagem dos anéis de crescimento das toras de madeiras é feita através do uso de redes neurais profundas adaptadas para a tarefa de regressão. Diversos experimentos foram conduzidos para avaliar o desempenho das redes propostas, bem como a utilidade de uma das bases criadas.

Esta pesquisa utilizou como um de seus principais elementos a base de dados criada e disponibilizada por (DECELLE; JALILIAN, 2020). No entanto, a base de dados original contava com imagens de alta resolução, sem padrão de tamanho ou distribuição em conjuntos de treinamento e validação. Nesse sentido, parte da pesquisa se voltou em adaptar a base para o padrão Cityscapes (CORDTS *et al.*, 2016), que é amplamento utilizado no processo de avaliação de desempenho de redes neurais profundas. Esta adaptação tem como objetivo facilitar a reprodutibilidade dos experimentos. Adicionalmente, uma base de dados voltada para a contagem dos anéis de crescimento das toras de madeira foi criada. Esta base utiliza as mesmas imagens encontradas em (DECELLE; JALILIAN, 2020), adicionando um arquivo de texto com as quantidades de anéis de crescimento contadas manualmente para cada imagem. A base de dados para contagem de anéis possui duas variações, onde uma delas contém as imagens originais e a outra contém as imagens já segmentadas.

Um dos desafios encontrados é a tarefa de identificar a tora de madeira central em imagens. Para tanto, este trabalho utilizou a técnica de segmentação semântica. Especificamente, são conduzidos testes utilizando arquiteturas de redes neurais profundas baseadas em CNNs e baseadas em Transformers. As redes baseadas em CNNs utilizadas são a U-Net e a FastFCN, enquanto as redes baseadas em Transformers são a SegFormer, a Swin e o Twins. Nesta tarefa, as redes baseadas em Transformers apresentaram resultados muito promissores, superando as CNNs tanto nas métricas de precisão quanto na estabilidade ao longo das diferentes bases de dados treinadas. Do ponto de vista de uso de *hardware*, as arquiteturas baseadas em Transformers também apresentaram resultados promissores, onde a rede com o menor tempo de inferência foi o Twins. De modo geral, esta parte do trabalho realiza uma comparação entre redes neurais profundas modernas, baseadas no mecanismo de atenção, e redes neurais profundas baseadas em convoluções. Atingindo métricas de precisão acima

de 95%, as redes baseadas em Transformers se mostraram aptas a realizar a tarefa de identificação de toras de madeiras em imagens.

O outro desafio encontrado nesta pesquisa é a estimativa de anéis de crescimento encontrados nas toras de madeira. Para esta tarefa, utilizou-se da família de redes neurais profunads ResNet (HE, K. *et al.*, 2016). As variações ResNet18, ResNet34, ResNet50 e ResNet101 foram modificadas para que pudessem realizar a tarefa de regressão. Os experimentos desta etapa demonstraram que não é necessario utilizar as variações mais profundas para atingir bons resultados. Na verdade, o melhor desempenho foi encontrado através da ResNet34 quando treinada na base de imagens segmentada. Os resultados experimentais demonstramram resultados promissores na regressão da quantidade de anéis, onde a ResNet34 atingiu as métricas MSE 195.29, MAE 12.00 e R2-Score 0.60. Este resultados também indicam que a abordagem multi etapas, segmentação e regressão, baseada em redes neurais profundas é promissora e deve ser explorada mais a fundo.

De modo geral, a pesquisa se mostrou muito positiva e produziu resultados animadores. As redes neurais profundas propostas foram capazes de atingir os requisitos funcionais e não funcionais estipulados. A redes baseadas em Transformers utilizadas na etapa de segmentação se apresentaram como um bom avanço tecnológico em relação as redes baseadas em CNNs, sem custos adicionais de *hardware* ou tempo de inferência. As redes ResNet, de forma similar, se apresentaram como arquiteturas extremamente adaptáveis ao serem empregadas na atividade de contagem de anéis de crescimento.

6.1 TRABALHOS FUTUROS

Partindo de todas as análises feitas e dificuldades encontradas no desenvolvimento deste trabalho, algumas tarefas foram propostas como continuações e trabalhos correlatos. Naturalmente, o primeiro trabalho futuro indicado é a expansão da base de dados. Apesar de possuir tamanho razoável, para tarefas de visão computacional envolvendo redes profundas as bases se mostraram insuficientes, principalmente no âmbito da contagem de anéis. As marcações envolvendo a contagem dos anéis de crescimento também poderiam ser revisadas e expandidas por pesquisadores da área.

Este trabalho mostrou que existe um claro avanço na performance quando utilizamos técnicas baseadas em Transformers para a segmentação. Como este tema é bastante recente, outras arquiteturas baseadas em Transformers, utilizando elementos mais sofistifcados deveriam ser empregadas. Ainda, redes com diferentes propostas como LLMs (*Large Language Models*) e modelos generativos (GenAI) devem ser testadas neste âmbito. Arquiteturas como o SwinV2 (LIU, Z. *et al.*, 2022) e BEiT-3 (WANG, Wenhui *et al.*, 2022) são bons exemplos de redes que podem contribuir com pesquisas futuras.

Como a área da segmentação semântica se beneficiou muito do uso de Transformers, também seria interessante avaliar o uso destas redes na tarefa de contagem de anéis de crescimento. Um bom tema de trabalho futuro pode ser aplicar os mesmos Transformer encontrados neste trabalho (SegFormer, Swin e Twins) na tarefa de regressão e comparar os resultados com as ResNets.

Por fim, uma ótima contribuição também poderia ser avaliar o uso de imagens geradas artificialmente, por outras redes neurais profundas, para ampliar a base de dados deste trabalho. Este aumento artificial da base de dados poderia beneficiar a performance das redes já vistas aqui.

REFERÊNCIAS

(US), Forest Products Laboratory. **Wood handbook: wood as an engineering material**. [*S.l.*]: The Laboratory, 2021.

ALI, Muhammad Shoaib. **Flattening CNN layers for Neural Network and basic concepts**. [*S.l.*]: Medium, 2022. Disponível em: https://medium.com/@muhammadshoaibali/flattening-cnn-layers-for-neural-network-694a232eda6a.

ALZELEY, Omar; ALJEDDANI, Sadiah. **Machine Learning Approach and Extreme Value Theory to Correlated Stochastic Time Series with Application to Tree Ring Data**. [*S.l.*: *s.n.*], 2023. arXiv: 2301.11488 [stat.ML].

ARTHUR, David; VASSILVITSKII, Sergei. K-means++ the advantages of careful seeding. *In*: PROCEEDINGS of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. [*S.l.*: *s.n.*], 2007. P. 1027–1035.

ATIENZA, Rowel. **Vision Transformer and its Applications**. Youtube. 2022. Disponível em: https://www.youtube.com/watch?v=hPb6A92LROc&ab_channel=OpenDataScience.

BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 39, n. 12, p. 2481–2495, 2017.

BOTOSSO, Paulo Cesar; MATTOS, Patrícia Póvoa de. **Conhecer a Idade das Árvores: Importância e Aplicação**. Embrapa. 2002. Disponível em: https://www.infoteca.cnptia.embrapa.br/bitstream/doc/280995/1/doc75.pdf.

BOYKOV, Y.; VEKSLER, O.; ZABIH, R. Fast approximate energy minimization via graph cuts. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 23, n. 11, p. 1222–1239, 2001.

CARION, Nicolas; MASSA, Francisco; SYNNAEVE, Gabriel; USUNIER, Nicolas; KIRILLOV, Alexander; ZAGORUYKO, Sergey. End-to-End Object Detection with Transformers. *In*: VEDALDI, Andrea; BISCHOF, Horst; BROX, Thomas;

FRAHM, Jan-Michael (Ed.). **Computer Vision – ECCV 2020**. Cham: Springer International Publishing, 2020. P. 213–229.

CERDA, Mauricio; HITSCHFELD-KAHLER, Nancy; MERY, Domingo. Robust Tree-Ring Detection. *In*: MERY, Domingo; RUEDA, Luis (Ed.). **Advances in Image and Video Technology**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. P. 575–585.

CHAN, T.F.; VESE, L.A. Active contours without edges. **IEEE Transactions on Image Processing**, v. 10, n. 2, p. 266–277, 2001.

CHEN, Liang-Chieh; PAPANDREOU, George; SCHROFF, Florian; ADAM, Hartwig. **Rethinking Atrous Convolution for Semantic Image Segmentation**. [*S.l.*: *s.n.*], 2017. arXiv: 1706.05587 [cs.CV].

CHENG, Bowen; MISRA, Ishan; SCHWING, Alexander G.; KIRILLOV, Alexander; GIRDHAR, Rohit. Masked-Attention Mask Transformer for Universal Image Segmentation. *In*: PROCEEDINGS of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2022. P. 1290–1299.

CHIRYSHEV, Yurii V.; KRUGLOV, Artem V.; ATAMANOVA, Anastasia S. Automatic Detection of Round Timber in Digital Images Using Random Decision Forests Algorithm. *In*: PROCEEDINGS of the 1st International Conference on Control and Computer Vision. Singapore, Singapore: Association for Computing Machinery, 2018. (ICCCV '18), p. 39–44.

CHOLLET, Francois. **Deep learning with Python**. [*S.l.*]: Simon e Schuster, 2021.

CHOLLET, Francois. Xception: Deep Learning With Depthwise Separable Convolutions. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jul. 2017.

CHU, Xiangxiang; TIAN, Zhi; WANG, Yuqing; ZHANG, Bo; REN, Haibing; WEI, Xiaolin; XIA, Huaxia; SHEN, Chunhua. Twins: Revisiting the Design of Spatial Attention in Vision Transformers. *In*: RANZATO, M.; BEYGELZIMER, A.; DAUPHIN, Y.; LIANG, P.S.; VAUGHAN, J. Wortman (Ed.). **Advances in Neural Information Processing Systems**. [*S.l.*]: Curran Associates, Inc., 2021. v. 34, p. 9355–9366.

CHU, Xiangxiang; TIAN, Zhi; ZHANG, Bo; WANG, Xinlong; SHEN, Chunhua. **Conditional Positional Encodings for Vision Transformers**. [*S.l.*: *s.n.*], 2023. arXiv: 2102.10882 [cs.CV].

CONTRIBUTORS, MMSegmentation. **MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark**. [*S.l.*: *s.n.*], 2020. https://github.com/open-mmlab/mmsegmentation.

CORDTS, Marius; OMRAN, Mohamed; RAMOS, Sebastian; REHFELD, Timo; ENZWEILER, Markus; BENENSON, Rodrigo; FRANKE, Uwe; ROTH, Stefan; SCHIELE, Bernt. The Cityscapes Dataset for Semantic Urban Scene Understanding. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2016.

CORKE, Peter. **Robotics, Vision and Control: Fundamental Algorithms in Python.[3rd ed.]** Cham, Switzerland: Springer, 2023. (Springer Tracts in Advanced Robotics).

DAVIES, E.R. **Computer Vision (Fifth Edition)**. Fifth Edition. [*S.l.*]: Academic Press, 2018. ISBN 978-0-12-809284-2.

DECELLE, Rémi; JALILIAN, Ehsaneddin. Neural Networks for Cross-Section Segmentation in Raw Images of Log Ends. *In*: 2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS). [*S.l.*: *s.n.*], 2020. P. 131–137.

DENG, Jia; DONG, Wei; SOCHER, Richard; LI, Li-Jia; LI, Kai; FEI-FEI, Li. ImageNet: A large-scale hierarchical image database. *In*: 2009 IEEE Conference on Computer Vision and Pattern Recognition. [*S.l.*: *s.n.*], 2009. P. 248–255.

DOSOVITSKIY, Alexey *et al.* **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale**. [*S.l.*: *s.n.*], 2021. arXiv: 2010.11929 [cs.CV].

DUCHI, John; HAZAN, Elad; SINGER, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of machine learning research**, v. 12, n. 7, 2011.

EVERINGHAM, Mark; ESLAMI, SM Ali; VAN GOOL, Luc; WILLIAMS, Christopher KI; WINN, John; ZISSERMAN, Andrew. The pascal visual object classes challenge: A

retrospective. **International journal of computer vision**, Springer, v. 111, p. 98–136, 2015.

FELZENSZWALB, Pedro F; HUTTENLOCHER, Daniel P. Efficient graph-based image segmentation. **International journal of computer vision**, Springer, v. 59, p. 167–181, 2004.

FENG, Xin; JIANG, Youni; YANG, Xuejiao; DU, Ming; LI, Xin. Computer vision algorithms and hardware implementations: A survey. **Integration**, v. 69, p. 309–320, 2019. ISSN 0167-9260.

FORSYTH, David A; PONCE, Jean. **Computer vision: a modern approach**. [*S.l.*]: Prentice Hall, 2011.

FORTIN, Jean-Michel; GAMACHE, Olivier; GRONDIN, Vincent; POMERLEAU, François; GIGUÈRE, Philippe. Instance Segmentation for Autonomous Log Grasping in Forestry Operations. *In*: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [*S.l.*: *s.n.*], 2022. P. 6064–6071.

GALSGAARD, Bo; LUNDTOFT, Dennis H.; NIKOLOV, Ivan; NASROLLAHI, Kamal; MOESLUND, Thomas B. Circular Hough Transform and Local Circularity Measure for Weight Estimation of a Graph-Cut Based Wood Stack Measurement. *In*: 2015 IEEE Winter Conference on Applications of Computer Vision. [*S.l.*: *s.n.*], 2015. P. 686–693.

GLOROT, Xavier; BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. *In*: TEH, Yee Whye; TITTERINGTON, Mike (Ed.). **Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics**. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010. v. 9. (Proceedings of Machine Learning Research), p. 249–256.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [*S.l.*]: MIT Press, 2016. `http://www.deeplearningbook.org`.

GOTTSCHLICH, Carsten. Curved-Region-Based Ridge Frequency Estimation and Curved Gabor Filters for Fingerprint Image Enhancement. **IEEE Transactions on Image Processing**, v. 21, n. 4, p. 2220–2227, 2012.

GUTS, Yuriy. **Yuriy Guts. TARGET LEAKAGE IN MACHINE LEARNING**. Youtube. 2018. Disponível em: `https://www.youtube.com/watch?v=dWhdWxgt5SU&ab_channel=AIUkraineConference`.

HE, Kaiming; GKIOXARI, Georgia; DOLLAR, Piotr; GIRSHICK, Ross. Mask R-CNN. *In*: PROCEEDINGS of the IEEE International Conference on Computer Vision (ICCV). [*S.l.*: *s.n.*], out. 2017.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2016.

HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *In*: PROCEEDINGS of the IEEE International Conference on Computer Vision (ICCV). [*S.l.*: *s.n.*], dez. 2015.

HE, Tong; ZHANG, Zhi; ZHANG, Hang; ZHANG, Zhongyue; XIE, Junyuan; LI, Mu. **Bag of Tricks for Image Classification with Convolutional Neural Networks**. [*S.l.*: *s.n.*], 2018. arXiv: `1812.01187 [cs.CV]`.

HENDRYCKS, Dan; GIMPEL, Kevin. **Gaussian Error Linear Units (GELUs)**. [*S.l.*: *s.n.*], 2023. arXiv: `1606.08415 [cs.LG]`.

HERBON, Christopher. **The HAWKwood Database**. [*S.l.*: *s.n.*], 2014. arXiv: `1410.4393 [cs.CV]`.

HINTON, Geoffrey. **Lecture 6a - Overview of mini-batch gradient descent**. [*S.l.*]: University of Toronto, 2012. Disponível em: `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`.

HOWARD, Andrew G.; ZHU, Menglong; CHEN, Bo; KALENICHENKO, Dmitry; WANG, Weijun; WEYAND, Tobias; ANDREETTO, Marco; ADAM, Hartwig. **MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications**. [*S.l.*: *s.n.*], 2017. arXiv: `1704.04861 [cs.CV]`.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. **The Journal of Physiology**, v. 195, n. 1, p. 215–243, 1968. eprint:

https:
//physoc.onlinelibrary.wiley.com/doi/pdf/10.1113/jphysiol.1968.sp008455.

HWANG, Sung-Wook; SUGIYAMA, Junji. Computer vision-based wood identification and its expansion and contribution potentials in wood science: A review. **Plant Methods**, v. 17, n. 1, p. 47, abr. 2021. ISSN 1746-4811.

IANDOLA, Forrest N.; HAN, Song; MOSKEWICZ, Matthew W.; ASHRAF, Khalid; DALLY, William J.; KEUTZER, Kurt. **SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size**. [*S.l.*: *s.n.*], 2016. arXiv: 1602.07360 [cs.CV].

IBGE. **PEVS - Produção da Extração Vegetal e da Silvicultura**. 2022. Disponível em: https://www.ibge.gov.br/estatisticas/economicas/agricultura-e-pecuaria/9105-producao-da-extracao-vegetal-e-da-silvicultura.html.

IMAÑA ENCINAS, José; SILVA, Gilson Fernandes da;
PINTO, José Roberto Rodrigues. Idade e crescimento das árvores. Universidade de Brasília, Departamento de Engenharia Florestal, 2005.

ISLAM, Md Amirul; JIA, Sen; BRUCE, Neil D. B. **How Much Position Information Do Convolutional Neural Networks Encode?** [*S.l.*: *s.n.*], 2020. arXiv: 2001.08248 [cs.CV].

ISOLA, Phillip; ZHU, Jun-Yan; ZHOU, Tinghui; EFROS, Alexei A. Image-To-Image Translation With Conditional Adversarial Networks. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jul. 2017.

KAINZ, Bernhard. **Deep Learning – Equivariance and Invariance**. Imperial College London. 2020. Disponível em:
https://www.doc.ic.ac.uk/~bkainz/teaching/DL/notes/equivariance.pdf.

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

KITCHENHAM, Barbara. Procedures for performing systematic reviews. **Keele, UK, Keele University**, Citeseer, v. 33, n. 2004, p. 1–26, 2004.

KRIZHEVSKY, Alex; HINTON, Geoffrey *et al.* Learning multiple layers of features from tiny images. Toronto, ON, Canada, 2009.

KUKIL. **Intersection over Union (IoU) in Object Detection and Segmentation**. [*S.l.*]: Learn Open CV, 2022. Disponível em: `https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/#IoU-in-Image-Segmentattion`.

LATEEF, Fahad; RUICHEK, Yassine. Survey on semantic segmentation using deep learning techniques. **Neurocomputing**, v. 338, p. 321–348, 2019. ISSN 0925-2312.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.

LEMOS, Yessica Maria Valencia *et al.* Inspeção automática de defeitos em ovos comerciais usando visão computacional, 2021.

LI, Fei-Fei. **Convolutional Neural Networks**. [*S.l.*]: Stanford University, 2023. Disponível em: `https://cs231n.github.io/convolutional-networks/`.

LI, Fei-Fei. **Neural Networks**. [*S.l.*]: Stanford University, 2023. Disponível em: `https://cs231n.github.io/neural-networks-1/`.

LIN, Guosheng; MILAN, Anton; SHEN, Chunhua; REID, Ian. RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jul. 2017.

LIN, Tianyang; WANG, Yuxin; LIU, Xiangyang; QIU, Xipeng. A survey of transformers. **AI Open**, v. 3, p. 111–132, 2022. ISSN 2666-6510.

LIN, Tianyang; WANG, Yuxin; LIU, Xiangyang; QIU, Xipeng. A survey of transformers. **AI Open**, v. 3, p. 111–132, 2022. ISSN 2666-6510.

LIN, Tsung-Yi; DOLLÁR, Piotr; GIRSHICK, Ross; HE, Kaiming; HARIHARAN, Bharath; BELONGIE, Serge. **Feature Pyramid Networks for Object Detection**. [*S.l.*: *s.n.*], 2017. arXiv: `1612.03144 [cs.CV]`.

LIN, Tsung-Yi *et al.* **Microsoft COCO: Common Objects in Context**. [*S.l.*: *s.n.*], 2015. arXiv: 1405.0312 [cs.CV].

LIU, Wei; ANGUELOV, Dragomir; ERHAN, Dumitru; SZEGEDY, Christian; REED, Scott; FU, Cheng-Yang; BERG, Alexander C. SSD: Single Shot MultiBox Detector. *In*: LEIBE, Bastian; MATAS, Jiri; SEBE, Nicu; WELLING, Max (Ed.). **Computer Vision – ECCV 2016**. Cham: Springer International Publishing, 2016. P. 21–37.

LIU, Ze; LIN, Yutong; CAO, Yue; HU, Han; WEI, Yixuan; ZHANG, Zheng; LIN, Stephen; GUO, Baining. Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows. *In*: PROCEEDINGS of the IEEE/CVF International Conference on Computer Vision (ICCV). [*S.l.*: *s.n.*], out. 2021. P. 10012–10022.

LIU, Ze *et al.* **Swin Transformer V2: Scaling Up Capacity and Resolution**. [*S.l.*: *s.n.*], 2022. arXiv: 2111.09883 [cs.CV].

LONG, Jonathan; SHELHAMER, Evan; DARRELL, Trevor. **Fully Convolutional Networks for Semantic Segmentation**. [*S.l.*: *s.n.*], 2015. arXiv: 1411.4038 [cs.CV].

MA, Jianqi; SHAO, Weiyuan; YE, Hao; WANG, Li; WANG, Hong; ZHENG, Yingbin; XUE, Xiangyang. Arbitrary-Oriented Scene Text Detection via Rotation Proposals. **IEEE Transactions on Multimedia**, v. 20, n. 11, p. 3111–3122, 2018.

MAJIN ERAZO, Jhon Jamilton *et al.* Desenvolvimento de um sistema de contagem e classificação de veıculos utilizando redes neurais convolucionais, 2021.

MARTINSEN, Iver; HARBITZ, Alf; BIANCHI, Filippo Maria. Age prediction by deep learning applied to Greenland halibut (Reinhardtius hippoglossoides) otolith images. Edição: Xiaowei Li. **PLOS ONE**, Public Library of Science (PLoS), v. 17, n. 11, e0277244, nov. 2022. ISSN 1932-6203.

MOHAMED, Mejri; RICHARD, Antoine; PRADALIER, Cedric. **A Study on Trees's Knots Prediction from their Bark Outer-Shape**. [*S.l.*: *s.n.*], 2020. arXiv: 2010.03173 [cs.CV].

MOTTAGHI, Roozbeh; CHEN, Xianjie; LIU, Xiaobai; CHO, Nam-Gyu; LEE, Seong-Whan; FIDLER, Sanja; URTASUN, Raquel; YUILLE, Alan. The Role of Context for Object Detection and Semantic Segmentation in the Wild. *In*: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], 2014.

NACK, Felipe Alfredo. Sistema de medição do volume de toras de madeira utilizando visão computacional. *In*.

NACK, Felipe Alfredo; STIVANELLO, Maurício Edgar; STEMMER, Marcelo Ricardo. **Modern Wood Segmentation**. [*S.l.*]: GitHub, 2024. `https://github.com/NackFelipe/ModernWoodSegmentation`.

NEUHOLD, Gerhard; OLLMANN, Tobias; ROTA BULO, Samuel; KONTSCHIEDER, Peter. The Mapillary Vistas Dataset for Semantic Understanding of Street Scenes. *In*: PROCEEDINGS of the IEEE International Conference on Computer Vision (ICCV). [*S.l.*: *s.n.*], out. 2017.

NIKLAS, Karl J; SPATZ, Hanns-Christof. Worldwide correlations of mechanical properties and green wood density. **American Journal of Botany**, Wiley Online Library, v. 97, n. 10, p. 1587–1594, 2010.

NINDEL, Thomas K.; HAFIDI, Mohcen; ISER, Tomáš; WILKIE, Alexander. **Automatic inference of a anatomically meaningful solid wood texture from a single photograph**. [*S.l.*: *s.n.*], 2023. arXiv: `2302.01820 [cs.GR]`.

OHLANDER, Ron; PRICE, Keith; REDDY, D. Raj. Picture segmentation using a recursive region splitting method. **Computer Graphics and Image Processing**, v. 8, n. 3, p. 313–333, 1978. ISSN 0146-664X.

PARMAR, Niki; VASWANI, Ashish; USZKOREIT, Jakob; KAISER, Lukasz; SHAZEER, Noam; KU, Alexander; TRAN, Dustin. Image Transformer. *In*: DY, Jennifer; KRAUSE, Andreas (Ed.). **Proceedings of the 35th International Conference on Machine Learning**. [*S.l.*]: PMLR, out. 2018. v. 80. (Proceedings of Machine Learning Research), p. 4055–4064.

POLÁČEK, Miroslav; ARIZPE, Alexis; HÜTHER, Patrick; WEIDLICH, Lisa; STEINDL, Sonja; SWARTS, Kelly. Automation of tree-ring detection and measurements using deep learning. **Methods in Ecology and Evolution**, Wiley, v. 14, n. 9, p. 2233–2242, jul. 2023. ISSN 2041-210X.

RASCHKA, Sebastian. **L8.7.1 OneHot Encoding and Multi-category Cross Entropy**. Youtube. 2021. Disponível em: `https://www.youtube.com/watch?v=4n71-tZ94yk&ab_channel=SebastianRaschka`.

RASCHKA, Sebastian; LIU, Yuxi Hayden; MIRJALILI, Vahid; DZHULGAKOV, Dmytro. **Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python**. [*S.l.*]: Packt Publishing Ltd, 2022.

RAVINDRAN, Prabu; OWENS, Frank C.; WADE, Adam C.; VEGA, Patricia; MONTENEGRO, Rolando; SHMULSKY, Rubin; WIEDENHOEFT, Alex C. Field-Deployable Computer Vision Wood Identification of Peruvian Timbers. **Frontiers in Plant Science**, v. 12, 2021. ISSN 1664-462X.

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2016.

REUWSAAT, Eliza Chaves Loschiavo; CARDOSO, Luiz Felipe Santos; LIMA, Ana Paula Garcia de; SALGADO, Jéssica Carneiro; SOUSA, WILZA CARLA SANTOS E; JESUS BARBOSA, Lucas de; PROTÁSIO, Thiago De Paula; GOULART, SELMA LOPES. CLASSIFICAÇÃO DE TORAS E QUALIDADE DA MADEIRA SERRADA DE ESPÉCIES AMAZÔNICAS. Galoá, 2017.

RIBEIRO, J. **Avaliação ambiental econômica da produção de madeira de espécie nativa em dois municípios na Amazônia Brasileira. 2008. 99 f**. 2008. Tese (Doutorado) – Dissertação (Mestrado em Saúde Ambiental). Faculdade de Saúde Pública da ...

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. *In*: NAVAB, Nassir; HORNEGGER, Joachim; WELLS, William M.; FRANGI, Alejandro F. (Ed.). **Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015**. Cham: Springer International Publishing, 2015. P. 234–241.

RUSSELL, Stuart J; NORVIG, Peter. **Artificial intelligence a modern approach 4th edition**. [*S.l.*]: London, 2021.

SAMDANGDECH, Noppawat; PHIPHOBMONGKOL, Suebskul. Log-End Cut-Area Detection in Images Taken from Rear End of Eucalyptus Timber Trucks. *In*: 2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE). [*S.l.*: *s.n.*], 2018. P. 1–6.

SAQIB, Muhammad; KHAN, Sultan Daud; SHARMA, Nabin; BLUMENSTEIN, Michael. Crowd Counting in Low-Resolution Crowded Scenes Using Region-Based Deep Convolutional Neural Networks. **IEEE Access**, v. 7, p. 35317–35329, 2019.

SCHARR, Hanno. **Optimal operators in digital image processing [Elektronische Ressource]** /. Set. 2014. Tese (Doutorado).

SCHRAML, Rudolf; UHL, Andreas. Similarity Based Cross-Section Segmentation in Rough Log End Images. *In*: ILIADIS, Lazaros; MAGLOGIANNIS, Ilias; PAPADOPOULOS, Harris (Ed.). **Artificial Intelligence Applications and Innovations**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. P. 614–623.

SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. FaceNet: A Unified Embedding for Face Recognition and Clustering. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2015.

SHEN, Wei; GUO, Yilu; WANG, Yan; ZHAO, Kai; WANG, Bo; YUILLE, Alan L. Deep Regression Forests for Age Estimation. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2018.

SILVA, José Luís; BORDALO, Rui; PISSARRA, José; PALACIOS, Paloma de. Computer Vision-Based Wood Identification: A Review. **Forests**, v. 13, n. 12, 2022. ISSN 1999-4907.

SIMONYAN, Karen; ZISSERMAN, Andrew. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. [*S.l.*: *s.n.*], 2015. arXiv: `1409.1556 [cs.CV]`.

SINGH, Ankita; SINGH, Pawan. Image Classification: A Survey. **Journal of Informatics Electrical and Electronics Engineering (JIEEE)**, v. 1, n. 2, p. 1–9, nov. 2020.

SNIF. **Boletim Snif 2022**. 2022. Disponível em: `https://snif.florestal.gov.br/images/pdf/publicacoes/Boletim-SNIF-2022_v24-01-2023.pdf`.

SPRINGER, Sebastian; GLIELMO, Aldo; SENCHUKOVA, Angelina; KAUPPI, Tomi; SUURONEN, Jarkko; ROININEN, Lassi; HAARIO, Heikki; HAUPTMANN, Andreas.

Reconstruction and segmentation from sparse sequential X-ray measurements of wood logs. [*S.l.*: *s.n.*], 2023. arXiv: 2206.09595 [eess.SP].

SRIVASTAVA, Nitish; HINTON, Geoffrey; KRIZHEVSKY, Alex; SUTSKEVER, Ilya; SALAKHUTDINOV, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to Sequence Learning with Neural Networks. *In*: GHAHRAMANI, Z.; WELLING, M.; CORTES, C.; LAWRENCE, N.; WEINBERGER, K.Q. (Ed.). **Advances in Neural Information Processing Systems**. [*S.l.*]: Curran Associates, Inc., 2014. v. 27.

SZEGEDY, Christian; VANHOUCKE, Vincent; IOFFE, Sergey; SHLENS, Jon; WOJNA, Zbigniew. Rethinking the Inception Architecture for Computer Vision. *In*: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], 2016. P. 2818–2826.

SZELISKI, Richard. Introduction. *In*: COMPUTER Vision: Algorithms and Applications. Cham: Springer International Publishing, 2022. P. 1–26. ISBN 978-3-030-34372-9.

SZÜCS, Carlos Alberto; TEREZO, Rodrigo Figueiredo; VALLE, Ângela do; MORAES, Poliana Dias de. **Estruturas de madeira**. Universidade Federal de Santa Catarina (UFSC). 2015. Disponível em: https://moodle.ufsc.br/pluginfile.php/1313798/mod_resource/content/0/Apostilamadeiras2015-1.pdf.

TIECKER GUSTAVO; MAZZOCHIN, João Victor Costa. **Segmentação e contagem de troncos de madeira utilizando deep learning e processamento de imagens**. 2021. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas) – Universidade Tecnológica Federal do Paraná (UTFP), Pato Branco.

WANG, Wenhai; XIE, Enze; LI, Xiang; FAN, Deng-Ping; SONG, Kaitao; LIANG, Ding; LU, Tong; LUO, Ping; SHAO, Ling. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction Without Convolutions. *In*: PROCEEDINGS of the IEEE/CVF International Conference on Computer Vision (ICCV). [*S.l.*: *s.n.*], out. 2021. P. 568–578.

WANG, Wenhui *et al.* **Image as a Foreign Language: BEiT Pretraining for All Vision and Vision-Language Tasks**. [*S.l.*: *s.n.*], 2022. arXiv: 2208.10442 [cs.CV].

WANGENHEIM, Aldo von. **Aula 04 - Convolução, Morfologia Matemática e Filtros**. Youtube. 2019. Disponível em: https://www.youtube.com/watch?v=mSb_rGWfeWM&ab_channel=AldovonWangenheim.

WEINBERGER, Kilian Q; BLITZER, John; SAUL, Lawrence. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *In*: WEISS, Y.; SCHÖLKOPF, B.; PLATT, J. (Ed.). **Advances in Neural Information Processing Systems**. [*S.l.*]: MIT Press, 2005. v. 18.

WIMMER, Georg; SCHRAML, Rudolf; HOFBAUER, Heinz; PETUTSCHNIGG, Alexander; UHL, Andreas. Two-Stage CNN-Based Wood Log Recognition. *In*: GERVASI, Osvaldo *et al.* (Ed.). **Computational Science and Its Applications – ICCSA 2021**. Cham: Springer International Publishing, 2021. P. 115–125.

WU, Huikai; ZHANG, Junge; HUANG, Kaiqi; LIANG, Kongming; YU, Yizhou. **FastFCN: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation**. [*S.l.*: *s.n.*], 2019. arXiv: 1903.11816 [cs.CV].

XIAO, Tete; LIU, Yingcheng; ZHOU, Bolei; JIANG, Yuning; SUN, Jian. **Unified Perceptual Parsing for Scene Understanding**. [*S.l.*: *s.n.*], 2018. arXiv: 1807.10221 [cs.CV].

XIE, Enze; WANG, Wenhai; YU, Zhiding; ANANDKUMAR, Anima; ALVAREZ, Jose M.; LUO, Ping. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. *In*: RANZATO, M.; BEYGELZIMER, A.; DAUPHIN, Y.; LIANG, P.S.; VAUGHAN, J. Wortman (Ed.). **Advances in Neural Information Processing Systems**. [*S.l.*]: Curran Associates, Inc., 2021. v. 34, p. 12077–12090.

YAMASHITA, Rikiya; NISHIO, Mizuho; DO, Richard Kinh Gian; TOGASHI, Kaori. Convolutional neural networks: an overview and application in radiology. **Insights into Imaging**, v. 9, n. 4, p. 611–629, ago. 2018. ISSN 1869-4101.

YANG, Wenxian; CAI, Jianfei; ZHENG, Jianmin; LUO, Jiebo. User-Friendly Interactive Image Segmentation Through Unified Combinatorial User Inputs. **IEEE Transactions on Image Processing**, v. 19, n. 9, p. 2470–2479, 2010.

YOSINSKI, Jason; CLUNE, Jeff; BENGIO, Yoshua; LIPSON, Hod. How transferable are features in deep neural networks? *In*: GHAHRAMANI, Z.; WELLING, M.;

CORTES, C.; LAWRENCE, N.; WEINBERGER, K.Q. (Ed.). **Advances in Neural Information Processing Systems**. [*S.l.*]: Curran Associates, Inc., 2014. v. 27.

ZHANG, Hang; DANA, Kristin; SHI, Jianping; ZHANG, Zhongyue; WANG, Xiaogang; TYAGI, Ambrish; AGRAWAL, Amit. Context Encoding for Semantic Segmentation. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jun. 2018.

ZHAO, Hengshuang; SHI, Jianping; QI, Xiaojuan; WANG, Xiaogang; JIA, Jiaya. Pyramid Scene Parsing Network. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jul. 2017.

ZHOU, Bolei; ZHAO, Hang; PUIG, Xavier; FIDLER, Sanja; BARRIUSO, Adela; TORRALBA, Antonio. Scene Parsing Through ADE20K Dataset. *In*: PROCEEDINGS of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [*S.l.*: *s.n.*], jul. 2017.

ZHOU, Bolei; ZHAO, Hang; PUIG, Xavier; XIAO, Tete; FIDLER, Sanja; BARRIUSO, Adela; TORRALBA, Antonio. Semantic Understanding of Scenes Through the ADE20K Dataset. **International Journal of Computer Vision**, v. 127, n. 3, p. 302–321, mar. 2019. ISSN 1573-1405.