UNIVERSIDADE NOVE DE JULHO - UNINOVE
PROGRAMA DE PÓS GRADUAÇÃO EM INFORMÁTICA E GESTÃO DO CONHECIMENTO - PPGI

ANDERSON HENRIQUE ROLLO DE SOUZA

APRENDIZAGEM DE FILTROS CONEXOS POR REDES NEURAIS USANDO ÁRVORES DE COMPONENTES

São Paulo
2025

ANDERSON HENRIQUE ROLLO DE SOUZA

APRENDIZAGEM DE FILTROS CONEXOS POR REDES NEURAIS USANDO ÁRVORES DE COMPONENTES

Dissertação submetida à Universidade Nove de Julho - UNINOVE, em cumprimento aos requisitos para obter o grau de Mestre em Informática e Gestão do Conhecimento.

Prof. Orientador: Dr. Wonder Alexandre Luz Alves

São Paulo
2025

Souza, Anderson Henrique Rollo de.
Aprendizagem de Filtros Conexos por Redes Neurais usando Árvores de Componentes. / Anderson Henrique Rollo de Souza. 2025.
89 f.
Dissertação (Mestrado) - Universidade Nove de Julho - UNINOVE, São Paulo, 2025.

Orientador (a): Prof. Dr. Wonder Alexandre Luz Alves.
1. Filtro conexo. 2. Árvore morfológica. 3. Filtragem de árvores morfológicas.
I. Alves, Wonder Alexandre Luz. II. Título.
CDU 004

UNINOVE
Universidade Nove de Julho

PARECER – EXAME DE DEFESA

Parecer da Comissão Examinadora designada para o exame de defesa do Programa de Pós-Graduação em Informática e Gestão do Conhecimento a qual se submeteu o aluno Anderson Henrique Rollo de Souza.

Tendo examinado o trabalho apresentado para obtenção do título de "Mestre em Informática e Gestão do Conhecimento", com dissertação intitulada "Aprendizagem de filtros conexos baseada em redes neurais usando árvores de componentes", a Comissão Examinadora considerou o trabalho:

(X) Aprovado **( ) Aprovado condicionalmente**
( ) Reprovado com direito a novo exame **( ) Reprovado**

EXAMINADORES

Prof. Dr. Wonder Alexandre Luz Alves - Uninove (Orientador)

Wonder A. L. Alves

Prof. Dr. Leonardo Nogueira Matos – UFS (Membro Externo)

gov.br Documento assinado digitalmente
LEONARDO NOGUEIRA MATOS
Data: 05/02/2025 15:40:07-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Sidnei Alves de Araújo – Uninove (Membro Externo)

gov.br Documento assinado digitalmente
SIDNEI ALVES DE ARAUJO
Data: 06/02/2025 23:02:24-0300
Verifique em https://validar.iti.gov.br

São Paulo, 05 de fevereiro de 2025

DEDICATÓRIA

Dedicado a Osvaldo e a Carolina.

AGRADECIMENTOS

Agradeço ao Senhor dos Exércitos, pois n'Ele me movo, existo e sou (At. 17.28).

Ao meu orientador, Prof. Dr. Wonder Alexandre Luz Alves, pela paciência para comigo e pela contribuição neste trabalho.

Aos professores, Prof. Dr. Sidnei Alves de Araújo, Prof. Dr. Leonardo Nogueira Matos e Prof. Dr. Peterson Adriano Belan, membros da banca de defesa e qualificação, pelas contribuições sugeridas neste trabalho.

Ao meu amigo, Prof. Dr. Hélio Doyle Pereira da Silva, pelo incentivo ao estudo desde o período de graduação em Ciência da Computação e pelas contribuições nas minhas apresentações de defesa e qualificação.

Ao meu amigo Lucas de Paula Oliveira Santos, companheiro de jornada no mestrado e pela contribuição com o sistema Linux.

À Universidade Nove de Julho (UNINOVE), pela oportunidade da concessão da bolsa de estudos.

EPÍGRAFE

"Não é o conhecimento, mas o ato de aprender; não a posse, mas o ato de chegar lá, que concede maior satisfação."

Carl Friedrich Gauss
Matemático

RESUMO

Filtros conexos são amplamente reconhecidos por sua capacidade de preservar contornos em imagens. Uma abordagem comum para implementá-los utiliza representações de imagens baseadas em árvores de componentes, que permitem calcular atributos característicos dos componentes conexos representados pelos nós da árvore. Esses atributos podem ser usados para filtrar determinados nós com base em limiares e, posteriormente, reconstruir a imagem filtrada. Apesar de sua relevância, a literatura apresenta poucas iniciativas que integram diretamente a aprendizagem automática de filtros conexos no âmbito de redes neurais. Nesta dissertação, propõe-se uma abordagem inovadora para otimizar a filtragem em árvores de componentes, integrando-as diretamente ao processo de aprendizagem das redes neurais. Em vez da tradicional função booleana usada para selecionar os nós, a abordagem emprega uma função contínua e parametrizada, atendendo aos requisitos do treinamento em redes neurais. Os experimentos realizados demonstram que o método proposto é capaz de aprender filtros conexos de maneira eficaz, com desempenho consistente em diferentes conjuntos de imagens, atributos e configurações de treinamento, consolidando sua aplicabilidade e eficiência.

Palavras-chave: filtro conexo, árvore morfológica, filtragem de árvores morfológicas.

ABSTRACT

Connected filters are widely recognized for their ability to preserve contours in images. A common approach to implementing them uses image representations based on component trees, which allow the calculation of characteristic attributes of the connected components represented by the tree nodes. These attributes can be used to filter specific nodes based on thresholds and then reconstruct the filtered image. Despite their relevance, the literature contains few initiatives that directly integrate the machine learning of connected filters within the context of neural networks. This dissertation proposes an innovative approach to optimize filtering in component trees by integrating them directly into the neural network learning process. Instead of the traditional boolean function used to select the nodes, the approach employs a continuous and parameterized function, meeting the requirements for neural network training. The experiments conducted demonstrate that the proposed method effectively learns connected filters, showing consistent performance across different image datasets, attributes, and training configurations, thereby consolidating its applicability and efficiency.

Keywords: filter connected, morphological tree, morphological tree filtering.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

CNN	Acrônimo para a expressão em inglês *Convolutional Neural Network* (em português, Redes Neurais Convolucionais).
CC	Componente conexo.
MNIST	Acrônimo para a expressão em inglês *Modified National Institute of Standards and Technology*, sendo um conjunto de imagens de dígitos escritos à mão, onde cada imagem é uma representação de um dígito 0 a 9.
pixel	Acrônimo para a expressão em inglês *picture element* (em português, elemento da imagem).
SSIM	Acrônimo para a expressão em inglês *Structural Similarity Index* (em português, Índice de Similaridade Estrutural).
PSNR	Acrônimo para a expressão em inglês *Peak Signal-to-Noise Ratio* (em português, Razão de Pico a Ruído do Sinal).
MAE	Acrônimo para a expressão em inglês *Mean Absolute Error* (em português, Erro Absoluto Médio).
PPD	Acrônimo para *Percentage of Pixel Difference* (em português, Percentual de Diferença entre *Pixels*).
SFS	Acrônimo para a expressão em inglês *Sequential Forward Selection* (em português, Seleção Sequencial para Frente).

LISTA DE SÍMBOLOS

Conceitos básicos

$\mathbb{Z}$	Conjunto dos números inteiros.
$\emptyset$	Conjunto vazio.
$\in$	Pertence ao conjunto.
$\notin$	Não pertence ao conjunto.
$\cap$	Operação de intersecção entre conjuntos.
$\cup$	Operação de união entre conjuntos.
$\subseteq$	É subconjunto de.
$\supseteq$	É superconjunto de.
$\setminus$	Operação de subtração entre conjuntos.
$\forall$	Para todo.
$\exists$	Existe.

Imagens

$\mathcal{D}$	Subconjuntos de $\mathbb{Z}^2$ utilizado para representar o domínio de uma imagem.
$\mathbb{K}$	Subconjuntos de $\mathbb{Z}$ que representa contradomínio da imagem.
$\mathcal{P}(\mathcal{D})$	Conjunto dos subconjuntos de $\mathcal{D}$.
$\mathcal{F}(\mathcal{D})$	Conjunto das imagens definidas com o domínio em $\mathcal{D}$ e o contradomínio em $\mathbb{K}$.
$p, q, e \in \mathcal{D}$	Variáveis que representam pares (x, y) chamados de *pixels*.
f e g	Variáveis que representam imagens.
$f(p)$	Nível de cinza em f de um *pixel* p.
$\mathcal{X}$	Variável utilizada para representar subconjuntos de $\mathcal{D}$.
$\mathcal{A}$	Relação de adjacência sobre $\mathcal{D}$.
$\mathcal{A}_4$	Relação de adjacência vizinhança-4 sobre $\mathcal{D}$.
$\mathcal{A}_8$	Relação de adjacência vizinhança-8 sobre $\mathcal{D}$.
$(p_x, p_y) \in \mathcal{D}$	Notação utilizada para representar as coordenadas horizontal e vertical.
$f(p)$	Função f aplicada no *pixel* p, ou seja, nível de cinza de f para o *pixel* p.
$(p, q) \in \mathcal{A}$	Par de *pixels* adjacentes ou vizinhos.
$\mathcal{CC}(\mathcal{X})$	Conjuntos dos componentes conexos de $\mathcal{X} \subseteq \mathcal{D}$.
$\mathcal{A}(p)$	Conjunto de todos dos *pixels* vizinhos a p dado pela adjacência $\mathcal{A}$.
λ, α, β	Valores de níveis de cinza.

LISTA DE SÍMBOLOS

Árvores

c	Variável que representa um componente conexo.
$\mathcal{X}_\lambda(f)$	Conjunto de nível inferior de f de valor λ, isto é $\mathcal{X}_\lambda(f) = \{p \in \mathcal{D} : f(p) \leq \lambda\}$.
$\mathcal{X}^\lambda(f)$	Conjunto de nível superior de f de valor λ, isto é $\mathcal{X}^\lambda(f) = \{p \in \mathcal{D} : f(p) \geq \lambda\}$.
$\mathcal{L}(f)$	Conjunto dos CCs dos conjuntos de níveis inferiores de $f(p)$.
$\mathcal{U}(f)$	Conjunto dos CCs dos conjuntos de níveis superiores de $f(p)$.
$\mathcal{SC}(\mathcal{L}(f), p)$	Menor CC de $\mathcal{L}(f)$ contendo p.
$\mathcal{SC}(\mathcal{U}(f), p)$	Menor CC de $\mathcal{U}(f)$ contendo p.
$\mathcal{SC}(\mathcal{T}, p)$	Menor componente conexo de um *pixel* $p \in \mathcal{D}$ em uma árvore $\mathcal{T}$.
$\mathcal{T}$	Variável utilizada para representar um árvore.
$\mathcal{T}_f$	Notação utilizada para representar um árvore $\mathcal{T}$ construída a partir da imagem f.
$(\mathcal{L}(f), \subseteq)$	Árvore de componentes inferiores (*min-tree*).
$(\mathcal{U}(f), \subseteq)$	Árvore de componentes superiores (*max-tree*).
$\kappa : \mathcal{P}(\mathcal{D}) \to \mathbb{R}$	Função que representa um atributo extraído de um CC.
$level : \mathcal{P}(\mathcal{D}) \to \mathbb{K}$	Função que associa CCs em níveis de cinzas.
$\mathrm{Rec}\,(\mathcal{T}_f)$	Reconstrução da imagem f a partir da árvore $\mathcal{T}_f$.
κ, $\kappa_{(\tau)}$	Variável que representa um atributo extraído de uma árvore morfológica.
τ	Variável que representa um nó $\tau \in \mathcal{T}$.
$\hat{\tau}$	Variável que representa um nó compacto que contém os *pixels* CNPs de um nó $\tau \in \mathcal{T}$.
τ_p	Representa um nó pai (parent) de um nó τ em uma árvore $\mathcal{T}_f$.

LISTA DE SÍMBOLOS

Rede Neural

x	Vetor de entrada de uma rede neural.
W	Vetor de pesos correspondentes aos valores de entrada x_i de uma rede neural).
y	Saída de um de uma rede neural).
b	*Bias*.
x_i	i-ésima entrada.
w_i^{anterior}	Peso correspondente à i-ésima entrada na iteração anterior.
$\mathrm{W}^{(l)}$	Matriz de pesos associada à camada l em uma rede neural ou rede reural de multiplas camadas.
$\mathrm{z}^{(l)}$	Vetor que representa a entrada ponderada para a camada l m uma rede neural ou rede reural de multiplas camadas.
$\mathrm{a}^{(l-1)}$	Vetor de ativações provenientes da camada anterior $l-1$ em rede reural de multiplas camadas.
σ	Função de ativação aplicada.
y	Saída real desejada.
$\hat{\mathrm{y}}$	Saída predita pela rede neural.
E_{mse}	Função de custo conhecida como Erro Quadrático Médio (MSE).
η	Taxa de aprendizagem.

LISTA DE SÍMBOLOS

Filtragem

$\mathbf{s}$	Função característica geral aplicada na filtragem de árvores morfológicas.
$\mathbf{s}_{(\tau,\kappa,i)}$	Função característica usada para verificar se um nó τ será mantido ou removido, baseada no atributo κ e no limiar i.
τ_a	Nó ancestral de um nó τ em uma árvore $\mathcal{T}_f$.
τ_d	Nó descendente de τ na árvore $\mathcal{T}_f$.
$\mathcal{T}_g$	Representa uma árvore construída a partir de uma imagem g.
$[\mathrm{Rec}\,(\mathcal{T}_f, \kappa, i)]$	Função de reconstrução utilizada após a filtragem de uma árvore $\mathcal{T}_f$ com base no atributo κ e no limiar i.
$\mathcal{SC}\,(\mathcal{T}_f, p)$	Menor componente conexo de um *pixe* p em uma árvore $\mathcal{T}_f$.

LISTA DE SÍMBOLOS

Metodologia

$\mathbf{s_{w,b}}$	Função parametrizada, utilizada na filtragem da árvore *max-tree*, onde w representa o vetor de pesos e b é o *bias* .
$\mathcal{T}_f$	Representa a árvore construída a partir da imagem f.
σ	Função de ativação sigmoide.
$\mathbf{x}_{i\tau}$	Vetor de atributos associados a um nó τ na árvore T_f.
$\mathbf{w}$	Vetor de pesos utilizado na função $\mathbf{s_{w,b}}$.
$\mathbf{b}$	*bias* utilizado na função $\mathbf{s_{w,b}}$.
f_y	Representa a imagem desejada.
f_{out}	Imagem reconstruída após a aplicação da filtragem e da função $\mathbf{s_{w,b}}$.
$\mathbf{X}$	Representa a matriz de aributos da árvore.
E	Função de custo utilizada durante a retropropagação.
$\frac{\partial E}{\partial f_{\text{out}}}$	Derivada da função de custo em relação à saída da rede.
$\frac{\partial E}{\partial \theta}$	Derivada da função de custo em relação aos parâmetros θ da rede neural.
$\frac{\partial E}{\partial f}$	Derivada da função de custo em relação à entrada da rede neural.
$[\text{Rec}(\mathcal{T}_f, \mathbf{s_{w,b}})]$	Representa a função de reconstrução Rec aplicada sobre uma árvore $\mathcal{T}_f$ (construída a partir da imagem f) e a função parametrizada $\mathbf{s_{w,b}}$, onde $\mathbf{w}$ é o vetor de pesos e $\mathbf{b}$ é o *bias* .
θ	Representa o conjunto de parâmetros da rede neural, incluindo pesos $\mathbf{w}$ e *bias* $\mathbf{b}$.

1 INTRODUÇÃO

1.1 Contextualização

A área de visão computacional testemunhou avanços significativos nas últimas décadas, permitindo novas aplicações em vários campos (MANAKITSA et al., 2024). Atualmente, existem sistemas de visão computacional complexos, como sistemas de reconhecimento facial (KORTLI et al., 2020), sistemas de visão em carros autônomos (TAN et al., 2024), sistemas de vigilância (SALEH et al., 2024) e muitos outros.

Essas aplicações modernas, em geral, exigem um entendimento profundo do conteúdo das imagens, incluindo tarefas como detecção e reconhecimento de objetos (ZHANG; SHI; YANG, 2015). Os avanços mais recentes em métodos de processamento de imagens dependem da utilização de um grande número de características de imagem aliadas a técnicas de aprendizado de máquina (JORDAN; MITCHELL, 2015).

Um exemplo bem-sucedido para essas tarefas de aprendizado são as Redes Neurais Convolucionais (do inglês, CNN - *Convolutional Neural Network*). As CNNs pertencem ao aprendizado profundo, que é uma subcategoria do aprendizado de máquina (KHANAM et al., 2024). Com o avanço do hardware dos computadores, a capacidade computacional aumentou, permitindo o desenvolvimento de arquiteturas mais profundas que alcançaram notável sucesso em várias tarefas de visão computacional (HE et al., 2016; SIMONYAN; ZISSERMAN, 2014).

Apesar da eficácia das CNNs, para algumas tarefas de processamento de imagens, as convoluções não são ideais. Embora as convoluções sejam poderosas para muitas aplicações em visão computacional, a Morfologia Matemática (MM) oferece uma abordagem complementar que pode ser mais eficaz em tarefas específicas que envolvem estruturas e formas, especialmente quando se trabalha com imagens binárias (AOUAD; TALBOT, 2022).

Essa abordagem complementar se tornou ainda mais evidente com a ascensão das arquiteturas de CNN no início da década de 2010, que motivou a exploração da integração de operações morfológicas com CNN (HERMARY et al., 2022). Embora essa aprendizagem de redes neurais morfológicas não seja um tema novo, tendo sido introduzida no final da década de 1980 (WILSON, 1989), a similaridade entre as operações das camadas convolucionais em CNNs e as camadas morfológicas (conceito adotado pelos pesquisadores) em redes neurais morfológicas (MONDAL et al., 2019) reside na sua finalidade comum de detecção e extração de padrões complexos a partir de dados de entrada, por meio da aprendizagem.

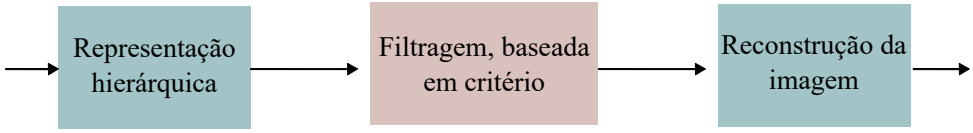
As camadas convolucionais em CNNs e as camadas morfológicas em redes morfológicas operam em nível de *pixels*. Nas camadas convolucionais, ocorre a aplicação de *kernels* à imagem, sendo computadas operações lineares utilizando tais *kernels* (ALZUBAIDI et

al., 2021). Por sua vez, nas camadas morfológicas em CNN, são computadas operações não lineares que aplicam elementos estruturantes sobre os *pixels* de entrada (SHEN; ZHONG; SHIH, 2019). Em ambos os casos, tanto os *kernels* quanto os elementos estruturantes são aprendidos durante o treinamento dessas redes.

Diferente do processamento de imagens por meio das operações lineares e não lineares, que ajustam os valores dos *pixels* com base nas informações de seus vizinhos, na literatura existe uma classe especial de estratégias de filtragem conhecidas como filtros conexos. Os filtros conexos são ferramentas de filtragem que atuam mesclando zonas planas (componentes conexos) (SALEMBIER; OLIVERAS, 1996). Essa abordagem se distingue por operar em um nível superior, manipulando diretamente as regiões da imagem, sem a dependência de *kernels* ou elementos estruturantes (WILKINSON et al., 2009). Em outras palavras, os filtros conexos permitem uma única operação, a exclusão de componentes conexos, garantindo assim que novos contornos não sejam deslocados nem criados (PERRET et al., 2014).

Para a construção de filtros conexos, existem diversas abordagens. Do ponto de vista prático, as estratégias mais eficazes dependem do método de reconstrução da imagem (imagem filtrada) ou à filtragem da árvore (SALEMBIER; OLIVERAS, 1996). A estratégia de filtragem da árvore baseia-se em uma representação hierárquica da imagem de entrada. Ou seja, uma árvore, calculada em um passo inicial. Em seguida, a simplificação é obtida através da seleção dos nós da árvore (baseada em um critério de filtragem) e, por fim, a imagem de saída é construída a partir da árvore filtrada (WILKINSON et al., 2009). A seguir, um esquema geral de filtragem é ilustrado na Figura 1.1.

Figura 1.1 – *Filtros conexos com representação hierárquica.Esse processo envolve três principais etapas, a primeira, é construída uma representação hierárquica da imagem, em seguida, realiza-se a filtragem, onde cada nó é analisado para decidir quais devem ser mantidos ou removidos conforme um critério, por fim, a imagem filtrada é reconstruída.*



Fonte: Adaptado de Salembier, Oliveras e Garrido (1998).

Uma representação hierárquica de imagem que se destaca na literatura é a árvore de componentes (JONES, 1997). A árvore de componentes é uma estrutura hierárquica que representa componentes conexos de diferentes níveis de cinza de uma imagem. Apesar do grande avanço nas pesquisas ao longo das duas últimas décadas envolvendo árvores de componentes, especialmente em tarefas de filtragem (XU; GÉRAUD; NAJMAN, 2012) e segmentação de imagens (PASSAT et al., 2011), e trabalhos recentes (WANG et al., 2022; RYU; TRAGER; WILKINSON, 2023; FAEZI; PELETIER; WILKINSON, 2024), poucos

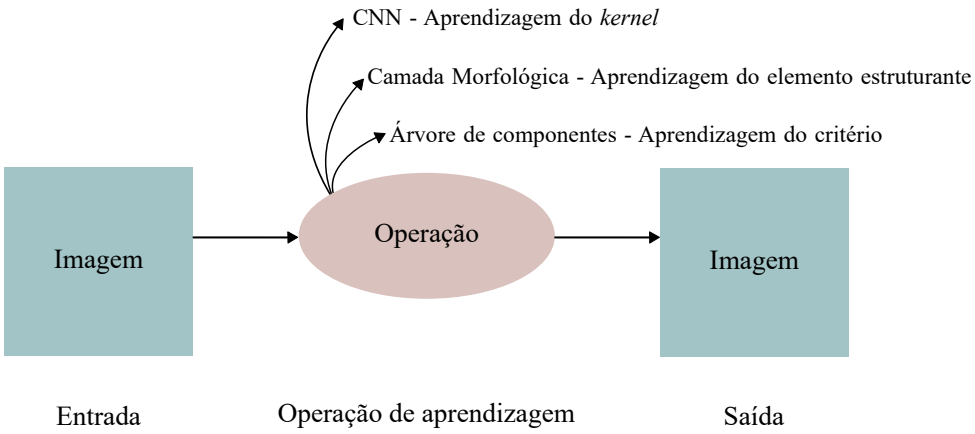
estudos abordam a integração dessa árvore com métodos de otimização contínua em redes neurais.

As pesquisas na literatura se concentram na aplicação de CNNs para segmentação, utilizando atributos dos nós da árvore de componentes como filtros ou características complementares. Em particular, Cabrera et al. (2021) mostraram que os atributos dos nós podem ser utilizados para gerar mapas de características. Da mesma forma, Meyer et al. (2022) empregaram atributos da árvore na CNN e compararam diferentes estratégias de seleção de atributos e nós para segmentação. Dessa maneira, é possível observar que a árvore é utilizada em uma etapa de pré-processamento, sem ser integrada à rede.

Diferentemente da abordagem anterior, a pesquisa de Perret e Cousty (2022) propôs um método para projetar funções de custo em árvores de componentes, otimizáveis por meio do algoritmo de gradiente descendente. O artigo destacou a incompatibilidade inicial entre as árvores de componentes devido à natureza combinatorial das decomposições topológicas. Para superar essa limitação, os autores demonstraram como o vetor dos níveis de cinza pode ser diferenciado em relação aos valores dos *pixel*s da imagem. A função de custo proposta controlou o número de nós folhas na árvore, funcionando como um critério de seleção de nós.

Considerando as informações apresentadas, é possível observar que a integração da árvore de componentes com abordagens de redes neurais ainda enfrenta desafios. Isso se aplica, em particular, aos métodos de otimização contínua e à necessidade de uma função diferenciável para essa estrutura. Além disso, há um número limitado de pesquisas científicas sobre o tema. Por essas razões, nesta pesquisa, é explorada a aprendizagem de filtros conexos (critério de seleção de nós) utilizando redes neurais. Dessa forma, para simplificar a visualização da aprendizagem desta pesquisa, a Figura 1.2 ilustra essa ideia.

Figura 1.2 – *Representação das operações de aprendizagem. A ilustração é apresentada de forma singular para simplificação. Contudo, na prática, utiliza-se um conjunto de imagens, envolvendo a aprendizagem de kernels e elementos estruturantes.*



Fonte: Elaborado pelo autor (2024).

A Figura 1.2 ilustra um diagrama simplificado do processo de aprendizado aplicado a imagens, compreendendo três etapas principais. A imagem de entrada representa a imagem original submetida ao processamento. A operação de aprendizagem constitui o núcleo do processo, onde ocorrem transformações e extrações de características. Por fim, a imagem de saída resulta das operações realizadas, podendo manifestar-se como uma nova imagem, um conjunto de características ou uma classificação.

No contexto das CNNs, a aprendizagem do *kernel* refere-se ao aprendizado dos filtros ideais que a rede utiliza para extrair características relevantes da imagem de entrada. Por outro lado, a camada morfológica aborda a aprendizagem do elemento estruturante, que envolve a seleção de formas apropriadas para operações morfológicas, visando à extração ou remoção de componentes da imagem. Adicionalmente, a árvore de componentes relaciona-se à aprendizagem do critério, utilizando uma estrutura hierárquica. Sendo assim, o foco deste estudo baseia-se na aprendizagem desse critério utilizando redes neurais.

1.2 Motivação

A motivação para este trabalho decorre do reconhecimento da abordagem inexplorada da integração da árvore de componentes com redes neurais. Embora a árvore de componentes tenha se demonstrado eficaz em tarefas como filtragem e segmentação de imagens, a integração dessa representação hierárquica com redes neurais permanece uma área pouco abordada pela comunidade científica. Neste contexto, nossa pesquisa busca investigar essa integração, focando na aprendizagem de filtros conexos.

1.3 Objetivos

O objetivo principal desta pesquisa é explorar a aprendizagem de filtros conexos para a filtragem de árvores morfológicas por redes neurais. De forma mais específica, pretende-se:

a) Desenvolver um método para otimizar a filtragem de árvores morfológicas, aprendendo critérios baseados em filtros conexos treinados por redes neurais, garantindo a compatibilidade do método com as propriedades e requisitos de aprendizado dessas redes, como a utilização de uma função contínua.

b) Investigar o impacto da variação dos parâmetros da árvore morfológica na performance da rede neural, por meio de experimentos com diferentes configurações e análise dos resultados obtidos.

c) Avaliar o desempenho do método proposto, direcionado para a tarefa de filtragem de imagens, por meio da realização de experimentos em um conjunto diversificado de dados de imagens, utilizando métricas de desempenho.

d) Avaliar o desempenho do método proposto com a escolha de atributos para

a tarefa de filtragem de imagens.

1.4 Contribuições da Pesquisa

Neste trabalho, as principais contribuições apresentadas são:

- Propõe uma abordagem inovadora que integra a filtragem de árvores morfológicas diretamente ao processo de aprendizagem das redes neurais, permitindo que os filtros conexos sejam aprendidos de forma otimizada.
- Substitui a tradicional função booleana, usada para selecionar os nós da árvore por uma função contínua, diferenciável e parametrizada, o que possibilita a otimização via gradiente descendente.
- Investiga o impacto da variação dos parâmetros da árvore morfológica sobre a performance da rede neural, fornecendo *insights* relevantes sobre a influência desses parâmetros na filtragem de imagens.
- Avalia experimentalmente o método proposto por meio de testes com conjuntos diversificados de imagens e métricas de desempenho, inclusive considerando a escolha de atributos para a filtragem, demonstrando sua eficácia e aplicabilidade prática.

1.5 Organização do Trabalho

O desenvolvimento deste trabalho está estruturado em cinco capítulos adicionais, cada um explorando temas específicos, conforme descrito a seguir:

- Capítulo 2 (Fundamentação Teórica): nesse Capítulo, são apresentados conceitos fundamentais relacionados a imagens, exploração de atributos, filtragem de árvores morfológicas e redes neurais. Esses tópicos proporcionam a base necessária para uma compreensão aprofundada dos temas abordados ao longo desta dissertação.
- Capítulo 3 (Materiais e Métodos): nesse Capítulo, é descrito o processo de aprendizagem dos filtros conexos (critério de seleção de nós) utilizando redes neurais.
- Capítulo 4 (Resultados): nesse Capítulo, são apresentados os resultados obtidos em dois experimentos.
- Capítulo 5 (Conclusões): nesse Capítulo, são apresentadas as conclusões do trabalho realizado até o momento, além dos próximos passos a serem seguidos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, é apresentada a fundamentação teórica. São abordados conceitos e notações fundamentais relacionados a filtros conexos, imagens, exploração de atributos, filtragem de árvores morfológicas e conceitos sobre redes neurais. Vale destacar que parte da notação utilizada para os conceitos de imagens foi baseada nos trabalhos de Gobber (2021), Silva (2017), Alves (2015) e Morimitsu (2015).

2.1 Filtros Conexos

Filtros conexos (HEIJMANS, 1999; SALEMBIER; SERRA, 1995; BREEN; JONES, 1996) são ferramentas fundamentais na área de morfologia matemática para processamento de imagens (PERRET et al., 2014). Eles operam sobre componentes conexos, definidos como subconjuntos máximos de *pixels* em uma imagem, onde qualquer par de *pixels* está conectado por um caminho contínuo. O princípio central desses filtros é remover ou preservar componentes conexos inteiros sem criar ou deslocar contornos (SALEMBIER; WILKINSON, 2009; NAJMAN; TALBOT, 2013). Assim, eles possuem propriedades de preservação de contorno.

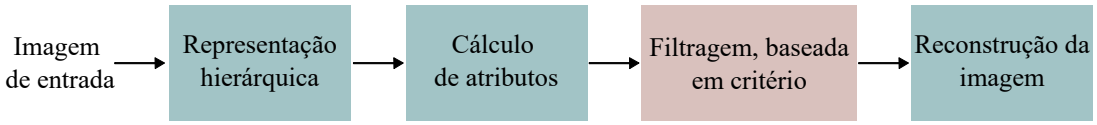
Ao contrário dos filtros clássicos que utilizam estruturas predefinidas, como *kernels* ou elementos estruturantes, os filtros conexos empregam as estruturas presentes na própria imagem de entrada para realizar a filtragem. Isso significa que nenhuma nova estrutura ou distorção é introduzida na imagem de saída, resultando em melhor preservação de contornos e menos distorções (SALEMBIER; OLIVERAS, 1996).

Na literatura, os filtros conexos são amplamente utilizados em aplicações como filtragem (SALEMBIER; OLIVERAS; GARRIDO, 1998), segmentação (JONES, 1999) e diversas outras. Em imagens binárias, por exemplo, permitem a remoção ou manutenção de componentes conexos inteiros com base em critérios como área ou forma (SALEMBIER; OLIVERAS; GARRIDO, 1998; SALEMBIER; GARRIDO, 2000). Para imagens em níveis de cinza, filtros conexos utilizam representações hierárquicas, como árvores de componentes, *max-tree* e *min-tree* (essas serão exploradas posteriormente), a árvore binária de partição (SALEMBIER; GARRIDO, 2000) ou a árvore de formas (MONASSE; GUICHARD, 2000), garantindo conectividade estrutural de forma hierárquica durante o processamento.

Para a construção de filtros conexos, existem diversas abordagens. Do ponto de vista prático, as estratégias mais eficazes dependem do método de reconstrução da imagem (imagem filtrada) ou da filtragem da árvore (SALEMBIER; OLIVERAS, 1996). A estratégia de filtragem da árvore baseia-se em uma representação hierárquica da imagem de entrada, ou seja, uma árvore construída em um passo inicial. Dessa forma, um esquema geral de definição para filtros conexos utilizando uma árvore envolve quatro etapas

(PERRET et al., 2014), a construção da representação hierárquica da imagem (árvore), o cálculo de atributos em cada nó da representação, a seleção de nós (filtragem) relevantes com base nesses atributos e a reconstrução da imagem a partir da árvore filtrada. A seguir, a Figura 2.1 ilustra essas etapas.

Figura 2.1 – *Fluxograma do processo de filtragem baseado em árvores*



Fonte: Adaptado de Salembier, Oliveras e Garrido (1998).

Na Figura 2.1, a árvore é construída a partir de uma imagem de entrada. Em seguida, os atributos dos nós são calculados e a filtragem é realizada. Por fim, a imagem é reconstruída com base na árvore atualizada, conforme o critério de filtragem.

A criação de filtros conexos mais avançados, como aqueles baseados na filtragem de árvores, exige a aplicação de critérios que definem quais nós devem ser preservados ou removidos.

2.2 Imagens como funções

Uma imagem pode ser definida de diversas formas. Uma imagem f é representada como uma função que mapeia uma grade retangular finita para um conjunto de níveis de cinza $\mathbb{K} = \{0, 1, \ldots, K\}$, ou seja

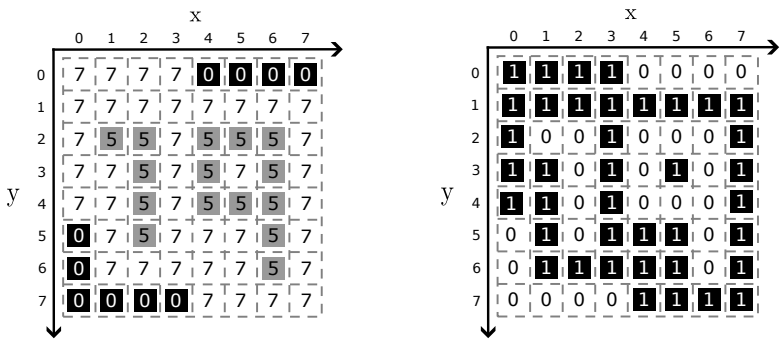
$$f : \mathcal{D} \subset \mathbb{Z}^2 \to \mathbb{K} = \{0, 1, \ldots, K\}. \tag{2.1}$$

Dessa forma, cada elemento $p = (p_x, p_y) \in \mathcal{D}$ é chamado de *pixel*, sendo representado por um par ordenado ou coordenada, consistindo de uma linha e uma coluna. A aplicação de f em p, denotada por $f(p)$, indica a intensidade do *pixel* p na imagem f. Além disso, uma imagem é considerada binária quando os *pixels* assumem apenas dois valores, isto é, $\mathbb{K} = \{0, 1\}$. Nesse caso, ela pode ser representada como o conjunto dos *pixels* com valor 1, ou seja

$$\mathcal{X} = \{p \in \mathcal{D} : f(p) = 1\} \subseteq \mathcal{D}. \tag{2.2}$$

Assim, os *pixels* de imagens binárias que satisfazem a condição $f(p) = 1$ são considerados *pixels* de objeto, enquanto aqueles que atendem à condição $f(p) = 0$ são chamados de *pixels* de fundo. Na Figura 2.2, apresenta-se um exemplo de uma imagem em níveis de cinza e de uma imagem binária representadas como funções.

Figura 2.2 – *Representação de uma imagem 8 × 8 em níveis de cinza e binária*



(a) *Imagem em níveis de cinza.*

(b) *Imagem binária.*

Fonte: Elaborado pelo autor (2024).

2.2.1 Adjacência

Em determinados cenários de processamento de imagens, a informação contida em um único *pixel* pode ser insuficiente para representar as características de interesse. Assim, a literatura apresenta o conceito de conectividade entre *pixels*. Nesse contexto, essa conectividade é representada por uma relação de adjacência. Uma relação de adjacência $\mathcal{A}$ é definida como uma relação binária entre os *pixels* de $\mathcal{D}$, isto é

$$\mathcal{A} \subseteq \mathcal{D} \times \mathcal{D}. \tag{2.3}$$

Desta forma, se $(p, q) \in \mathcal{A}$, então os *pixels* p e q são adjacentes ou vizinhos.

Embora existam diversas formas de se construir uma adjacência, neste trabalho são utilizadas apenas relações de adjacência circulares e simétricas. Essas relações são definidas como

$$\mathcal{A} = \{(p, q) : p, q \in \mathcal{D}, d(p, q) \leq r\}, \tag{2.4}$$

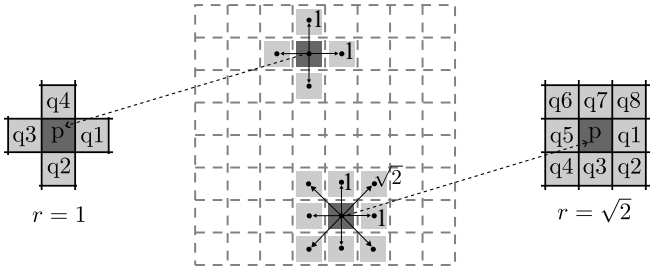
onde $d(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$ é a distância Euclidiana entre os *pixels* p e q, e r é um parâmetro de raio que representa a menor circunferência na qual ambos estão contidos.

Neste trabalho, utilizam-se duas adjacências bastante conhecidas para imagens 2D, $\mathcal{A}_4$, ou vizinhança-4, onde $r = 1$; e $\mathcal{A}_8$, ou vizinhança-8, onde $r = \sqrt{2}$. A partir de então, denota-se por

$$\mathcal{A}_{\mathcal{X}}(p) = \{q : p, q \in \mathcal{X}, p \text{ é vizinho de } q\}, \tag{2.5}$$

o conjunto que contém todos os *pixels* vizinhos a um dado *pixel* $p \in \mathcal{X}$. A seguir, a Figura 2.3 ilustra um exemplo das relações de adjacência.

Figura 2.3 – *Vizinhança mais comum entre pixels. $\mathcal{A}_4$ ou vizinhança-4 à esquerda da grade (ao centro). $\mathcal{A}_8$ ou vizinhança-8 à direita da grade.*



Fonte: Elaborado pelo autor (2024).

2.2.2 Caminho digital

Conforme definido pelos autores Gonzalez e Woods (2008), um caminho digital do *pixel* p, com coordenadas (p_x, p_y), ao *pixel* q, com coordenadas (q_s, q_t), é uma sequência de *pixels* distintos com coordenadas $(p_{x_0}, p_{y_0}), (p_{x_1}, p_{y_1}), \ldots, (p_{x_n}, p_{y_n})$, onde

$$(p_{x_0}, p_{y_0}) = (p_x, p_y) \quad \text{e} \quad (p_{x_n}, p_{y_n}) = (q_s, q_t), \tag{2.6}$$

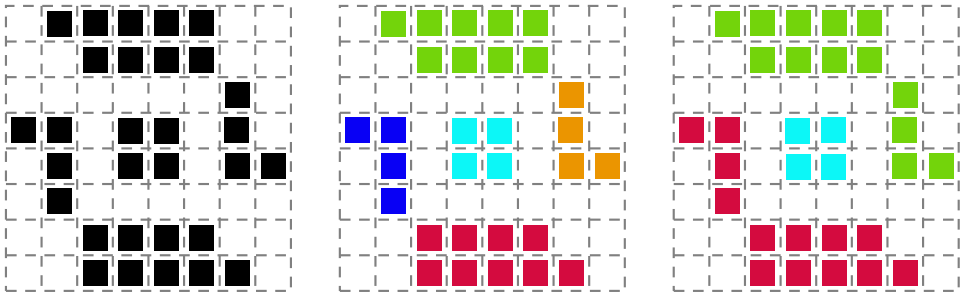
e os *pixels* (p_{x_i}, p_{y_i}) e $\left(p_{x_{i-1}}, p_{y_{i-1}}\right)$ são adjacentes para $1 \leq i \leq n$. Isso significa que os *pixels* estão próximos uns dos outros de acordo com algum critério de vizinhança, como $\mathcal{A}_4$ ou $\mathcal{A}_8$.

2.2.3 Componente conexo

Em processamento de imagens, um componente conexo refere-se a um conjunto de maximal *pixels* que são conectados ou adjacentes com base em um critério específico, (ALVES, 2015). Assim, podemos definir formalmente um componente conexo ($\mathcal{CC}$) $\mathcal{C}$, $\mathcal{X} \subseteq \mathcal{D}$, tal que para quaisquer dois *pixels* $p, q \in \mathcal{C}$, existe um caminho de p até q em $\mathcal{C}$. A notação $\mathcal{CC}(\mathcal{X}, \mathcal{A})$ representa o conjunto de todos os componentes conexos de $\mathcal{X}$ construídos de acordo com uma adjacência $\mathcal{A}$, que define a forma como a conectividade entre os *pixels* é estabelecida. Para cada tipo de adjacência, o conjunto de componentes conexos pode ser diferente, como ilustrado na Figura 2.4 para as adjacências $\mathcal{A}_4$ e $\mathcal{A}_8$.

Podemos observar exemplos de componentes conexos em uma imagem na Figura 2.4. Ao utilizar o critério de vizinhança $\mathcal{A}_4$, obtemos cinco componentes conexos distintos, *pixels* em ciano, *pixels* em magenta, *pixels* azul escuro, *pixels* em azul claro e *pixels* verdes exibidos na Figura 2.4(b). Já ao utilizar o critério de vizinhança $\mathcal{A}_8$, obtemos três componentes conexos distintos: *pixels* em ciano, *pixels* em azul claro e *pixels* em verde, como mostrado na Figura 2.4(c).

Figura 2.4 – *Comparação de componentes conexos: (b) cinco componentes com $\mathcal{A}_4$ e (c) três com $\mathcal{A}_8$. Essa comparação demonstra como a escolha da vizinhança pode afetar significativamente a quantidade dos componentes conexos identificados na imagem.*



(a) $\mathcal{X} \subseteq \mathcal{D}$ **(b)** $CC(\mathcal{X}, \mathcal{A}_4)$ **(c)** $CC(\mathcal{X}, \mathcal{A}_8)$

Fonte: Elaborado pelo autor (2024).

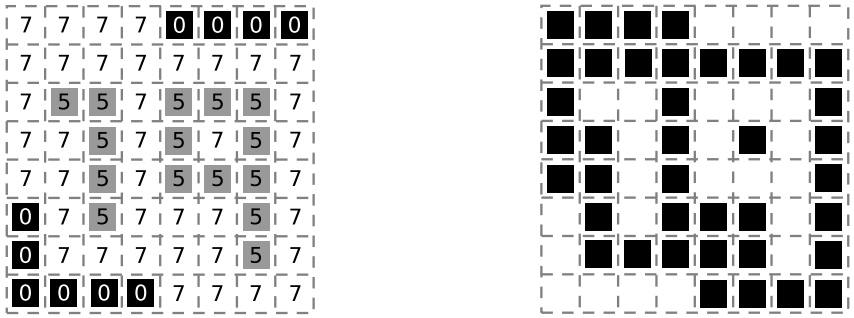
2.2.4 Limiarização

Limiarização é um filtro de imagens que visa transformar uma imagem em níveis de cinza em uma imagem binária, classificando os seus *pixels*. Isso é alcançado estabelecendo um valor limite no nível de cinza da imagem e classificando todos os *pixels* com intensidade acima ou abaixo desse limite como pertencentes ao objeto da imagem. Por exemplo, considere uma imagem em níveis de cinza f e seja λ um valor de limiar. A imagem binária resultante da operação de limiarização é dada por

$$g = \{p \in \mathcal{D} : f(p) \geq \lambda\} \quad \text{ou} \quad g = \{p \in \mathcal{D} : f(p) \leq \lambda\}. \tag{2.7}$$

A seguir, na Figura 2.5, é apresentado um exemplo de limiarização.

Figura 2.5 – *Exemplo da operação de limiarização em uma imagem em níveis de cinza.*



(a) f (b) g

Fonte: Elaborado pelo autor (2024).

Na Figura 2.5 é apresentado um exemplo de limiarização com um limiar $\lambda = 7$ e a

operação $\geqslant$. Neste processo, apenas os *pixels* cuja intensidade é maior ou igual a 7 são mantidos no conjunto que representa a imagem binária resultante (Figura 2.5(b)).

2.3 Representação de imagens por meio de árvores morfológicas

2.3.1 Conjunto de níveis

A operação de limiarização é utilizada para definir os conjuntos de nível de uma imagem. O conjunto de nível superior ou inferior corresponde a um conjunto de *pixels* da imagem que possuem valores acima ou abaixo de um determinado limiar $\lambda \in \mathbb{K}$ (ALVES, 2015). De forma mais precisa, dada uma imagem em níveis de cinza f, o conjunto de nível superior de valor λ é definido como

$$\mathcal{X}^{\lambda}(f) = \{p \in \mathcal{D} : f(p) \geq \lambda\}, \tag{2.8}$$

e o conjunto de nível inferior de valor λ é definido como

$$\mathcal{X}_{\lambda}(f) = \{p \in \mathcal{D} : f(p) < \lambda\}. \tag{2.9}$$

Para garantir a consistência nas operações que utilizam os conjuntos de níveis superiores e inferiores de uma imagem, é necessário manter valores maiores ou iguais a λ para o conjunto de nível superior e valores menores ou iguais a λ para o conjunto de nível inferior. Essa consistência é fundamental para a construção da árvore de componentes de uma imagem (a definição de árvore será apresentada posteriormente).

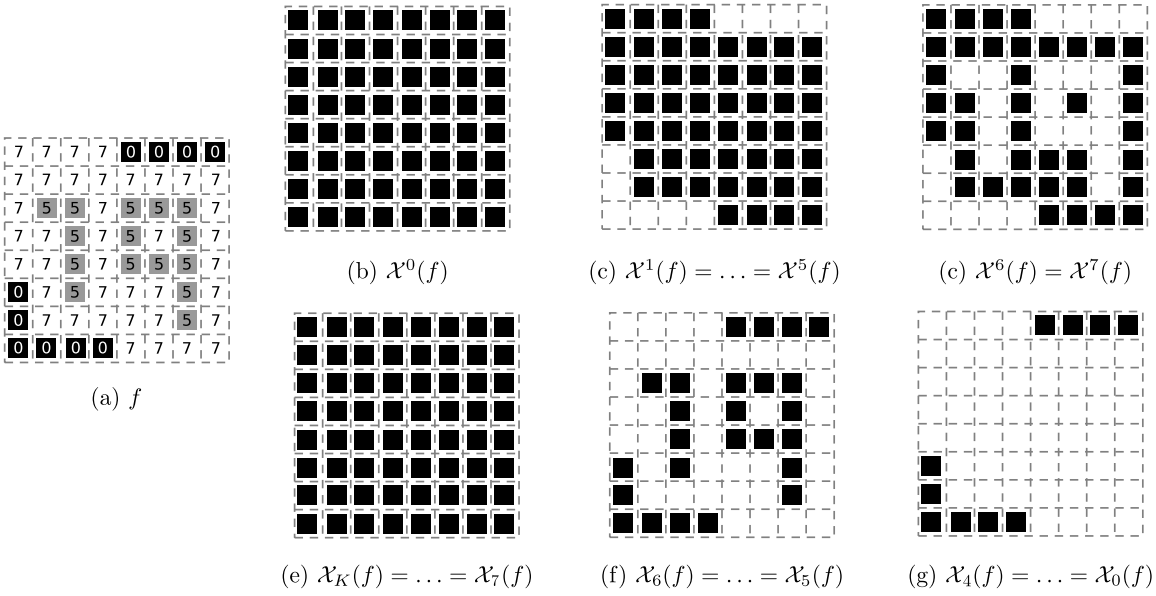
Nesse contexto, os conjuntos de níveis superiores são aninhados (ou seja, formam uma sequência de conjuntos em que cada conjunto está contido no próximo) pela relação de inclusão, formando uma hierarquia representada por $\mathcal{X}^0(f) \subseteq \mathcal{X}^1(f) \subseteq \ldots \subseteq \mathcal{X}^K(f)$, enquanto os conjuntos de níveis inferiores seguem a sequência $\mathcal{X}_0(f) \supseteq \mathcal{X}_1(f) \supseteq \ldots \supseteq \mathcal{X}_K(f)$. Para facilitar a compreensão, a Figura 2.6 apresenta um exemplo ilustrativo dos conjuntos de níveis superiores e inferiores.

As imagens na Figura 2.6 ilustram a variação dos conjuntos de níveis para diferentes valores de λ. As imagens são organizadas em duas linhas ao lado da imagem f (Figura 2.6(a)), a linha superior representa os conjuntos de níveis superiores (Figura 2.6(b), Figura 2.6(c), Figura 2.6(d)) e a linha inferior representa os conjuntos de níveis inferiores (Figura 2.6(e), Figura 2.6(f), Figura 2.6(g)).

2.3.2 Reconstrução de imagens

Uma imagem f pode ser reconstruída a partir de seus conjuntos de níveis devido à estrutura de aninhamento desses conjuntos. Assim, na teoria dos conjuntos, um conjunto

Figura 2.6 – *Conjuntos de níveis superiores (primeira linha) e níveis inferiores (segunda linha) para uma dada imagem f. As imagens em cada linha estão aninhadas em relação à operação de inclusão, da direita para a esquerda.*



Fonte: Elaborado pelo autor (2024).

A está incluso em um conjunto B se todos os *pixels* de A também são *pixels* de B. Formalmente, isso pode ser descrito como $A \subseteq B$.

Dada a afirmação acima, considerando dois conjuntos de níveis de uma imagem, α e $\beta \in \mathbb{K}$, se temos dois níveis de cinza α e β, tal que $\alpha > \beta$, podemos concluir que $\mathcal{X}^\beta(f) \subseteq \mathcal{X}^\alpha(f)$. Ou seja, todos os *pixels* no conjunto de nível β estão contidos no conjunto de nível α. Isso ocorre porque, se um *pixel* tem um nível de cinza $f(q)$, todos os *pixels* que possuem um nível de cinza inferior a $f(q)$ estarão em um conjunto de nível inferior ao do *pixel* q, o que significa que o *pixel* q não estará nesses conjuntos. Sendo assim, ao selecionar um *pixel* $q \in \mathcal{D}$ e o conjunto de nível $\mathcal{X}^{f(q)}(f) = \{p \in \mathcal{D} : f(p) < f(q)\}$, podemos concluir que q não pertence a esse conjunto ($q \notin \mathcal{X}^{f(q)}(f)$) e também inferir que o mesmo q não pertence aos conjuntos de níveis inferiores, isto é, $q \notin \mathcal{X}^{f(q)-1}(f) \supseteq \mathcal{X}^{f(q)-2}(f) \supseteq \ldots \supseteq \mathcal{X}^0(f)$. Dessa forma, uma maneira de reconstruir a intensidade de um *pixel* em uma imagem é através dos conjuntos de níveis inferiores e superiores. Assim, podemos usar a relação a seguir

$$f(q) = \min = \{\lambda \in \mathbb{K} : q \in \mathcal{X}_\lambda(f)\}, \tag{2.10}$$

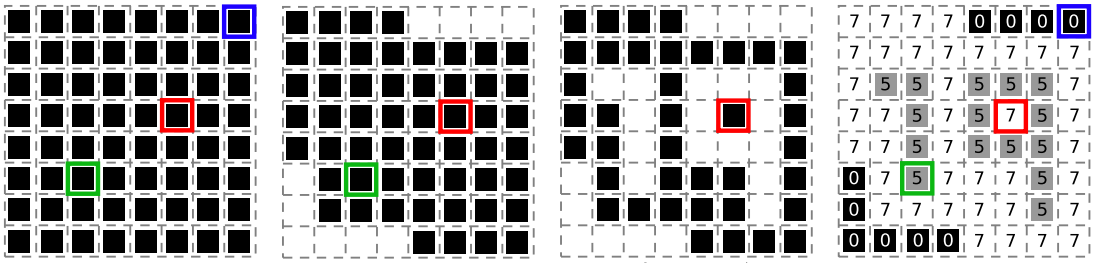
para encontrar a intensidade de um *pixel* q a partir dos seus conjuntos de níveis inferiores. Por exemplo, se o conjunto de níveis inferiores de um *pixel* q for $\mathcal{X}^3(f)$, que representa os *pixels* cujos níveis de cinza são menores que 3, então o nível de cinza do *pixel* q será 4. De maneira similar, podemos recuperar a intensidade de um *pixel* q usando apenas os

conjuntos de níveis superiores através da seguinte relação,

$$f(q) = \max = \{\lambda \in \mathbb{K} : q \in \mathcal{X}_\lambda(f)\}. \tag{2.11}$$

Por exemplo, se o conjunto de níveis superiores de um *pixel* q for $\mathcal{X}^4(f)$, que representa os *pixels* cujos níveis de cinza são maiores que 4, então o nível de cinza do *pixel* q será 5. Uma das formas de ilustrar o processo de recuperação das intensidades dos *pixels* é por meio da Figura 2.7, a qual apresenta exemplos de *pixels* selecionados e suas intensidades correspondentes.

Figura 2.7 – *Exemplo de como reconstruir uma imagem usando seus conjuntos de níveis superiores. Na imagem inicial, da esquerda para direita, são marcados os pixels pertencentes ao conjunto (a) $\mathcal{X}^0(f)$. Conforme avançamos pelos conjuntos, a partir do conjunto (b) $\mathcal{X}^1(f)$, o pixel azul é removido da imagem, enquanto que a partir do conjunto (c) $\mathcal{X}^6(f)$, o pixel verde é removido. Por fim, a imagem em níveis de cinza é reconstruída.*



(a) $\mathcal{X}^0(f)$ (b) $\mathcal{X}^1(f) = \ldots = \mathcal{X}^5(f)$ (c) $\mathcal{X}^6(f) = \mathcal{X}^7(f)$ (d) f

Fonte: Elaborado pelo autor (2024).

A Figura 2.7 destaca três *pixels* que são identificados como p (marcado em azul), q (marcado em verde) e r (marcado em vermelho). O *pixel* p pertence apenas ao conjunto $\mathcal{X}^0(f)$ (Figura 2.7(a)), portanto, sua intensidade é $f(p) = 0$. O *pixel* q pertence aos conjuntos $\mathcal{X}^0(f) \subseteq \mathcal{X}^1(f) = \mathcal{X}^2(f) = \mathcal{X}^3(f) = \mathcal{X}^4(f) = \mathcal{X}^5(f)$ (Figura 2.7(b)), mas não pertence ao conjunto $\mathcal{X}^6(f)$ (Figura 2.7(c)). Como seu conjunto de nível superior máximo é 5, então $f(q) = 5$. De maneira semelhante, o *pixel* r tem $\mathcal{X}^7(f)$ como o conjunto de nível superior máximo, e, consequentemente, $f(r) = 7$. Embora apenas três exemplos sejam destacados, o mesmo princípio se aplica a todos os *pixels*, dependendo apenas do conjunto de nível superior com o valor máximo para encontrar sua intensidade correspondente.

2.3.3 Árvores

Antes de apresentar aspectos específicos das árvores morfológicas, nesta Subseção são apresentados alguns elementos relacionados à estrutura de dados árvore.

Uma árvore é um tipo abstrato de dados que armazena elementos de forma hierárquica. Ela consiste em um conjunto de nós interconectados de forma organizada, em que

cada nó pode ter zero ou mais nós filhos, criando uma estrutura de ramificação similar a uma árvore botânica, (CORMEN et al., 2022).

Cada elemento em uma árvore possui um elemento pai e zero ou mais elementos filhos, exceto o elemento superior. A árvore é frequentemente visualizada colocando os elementos dentro de nós e desenhando conexões entre pais e filhos. O elemento superior é a raiz da árvore, desenhada como o elemento mais alto, com outros elementos conectados abaixo, ao contrário de uma árvore botânica real. A terminologia para estruturas de dados de árvores deriva das árvores genealógicas, com termos como pai, filho, ancestral e descendente sendo comuns para descrever essas relações. A seguir são apresentadas as definições dos principais componentes de uma árvore hierárquica.

- **Raiz** (*Root*): A raiz é o nó superior da árvore e não possui um nó pai. É a origem da estrutura hierárquica e é frequentemente representada como o ponto de partida para acessar os nós subsequentes.
- **Filhos** (*Children*): Os filhos de um nó são os nós diretamente conectados a ele na geração seguinte. Cada nó pode ter zero, um ou vários filhos, dependendo da estrutura da árvore.
- **Pai** (*Parent*): O pai de um nó é o nó imediatamente acima dele na hierarquia. Em outras palavras, o nó filho está ligado ao nó pai. Um nó pode ter apenas um nó pai.
- **Ancestrais** (*Ancestors*): Os ancestrais de um nó são todos os nós na cadeia hierárquica que leva do nó até a raiz. Em outras palavras, são os nós pais, avós, bisavós, e assim por diante, até chegar à raiz.
- **Descendentes** (*Descendants*): Os descendentes de um nó são todos os nós que estão abaixo dele na hierarquia, incluindo seus filhos, os filhos de seus filhos e assim por diante.
- **Folhas** (*Leaves*): As folhas são os nós que não têm filhos. São os nós terminais da árvore, que não se ramificam mais em direção aos filhos.
- **Nível** (*Level*): O nível de um nó é a sua distância em relação à raiz. A raiz está no nível 0, seus filhos estão no nível 1, os filhos dos filhos estão no nível 2 e assim por diante.
- **Subárvore** (*Subtree*): Uma subárvore é uma parte da árvore que inclui um nó e todos os seus descendentes, juntamente com as conexões entre eles.

2.3.4 Árvore de componentes

A árvore de componentes (JONES, 1997) é uma representação da imagem em níveis de cinza através da relação hierárquica entre seus componentes conexos em diferentes

limiares de intensidade. Dessa maneira, conforme descrito por Alves (2015), a árvore de componentes de uma imagem f é construída a partir de todos os componentes conexos presentes em todos os conjuntos de níveis, superiores ou inferiores, dessa imagem. Sendo assim, para definirmos a árvore de componentes, considere $\mathcal{L}(f)$ a família dos $\mathcal{CC}s$ dos conjuntos de níveis inferiores de f, isto é

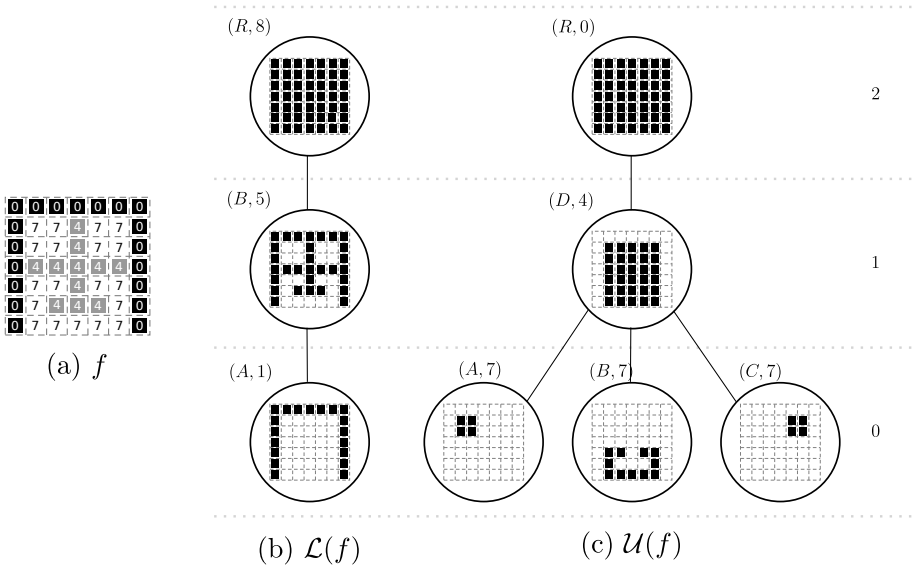
$$\mathcal{L}(f) = \{\mathcal{C} \in \mathcal{CC}(\mathcal{X}_\lambda(f), \mathcal{A}) : \lambda \in \mathbb{K}\}. \tag{2.12}$$

De forma análoga, considere $\mathcal{U}(f)$ a família de $\mathcal{CC}s$ dos conjuntos de níveis superiores de f, ou seja

$$\mathcal{U}(f) = \{\mathcal{C} \in \mathcal{CC}(\mathcal{X}^\lambda(f), \mathcal{A}) : \lambda \in \mathbb{K}\}. \tag{2.13}$$

Dado as definições Equação 2.12 e Equação 2.13 , Alves (2015) destaca que, se considerarmos dois níveis $\lambda_1 \geq \lambda_2$, onde $A \in \mathcal{CC}(\mathcal{X}^{\lambda_1}(f), \mathcal{A})$ e $B \in \mathcal{CC}(\mathcal{X}^{\lambda_2}(f), \mathcal{A})$, então temos duas possibilidades, ou $A \cap B = \emptyset$, indicando que A e B são disjuntos, ou $A \subseteq B$, indicando que A está contido em B. Da mesma forma, para os conjuntos de níveis inferiores, se $A \in \mathcal{CC}(\mathcal{X}_{\lambda_1}(f), \mathcal{A})$ e $B \in \mathcal{CC}(\mathcal{X}_{\lambda_2}(f), \mathcal{A})$, então ou $A \cap B = \emptyset$ ou $B \subseteq A$. Isso mostra que A e B são ou disjuntos ou aninhados. Assim, os conjuntos das famílias dos $\mathcal{CC}s$ dos níveis superiores $\mathcal{U}(f)$ e dos níveis inferiores $\mathcal{L}(f)$, junto com a relação de inclusão $\subseteq$, definem a árvore de componentes dos conjuntos de níveis superiores $(\mathcal{U}(f), \subseteq)$ e a árvore de componentes dos conjuntos de níveis inferiores $(\mathcal{L}(f), \subseteq)$, como ilustrado na Figura 2.8.

Figura 2.8 – *Exemplos de árvores de componentes com os conjuntos de níveis superiores $\mathcal{U}(f)$ e inferiores $\mathcal{L}(f)$.*



Fonte: Elaborado pelo autor (2024).

Na árvore de componentes construída com base no conjunto $\mathcal{L}(f)$ da Figura 2.8 (b), possui os nós $(R, 8), (B, 5)$ e $(A, 1)$, onde a $A \subset B$ e $B \subset R$, e consequentemente, $A \subset R$. O nó A tem altura 0, o nó B tem altura 1 e o nó R tem altura 2 nesta árvore de componentes. Similarmente, a árvore de componentes construída com base no conjunto $\mathcal{U}(f)$ (b) da Figura 2.8, os nós $(R, 0), (D, 4), (C, 7), (B, 7)$ e $(A, 7)$ estão presentes e $A \subset D$, $B \subset D$, $C \subset D$, e $D \subset R$. Nesta árvore de componentes, os nós A, B e C têm altura 0 , o nó D tem altura 1 e R tem altura 2.

A partir de agora, denotamos por $\mathcal{T}$ a árvore de componentes obtida dos conjuntos de níveis inferior ou superior, ou seja, $\mathcal{T} = (\mathcal{L}(f))$ ou $\mathcal{T} = (\mathcal{U}(f))$. Os nós de uma árvore de componentes $\mathcal{T}$ também são representados por τ. Além disso, a notação da árvore de componentes e de seus nós será estendida para a árvore compacta apresentada na próxima subseção.

2.3.5 *Max-tree* e *Min-tree*

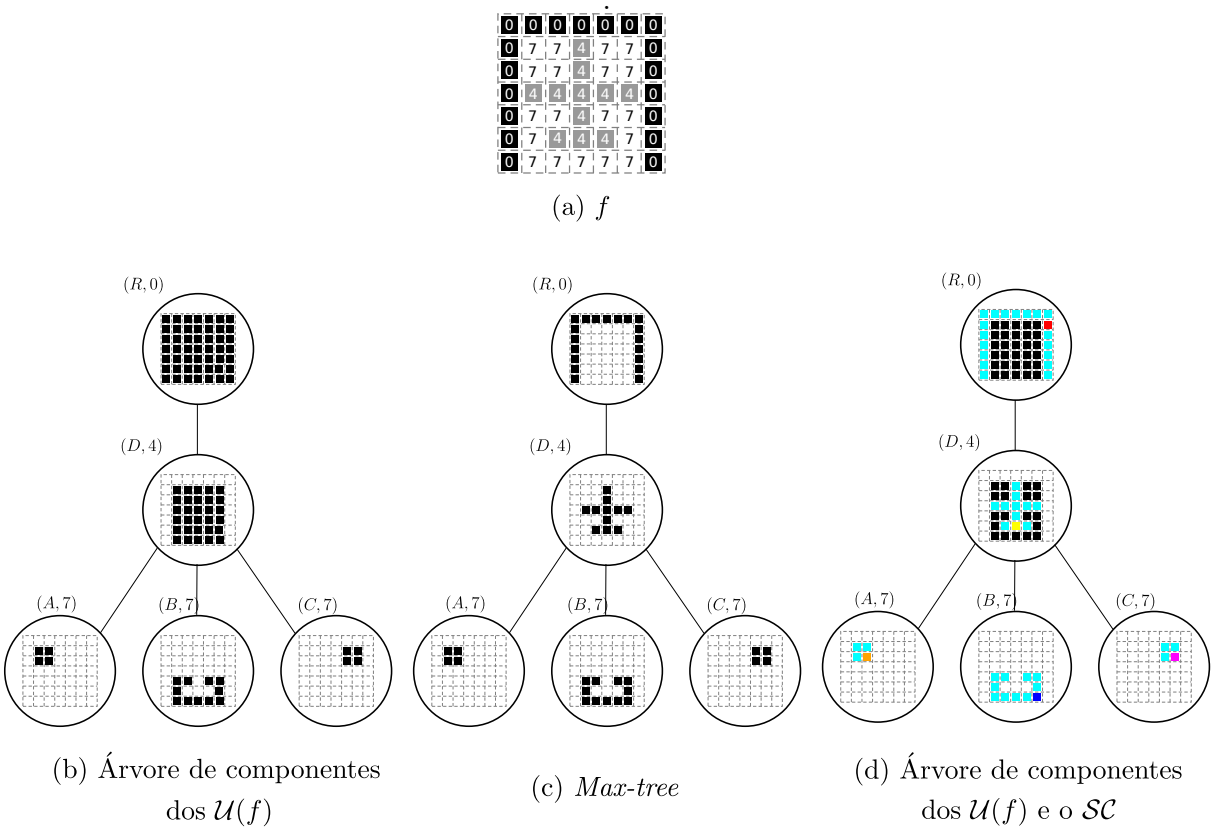
Como visto anteriormente, a árvore de componentes é composta por nós que representam componentes conexos de níveis superiores ou inferiores da imagem, acompanhados de suas intensidades. Esses nós possuem relações de parentesco, em que um nó é pai de outro se o seu componente conexo contém todos os *pixels* do componente do filho. Dessa forma, o armazenamento de todos os conjuntos de níveis na árvore de componentes pode ser custoso em termos de memória computacional. Como os *pixels* de um nó filho estão sempre contidos no nó pai, é possível armazenar apenas os *pixels* que estão no nó pai, mas não nos nós descendentes, o que é suficiente para reconstruir o $\mathcal{CC}$ que este nó representa. Dessa forma, essa abordagem pode ser aplicada a qualquer nó da árvore, tornando-a uma forma eficiente de armazenamento de *pixels* para todos os nós.

Conforme demonstrado por Salembier, Oliveras e Garrido (1998), essa estrutura de dados compacta e não redundante é conhecida como *max-tree* quando a árvore de componentes é construída sobre o conjunto de níveis superiores, e como *min-tree* quando a árvore é construída sobre o conjunto de níveis inferiores. Essas estruturas de dados não são redundantes, pois um *pixel* $p \in \mathcal{D}$ está armazenado apenas no menor componente conexo ($\mathcal{CC}$) que o contém. Além disso, a relação de parentesco entre os vértices da árvore faz com que o *pixel* p seja associado a todos os $\mathcal{CC}s$ que são seus ancestrais na árvore, levando-nos à definição do menor componente.

O menor componente conexo de um *pixel* $p \in \mathcal{D}$ em uma árvore $\mathcal{T}$, denotado como $\mathcal{SC}(\mathcal{T}, p)$, é o componente conexo que contém p e possui o menor número de *pixels* (ALVES, 2015). Em outras palavras, o menor $\mathcal{CC}$ de um *pixel* é aquele que contém esse *pixel* e não possui nenhum $\mathcal{CC}$ menor dentro da árvore que também contenha esse mesmo *pixel*. Para facilitar a compreensão dessa definição, a Figura 2.9 apresenta uma representação visual comparativa entre a árvore de componentes e a *max-tree* e uma árvore de componentes

com destaque ao $\mathcal{SC}$.

Figura 2.9 – *Comparação entre árvore de componentes e max-tree com destaque dos menores componentes conexos ($\mathcal{SC}$) da imagem f.*



(a) f

(b) Árvore de componentes dos $\mathcal{U}(f)$

(c) *Max-tree*

(d) Árvore de componentes dos $\mathcal{U}(f)$ e o $\mathcal{SC}$

Fonte: Elaborado pelo autor (2024).

A Figura 2.9 ilustra uma comparação entre estruturas de dados para representação de imagens, a árvore de componentes, a *max-tree*, e uma árvore de componentes destacando os menores componentes conexos ($\mathcal{SC}$). Na Figura 2.9(b), apresenta-se uma árvore de componentes construída a partir dos níveis superiores da imagem f, onde cada nó representa um componente conexo. Observa-se que os *pixels* de um nó são repetidos em seus nós ancestrais, resultando em redundância. A Figura 2.9(c) exibe uma *max-tree*, uma estrutura compacta e não redundante, onde apenas os *pixels* adicionais necessários para representar o componente de cada nó são armazenados. Por fim, na Figura 2.9(d), destaca-se a associação dos menores componentes conexos ($\mathcal{SC}$) aos *pixels* coloridos: p (vermelho), q (amarelo), r (laranja), s (azul) e t (rosa). Esses *pixels* estão associados aos seguintes $\mathcal{SC}$ na *max-tree*: $\mathcal{SC}(\mathcal{T}, p) = R$, $\mathcal{SC}(\mathcal{T}, q) = D$, $\mathcal{SC}(\mathcal{T}, r) = A$, $\mathcal{SC}(\mathcal{T}, s) = B$ e $\mathcal{SC}(\mathcal{T}, t) = C$. Essa representação evidencia a eficiência da *max-tree* e a relação entre os *pixels* e os menores componentes conexos na estrutura.

Por fim, em complemento ao $\mathcal{SC}$, um *pixel* é considerado *pixel* de nó compacto (do inglês, CNP - *Compact Node Pixel*) de um componente específico $\tau \in \mathcal{T}$ se, e somente se,

τ é o menor CC de p na árvore $\mathcal{T}$, como discutido por Alves (2015). Ou seja, $\tau = \mathcal{SC}(\mathcal{T}, p)$. Nesse contexto, denotamos o conjunto de *pixels* armazenados em um nó $\hat{\tau}$ como sendo composto apenas pelos CNPs de τ, então, $\hat{\tau} = \{p \in \mathcal{D} : \tau = \mathcal{SC}(\mathcal{T}, p)\}$.

2.3.6 Reconstrução de árvores

Conforme evidenciado na Equação 2.10 e na Equação 2.11, a reconstrução de uma imagem pode ser realizada a partir dos seus conjuntos de níveis. Para aplicar esse conceito utilizando uma árvore, definem-se as funções $level_{\mathcal{L}}(C) : \mathcal{L}(C) \to \mathbb{K}$ e $level_{\mathcal{U}}(C) : \mathcal{U}(C) \to \mathbb{K}$ (ALVES, 2015). Essas funções, derivadas das equações mencionadas, mapeiam os componentes conexos das árvores em níveis de cinza como

$$level_{\mathcal{L}}(C) = \min\{\lambda : C \in \mathcal{CC}(\mathcal{X}_{\lambda}(f), \mathcal{A}), \lambda \in \mathbb{K}\} \text{ e} \tag{2.14}$$

$$level_{\mathcal{U}}(C) = \max\{\lambda : C \in \mathcal{CC}(\mathcal{X}^{\lambda}(f), \mathcal{A}), \lambda \in \mathbb{K}\}. \tag{2.15}$$

Para simplificar a notação, a nomenclatura *level* será adotada para todas essas funções a partir deste ponto. Em adicional, partir de então, denotamos por $\mathcal{T}_f$ uma árvore construída a partir de uma imagem $f \in \mathcal{F}(\mathcal{D})$. Assim, partindo da árvore $\mathcal{T}_f$, é possível reconstruir a imagem f, conforme demonstrado por Alves (2015), usando a expressão

$$\forall p \in \mathcal{D}, f(p) = level(\mathcal{SC}(\mathcal{T}_f, p)). \tag{2.16}$$

Dessa maneira, essa operação é escrita como $f = Rec(\mathcal{T})$. Então, ao se tratar de uma $\text{Rec}((\mathcal{L}(f), \subseteq))$ ou $\text{Rec}((\mathcal{U}(f), \subseteq))$, estas operações são denominadas reconstrução inferior e reconstrução superior, respectivamente (ALVES, 2015).

2.4 Atributos

Um atributo, denotado por κ, é uma função que mapeia um conjunto de *pixels* e retorna um valor numérico, descrevendo alguma característica quantitativa desse conjunto, sendo formalmente representado por $\kappa : \mathcal{P}(\mathcal{D}) \to \mathbb{R}$ (ALVES, 2015). Exemplos de atributos comuns encontrados na literatura incluem área, altura, largura, perímetro, entre outros. A partir de então, as notações κ ou $\kappa_{(\tau)}$ são usadas para representar o valor do atributo para um nó específico $\tau \in \mathcal{T}$ ou nós da $\mathcal{T}$.

2.4.1 Classes de Atributos

Um atributo é considerado crescente se, e somente se, para qualquer nó τ na árvore $\mathcal{T}$, o valor do atributo $\kappa_{(\tau)}$ é menor ou igual ao valor do mesmo atributo para o nó pai τ_p (XU, 2013). Dessa maneira, o atributo crescente pode ser representado matematicamente

como

$$\forall \tau \in \mathcal{T}, \kappa_{(\tau)} \leqslant \kappa_{(\tau_p)}, \tag{2.17}$$

caso contrário, κ é classificado como um atributo não crescente. Exemplos de atributos crescentes incluem área, volume, entre outros. Por sua vez, atributos não crescentes englobam características como perímetro, circularidade, entre outros (SALEMBIER; OLIVERAS; GARRIDO, 1998; SALEMBIER; WILKINSON, 2009).

2.5 Atributos específicos

Embora a literatura apresente inúmeros atributos que podem ser extraídos dos nós de uma árvore, neste trabalho optou-se por focar nos atributos a serem definidos a seguir e que são considerados suficientes para abordar as aplicações discutidas em detalhes no Capítulo 4. É importante ressaltar que, na literatura, os atributos são associados a regiões e objetos. No entanto, devido à ênfase específica deste trabalho, as menções a regiões e objetos foram substituídas pelos cálculos de atributos, especialmente em nós. Também é importante destacar que parte das definições dos atributos foi baseada no trabalho de Burger e Burge (2022).

- Área: o número total de *pixels* que compõem o nó τ. Sendo definida pela seguinte equação

$$\kappa_{\text{área}(\tau)} = |\tau| = m_{00}(\tau) = \sum_{(u,v)\in\tau} 1. \tag{2.18}$$

- Volume: O atributo volume é área de um nó multiplicaod pelo seu nível, sendo calculado como

$$\kappa_{\text{volume}(\tau)} = |\tau| \times level(\tau). \tag{2.19}$$

- Perímetro: o perímetro é a soma das distâncias entre todos os *pixels* que compõem a borda de um componente conexo. O perímetro é calculado medindo a soma das distâncias entre *pixels* consecutivos na borda de uma região. Ao utilizar uma vizinhança $\mathcal{A}_4$, o comprimento medido do contorno (exceto quando esse comprimento é 1) será maior do que seu comprimento real. Por outro lado, para $\mathcal{A}_4$, os segmentos horizontais e verticais com 1, e os segmentos diagonais com $\sqrt{2}$. Desse modo, o atributo perímetro de um nó pode ser determinado como

$$\kappa_{\text{perímetro}(\tau)} = \sum_{p\in\partial\tau} \text{comprimento}(p_x, p_y), \tag{2.20}$$

com

$$\text{comprimento}(p_x, p_y) = \begin{cases} 1 & \text{se } \mathcal{A}_4, \mathcal{A}_8 \text{ para deslocamento horizontal e vertical,} \\ \sqrt{2} & \text{se } \mathcal{A}_8 \text{ para deslocamento diagonal,} \end{cases} \tag{2.21}$$

onde $\partial\tau$ representa os *pixels* do contorno do nó τ.

- Caixa Delimitadora: é o retângulo mínimo que contém todos os *pixels* de τ, define suas dimensões. O atributo caixa delimitadora pode ser descrito usando a seguinte expressão

$$\kappa_{\text{ caixa delimitadora}(\tau)} = \langle x_{min}, x_{max}, y_{min}, y_{max} \rangle. \tag{2.22}$$

- Largura e Altura: o atributo largura e a altura da caixa delimitadora podem ser definidas com base nos valores dos extremos da caixa delimitadora. Como os extremos são dados por (x_{max}, y_{max}) e (x_{min}, y_{min}), a largura e a altura podem ser calculadas como

$$\kappa_{\text{ largura}(\tau)} = x_{\max} - x_{\min} + 1 \text{ e } \kappa_{\text{ altura}(\tau)} = y_{\max} - y_{\min} + 1. \tag{2.23}$$

- Momentos Ordinários: caracterizam a distribuição dos *pixels* do nó. São calculados da seguinte forma

$$m_{pq}(\tau) = \sum_{p \in \tau} x^p \times y^q. \tag{2.24}$$

- Centróide: a centróide de um nó τ é definido como o ponto médio das coordenadas dos *pixels* pertencentes ao nó. As coordenadas da centróide são calculadas por

$$\bar{x} = \frac{m_{10}(\tau)}{m_{00}(\tau)}, \quad \bar{y} = \frac{m_{01}(\tau)}{m_{00}(\tau)}, \tag{2.25}$$

onde os momentos de primeira ordem são definidos como

$$m_{10}(\tau) = \sum_{(x,y) \in \tau} x, \tag{2.26}$$

$$m_{01}(\tau) = \sum_{(x,y) \in \tau} y. \tag{2.27}$$

- Momentos Centrais: medem a distribuição dos *pixels* em relação a centroide, eliminando a dependência de posição. O momento central correspondente de ordem p, q é definido como

$$\mu_{pq}(\tau) = \sum_{(x,p) \in \tau} (x - \bar{x})^p (y - \bar{y})^q. \tag{2.28}$$

- Momentos Centrais Normalizados: momentos centrais normalizados garantem invariância em relação à escala. Estes são calculado da seguinte maneira

$$\eta_{pq}(\tau) = \frac{\mu_{pq}(\tau)}{\mu_{00}(\tau)^{(p+q)/2+1}}. \tag{2.29}$$

- Orientação e Eixos Principais: orientação de um nó descreve a direção do seu eixo principal, que é o eixo que passa pelo centroide do nó e se alinha com a parte mais larga desse nó. Este eixo representa a direção principal ao longo da qual o nó é estendida, e a orientação é tipicamente medida como o ângulo θ em relação a um eixo de referência, geralmente o eixo horizontal. O atributo orientação pode então ser calculada usando a equação

$$\kappa_{\text{ orientação}(\tau)} = \frac{1}{2}\arctan\left(\frac{2\mu_{11}(\tau)}{\mu_{20}(\tau) - \mu_{02}(\tau)}\right). \tag{2.30}$$

  O intervalo do ângulo resultante está em radianos, isto é $[-\pi/2, \pi/2]$. Já os eixos, eixo maior e eixo menor são claculados como

$$\kappa_{\text{ comprimento do eixo maior}(\tau)} = 2\sqrt{\frac{\lambda_1}{|\tau|}}, \quad \kappa_{\text{ comprimento do eixo menor}(\tau)} = 2\sqrt{\frac{\lambda_2}{|\tau|}}, \tag{2.31}$$

  onde λ_1 e λ_2 são os autovalores da matriz de covariância.

- Razão: é relação entre o eixo maior e o eixo menor, indicando a forma da região, calculado pela seguinte equação

$$\kappa_{\text{ razão}(\tau)} = \frac{\kappa_{\text{comprimento do eixo maior}(\tau)}}{\kappa_{\text{comprimento do eixo menor}(\tau)}}. \tag{2.32}$$

- Média: o valor médio dos *pixels* do nó, expresso pela seguinte maneira

$$\kappa_{\text{ média}(\tau)} = \frac{1}{|\tau|}\sum_{(x,y)\in\tau} f(x,y). \tag{2.33}$$

- Variância: a variância mede a dispersão dos valores dos *pixels* em relação à média. Formalmente é calculada como

$$\kappa_{\text{ variância}(\tau)} = \frac{1}{|\tau|}\sum_{(x,y)\in\tau} (f(x,y) - \kappa_{\text{ média}(\tau)})^2. \tag{2.34}$$

- Desvio Padrão: mede a dispersão dos valores dos *pixels* ao redor da média, podendo então ser calculado como

$$\kappa_{\text{ desvio padrão}(\tau)} = \sqrt{\kappa_{\text{ variância}(\tau)}}. \tag{2.35}$$

- Compactidade: quantifica o quão próximo a forma da região é de um círculo perfeito. Este atributo é obtido pela seguinte equação

$$\kappa_{\text{ compacidade}(\tau)} = 4\pi \cdot \frac{\kappa_{\text{ área}(\tau)}}{\left(\kappa_{\text{ perímetro}(\tau)}\right)^2}. \tag{2.36}$$

- Excentricidade: a excentricidade mede a relação entre os eixos maior e menor do nó. O atributo excentricidade em relação ao nó é dada por

$$\kappa_{\text{ excentricidade}(\tau)} = \frac{\sqrt{\mu_{20} + \mu_{02} + \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}}{\sqrt{\mu_{20} + \mu_{02} - \sqrt{(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2}}}. \tag{2.37}$$

- Retangularidade: a retangularidade é uma medida que quantifica o quão próximo um τ se assemelha a um retângulo. O atributo retangularidade de um nó pode ser calculada usando a seguinte equação

$$\kappa_{\text{ retangularidade}(\tau)} = \frac{\kappa_{\text{ área}(\tau)}}{\kappa_{\text{ largura}(\tau)} \times \kappa_{\text{ altura}(\tau)}}. \tag{2.38}$$

- Momentos de Hu: os Momentos de Hu são invariantes geométricos que capturam características de forma de um componente conexo, independentemente de transformações como translação, rotação e escala. Os sete Momentos de Hu podem ser calculados a partir dos momentos centrais. Eles são calculados como

$$\begin{aligned}
\phi_1 =& \bar{\mu}_{20} + \bar{\mu}_{02}, \\
\phi_2 =& \left(\bar{\mu}_{20} - \bar{\mu}_{02}\right)^2 + 4\bar{\mu}_{11}^2, \\
\phi_3 =& \left(\bar{\mu}_{30} - 3\bar{\mu}_{12}\right)^2 + \left(3\bar{\mu}_{21} - \bar{\mu}_{03}\right)^2, \\
\phi_4 =& \left(\bar{\mu}_{30} + \bar{\mu}_{12}\right)^2 + \left(\bar{\mu}_{21} + \bar{\mu}_{03}\right)^2, \\
\phi_5 =& \left(\bar{\mu}_{30} - 3\bar{\mu}_{12}\right) \cdot \left(\bar{\mu}_{30} + \bar{\mu}_{12}\right) \cdot \left[\left(\bar{\mu}_{30} + \bar{\mu}_{12}\right)^2 - 3\left(\bar{\mu}_{21} + \bar{\mu}_{03}\right)^2\right] + \\
& \left(3\bar{\mu}_{21} - \bar{\mu}_{03}\right) \cdot \left(\bar{\mu}_{21} + \bar{\mu}_{03}\right) \cdot \left[3\left(\bar{\mu}_{30} + \bar{\mu}_{12}\right)^2 - \left(\bar{\mu}_{21} + \bar{\mu}_{03}\right)^2\right], \\
\phi_6 =& \left(\bar{\mu}_{20} - \bar{\mu}_{02}\right) \cdot \left[\left(\bar{\mu}_{30} + \bar{\mu}_{12}\right)^2 - \left(\bar{\mu}_{21} + \bar{\mu}_{03}\right)^2\right] + \\
& 4\bar{\mu}_{11} \cdot \left(\bar{\mu}_{30} + \bar{\mu}_{12}\right) \cdot \left(\bar{\mu}_{21} + \bar{\mu}_{03}\right), \\
\phi_7 =& \left(3\bar{\mu}_{21} - \bar{\mu}_{03}\right) \cdot \left(\bar{\mu}_{30} + \bar{\mu}_{12}\right) \cdot \left[\left(\bar{\mu}_{30} + \bar{\mu}_{12}\right)^2 - 3\left(\bar{\mu}_{21} + \bar{\mu}_{03}\right)^2\right] + \\
& \left(3\bar{\mu}_{12} - \bar{\mu}_{30}\right) \cdot \left(\bar{\mu}_{21} + \bar{\mu}_{03}\right) \cdot \left[3\left(\bar{\mu}_{30} + \bar{\mu}_{12}\right)^2 - \left(\bar{\mu}_{21} + \bar{\mu}_{03}\right)^2\right].
\end{aligned} \tag{2.39}$$

2.6 Processo de filtragem e estratégias de filtragem

Conforme apresentado na Seção 2.1, a aplicação de filtros conexos por meio de árvores envolve quatro etapas: a construção da representação hierárquica da imagem, o cálculo de atributos em cada nó da representação, a seleção de nós (filtragem) relevantes com base nesses atributos e a reconstrução da imagem a partir da árvore filtrada. As etapas de filtragem e reconstrução serão detalhadas em subseções posteriores. A Figura 2.1, apresentada anteriormente, ilustra essas etapas.

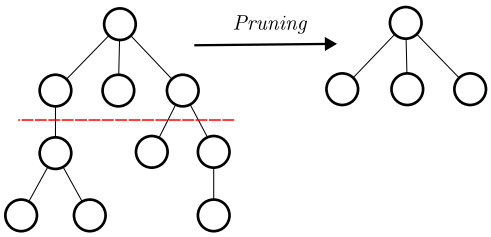
Uma vez que a árvore é construída, na etapa do fluxograma exposto acima na Figura 2.1, o processo conhecido como filtragem é apresentado. Essa etapa é central para todo o processo. Com base nos nós a serem filtrados, as estratégias de filtragem baseadas em árvores podem ser categorizadas em dois tipos principais, *pruning* e *nonpruning*.

2.7 Estratégias de filtragem

2.7.1 *Pruning*

As estratégias de *pruning* envolvem a remoção de subárvores inteiras que estão enraizadas em nós específicos, preservando os nós acima daqueles selecionados para remoção, efetivamente cortando os sub-ramos da árvore (SALEMBIER; OLIVERAS; GARRIDO, 1998; SALEMBIER; WILKINSON, 2009). Ou seja, para qualquer nó filtrado por uma estratégia de *pruning*, todos os seus descendentes são removidos. Uma ilustração de *pruning* de árvore pode ser vista na Figura 2.10.

Figura 2.10 – *Estratégia de pruning*



Fonte: Adaptado de Xu (2013).

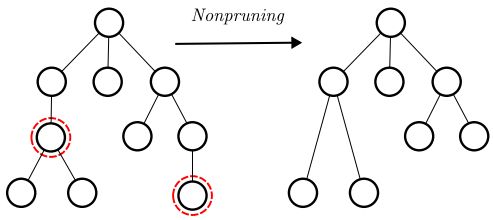
Na Figura 2.10, dado um limiar i e um atributo κ qualquer, os nós com valores abaixo do limiar são removidos, indicados pela linha vermelha tracejada.

2.7.2 *Nonpruning*

Por outro lado, a estratégia de *nonpruning* pode preservar todos ou alguns dos nós descendentes de um nó filtrado, que seria removido por não satisfazer o critério defi-

nido (SALEMBIER; OLIVERAS; GARRIDO, 1998; SALEMBIER; WILKINSON, 2009). Sendo assim, a relação de parentesco dos nós descendentes em relação ao nó removido é atualizada, passando a ser conectada ao nó ancestral do nó removido. A seguir, na Figura 2.11, é apresentada a estratégia de *nonpruning*.

Figura 2.11 – *Estratégia de nonpruning*



Fonte: Adaptado de Xu (2013).

De acordo com as definições das estratégias de *pruning* e *nonpruning*, é possível observar que a escolha da estratégia de filtragem está correlacionada com a propriedade do atributo (XU, 2013), sendo os atributos crescentes e não crescentes, discutidos na Subseção 2.4.1.

2.7.3 Regras de filtragem de árvores para atributos crescentes

Ao selecionar um atributo crescente, a remoção ou preservação de um nó pode ser interpretada como uma função booleana a um atributo escolhido dos nós da árvore. Assim, isso pode ser formalmente expresso pela função característica $\mathbf{s} : \mathcal{T}_f \to \{0, 1\}$, que indica os nós a serem mantidos na árvore, definida da seguinte maneira, sendo $\forall \tau \in \mathcal{T}_f$,

$$\mathbf{s}_(\tau, \kappa, i) = \begin{cases} 1, & \text{se } \kappa_{(\tau)} \geq i, \\ 0, & \text{caso contrário,} \end{cases} \tag{2.40}$$

onde i é o limiar que determina se um dado nó $\tau \in \mathcal{T}$ deve ser removido ou preservado. Se o valor de um dado atributo estiver abaixo de um limiar, o nó será removido e vice-versa.Na Subseção a seguir, as regras de filtragem serão discutidas em detalhes, abordando as regras Mínima, Máxima, Direta e Subtrativa.

2.7.4 Regras de filtragem de árvores para atributos não crescentes

A filtragem por atributo não crescente também é realizada através da função característica $\mathbf{s} : \mathcal{T}_f \to \{0, 1\}$, no entanto, a remoção ou preservação leva em conta o critério a ser utilizado de acordo com cada regra escolhida de filtragem. Então, Salembier, Oliveras e Garrido (1998) descrevem várias regras distintas utilizadas para filtrar árvores morfológicas, incluindo a regra Mínima, a regra Máxima e a regra Direta. Sendo essas estratégias

de *pruning*. Além dessas regras, a regra Subtrativa, Urbach e Wilkinson (2002), é uma estratégia de *nonpruning*. A seguir, são apresentados detalhes das regras de filtragem de árvores morfológicas.

- **Regra Mínima:** Um nó $\tau \in \mathcal{T}_f$ é removido se $\kappa_{(\tau)} < i$, ou se qualquer ancestral τ_a de τ possuir $\kappa_{(\tau_a)} < i$. Dada a condição desta regra, a filtragem $\forall \tau \in \mathcal{T}_f$ é expressa pela seguinte condição

$$\mathbf{s}(\tau, \kappa, i) = \begin{cases} 1, & \text{se } \kappa(\tau) \geq i \text{ e } \kappa(\tau_a) \geq i \text{ para todo ancestral } \tau_a \text{ de } \tau, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.41}$$

- **Regra Máxima:** Para um nó $\tau \in \mathcal{T}_f$, esse é removido somente quando as seguintes condições forem estritamente satisfeitas, $\kappa_{(\tau)} < i$ e todos os descendentes τ_d também satisfazem $\kappa_{\tau_d} < i$. Seguindo a condição desta regra, a filtragem $\forall \tau \in \mathcal{T}_f$ é determinada pela seguinte maneira

$$\mathbf{s}(\tau, \kappa, i) = \begin{cases} 1, & \text{se } \kappa(\tau) \geq i \text{ ou } \exists \tau_d \text{ descendente de } \tau \text{ tal que } \kappa(\tau_d) \geq i, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.42}$$

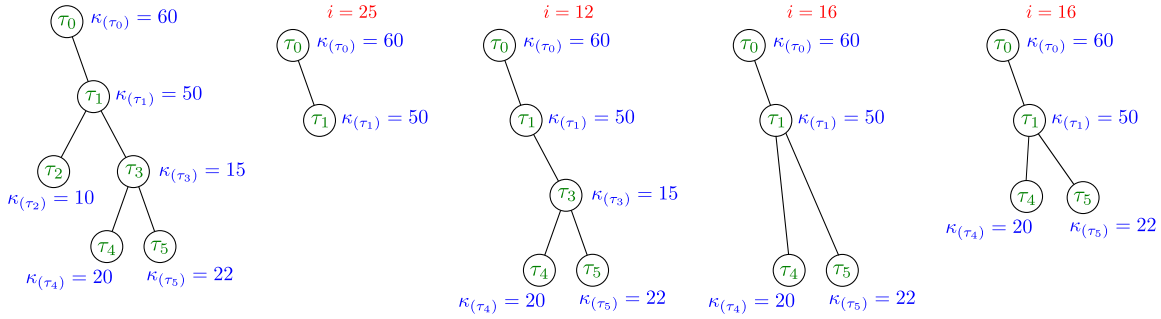
- **Regra Direta:** Um nó $\tau \in \mathcal{T}_f$ é removido da árvore se, e somente se, a condição $\kappa_{(\tau)} < i$ for atendida. A regra Direta não envolve outras condições além dessa simples comparação. Sendo assim, baseado nesta condição esta regra pode ser descrita pela seguinte maneira

$$\mathbf{s}(\tau, \kappa, i) = \begin{cases} 1, & \text{se } \kappa(\tau) \geq i, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.43}$$

- **Regra Subtrativa:** A regra Subtrativa é semelhante à regra Direta, mas com uma diferença crucial, quando um nó τ é removido, os níveis de cinza de todos os seus descendentes não removidos são ajustados para o nível de cinza do ancestral removido. Como esta regra é similar a regra Direta, neste caso então a filtragem pode ser realizada através da Equação 2.43.

A seguir, um exemplo das regras de filtragem é ilustrado na Figura 2.12.

Figura 2.12 – *Um exemplo de filtragem de árvores. Da esquerda para direita, árvore de entrada e árvores filtradas utilizando diferentes regras de filtragem para o atributo não crescente em conjunto com os devidos limiares utilizados representados pela cor vermelha.*



(a) Entrada (b) Mínima (c) Máxima (d) Direta (e) Subtrativa

Fonte: Elaborado pelo autor (2024).

A Figura 2.12 apresenta um exemplo prático de como as diferentes regras de filtragem (mínima, máxima, direta e subtrativa) atuam sobre uma árvore, utilizando um atributo não crescente (representado pelos valores numéricos nos nós em azul) e diferentes limiares (representados pelo valor de i em cada regra). Na Figura 2.12(b) - Mínima, a árvore é filtrada com um limiar de 25, resultando na remoção dos nós τ_2, τ_3, τ_4 e τ_5. Na Figura 2.12(c) - Máxima, a árvore é filtrada com um limiar de 12, resultando na remoção do nó τ_2. Na Figura 2.12(d) - Direta, a árvore é filtrada com um limiar de 16, onde os nós τ_2 e τ_3 são removidos, e as relações de parentesco dos nós τ_4 e τ_5 são atualizadas. Por fim, na Figura 2.12(e) - Subtrativa, a árvore é filtrada com um limiar de 16. Após a filtragem, os nós τ_2 e τ_3 são removidos, e a relação de parentesco dos nós τ_4 e τ_5 é atualizada, com ambos passando a ter o nível de cinza $\lambda = 2$.

2.7.5 Reconstrução de Árvores Filtradas

Na Subseção 2.3.6, discutimos como a reconstrução de uma imagem f pode ser realizada com base nos conjuntos de níveis que a compõem, utilizando a Equação 2.16. No entanto, não abordamos as estratégias específicas de filtragem aplicáveis às árvores morfológicas, sendo *pruning* e *nonpruning*. Para complementar, exploramos as estratégias de filtragem de árvores morfológicas e as diferentes regras de filtragem. Baseado nisso, a seguir o foco será a reconstrução de árvores filtradas.

Após a aplicação de uma filtragem à árvore morfológica original $\mathcal{T}_f$, obtém-se uma árvore filtrada $\mathcal{T}_g$, composta pelos nós que atendem aos critérios de filtragem estabelecidos. A função de reconstrução Rec permite recompor a imagem utilizando os nós que permaneceram em $\mathcal{T}_g$. Dependendo da estratégia de filtragem adotada, seja *pruning* ou *nonpruning*, a reconstrução pode ser realizada por duas diferentes equações a serem apre-

sentadas. Para as estratégias de *pruning*, com a utilização de atributos crescentes, as regras Máxima, Mínima e Direta podem ser reconstruídas pela seguinte equação

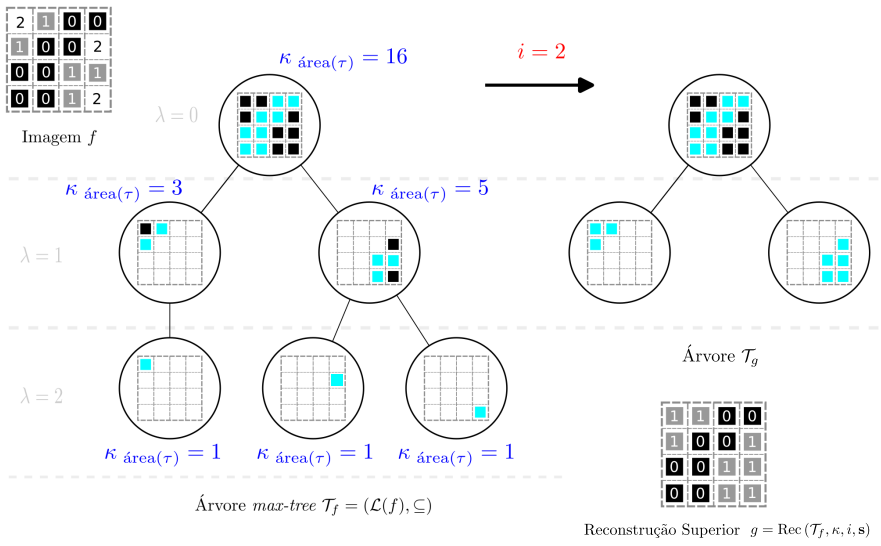
$$[\mathrm{Rec}\,(\mathcal{T}_f, \kappa, i)]\,(p) = \begin{cases} level\,(\mathcal{SC}\,(\mathcal{T}_f, p))\,, & \text{se } \mathbf{s}\,(\mathcal{SC}\,(\mathcal{T}_f, p)\,, \kappa, i) = 1, \\ level(\tau_a), & \text{caso contrário,} \end{cases} \tag{2.44}$$

onde τ_a é o primeiro nó ancestral que satisfaz a função $\mathbf{s}$, a qual depende do critério de filtragem utilizado nas regras de filtragem (Mínima, Máxima e Direta). Além disso, é importante destacar que, ao escolher qualquer uma das quatro regras de filtragem com atributos crescentes, o resultado da imagem reconstruída será o mesmo para todas as regras. No entanto, quando os atributos não são crescentes, o resultado da imagem reconstruída será diferente devido às características de cada regra, como explorado anteriormente. Por sua vez, para as estratégias de *nonpruning*, como a regra Subtrativa, a reconstrução é realizada conforme descrito na equação abaixo

$$[\mathrm{Rec}\,(\mathcal{T}_f, \kappa, i, \mathbf{s})]\,(p) = \sum_{\tau \in \mathcal{T}_f : p \in \tau} |level(\tau) - level(\mathrm{Parent}(\tau))| \times \mathbf{s}(\tau, \kappa, i), \tag{2.45}$$

onde $\mathrm{Parent}(\tau)$ é o nó pai de τ, e τ é preservado após a filtragem, relembrando que $\mathbf{s}$ neste caso é a regra Subtrativa. Nesta abordagem, a reconstrução leva em conta as diferenças de nível entre cada nó e seu pai e, em adicional, esta equação para a reconstrução contempla tanto os atributos crescentes como não crescentes. A seguir, a Figura 2.13 apresenta um exemplo visual de uma operação de *pruning* e reconstrução aplicada a uma árvore *max-tree* para uma dada imagem f.

Figura 2.13 – *Exemplo da operação de pruning e reconstrução.*



Fonte: Elaborado pelo autor (2024).

Na Figura 2.13, temos uma imagem f de dimensões 4×4 com três níveis de intensidade. A partir dessa imagem, é construída a árvore *max-tree* $\mathcal{T}_f$, onde os valores do atributo área são destacados para cada nó. Em seguida, é aplicada uma filtragem utilizando a regra Direta com um limiar de 2, resultando na árvore filtrada denominada $\mathcal{T}_g$. A imagem final g é então reconstruída substituindo os valores dos *pixels* pertencentes aos nós removidos pelo nível do menor componente que os contém. Esse processo é realizado com base na árvore filtrada $\mathcal{T}_g$, no atributo κ, no limiar i escolhido, e na função $\mathbf{s}$, que segue uma das regras de filtragem previamente definidas, nesta ocasião a regra Direta.

2.8 Redes Neurais Artificiais

As redes neurais artificiais (RNAs) são modelos matemáticos inspirados na biologia, projetados para simular a forma como o cérebro humano processa informações (AGATONOVIC-KUSTRIN; BERESFORD, 2000). Elas são compostas por camadas de unidades chamadas neurônios artificiais, que recebem entradas, processam essas informações por meio de funções de ativação e geram saídas (GOODFELLOW; BENGIO; COURVILLE, 2016). Cada neurônio artificial, que pode ser comparado a uma célula na rede, recebe múltiplas entradas e, após o processamento, gera uma saída específica (JAIN; MAO; MOHIUDDIN, 1996).

O aprendizado dessas redes é realizado por meio de ajustes nos pesos das conexões entre os neurônios, com o objetivo de minimizar o erro entre a saída prevista e a saída real, utilizando algoritmos como o de retropropagação (GOODFELLOW; BENGIO; COURVILLE, 2016).

O modelo mais simples de rede neural é o *Perceptron*, uma rede de uma única camada que pode ser usada para resolver problemas lineares. No entanto, redes mais complexas, como as redes neurais multicamadas (MLP), são capazes de lidar com problemas não lineares e aprender representações mais complexas (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.8.1 *Perceptron*

O *Perceptron* foi uma das primeiras arquiteturas de redes neurais criadas, desenvolvido por Frank Rosenblatt em 1958 (ROSENBLATT, 1958). Trata-se de um modelo de rede neural de camada única, composto por um conjunto de entradas, um peso para cada entrada, uma função de soma ponderada e uma função de ativação. Sua principal inovação foi a introdução do processo de aprendizado supervisionado, no qual os pesos das conexões entre os neurônios são ajustados de acordo com os erros cometidos nas previsões Colliot (2023). Matematicamente, o *Perceptron* pode ser descrito da seguinte forma. Dados $x_1, x_2, \ldots, x_n$ como entradas, e $w_1, w_2, \ldots, w_n$ como os respectivos pesos, a saída y

do *Perceptron* é dada por

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right), \tag{2.46}$$

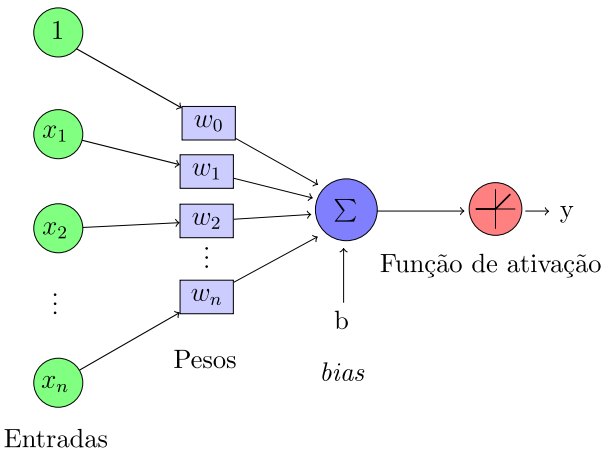
onde σ é a função de ativação e b é o viés. O ajuste dos pesos é realizado com base no erro entre a saída prevista e a saída desejada, utilizando o algoritmo de aprendizado do *Perceptron*. Esse algoritmo ajusta os pesos de forma iterativa, de acordo com a fórmula

$$w_i \leftarrow w_i + \Delta w_i \tag{2.47}$$

onde $\Delta w_i = \eta(t-y)x_i$, sendo t o valor alvo (verdadeiro) da saída, y a saída do *Perceptron*, η a taxa de aprendizado e x_i o valor da entrada. Esse processo de ajuste permite que o *Perceptron* aprenda a classificar corretamente os dados apresentados.

A seguir, a Figura 2.14 apresenta uma representação gráfica de um *Perceptron*.

Figura 2.14 – *Modelo gráfico de um Perceptron simples.*



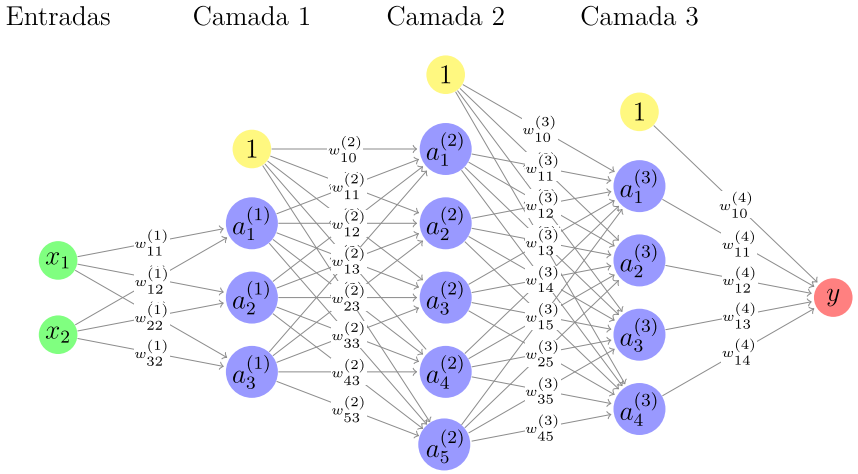
Fonte: Elaborado pelo autor (2024).

Na Figura 2.14, a estrutura de um *Perceptron* é ilustrada, onde cada elemento de entrada está conectado a um neurônio. Esses elementos são combinados com os pesos e o *bias* (também conhecidos como parâmetros do modelo), e, em seguida, são passados por uma função de ativação para gerar uma decisão final.

2.8.2 *Perceptron* Multicamadas

Ao contrário do *Perceptron* simples, o *Perceptron* multicamadas (ou *MultiLayer Perceptron* - MLP) possui uma arquitetura mais complexa, com várias camadas ocultas intermediárias. A Figura 2.15 ilustra um exemplo de MLP.

Figura 2.15 – *Representação de uma rede MLP. A soma e os nós de não linearidade foram omitidos para simplificar a figura.*



Fonte: Elaborado pelo autor (2024).

Na Figura 2.15, observamos as camadas de entrada representadas pelos nós x_1 e x_2 (em verde), que correspondem às variáveis de entrada do modelo. As camadas ocultas são compostas pelos neurônios $a_1^{(1)}, a_2^{(1)}, a_3^{(1)}, a_4^{(1)}$ (em azul), e há um nó de *bias* (em amarelo), identificado como 1. A segunda camada oculta (Camada 2) contém os neurônios $a_1^{(2)}, a_2^{(2)}, a_3^{(2)}, a_4^{(2)}, a_5^{(2)}$ e outro nó de *bias*. A terceira camada oculta (Camada 3) contém os neurônios $a_1^{(3)}, a_2^{(3)}, a_3^{(3)}, a_4^{(3)}$ e o outro nó de *bias*. A camada de saída é representada por um único neurônio y (em vermelho), que fornece o resultado final da rede. As conexões entre os neurônios, representadas por linhas, são associadas a pesos $w_{ij}^{(l)}$, que são ajustados durante o treinamento da rede. Na mesma figura, os neurônios de uma camada oculta estão conectados a todos os neurônios da camada anterior e da camada seguinte. Para a primeira camada oculta, as conexões são feitas diretamente com as entradas. Cada conexão entre o neurônio i da camada $l-1$ e o neurônio j da camada l é associada ao peso $w_{ij}^{(l)}$, e esses pesos são organizados em uma matriz $\mathrm{W}^{(l)}$. A transformação linear que ocorre em cada camada é expressa pela equação

$$\mathrm{z}^{(l)} = \mathrm{W}^{(l)} \cdot \mathrm{a}^{(l-1)}, \tag{2.48}$$

onde $\mathrm{z}^{(l)}$ é o vetor de entradas ponderadas para a camada l, $\mathrm{W}^{(l)}$ é a matriz de pesos da camada l, e $\mathrm{a}^{(l-1)}$ é o vetor de ativações da camada anterior $l-1$. Após essa transformação, aplica-se uma função de ativação não linear.

$$\mathrm{a}^{(l)} = \sigma(\mathrm{z}^{(l)}), \tag{2.49}$$

onde σ é a função de ativação aplicada elemento por elemento ao vetor $\mathrm{z}^{(l)}$, resultando no vetor de ativações $\mathrm{a}^{(l)}$.

Cada camada de uma RNA realiza a computação de uma ativação com base nas suas entradas e na função de ativação não linear associada. A seguir, discutiremos as funções de ativação utilizadas nas redes neurais.

2.8.3 Funções de ativação e não linearidade

As funções de ativação desempenham um papel crucial nas RNAs, aprendendo os recursos abstratos por meio de transformações não lineares (DUCH; JANKOWSKI, 1999). Elas introduzem não linearidade à rede, permitindo a representação de relações complexas entre entradas e saídas. Para isso, a saída de cada *Perceptron* passa por uma função de ativação, que decide se o neurônio deve ser ativado ou não (NWANKPA et al., 2018). A seguir, três funções não lineares comuns são apresentadas e suas representações são ilustradas na Figura 2.16 pouco mais abaixo.

- Sigmóide: A função de ativação *Sigmóide* (LECUN; BENGIO; HINTON, 2015), também conhecida como função logística, é uma função de ativação não linear comumente usada em RNAs. A função *Sigmóide* é usada para representar uma distribuição de probabilidade sobre um intervalo de 0 a 1. A função *Sigmóide* é definido como

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{2.50}$$

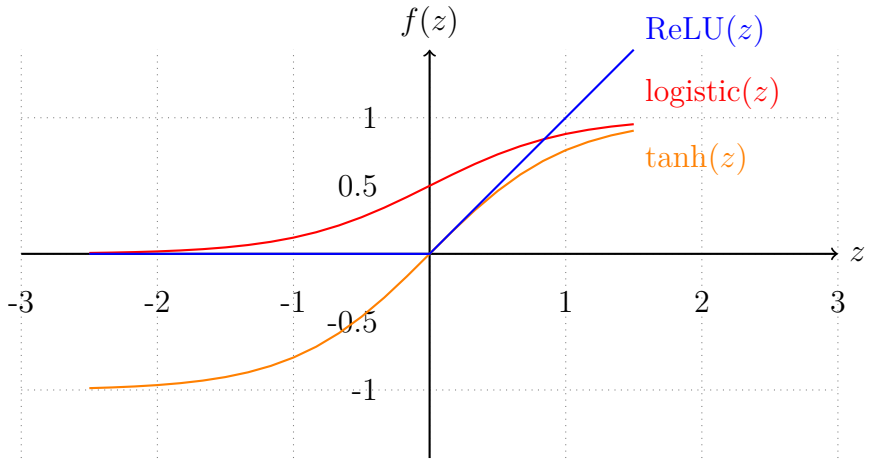
- Tangente Hiperbólica: A função tangente hiperbólica (LECUN; BENGIO; HINTON, 2015) é outro tipo de função de ativação usada em RNAs. A função tangente hiperbólica também conhecida como função *tanh*, é uma função mais suave centrada em zero cujo intervalo está entre -1 a 1, portanto, a saída da função tanh é dada por

$$tanh(z) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}}. \tag{2.51}$$

- Unidade Linear Retificada: A função de ativação unidade linear retificada (do inglês, *Rectified Linear Unit* - ReLU) (NAIR; HINTON, 2010) é uma função simples que é a função de identidade para entrada positiva e zero para entrada negativa e dado como

$$Relu(z) = max(0, z). \tag{2.52}$$

Figura 2.16 – *Algumas das funções de ativação mais comuns, Sigmóide, tanh, ReLU.*



Fonte: Elaborado pelo autor (2024).

2.8.4 Função de custo

As RNAs requerem uma métrica para avaliar o processo de aprendizado. Nesse contexto, a função de custo, também conhecida como função de erro, desempenha o papel de avaliador de desempenho da RNA. Na literatura, uma das funções de custo mais comuns para problemas de regressão é conhecida como Erro Quadrático Médio (do inglês, *Mean Squared Error* - MSE), que é denotada como

$$E_{mse} = \frac{1}{M} \sum_{\mathcal{D}} \frac{1}{2} (\mathrm{y} - \hat{\mathrm{y}})^2 , \tag{2.53}$$

onde $\mathcal{D}$ é um conjunto de dados de M amostras. Cada amostra consiste em uma entrada x e sua saída desejada associada y. Aqui, ŷ representa a saída predita pela rede neural para a amostra de entrada x. Além disso, a função de custo E_{mse} torna-se reduzida, ou seja, $E_{mse} \approx 0$, precisamente quando a saída prevista y(x) é aproximadamente igual à saída real, ŷ, para todas as entradas de treinamento x. Então a rede pode encontrar os pesos e *bias* de modo que $E_{mse} \approx 0$. Por outro lado, se o custo E_{mse} for grande, isso implica que a saída prevista y(x) se desvia significativamente da saída real para um número considerável de amostras de entrada.

2.8.5 Retropropagação

Durante a aprendizagem de uma rede neural, o processo de propagação direta ou propagação para frente inicia-se quando uma entrada x fornece informações iniciais, propagando-se através das camadas ocultas até finalmente produzir uma saída ŷ. Du-

rante essa fase, a propagação pode continuar até que a função de custo seja reduzida (LECUN; BENGIO; HINTON, 2015).

Por sua vez, durante o mesmo processo de aprendizagem, o algoritmo de retropropagação (RUMELHART; HINTON; WILLIAMS, 1986) permite que a informação do erro da propagação seja enviada para trás através da rede para calcular o gradiente (LECUN; BENGIO; HINTON, 2015). Em outras palavras, é o algoritmo do gradiente descendente, usado de maneira eficiente para calcular os gradientes automaticamente em apenas duas passagens pela rede: uma para frente e uma para trás, sendo então capaz de calcular o gradiente do erro da rede em relação a cada parâmetro do modelo (GÉRON, 2022).

Já a atualização de pesos na otimização através do gradiente descendente é similar à Equação 2.47, para compreender um pequeno passo na direção negativa do gradiente. Em um contexto relacionado, o algoritmo de retropropagação apresenta semelhanças com a regra de aprendizado do *Perceptron*, conforme expresso na Equação 2.47, onde a ideia central nesta abordagem é ajustar cada peso da rede proporcionalmente ao erro $E = \hat{y} - y$ e à entrada. Dessa maneira, mesmo que em MLPs seja comum considerar diferentes tipos de funções de custo, o mesmo conceito de aprendizado (Equação 2.47) se aplica aqui, qual busca modificar os pesos para minimizar o erro da função de custo, conforme demonstrado pela equação a seguir

$$\mathrm{W}_t = \mathrm{W}_{t-1} - \eta \frac{\partial E(\mathrm{W}_{t-1})}{\partial \mathrm{W}}, \tag{2.54}$$

onde W_t representa a matriz de pesos da RNA no instante de tempo t, onde W é a matriz de pesos e t é o instante de tempo. W_{t-1} representa a matriz de pesos da RNA no instante de tempo anterior $(t-1)$. Já η a taxa de aprendizado. Por sua vez, $\partial E/\partial \mathrm{W}$ representa o gradiente da função de custo E (como a Equação 2.53) em relação à matriz de pesos W.

Ainda, no contexto do algoritmo de retropropagação, a regra da cadeia no cálculo diferencial é fundamental para calcular as derivadas de funções formadas pela composição de outras funções cujas derivadas são conhecidas. O cálculo de derivadas parciais possibilita determinar como cada peso na rede contribui para o erro total, permitindo, assim, aprimorar o desempenho da rede por meio de ajustes ponderados (LECUN; BENGIO; HINTON, 2015). A seguir, é apresentada a equação que calcula a fração do erro que pode ser atribuída a cada peso da rede, neste exemplo $w_{ij}^{(l)}$ é

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = -\frac{1}{m} \sum_{d=1}^{m} \left(y^{(d)} - \hat{y}^{(d)} \right) \frac{\partial \hat{y}^{(d)}}{\partial w_{ij}^{(l)}}, \tag{2.55}$$

onde m é o número de exemplos, $y^{(d)}$ é a saída desejada para o exemplo d, $\hat{y}^{(d)}$ é a saída calculada pela rede para o exemplo d. Essa equação mostra que cada peso é ajustado com base no erro $\left(y^{(d)} - \hat{y}^{(d)} \right)$ multiplicado pela sensibilidade da saída $\hat{y}^{(d)}$ em relação a esse peso. Em outras palavras, o ajuste do peso é proporcional à contribuição dele para o erro

total, conforme determinado pela regra da cadeia.

2.9 Seleção de atributos

A seleção de características ou atributos é uma etapa crucial que contribui para a compreensão dos dados, reduz a necessidade computacional, diminui os efeitos da maldição da dimensionalidade e melhora o desempenho de modelos de aprendizagem de máquina e aprendizagem profunda (CHANDRASHEKAR; SAHIN, 2014). O foco da seleção de atributos é selecionar um subconjunto de variáveis a partir dos dados de entrada que possa descrever eficientemente os dados, ao mesmo tempo em que reduz os efeitos do ruído ou de variáveis irrelevantes, e ainda assim forneça bons resultados de previsão (GUYON; ELISSEEFF, 2003). Técnicas de seleção de atributos podem ser aplicadas durante o pré-processamento de dados, permitindo uma redução eficiente da dimensionalidade (JOVIĆ; BRKIĆ; BOGUNOVIĆ, 2015), o que facilita a identificação dos melhores atributos para o treinamento dos modelos. Na literatura, embora o algoritmo de busca exaustiva para encontrar um subconjunto ótimo de características seja, na maioria dos casos, inviável, diversas estratégias de busca alternativas têm sido propostas para contornar essa limitação (JOVIĆ; BRKIĆ; BOGUNOVIĆ, 2015). A seleção de atributos pode ser feita através de técnicas como filtros, *wrappers* ou métodos baseados em *embedded*, que avaliam a importância dos atributos de acordo com seu impacto no desempenho do modelo (CHANDRASHEKAR; SAHIN, 2014).

2.9.1 Métodos *Wrappers*

Os métodos *wrappers* são uma abordagem de seleção de atributos em que a performance de um modelo é usada para avaliar a qualidade de diferentes subconjuntos de atributos (WAH et al., 2018). Em outras palavras, esses métodos embrulham o modelo de aprendizado, utilizando-o para testar diferentes combinações de atributos e selecionar aquele(s) que fornecem o melhor desempenho. Na literatura, um dos algoritmos pertencentes a essa classe de seleção de atributos é a Seleção Sequencial para Frente (do inglês, *Sequential Forward Selection* - SFS). O SFS é um algoritmo de seleção de características utilizado para escolher um subconjunto de atributos de um conjunto de dados com o objetivo de melhorar o desempenho de um modelo preditivo. O algoritmo começa utilizando apenas um dos atributos e tenta modelar os dados usando o modelo fornecido. Em seguida, ele seleciona o atributo que oferece a maior acurácia ou uma métrica de desempenho definida. Esse processo se repete até atingir um número determinado de atributos definido (SHAFIEE et al., 2021).

2.10 Conceitos relacionados à arquitetura da rede neural

Em complemento à Seção 2.8, serão apresentados os principais elementos e conceitos relacionados à arquitetura da rede neural e que são mencionados na Seção 4.

Número de camadas: em relação ao número de camadas, como mencionado na Seção anterior, a arquitetura proposta consiste em um modelo de *Perceptron* Simples. Ou seja, uma única camada é utilizada no modelo.

Taxa de aprendizagem: taxa de aprendizado (ver Subseção 2.8.5) é um parâmetro que impacta a convergência do modelo, controlando a velocidade com que os pesos da rede neural são ajustados durante o treinamento. Com valor positivo pequeno (geralmente entre 0.0 e 1.0), uma alta taxa pode acelerar o treinamento, mas também pode dificultar a convergência se os gradientes se tornarem muito pequenos ou grandes. Esse hiperparâmetro é crucial para o desempenho e a estabilidade da rede neural.

Função de Ativação: a função de ativação em uma rede neural determina como as entradas ponderadas afetam os neurônios em cada camada e o tipo de transformação na saída de cada neurônio. Ela tem grande impacto no treinamento, capacidade e desempenho da rede, incluindo sua convergência. Por isso, a escolha da função de ativação deve ser feita com base nas características do problema a ser resolvido. No contexto deste trabalho, utiliza-se a função de ativação Sigmóide, mencionada na Seção 2.8.3.

Tamanho do Lote: o tamanho do lote é um parâmetro importante no aprendizado de redes neurais, pois define o número de amostras usadas em uma única passagem de propagação e retropropagação. Ele impacta o equilíbrio entre velocidade de treinamento e precisão. Tamanhos de lote menores podem melhorar a precisão, mas aumentam os custos computacionais e o tempo de treinamento. Por outro lado, tamanhos maiores reduzem o tempo de treinamento, mas podem diminuir a precisão e aumentar o risco de sobreajuste. Além disso, o tamanho do lote afeta a convergência, o procedimento de otimização e a taxa de aprendizado do modelo.

Número de Épocas: o número de épocas determina quantas vezes uma rede neural passa por todo o conjunto de dados de treinamento. Ele afeta a convergência, o processo de otimização e a taxa de aprendizado. Mais épocas podem melhorar a precisão, mas podem levar ao sobreajuste, enquanto menos épocas aceleram o treinamento, mas podem prejudicar a generalização do modelo.

Normalização: o treinamento de redes neurais funciona melhor quando os dados estão em uma escala similar, ou seja, normalizados. Para isso, para o treinamento dos modelos discutidos nesta Seção, os dados foram normalizados com a normalização z. A normalização z, ou escore z, é uma técnica utilizada para transformar os dados de maneira que a média seja 0 e o desvio padrão seja 1. Formalmente, o escore z pode ser calculado como

$$z = \frac{x - \mu}{\sigma}, \tag{2.56}$$

onde x é o valor observado, μ é a média do conjunto de treino, σ é o desvio padrão do conjunto de treino.

2.10.1 Otimizador

O otimizador utilizado nos experimentos do Capítulo 4 foi o AdaMax, uma versão modificada do otimizador Adam (do inglês, *Adaptive Moment Estimation*), proposto por Kingma e Ba (2014), que é um método de otimização que utiliza médias móveis dos gradientes e dos quadrados dos gradientes para atualizar os parâmetros de um modelo durante o treinamento. Essas médias são inicializadas como zero e calculadas como

$$\mathbf{m}_t = \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \cdot g_t \quad \text{e} \quad \mathbf{v}_t = \beta_2 \cdot \mathbf{v}_{t-1} + (1 - \beta_2) \cdot g_t^2, \tag{2.57}$$

onde, $\mathbf{m}_t$ é o vetor das médias dos gradientes no tempo t, $\mathbf{v}_t$ é o vetor das médias dos quadrados dos gradientes no tempo t, já g_t é o gradiente da função de custo, β_1 é o peso atribuído à média dos gradientes, com valor sugerido de 0.9, β_2 é o peso atribuído à média dos quadrados dos gradientes, com valor sugerido de 0.999. Porém, os valores $\mathbf{m}_t$ e $\mathbf{v}_t$ podem estar desajustados no início do aprendizado, dificultando a busca por soluções melhores (RUDER, 2016). Isso acontece porque as primeiras iterações tendem a dar muito peso aos valores iniciais $\mathbf{m}_{t-1}$ e $\mathbf{v}_{t-1}$, que são zero, devido aos altos valores de atualização β_1 e β_2. Como resultado, $\mathbf{m}_t$ e $\mathbf{v}_t$ aumentam lentamente, o que pode atrasar o aprendizado. Para corrigir esse problema, os valores são ajustados de acordo com as seguintes equações

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \quad \text{e} \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t}, \tag{2.58}$$

onde $\hat{\mathbf{m}}_t$ representa o vetor corrigido das médias dos gradientes, $\hat{\mathbf{v}}_t$ representa o vetor corrigido das médias dos quadrados dos gradientes, $\mathbf{m}_t$ é o vetor das médias dos gradientes, $\mathbf{v}_t$ é o vetor das médias dos quadrados dos gradientes, e β_x^t é o valor de β_1 ou β_2 elevado ao número de épocas.Essa correção compensa o viés inicial, permitindo uma adaptação mais adequada dos parâmetros durante o treinamento, conforme explicado em Kingma e Ba (2014). Em relação à atualização dos parâmetros, ela é feita pela seguinte equação

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \cdot \hat{\mathbf{m}}_t, \tag{2.59}$$

onde, $\boldsymbol{\theta}$ representa o vetor de pesos, η é a taxa de aprendizagem. Já $\hat{\mathbf{m}}_t$ é o vetor corrigido das médias dos gradientes no tempo t, $\hat{\mathbf{v}}_t$ é o vetor corrigido das médias dos quadrados dos gradientes no tempo t, e ϵ é um termo adicionado para evitar a divisão por zero.

No mesmo artigo em que o otimizador Adam é proposto, os autores apresentam uma versão um pouco diferente chamada AdaMax (KINGMA; BA, 2014). Neste algoritmo, a norma ℓ_2 usada no cálculo de $\mathbf{v}_t$ (ver Equação 2.57) é substituída pela norma ℓ_∞. A seguir,

a notação para essa versão do AdaMax é apresentada,

$$\mathbf{m}_t = \beta_1 \cdot \mathbf{m}_{t-1} + (1 - \beta_1) \cdot g_t, \quad \mathbf{u}_t = \max\left(\beta_2 \cdot \mathbf{u}_{t-1}, |g_t|\right). \tag{2.60}$$

Nessa versão, a atualização dos parâmetros dá-se pela seguinte forma,

$$\theta_{t+1} = \theta_t - \eta \cdot \frac{\hat{\mathbf{m}}_\mathbf{t}}{\mathbf{u}_t}. \tag{2.61}$$

2.11 Métricas de avaliação

No Capítulo 4 as métricas de avaliação mencionadas a seguir são utilizadas.

2.11.1 Índice de Similaridade Estrutural

Em adicional ao MSE (ver Equação 2.53), a métrica Índice de Similaridade Estrutural (do inglês, *Structural Similarity Index* - SSIM) (WANG et al., 2004), também é utilizada. O SSIM é uma medida de similaridade estrutural entre duas imagens, e seu valor varia entre 0 e 1. Ele é calculado com base em três componentes principais, luminância, contraste e estrutura. A fórmula geral do SSIM é dada por

$$\text{SSIM} = \left[l(x,y)^\alpha \cdot c(x,y)^\beta \cdot s(x,y)^\gamma\right], \tag{2.62}$$

onde $\alpha > 0$, $\beta > 0$ e $\gamma > 0$ são parâmetros que ajustam a importância relativa das componentes de luminância ($l(x,y)$), contraste ($c(x,y)$) e estrutura ($s(x,y)$) das imagens. Essas componentes são definidas por,

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1}, \quad c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2}, \quad s(x,y) = \frac{2\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3}. \tag{2.63}$$

onde μ_i e σ_i são a média e o desvio padrão da intensidade dos *pixels* das imagens x e y, respectivamente. Já σ_{xy} é a covariância entre as imagens de referência e de teste x e y, respectivamente. Por fim, c_1, c_2 e c_3 são termos inseridos para evitar instabilidades numéricas.

2.11.2 *Peak Signal-to-Noise Ratio* (PSNR)

O *Peak Signal-to-Noise Ratio* - PSNR é uma métrica usada para medir a qualidade de uma imagem, comparando a imagem original com a imagem processada. A fórmula do PSNR é dada por

$$PSNR = 10 \times \log_{10}\left(\frac{MAX^2}{MSE}\right), \tag{2.64}$$

onde, MAX é o valor máximo possível de um *pixel* na imagem.

2.11.3 Erro Absoluto Médio (MAE)

O Erro Absoluto Médio (do inglês, *Mean Absolute Error* - MAE) é uma métrica que calcula o erro médio entre os valores reais e os valores previstos de um modelo. Ele é dado pela fórmula

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{2.65}$$

onde, n é o número total de amostras ou elementos no conjunto de dados. Já y_i é o valor real ou observado da i-ésima amostra. Por sua vez $\hat{y}_i$ é o valor previsto ou estimado pela modelagem para a i-ésima amostra. Por fim $|y_i - \hat{y}_i|$ é o valor absoluto da diferença entre o valor real e o valor previsto para a amostra i.

O MAE representa a média das diferenças absolutas entre os valores reais e os valores previstos, fornecendo uma medida direta de quão longe, em média, estão as previsões do modelo em relação aos valores reais.

2.11.4 Percentual da diferença entre *pixels*

O percentual da diferença entre *pixels* calcula a proporção de *pixels* que apresentam valores diferentes entre duas imagens, neste caso a imagem desejada f_y e a imagem predita f_{out}. Para isso, para cada linha da imagem, comparam-se os valores de cada *pixel* com os valores da imagem de referência. Em seguida, calcula-se a média das diferenças ao longo de todas as linhas e colunas da imagem. O resultado final é expresso em percentual. Esse percentual pode ser calculado formalmente pela seguinte maneira

$$PPD(f_y, f_{\text{out}}) = 100 \times \frac{1}{MN}\sum_{p=1}^{M}\sum_{q=1}^{N}[f_y(p,q) \neq f_{\text{out}}(p,q)], \tag{2.66}$$

onde M é o número de linhas da imagem e N é o número de colunas da imagem.

3 MATERIAIS E MÉTODOS

Neste Capítulo, é descrito o processo de aprendizagem dos filtros conexos (critério de seleção de nós) utilizando redes neurais. Antes dos detalhes do processo serem apresentados, no início deste Capítulo, são apresentadas algumas propriedades de funções matemáticas para melhor compreensão posterior.

3.1 Continuidade e Diferenciabilidade de funções

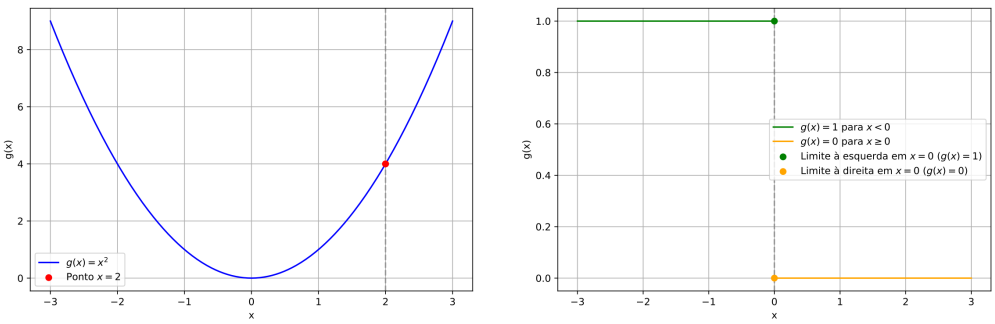
Na matemática, uma função g é contínua se for contínua em todos os pontos de seu domínio. Em termos geométricos, a continuidade implica que o gráfico da função não apresenta saltos ou interrupções (HUGHES-HALLETT; GLEASON; MCCALLUM, 2020). Ou seja, ao aproximar um ponto de ambos os lados, os valores da função convergem para o mesmo valor.

Seguindo a definição de Hughes-Hallett, Gleason e McCallum (2020), uma função $g(x)$ é contínua em um ponto a se o limite dos valores de $g(x)$ ao se aproximar de a pela esquerda ($x \to a^-$) e pela direita ($x \to a^+$) for igual ao valor da função no ponto a. Formalmente, $g(x)$ é contínua em a se

$$\lim_{x \to a} g(x) = g(a). \tag{3.1}$$

Para ilustrar este conceito, consideramos a função contínua $g(x) = x^2$. Na Figura 3.1(a), o gráfico demonstra que, ao aproximar o ponto $x = 2$ por ambos os lados, os valores de $g(x)$ convergem para o mesmo valor, ou seja, $g(2) = 4$. Isso demonstra a ausência de saltos ou interrupções.

Figura 3.1 – *Comparação entre funções contínua e descontínua.*



(a) Função Contínua

(b) Função Descontínua

Fonte: Elaborado pelo autor (2024).

Já a continuidade segundo Hughes-Hallett, Gleason e McCallum (2020), no entanto, é

apenas uma condição necessária, mas não suficiente, para a diferenciabilidade. Sendo uma função g é diferenciável se sua derivada existir em todos os pontos de seu domínio. Isso significa que a função deve possuir uma tangente bem definida em cada ponto. Funções não contínuas, como a função booleana, falham nesse critério, pois os saltos (Figura 3.1(b)) criam pontos onde a derivada não existe.

3.1.1 Funções Booleanas e Diferenciabilidade

As funções booleanas, por sua natureza, assumem apenas dois valores discretos, como 0 ou 1 . Esses valores não variam de maneira gradual com x, mas mudam de forma abrupta em pontos específicos do domínio. Por exemplo, considere uma função booleana $g(x)$ definida como

$$g(x) = \begin{cases} 1 & \text{se } x > a \\ 0 & \text{se } x \leq a \end{cases}. \tag{3.2}$$

Neste caso, há uma descontinuidade em $x = a$. Essa descontinuidade implica que não é possível calcular a derivada de $g(x)$ em $x = a$, e a derivada é zero ou inexistente nos outros pontos, tornando a função não diferenciável em todo o domínio. Ao observar a Figura 3.1(b), ao aproximar-se de $x = 0$ pela esquerda ($x \to 0^-$), os valores da função permanecem $g(x) = 1$. Ao aproximar-se pela direita ($x \to 0^+$), os valores mudam para $g(x) = 0$. Ou seja, os valores não convergem, evidenciando a descontinuidade. Essa descontinuidade torna a função não diferenciável em $x = 0$, pois a derivada não pode ser definida onde há uma ruptura.

3.2 Aprendizagem de Filtros Conexos para Filtragem de Árvores Morfológicas Baseada em Redes Neurais

Como discutido na Subseção 2.7.3 e Subseção 2.7.4, as regras de filtragem de árvores morfológicas podem ser vistas como uma aplicação de uma função booleana a um atributo específico dos nós da árvore. Essa função, representada pela função característica $\mathbf{s}$ previamente definida, determina quais nós devem ser mantidos na árvore e quais devem ser removidos. A função booleana avalia se o critério específico é satisfeito. Com base nessa avaliação, o nó será removido ou preservado, dependendo da regra aplicada (ver Subseção 2.7.4), do atributo selecionado e do limiar definido.

Neste contexto, a função booleana usada na filtragem da árvore não diferenciável. Isso ocorre devido à descontinuidade da função booleana, o que impede a existência de uma derivada em todos os pontos do seu domínio. A descontinuidade da função impede o cálculo da derivada em diversos pontos, inviabilizando o uso de algoritmos de otimização baseados em gradiente, como os utilizados em redes neurais, pois tais algoritmos dependem

da capacidade de calcular derivadas para realizar o aprendizado. (PERRET; COUSTY, 2022).

Com o intuito de superar a limitação associada à função booleana explorada anteriormente, este trabalho propõe uma expansão do critério de filtragem clássico da árvore para a aprendizagem dos filtros conexos (critério de seleção de nós) utilizando redes neurais. Com o uso da regra de filtragem Subtrativa, a função booleana é substituída por uma função contínua, diferenciável e parametrizada. Essa abordagem visa ampliar o processo de filtragem, garantindo que a operação possa ser otimizada através do gradiente descendente. Assim, a função proposta, denotada por $\mathbf{s}_{\mathbf{w},\mathbf{b}}$, é projetada para mapear os nós da árvore filtrada $\mathcal{T}_f$ em um intervalo real $[0, 1]$. A definição formal da função é apresentada como

$$\mathbf{s}_{\mathbf{w},\mathbf{b}} : \mathcal{T}_f \to [0, 1], \tag{3.3}$$

onde $\mathbf{w}$ representa os pesos e $\mathbf{b}$ representa o viés da camada da rede neural. Sendo assim, a função é definida como uma combinação linear entre os pesos $\mathbf{w}$ e o vetor de atributos dos nós, conforme formalizado a seguir, como sendo $\forall \tau \in \mathcal{T}_f$

$$\mathbf{s}_{\mathbf{w},\mathbf{b}}(\tau) = \sigma(\mathbf{x}_{i\tau} \times \mathbf{w} + \mathbf{b}), \tag{3.4}$$

onde σ é a função de ativação Sigmoide, conforme discutido na Subseção 2.8.3. O termo $\mathbf{x}_{i\tau}$ representa os atributos de um nó específico τ pertencente à árvore $\mathcal{T}_f$, atributos esses conforme discutidos na Seção 2.4. De forma mais clara, considerando uma matriz de atributos $\mathbf{X}$ associada à árvore $\mathcal{T}_f$, $\mathbf{x}_{i\tau}$ é um vetor que contém todos os atributos do nó τ na i-ésima linha da matriz $\mathbf{X}$. Para ilustrar, $\mathbf{x}_{i\tau}$ pode ser representado como

$$\mathbf{x}_{i\tau} = \begin{pmatrix} \mathbf{x}_{i0} & \mathbf{x}_{i1} & \mathbf{x}_{i2} & \cdots & \mathbf{x}_{in} \end{pmatrix}, \tag{3.5}$$

onde $\mathbf{x}_{i0}, \mathbf{x}_{i1}, \mathbf{x}_{i2}, \ldots, \mathbf{x}_{in}$ são os atributos específicos do nó τ, armazenados em diferentes colunas da matriz $\mathbf{X}$. Por sua vez, $\mathbf{w}$ representa o vetor de pesos, enquanto $\mathbf{b}$ denota um valor escalar (viés). Assim, a função paramétrica $\mathbf{s}_{\mathbf{w},\mathbf{b}}$ demonstra uma versatilidade em relação à função característica $\mathbf{s}$ mencionada anteriormente. Enquanto $\mathbf{s}$ estabelece um critério para manter um nó τ pertencente a $\mathcal{T}_f$ baseado na comparação direta entre um atributo κ e um limiar i (representado pela condição $\kappa_\tau \geq i$), $\mathbf{s}_{\mathbf{w},\mathbf{b}}$ utiliza parâmetros ajustáveis $\mathbf{w}$ e $\mathbf{b}$ com o uso da rede neural para mapear nós para um intervalo contínuo de valores, oferecendo uma abordagem mais flexível.

Em contraste com a reconstrução da imagem após a filtragem da árvore apresentada na Subseção 2.7.5, onde a seleção dos nós a serem removidos ou mantidos é feita com base na função $\mathbf{s}$, a reconstrução da imagem após a filtragem utilizada a função $\mathbf{s}_{\mathbf{w},\mathbf{b}}$, é

realizada da seguinte maneira $\forall p \in \mathcal{D}$

$$[\mathrm{Rec}(\mathcal{T}_f, \mathbf{s}_{\mathbf{w},\mathbf{b}})](p) = \sum_{\tau \in \mathcal{T}_f : p \in \tau} |level(\tau) - level(\mathrm{Parent}(\tau))| \times \mathbf{s}_{\mathbf{w},\mathbf{b}}(\tau). \tag{3.6}$$

O resultado dessa reconstrução será denotado por $f_{\text{ out}}$, representando a imagem obtida após a aplicação do processo de reconstrução.

3.3 Retropropagação e derivadas

Conforme é possível observar na Subseção 2.8.5, o algoritmo de retropropagação é crucial para otimizar os parâmetros de uma função através da propagação das derivadas do erro ao longo da arquitetura da rede neural. Sendo assim, em continuidade ao problema desta pesquisa, podemos assumir uma camada da rede neural sendo o f_{out} definido na Subseção anterior.

Com o objetivo de encontrar os melhores parâmetros durante o treinamento da rede neural, o algoritmo de retropropagação calcula a derivada da função de custo em relação à saída da camada ($\partial E / \partial f_{\text{ out}}$), sendo então calculada como

$$\frac{\partial E}{\partial f_{\text{ out}}} = \frac{1}{|\mathcal{D}|} \times 2 \sum_{p} (f_{\text{ out}}(p) - f_y(p)), \tag{3.7}$$

onde $f_y(p)$ um dado *pixel* da imagem desejada e $f_{\text{ out}}(p)$, um dado *pixel* da imagem reconstruída após a filtragem diferenciável. Essa derivada representa como a função de custo varia em relação à saída da camada.

Computado a derivada $\partial E / \partial f_{\text{out}}$, Equação 3.7, outras duas derivadas são calculadas adicionalmente, a derivada da função custo em relação aos parâmetros θ ($\partial E / \partial \theta$) e a derivada de função de custo em relação à entrada ($\partial E / \partial f$). Para calcular a derivada da função custo em relação aos parâmetros θ, aplica-se a regra da cadeia conforme explorado na Subseção 2.8.5, onde

$$\frac{\partial E}{\partial \theta} = \frac{\partial E}{\partial f_{\text{ out}}} \frac{\partial f_{\text{ out}}}{\partial \theta}, \tag{3.8}$$

e,

$$\frac{\partial f_{\text{ out}}}{\partial \theta} = [\frac{\partial f_{\text{ out}}}{\partial \mathbf{w}}, \frac{\partial f_{\text{ out}}}{\partial \mathbf{b}}]^T, \tag{3.9}$$

$$\begin{aligned} \frac{\partial f_{\text{ out}}}{\partial \mathbf{w}}(p) &= \sum_{\tau \in \mathcal{T}_f : p \in \tau} \frac{\partial s_{\mathbf{w},\mathbf{b}(\tau)}}{\partial \mathbf{w}} \times \mathbf{x}_\tau \times |level(\tau) - level(\mathrm{Parent}(\tau))|, \\ \frac{\partial f_{\text{ out}}}{\partial \mathbf{b}}(p) &= \sum_{\tau \in \mathcal{T}_f : p \in \tau} \frac{\partial s_{\mathbf{w},\mathbf{b}(\tau)}}{\partial \mathbf{b}} \times \mathbf{x}_\tau \times |level(\tau) - level(\mathrm{Parent}(\tau))|. \end{aligned} \tag{3.10}$$

A derivada da função de custo em relação aos parâmetros θ representa como a perda

varia com relação aos parâmetros da camada. Por sua vez, a atualização dos parâmetros, assim como na Equação 2.54, é definida como

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \frac{\partial E(\mathbf{w}_{t-1})}{\partial \mathbf{w}}, \tag{3.11}$$

Já o cálculo da derivada da função de custo em relação à entrada é mais difícil e requer alguma aproximação. Supondo que todos os *pixels* em um componente conexo (CC) variem seus valores de escala de cinza da mesma maneira, os próprios CCs não variam com as variações da entrada em si (PERRET; COUSTY, 2022). O cálculo em questão pode ser realizado como

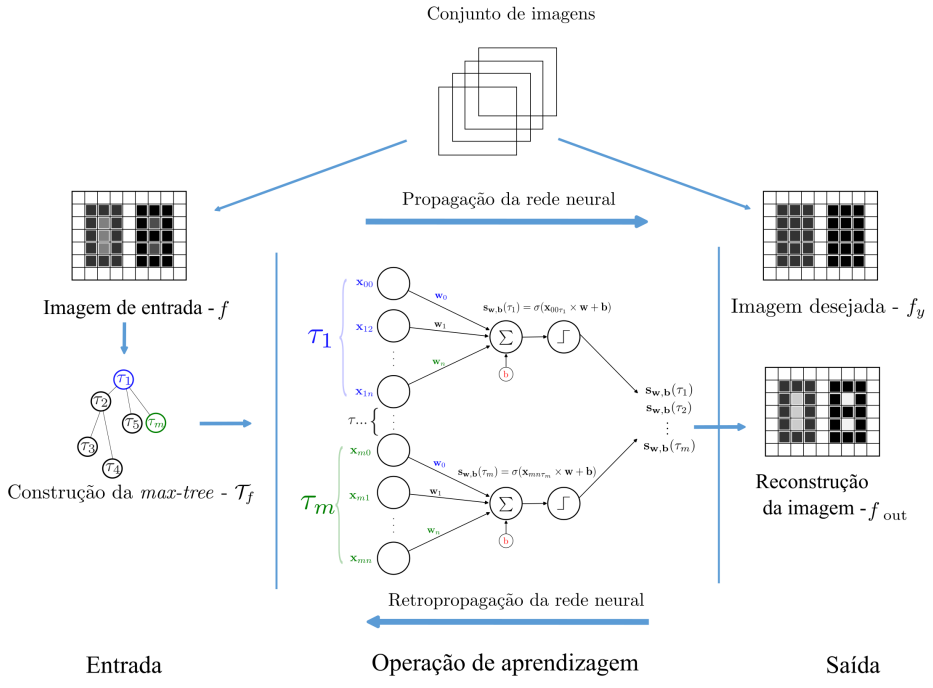
$$\frac{\partial E}{\partial f}(p) = \sum_{\tau \in \mathcal{T} : p \in \tau} \left(\sum_{q \in \tau} \frac{\partial E}{\partial f_{\text{ out}}}(q) \times \mathbf{s}_{\mathbf{w},\mathbf{b}}(\tau) \right). \tag{3.12}$$

Essa derivada determina como a função de custo varia em relação à entrada da camada e em relação à imagem f.

3.4 Representação visual

De acordo com as definições na Subseção 3.2, Subseção 3.3, a Figura 3.2 ilustra, a seguir, o processo de aprendizagem dos filtros conexos pela rede neural.

Figura 3.2 – *Representação visual do Problema de Pesquisa.*



Fonte: Elaborado pelo autor (2024).

Com o objetivo de simplificar a explicação do problema de pesquisa, a Figura 3.2 utiliza uma imagem de entrada f e uma imagem desejada f_y, em vez de conjuntos de imagens. Assim, a partir de uma imagem de entrada f, a construção da *max-tree* $\mathcal{T}_f$ é realizada. Em seguida, a computação da matriz atributos $\mathbf{X}$ é efetuada conforme discutido na Subseção 2.4.1. Com a matriz de atributos $\mathbf{X}$, o vetor de pesos $\mathbf{w}$ e o valor escalar do viés $\mathbf{b}$, esses são estabelecidos antes de serem utilizados na entrada da rede neural, a fase seguinte envolve o instanciamento da rede neural. Através da arquitetura de uma única camada da rede neural, que pode ser interpretada como uma rede Perceptron Simples (ver a Subseção 2.8.1) empilhada, o cálculo da função $\mathbf{s}_{\mathbf{w},\mathbf{b}}$ é realizado para cada nó pertencente à árvore através da Equação 3.4. Em seguida, cada saída da função $\mathbf{s}_{\mathbf{w},\mathbf{b}}$ é utilizada na reconstrução da imagem f_{out}. Obtida a imagem de saída f_{out}, através da Equação 3.6, o próximo passo é realizar o cálculo da função de custo por meio da Equação 2.53. A seguir, a função de custo é apresentada com notação atualizada para

$$E = \frac{1}{|\mathcal{D}|} \times 2 \sum_{p} (f_{\text{out}}(p) - f_y(p)), \tag{3.13}$$

onde f_y é a imagem desejada. Dessa maneira, o valor da função de custo será utilizado na fase de retropropagação mencionada a seguir. Durante a fase de retropropagação, são realizados diversos cálculos de derivadas, conforme explorado na Seção 3.3, incluindo a derivada da função de custo em relação à saída da camada ($\frac{\partial E}{\partial f_{\text{out}}}$) através da Equação 3.7, a derivada da função de custo em relação aos parâmetros θ ($\frac{\partial E}{\partial \theta}$) através da Equação 3.10 e a derivada da função de custo em relação à entrada ($\frac{\partial E}{\partial f}$) através da Equação 3.12. Em seguida, os parâmetros da rede neural são atualizados de acordo com a Equação 3.11. Dessa forma, o processo é reiniciado, utilizando a construção da nova *max-tree* a partir da imagem f_{out}, e se repete até o número de iterações configurado para o treinamento dessa rede neural. E assim, o objetivo é que a imagem f_{out} se aproxime ao máximo da imagem desejada f_y, como demonstrado na Figura 3.2 acima..

4 EXPERIMENTOS

Este Capítulo descreve os experimentos realizados para investigar a aprendizagem dos filtros conexos. Para isso, foi desenvolvida uma implementação em C++/Python, utilizando a biblioteca ctreelearn[1] para construir a árvore, calcular atributos e derivadas parciais. A aprendizagem dos filtros conexos proposta foi implementada com a biblioteca PyTorch (PASZKE et al., 2017), baseada em Python.

4.1 Experimento filtragem pelo atributo área e inércia

O objetivo deste experimento[2] foi realizar a aprendizagem dos filtros conexos de um conjunto de imagens através da rede neural. O modelo foi treinado para gerar imagens preditas a partir de imagens desejadas (imagens filtradas a partir de um atributo em conjunto com uma árvore para compor o conjunto de imagens). O experimento foi conduzido utilizando o conjunto de dados ICDAR 2024 *Occluded RoadText Competition* (ROADTEXT, 2024), criado especificamente para desafios de detecção e reconhecimento de texto em ambientes urbanos.

Para este experimento, foram selecionados dois atributos para a geração dos conjuntos de imagens, o atributo área (atributo crescente) e o atributo inércia (atributo não crescente). Cada um desses atributos foi aplicado de maneira distinta para a construção do conjunto de imagens utilizando a *max-tree* e a *min-tree*. Assim, foram obtidos quatro conjuntos de imagens para o treinamento. O primeiro conjunto de imagens utilizou a *max-tree* com filtragem baseada no atributo área, o segundo conjunto de imagens também usou a *max-tree*, mas com filtragem pelo atributo inércia. O terceiro conjunto de imagens foi gerado com a *min-tree* e o atributo área, e, por fim, o quarto conjunto de imagens foi obtido com o uso da *min-tree* e a filtragem pelo atributo inércia.

A construção de cada conjunto de imagens seguiu o seguinte procedimento, para o primeiro conjunto de imagens, a partir das imagens do conjunto de dados ICDAR, a árvore *max-tree* foi construída e filtrada para cada imagem, utilizando o atributo área, considerando apenas os valores dos nós superiores à área de 4.500. Para o segundo conjunto de imagens, com base no atributo inércia, a *max-tree* foi filtrada considerando apenas os valores de inércia dos nós superiores a 0.16. Vale lembrar que essa filtragem foi realizada com a utilização da função booleana discutida anteriormente (ver Subseção 2.7.3 e Subseção 2.7.4). Após a filtragem dessas árvores com os respectivos filtros, as imagens filtradas foram geradas. Como resultado, para cada conjunto de imagens (primeiro e segundo), obteve-se vários pares de imagens, onde cada par consiste na imagem de entrada e sua respectiva versão filtrada (imagem desejada) de acordo com o atributo selecionado. Dessa

[1] Disponível em https://github.com/wonderalexandre/ComponentTreeLearn.

[2] <https://github.com/Anderson-hrz/APRENDIZAGEM-DE-FILTROS-CONEXOS-POR-REDES-NEURAIS-USANDO-ARVORES-DE-COMPONENTES>.

forma, como mencionado, o objetivo é que o modelo aprenda os critérios de filtragem aplicados ao conjunto de imagens com base na imagem de entrada e na imagem desejada, gerando a reconstrução da imagem de saída correspondente a cada par. Além do procedimento realizado com a *max-tree*, o mesmo processo foi aplicado utilizando a *min-tree*. Para o terceiro conjunto de imagens, as imagens do conjunto de dados ICDAR foram filtradas através da *min-tree* com base no atributo área, considerando valores dos nós com área maior que 4.500. Para o quarto conjunto de imagens, com o uso do atributo inércia, a filtragem na *min-tree* foi realizada com valores de inércia dos nós maiores que 0.16. Após a filtragem dessas árvores com os respectivos filtros, as imagens filtradas foram geradas. Em ambos os casos, obteve-se, para cada conjunto de imagens (terceiro e quarto), um par composto pela imagem de entrada e pela imagem filtrada correspondente. Em relação à escolha dos limiares 4.500 e 0,16, ela decorreu da seguinte forma: após a construção de cada árvore para as imagens com a *max-tree*, calculou-se a média do atributo área de todos os nós de cada árvore. Em seguida, foi calculada a média geral, que corresponde à média das médias obtidas para cada árvore, resultando em um valor de 2.676. Com o objetivo de dificultar a aprendizagem dos filtros conexos e a inspeção visual das imagens desejadas geradas, escolheu-se o valor 4.500 para a filtragem das imagens desejadas. Em relação ao valor de 0.16, aplicou-se o mesmo princípio utilizado no cálculo das médias. No entanto, para esse caso específico, foi utilizado o valor obtido da média geral. Isso porque, ao aumentar esse valor para 0.16, as imagens filtradas (imagens desejadas) se tornavam completamente brancas ou pretas, dependendo da árvore utilizada. A seguir, a Figura 4.1 ilustra as imagens de entrada e imagens desejadas (filtradas).

Figura 4.1 – *Na primeira linha, da esquerda para a direita, estão as imagens filtradas com a max-tree. Na segunda linha, estão as imagens filtradas com a min-tree.*

(a) *Imagem original*

(b) *Max-tree, imagem reconstruída, atributo área.*

(c) *Max-tree, imagem reconstruída, atributo inércia.*

(d) *Imagem original*

(e) *Min-tree, imagem reconstruída, atributo área.*

(f) *Min-tree, imagem reconstruída, atributo inércia*

Fonte: Elaborado pelo autor (2024).

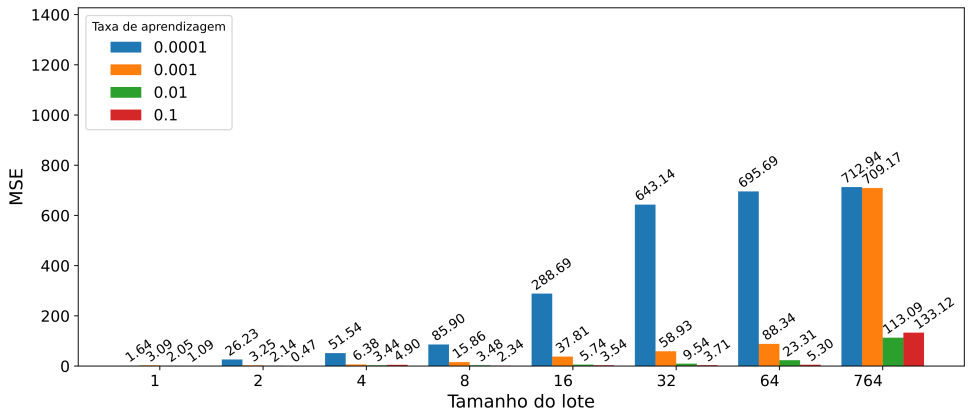
Em relação ao conjunto de imagens, para todas as configurações descritas anteriormente que envolvem o uso das árvores *max-tree* e *min-tree*, com os atributos área e inércia, foi mantida a mesma configuração do conjunto de dados. Assim, para cada conjunto, as imagens foram convertidas para níveis de cinza e compostas por imagens com dimensões de 270 × 480 *pixels*, correspondendo a 25% do tamanho original (1080 × 1920 *pixels*). O total de imagens disponíveis foi de 1.019, das quais o conjunto de imagens foi dividido em 25% (255 imagens) para o conjunto de teste, enquanto os 75% (764 imagens) restantes foram selecionados para o conjunto de treinamento.

Um processo de treinamento para cada conjunto de imagens (dos quatro conjuntos), consistindo em uma série de simulações experimentais iniciais, foi realizado para avaliar os parâmetros da rede (taxa de aprendizagem e tamanho do lote), a fim de definir uma configuração base para o aprendizado dos filtros conexos. A função de perda utilizada foi o MSE, com o otimizador AdaMax, e foi selecionado o número de 50 épocas para o treinamento. Simulações relacionadas aos tamanhos de lotes e taxas de aprendizagem foram realizadas. Os tamanhos dos lotes explorados foram 1, 2, 4, 8, 16, 32, 64 e 764 (tamanho do conjunto de imagens de treinamento). Já as taxas de aprendizagem testadas foram 0.1, 0.01, 0.001 e 0.0001. Por fim, destaca-se que, para o treinamento de todos os conjuntos de imagens e suas respectivas configurações de árvores e atributos, os modelos foram treinados utilizando todos os atributos discutidos anteriormente na Seção 2.4.

4.1.1 Análise de resultados e discussões

Na Figura 4.2, o gráfico mostra a relação entre o tamanho do lote e o MSE para diferentes taxas de aprendizagem, utilizando o primeiro conjunto de imagens de teste. Os resultados são obtidos a partir do conjunto de imagens de teste dos treinamentos.

Figura 4.2 – *MSE em função do tamanho do lote e taxa de aprendizado para a aprendizagem dos filtros conexos com a max tree e as imagens desejadas filtradas com o atrbituto área para o conjunto de imagens de teste.*



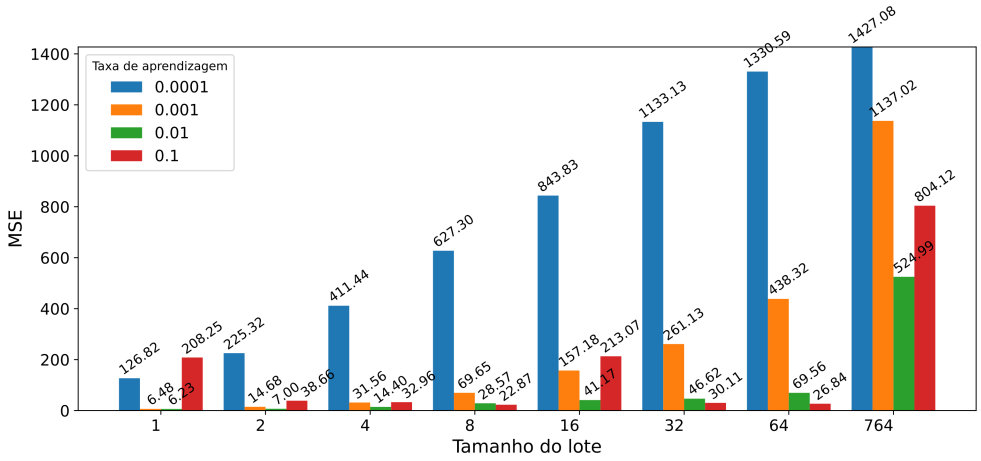
Fonte: Elaborado pelo autor (2024).

Ao analisar o gráfico apresentado na Figura 4.2, que compara modelos treinados com o primeiro conjunto de imagens utilizando a árvore *max-tree*, observam-se algumas tendências consistentes em relação ao tamanho do lote e à taxa de aprendizagem no desempenho do modelo. Em termos gerais, o MSE tende a diminuir com a redução do tamanho do lote, especialmente para tamanhos menores, como 1 e 2, onde o modelo apresenta os melhores resultados de MSE, indicando uma aprendizagem mais precisa.

Em relação à taxa de aprendizagem, os resultados mostram que taxas baixas, como 0.0001 e 0.001, não são tão eficazes em termos de MSE. A análise do gráfico na Figura 4.2 indica que, para taxas de aprendizado baixas (0.0001 e 0.001), o modelo apresenta MSE muito altos, sugerindo que a aprendizagem é lenta e pode não convergir adequadamente. Já para taxas de aprendizado como 0.01 e 0.1, os resultados são mais consistentes, com o modelo apresentando um desempenho notavelmente melhor.

A seguir, na Figura 4.3, o gráfico que mostra a relação entre o tamanho do lote e o MSE para diferentes taxas de aprendizado é apresentado, com base no segundo conjunto de imagens descrito anteriormente. Os resultados apresentados também correspondem ao conjunto de imagens de teste utilizado durante os treinamentos.

Figura 4.3 – *MSE em função do tamanho do lote e taxa de aprendizado para a aprendizagem dos filtros conexos com a max tree e as imagens desejadas filtradas com o atrbituto inércia para o conjunto de imagens de teste.*



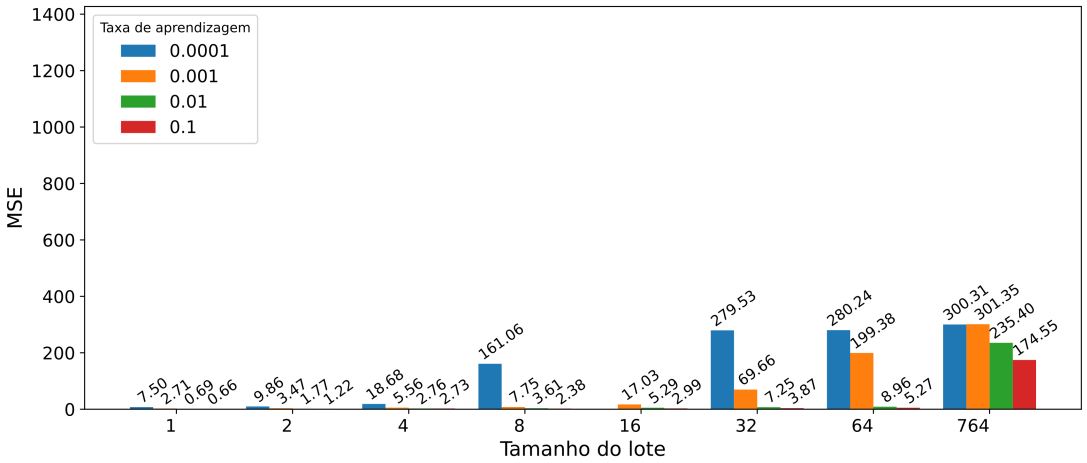
Fonte: Elaborado pelo autor (2024).

No gráfico da Figura 4.3, para o segundo conjunto de imagens, observa-se uma tendência semelhante no impacto do tamanho do lote e da taxa de aprendizado, quando comparado com o gráfico da Figura 4.2, mas com um desempenho geral inferior em relação ao primeiro conjunto de imagens. Taxas de aprendizagem baixas, como 0.0001, e altas, como 0.1 resultaram em MSE significativamente maiores, sugerindo que esses valores de taxa de aprendizado não são ideais. Embora as taxas intermediárias também apresentem um desempenho relevante em comparação com o primeiro conjunto de imagens (Figura

4.2), a taxa de aprendizado de 0.01 se destaca neste conjunto de imagens, com MSE relativamente mais baixos, especialmente para os tamanhos de lote 2 e 4.

Por sua vez, a Figura 4.4 também demonstra a relação entre o tamanho do lote e o MSE para diferentes taxas de aprendizado, para o terceiro conjunto de imagens, onde os valores apresentados correspondem ao conjunto de imagens de teste utilizado nos treinamentos.

Figura 4.4 – *MSE em função do tamanho do lote e taxa de aprendizado para a aprendizagem dos filtros conexos com a min tree e as imagens desejadas filtradas com o atrbituto área para o conjunto de imagens de teste.*



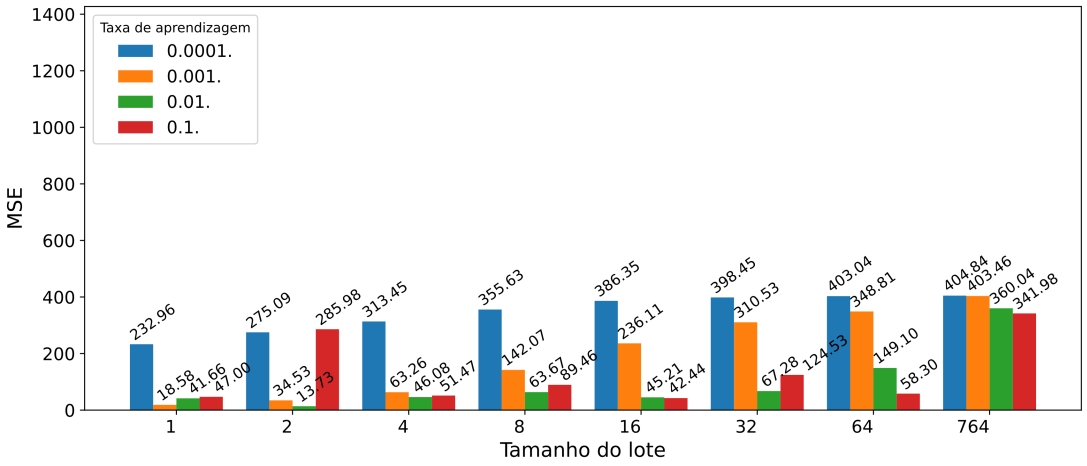
Fonte: Elaborado pelo autor (2024).

Ao analisar o gráfico apresentado na Figura 4.4 que apresenta os modelos treinados com o terceiro conjunto de imagens utilizando a árvore *min-tree*, observa-se tendências no desempenho, especialmente em relação ao tamanho do lote e à taxa de aprendizado. De maneira geral, os resultados indicam que a redução do tamanho do lote tende a reduzir o erro MSE, confirmando a observação de que lotes menores contribuem para uma aprendizagem mais precisa. Isso é visível nos valores de MSE para tamanhos de lote como 1 e 2, que apresentam os melhores resultados em comparação aos lotes maiores. Além disso, a taxa de aprendizado tem um impacto significativo no MSE. No caso do atributo terceiro conjunto de imagens, o modelo parece ser mais sensível a valores extremos de taxa de aprendizado, com valores de MSE significativamente mais elevados para taxas muito baixas 0.0001 ou altas 0.1.

Por outro lado, para o quarto conjunto de imagens, o modelo mantém uma performance relativamente consistente em taxas de aprendizado intermediárias, como 0.01 e 0.1, mas mostra um aumento no MSE em taxas muito baixas, como 0.0001. A comparação entre o terceiro conjunto de imagens, mostrado na Figura 4.4, e o quarto conjunto de imagens, apresentado no gráfico da Figura 4.5 abaixo, revela que, embora as tendências gerais de MSE em função do tamanho do lote e da taxa de aprendizado sejam seme-

lhantes, o modelo treinado com o quarto conjunto de imagens apresenta valores de MSE consistentemente mais altos do que o modelo com o terceiro conjunto de imagens. Isso se deve, obviamente, à diferença de filtragem entre os dois atributos, área e inércia, utilizados na construção do conjunto de imagens e, mais precisamente, na filtragem das imagens desejadas.

Figura 4.5 – *MSE em função do tamanho do lote e taxa de aprendizado para a aprendizagem dos filtros conexos com a min tree e as imagens desejadas filtradas com o atrbituto inércia para o conjunto de imagens de teste.*



Fonte: Elaborado pelo autor (2024).

A seguir, com base nos melhores resultados apresentados na Figura 4.2, Figura 4.3, Figura 4.4 e Figura 4.5, a Tabela 4.1 apresenta a análise dos melhores resultados da aprendizagem dos filtros conexos para cada um dos quatro conjuntos de imagens.

Árvore	Atributo filtrado	Tamanho do lote	Taxa de aprendizagem	MSE	MAE	PPD	PSNR
max-tree	Área	2	0.10	0,47	0,05	1,39	51,44
	Inércia	1	0.01	6,23	0,49	14,36	40,19
min-tree	Área	1	0.1	0,66	0,10	4,96	49,96
	Inércia	2	0.01	7,02	0,61	12,71	39,56

Tabela 4.1 – *Resultados do treinamento dos modelos mais eficazes utilizando todos os atributos disponíveis.*

Na Tabela 4.1, os resultados indicam que o modelo treinado com a *max-tree* e o atributo área (primeiro conjunto de imagens) usado para gerar as imagens desejadas, utilizando um tamanho de lote de 2 e taxa de aprendizagem de 0,10, obteve um bom desempenho em todas as métricas. O MSE 0,47 e o MAE 0,05 são baixos, sugerindo um erro mínimo na reconstrução das imagens. O PSNR 51,44 destaca a qualidade da imagem

reconstruída, com a similaridade entre a imagem desejada e a imagem predita. O PPD de 1,39 é baixo, o que indica uma boa fidelidade visual na reconstrução da imagem predita. No entanto, ao comparar com o modelo utilizando a *max-tree* e o atributo inércia (segundo conjunto de imagens) usado para gerar as imagens desejadas, observamos uma leve queda no desempenho. O MSE 6,23 e o MAE 0,49 indicam erros mais evidentes na reconstrução da imagem predita, e o PSNR 40,19. A diferença entre as imagens desejadas e as preditas é evidenciada pelo PPD de 14,36.

Ao analisar o modelo treinado com a *min-tree* e o atributo área (terceiro conjunto de imagens) usado para gerar as imagens desejadas, os resultados são semelhantes aos obtidos pela *max-tree* com o mesmo atributo (primeiro conjunto de imagens). O MSE 0,66 e o MAE 0,10 são baixos, e o PSNR 49,96 continua apresentando boa qualidade na reconstrução das imagens preditas. No entanto, o PPD de 4,96 é um pouco maior que o da *max-tree* com o mesmo atributo (primeiro conjunto de imagens), indicando que, embora a reconstrução seja ainda boa, há uma leve diferença entre as imagens desejadas e as imagens preditas utilizando essas árvores.

Por fim, o modelo treinado com a *min-tree* e o atributo inércia (quarto conjunto de imagens) usado para gerar as imagens desejadas teve um desempenho inferior ao dos outros modelos. O MSE de 7,02 e o MAE de 0,61 indicam que há um leve erro na reconstrução das imagens preditas, e o PSNR de 39,56 confirma que a qualidade da imagem reconstruída não foi tão boa quanto nos outros casos. O PPD de 12,71 também sugere uma diferença leve, perceptível e significativa, indicando que a combinação da *min-tree* com o atributo inércia não oferece os melhores resultados, especialmente quando comparado à *max-tree* e ao atributo área.

Embora os resultados apresentados na Tabela 4.1 tenham sido considerados satisfatórios, também foi realizado o treinamento do modelo com o objetivo de aprender os filtros conexos utilizando apenas um único atributo no treinamento. Para isso, foram utilizados os mesmos quatro conjuntos de imagens criados anteriormente. Mas, neste caso, optou-se por utilizar exclusivamente o atributo correspondente. Em contraste com a abordagem anterior, que considerou todos os atributos disponíveis, neste treinamento, cada conjunto de imagens foi associado a um único atributo. Para o primeiro conjunto, utilizou-se o atributo área, para o segundo, o atributo inércia, e assim por diante. Em relação ao treinamento, as mesmas configurações de hiperparâmetros utilizadas anteriormente foram mantidas (tamanho do lote, taxa de aprendizagem e número de épocas). A seguir, a Tabela 4.2 apresenta os resultados obtidos para o treinamento somente com seu respectivo atributo utilizado nos conjuntos de imagens para filtragem.

Árvore	Atributo filtrado	Tamanho do lote	Taxa de aprendizagem	MSE	MAE	PPD	PSNR
max-tree	Área	2	0.10	0,19	0,04	3,12	55,25
	Inércia	1	0.01	0,30	0,05	3,67	53,23
min-tree	Área	1	0.10	0,02	0,02	1,94	64,82
	Inércia	2	0.01	0,69	0,21	9,40	49,72

Tabela 4.2 – *Resultados do treinamento dos modelos com as melhores configurações (MSE e tamanho de lote) utilizando apenas um atributo disponível.*

Na Tabela 4.2, os modelos foram treinados utilizando apenas um único atributo para cada conjunto de imagens, conforme descrito anteriormente. Em comparação com os resultados da Tabela 4.1, observa-se que os modelos com atributos filtrados individualmente apresentam um resultado levemente melhor.

Para o modelo com a *max-tree* e o atributo área (primeiro conjunto de imagens), usado para gerar as imagens desejadas e no treinamento do modelo, os resultados mostram uma melhoria em relação à Tabela 4.1, com o MSE de 0,19 e o MAE de 0,04, valores mais baixos do que os anteriores, com MSE de 0,47 e MAE de 0,05. Essas métricas indicam que o erro na reconstrução das imagens geradas foi reduzido. O PSNR de 55,25 sugere que as imagens reconstruídas se igualam às imagens desejadas, refletindo uma precisão na reconstrução da imagem predita. No entanto, o PPD de 3,12 ainda indica uma leve diferença entre as imagens desejadas e as preditas.

O modelo com a *max-tree* e o atributo inércia (segundo conjunto de imagens), usados para gerar as imagens desejadas e no treinamento do modelo, obteve um desempenho consideravelmente melhor em comparação com os resultados da Tabela 4.1. O MSE de 0,30 e o MAE de 0,05 são mais baixos em comparação com os valores de MSE de 6,32 e MAE de 0,49 da tabela anterior, indicando um erro reduzido na reconstrução das imagens preditas. O PSNR de 53,23 confirma que as imagens reconstruídas estão muito próximas das imagens desejadas, refletindo boa precisão na reconstrução. O PPD de 3,67 também é consideravelmente menor em comparação com o valor da Tabela 4.1, que é 14,36, sugerindo uma diferença mais sutil entre as imagens desejadas e as preditas. Esses resultados indicam que o uso exclusivo do atributo inércia, neste caso, não prejudicou o desempenho do modelo, ao contrário, obteve bons resultados.

No caso do modelo com a *min-tree* e o atributo área (terceiro conjunto de imagens), usado para gerar as imagens desejadas e no treinamento do modelo, o desempenho foi positivo. O MSE de 0,02 e o MAE de 0,02 indicam um erro baixo na predição das imagens, comparados com o MSE de 0,66 e MAE de 0,10 da tabela anterior, refletindo precisão. O PSNR de 64,82 é elevado, sugerindo que a qualidade das imagens reconstruídas é quase idêntica às das imagens desejadas. Além disso, o PPD de 1,94 é baixo, o que significa que a percepção do erro na reconstrução foi mínima.

Para o modelo com a *min-tree* e o atributo inércia (quarto conjunto de imagens),

usado para gerar as imagens desejadas e no treinamento do modelo, o desempenho foi consideravelmente melhor em comparação com os resultados da Tabela 4.1. Na Tabela inicial, o modelo com a *min-tree* e o atributo inércia obteve um MSE de 5,26 e um MAE de 0,49, o que indicava uma perda significativa de precisão na reconstrução das imagens preditas. No entanto, ao comparar com os novos resultados, o MSE de 0,69 e o MAE de 0,21 observados nesta tabela (Tabela 4.2) indicam um erro ainda mais baixo, refletindo uma melhora na precisão da reconstrução das imagens. O PSNR de 49,72 dessa nova configuração é mais elevado do que o valor de 40,92 apresentado na tabela anterior, indicando uma melhoria na qualidade das imagens reconstruídas. Em relação ao PPD de 9,40, ele também é mais baixo em comparação com os 11,50 da tabela anterior, sugerindo uma percepção reduzida do erro na reconstrução. Esses resultados indicam que o uso exclusivo do atributo inércia teve um impacto positivo na qualidade da reconstrução das imagens. A seguir, a Figura 4.6 apresenta um exemplo de cada imagem do conjunto de imagens (primeiro e segundo).

Figura 4.6 – *Na primeira linha, está o primeiro conjunto de imagens, na segunda linha, o segundo conjunto. Em ambas as linhas, da esquerda para a direita, é apresentado a imagem de entrada, a imagem desejada, a imagem predita com o treinamento utilizando todos os atributos, e a imagem predita com o treinamento utilizando apenas um atributo.*

(a) *Imagem de entrada*

(b) *Imagem desejada*

(c) *Imagem reconstruída, todos atributos usados no treinamento*

(d) *Imagem reconstruída, somente o atributo área usado no treinamento.*

(e) *Imagem de entrada*

(f) *Imagem desejada*

(g) *Imagem reconstruída, todos atributos usados no treinamento*

(h) *Imagem reconstruída, somente o atributo inércia usado no treinamento*

Fonte: Elaborado pelo autor (2024).

Na Figura 4.6, observa-se que as imagens reconstruídas com o uso de todos os atributos no treinamento do modelo e aquelas reconstruídas utilizando apenas um único atributo são visualmente muito semelhantes. Além disso, como indicado na Tabela 4.1 e Tabela 4.2, as diferenças entre as métricas MSE, MAE, PPD e PSNR são mínimas. Em termos de resultados visuais das imagens preditas pelos modelos utilizando a *max-tree* e seus respectivos atributos na construção das imagens desejadas (área e inércia), as imagens

na Figura 4.6(c) e Figura 4.6(d) apresentam uma diferença praticamente imperceptível. O mesmo ocorre com as imagens na Figura 4.6(g) e Figura 4.6(h), que foram treinadas com todos os atributos e com apenas um único atributo, respectivamente. Em seguida, a Figura 4.7 apresenta um exemplo de cada imagem do conjunto de imagens (terceiro e quarto).

Figura 4.7 – *Na primeira linha, está o terceiro conjunto de imagens, na segunda linha, o quarto conjunto. Em ambas as linhas, da esquerda para a direita, é apresentado a imagem de entrada, a imagem desejada, a imagem predita com o treinamento utilizando todos os atributos, e a imagem predita com o treinamento utilizando apenas um atributo.*

(a) *Imagem de entrada* **(b)** *Imagem desejada* **(c)** *Imagem reconstruída, todos atributos usados no treinamento* **(d)** *Imagem reconstruída, somente o atributo área usado no treinamento.*

(e) *Imagem entrada* **(f)** *Imagem desejada* **(g)** *Imagem reconstruída, todos atributos usados no treinamento* **(h)** *Imagem reconstruída, somente o atributo inércia usado no treinamento*

Fonte: Elaborado pelo autor (2024).

De forma similar à Figura 4.6, na Figura 4.7, as imagens reconstruídas com todos os atributos e aquelas com apenas um único atributo durante o treinamento dos modelos são praticamente idênticas visualmente. Como indicado na Tabela 4.1 e Tabela 4.2, as diferenças nas métricas MSE, MAE, PPD e PSNR também são mínimas. As imagens na Figura 4.7(c) e Figura 4.7(d), assim como na Figura 4.7(g) e Figura 4.7(h), apresentam variações quase imperceptíveis, seja com todos os atributos ou com apenas um no treinamento dos modelos.

A partir dos quatro conjuntos de imagens gerados com essas árvores e atributos utilizados na filtragem das imagens desejadas, os resultados mostraram que a utilização de diferentes atributos de filtragem impacta levemente no desempenho do modelo, especialmente na reconstrução das imagens preditas. Os modelos treinados com a *max-tree* e o atributo área utilizados na filtragem das imagens desejadas mostraram um desempenho levemente superior, com menores valores de erro (MSE) e melhores métricas de qualidade da imagem (PSNR e SSIM). Já a combinação de *min-tree* e o atributo inércia utilizados na filtragem das imagens desejadas resultou em um desempenho levemente in-

ferior, especialmente quando comparado ao uso do atributo área. Além disso, ao treinar o modelo com apenas um único atributo por vez, observou-se uma leve melhora nos resultados, especialmente no caso do atributo área no treinamento do modelo. Isso sugere que, em determinadas condições, treinar o modelo com um único atributo pode reduzir a complexidade e melhorar a precisão na reconstrução das imagens preditas, embora as diferenças entre os métodos de treinamento, seja com todos os atributos ou apenas um, sejam mínimas em termos de métricas quantitativas. Em termos de desempenho visual, as imagens reconstruídas pelos modelos com diferentes abordagens de treinamento (todos os atributos versus um único atributo) apresentaram diferenças quase imperceptíveis. No entanto, a análise quantitativa das métricas de erro indicou que a escolha do atributo e da árvore utilizada tem um impacto leve, particularmente na redução de erros de reconstrução. Por fim, apesar da tentativa de dificultar o aprendizado dos filtros conexos, utilizando limiares elevados na construção das imagens desejadas, os resultados apresentados foram considerados satisfatórios.

4.2 Remoção de objetos e seleção dos melhores atributos

O experimento[3] em questão tem como objetivo explorar a remoção de objetos de imagens usando a aprendizagem dos filtros conexos, utilizando os melhores atributos para o treinamento e gerando a reconstrução da imagem de saída correspondente a cada par de imagens. Para isso, foram construídos 6 conjuntos de imagens, contendo diferentes tipos de parafusos, buchas e arruelas sobre uma superfície.

No processo de preparação dos conjuntos de imagens, as imagens foram obtidas por meio do seguinte procedimento, inicialmente, parafusos, buchas e/ou arruelas foram distribuídos sobre uma superfície, e então realizou-se a captura de uma imagem. Em seguida, procedeu-se à remoção manual dos parafusos, buchas e/ou arruelas dessa superfície, e uma nova captura de imagem foi realizada. Dessa forma, a captura de imagens seguiu-se por vários pares de imagens para cada conjunto de imagens. Mais a seguir, cada conjunto de imagens será detalhado para melhor compreensão. Ao final do processo de captura, obteve-se um total de 600 imagens, distribuídas pelos 6 conjuntos, sendo cada conjunto composto por 100 imagens. Em cada conjunto, 50 imagens correspondem àquelas denominadas imagens de entrada, nas quais todos os objetos escolhidos para compor o cenário permaneceram na superfície, enquanto as outras 50 imagens, chamadas de imagens desejadas, correspondem a situações nas quais um dos objetos foi removido manualmente do cenário. Como é possível observar, diferente do experimento anterior, neste experimento, as imagens desejadas foram obtidas manualmente, com a remoção dos objetos do cenário, ou seja, não foi realizada a construção das imagens desejadas utilizando a filtragem de uma

[3] <https://github.com/Anderson-hrz/APRENDIZAGEM-DE-FILTROS-CONEXOS-POR-REDES-NEURAIS-USANDO-ARVORES-DE-COMPONENTES>.

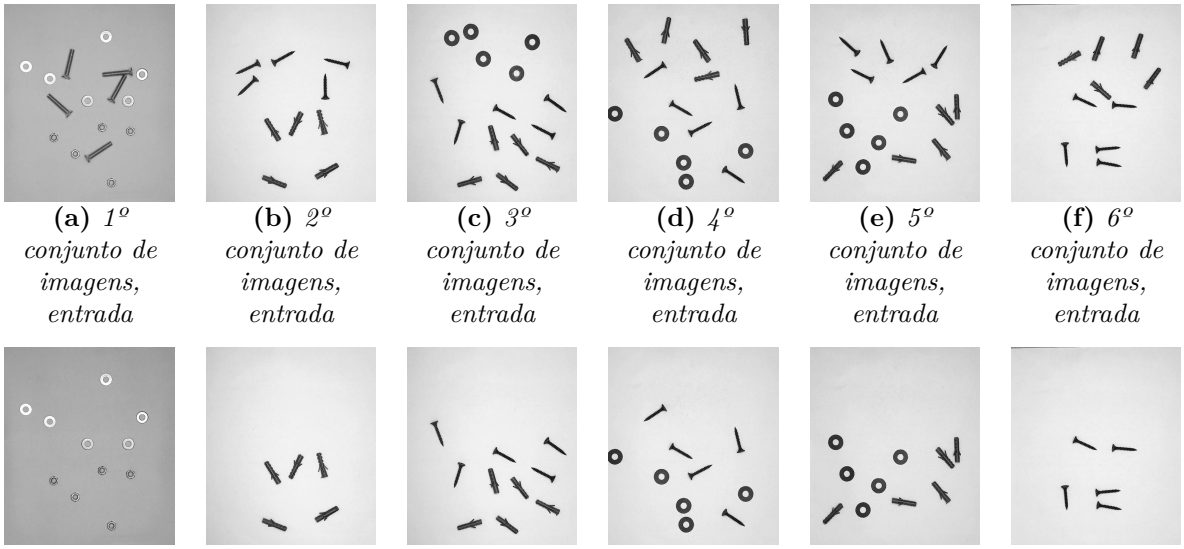
árvore. No entanto, para este experimento, como detalhado na Seção 3, a aprendizagem dos filtros conexo exige o uso de uma árvore. Assim, a árvore utilizada neste experimento foi a *min-tree*, devido à superfície branca sobre a qual as imagens foram geradas. A seguir, a Tabela 4.3 apresenta a distribuição dos conjuntos de imagens.

Conjunto de imagens	Objetos no cenário, imagens de entrada	Objeto a ser removido, imagens desejadas	Tipo de parafuso
1º	Parafusos, porcas e arruelas	Parafusos	Cabeça chata
2º	Parafusos, buchas	Parafusos	Cabeça trombeta
3º	Parafusos, porcas e arruelas	Arruelas	-
4º	Parafusos, buchas e arruelas	Buchas	-
5º	Parafusos, buchas e porcas	Parafusos	Cabeça trombeta
6º	Parafusos e buchas	Buchas	-

Tabela 4.3 – *Distribuição dos conjuntos de imagens, objetos presentes nos cenários e os objetos a serem removidos em cada imagem, utilizadas no treinamento dos modelos.*

Já na Figura 4.8 são apresentados os conjuntos de imagens e um par de amostras de imagens de cada conjunto. Em cada par, a imagem de entrada exibe os objetos no cenário, enquanto a imagem desejada mostra a versão do cenário com os objetos removidos.

Figura 4.8 – *Amostras dos conjuntos de imagens. Cada coluna representa um par de imagens dos conjuntos de imagens, sendo uma imagem de entrada e uma imagem desejada.*



(a) *1º conjunto de imagens, entrada* **(b)** *2º conjunto de imagens, entrada* **(c)** *3º conjunto de imagens, entrada* **(d)** *4º conjunto de imagens, entrada* **(e)** *5º conjunto de imagens, entrada* **(f)** *6º conjunto de imagens, entrada*

(g) *1º conjunto de imagens, desejada* **(h)** *2º conjunto de imagens, desejada* **(i)** *3º conjunto de imagens, desejada* **(j)** *4º conjunto de imagens, desejada* **(k)** *5º conjunto de imagens, desejada* **(l)** *6º conjunto de imagens, desejada*

Fonte: Elaborado pelo autor (2024)

Para a realização dos treinamentos, as imagens dos conjuntos foram convertidas para nível de cinza, com dimensões de 529 × 470 *pixels*, correspondendo a 40% do tamanho original(1324 × 1177 *pixels*). Para cada conjunto de imagens, 70% (35 imagens) do mesmo

foram usados para o treinamento e 30% (15 imagens) foram destinadas ao conjunto de teste. Em relação à avaliação do impacto da aprendizagem dos filtros conexos neste experimento, também utilizou-se a métrica MSE e o otimizador escolhido foi o AdaMax. O número de épocas para o treinamento foi estabelecido em 50. A taxa de aprendizagem foi fixada em 0,1 para todos os treinamentos, e o tamanho do lote foi definido como 1, essas configurações seguiram as tendências demonstradas no experimento anterior.

Conforme evidenciado pelo experimento anterior, foi realizado um comparativo entre o treinamento dos modelos utilizando todos os atributos e os treinamentos dos modelos com um único atributo. Esse comparativo demonstrou uma leve melhoria no desempenho dos modelos quando foi utilizado um único atributo. Para o experimento em questão, como mencionado acima, o objetivo é realizar a remoção de objetos de imagens usando a aprendizagem dos filtros conexo com os melhores atributos. Ou seja, não se limitar a um único atributo, mas selecionar os melhores atributos de forma mais abrangente e, em seguida, treinar os modelos com essa seleção. No entanto, antes de detalhar a escolha dos melhores atributos e apresentar os resultados do treinamento com esses atributos, foi realizado o treinamento com todos os atributos disponíveis, a fim de estabelecer uma base de comparação à qual será demonstrada pouco mais abaixo.

4.2.1 Análise de resultados e discussões

A seguir, são apresentados os resultados obtidos com o treinamento dos modelos com todos os atributos nos conjuntos de imagens, conforme demonstrado na Tabela 4.4, que exibe o desempenho da aprendizagem no conjunto de teste.

Conjunto de imagens	Atributos	MSE	MAE	PPD	PSNR	SSIM
1º	Todos	13,85	2,47	84,72	36,71	0,91
2º		21,05	1,99	82,16	34,89	0,95
3º		9,52	2,12	83,19	38,34	0,95
4º		17,62	2,92	87,94	35,66	0,91
5º		13,01	2,35	87,50	36,98	0,95
6º		16,10	2,37	85,15	36,06	0,92

Tabela 4.4 – *Resultados do treinamento dos modelos com a utilização de todos os atributos.*

Após a apresentação dos resultados dos treinamentos com os modelos com todos os atributos nos conjuntos de imagens na Tabela 4.4, a tabela a seguir exibe os resultados obtidos com os melhores atributos selecionados. A seleção dos atributos foi realizada conforme descrito na Subseção 2.9.1. Para esse processo, foram utilizadas 50 épocas para cada conjunto de imagens, com a taxa de aprendizagem mantida em 0,1 e o tamanho do lote definido como 1. A avaliação foi realizada utilizando o MSE, e o otimizador adotado foi o AdaMax. A seguir, são apresentados os melhores atributos, de acordo com a seleção realizada.

- Atributos selecionados no 1º conjunto de imagens: Área, Média, Altura, Momentos centrais de ordem 20, Momentos centrais de ordem 30, Momentos centrais de ordem 21, Momentos centrais de ordem 12, Orientação, Comprimento do eixo maior, Excentricidade, Momento de Hu de ordem 2, Momento de Hu de ordem 4, Momento de Hu de ordem 5 e Momento de Hu de ordem 7.
- Atributos selecionados no 2º conjunto de imagens: Área, Nível, Variância do nível, Desvio padrão, Retangularidade, Razão, Momentos centrais de ordem 30, Momentos centrais de ordem 12, Comprimento do eixo maior, Compacidade, Momento de Hu de ordem 1, Momento de Hu de ordem 2, Momento de Hu de ordem 3 e Momento de Hu de ordem 6.
- Atributos selecionados no 3º conjunto de imagens: Volume, Nível, Média do nível, Largura da caixa, Retangularidade, Razão, Momentos centrais de ordem 20, Comprimento do eixo maior, Comprimento do eixo menor, Compacidade, Momento de Hu de ordem 1, Momento de Hu de ordem 2, Momento de Hu de ordem 3 e Momento de Hu de ordem 6.
- Atributos selecionados no 4º conjunto de imagens: Área, Média do nível, Variância do nível, Largura da caixa, Razão, Momentos centrais de ordem 20, Momentos centrais de ordem 21, Momentos centrais de ordem 12, Comprimento do eixo maior, Comprimento do eixo menor, Momento de Hu de ordem 1, Momento de Hu de ordem 2, Momento de Hu de ordem 5 e Momento de Hu de ordem 6.
- Atributos selecionados no 5º conjunto de imagens: Área, Variância do nível, Retangularidade, Razão, Momentos centrais de ordem 20, Momentos centrais de ordem 11, Momentos centrais de ordem 03, Comprimento do eixo maior, Momento de Hu de ordem 1, Momento de Hu de ordem 2, Momento de Hu de ordem 3, Momento de Hu de ordem 5, Momento de Hu de ordem 6 e Momento de Hu de ordem 7.
- Atributos selecionados no 6º conjunto de imagens: Nível, Variância do nível, Desvio padrão, Largura da caixa, Momentos centrais de ordem 20, Momentos centrais de ordem 02, Momentos centrais de ordem 30, Comprimento do eixo maior, Compacidade, Momento de Hu de ordem 1, Momento de Hu de ordem 2, Momento de Hu de ordem 3, Momento de Hu de ordem 4 e Momento de Hu de ordem 6.

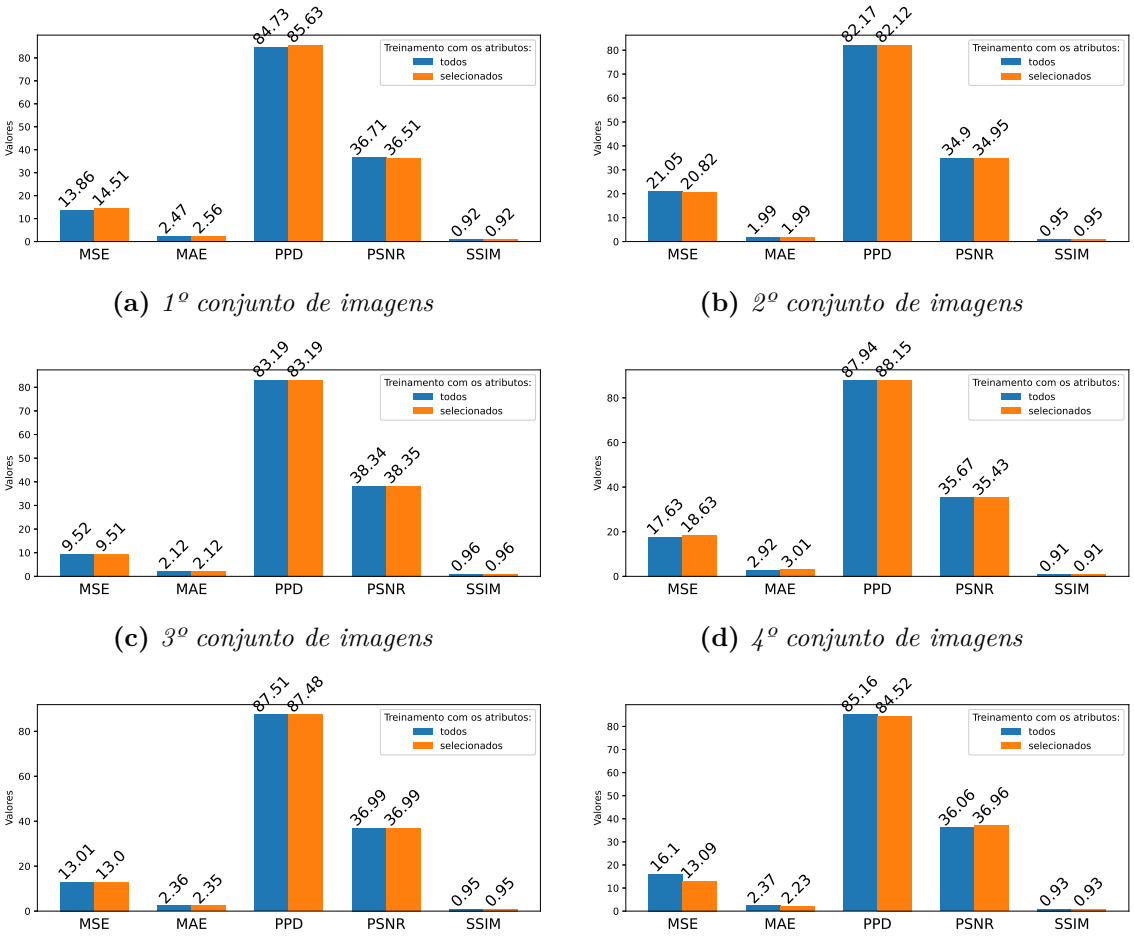
Com base nos melhores atributos selecionados na aprendizagem dos filtros conexos, a seguir a Tabela 4.5 apresenta os resultados do treinamento dos modelos.

Conjunto de imagens	Atributos	MSE	MAE	PPD	PSNR	SSIM
1º	Selecionados	14,51	2,56	85,62	36,51	0,92
2º		20,82	1,98	82,12	34,94	0,95
3º		9,50	2,12	83,19	38,35	0,95
4º		18,63	3,00	88,14	35,42	0,90
5º		13,00	2,35	87,48	36,99	0,95
6º		13,08	2,23	84,52	36,96	0,92

Tabela 4.5 – *Resultados do treinamento dos modelos com a utilização dos atributos selecionados.*

Além dos valores apresentados na Tabela 4.4 e na Tabela 4.5, a Figura 4.9 a seguir apresenta um gráfico de barras que compara as métricas MSE, MAE, PPD, PSNR e SSIM entre as duas modalidades dos modelos treinados.

Figura 4.9 – *Comparação entre os treinamentos dos conjuntos de imagens.*



(a) *1º conjunto de imagens*

(b) *2º conjunto de imagens*

(c) *3º conjunto de imagens*

(d) *4º conjunto de imagens*

(e) *5º conjunto de imagens*

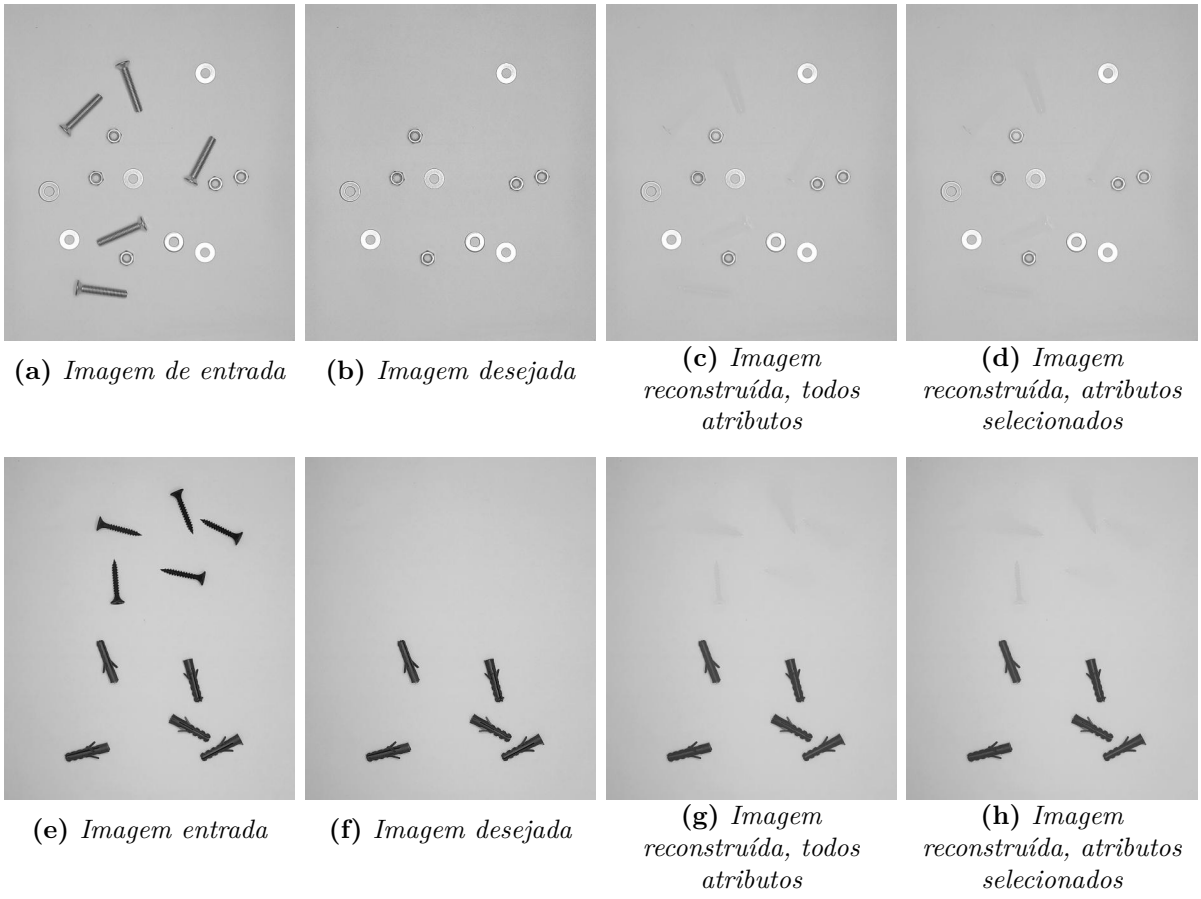
(f) *6º conjunto de imagens*

Fonte: Elaborado pelo autor (2024).

Ao comparar os resultados da Tabela 4.4 e Tabela 4.5, observa-se uma diferença discreta entre as duas abordagens de treinamento, conforme ilustrado na Figura 4.9. Em relação ao MSE, os modelos treinados com todos os atributos, no 1º e 4º conjunto de imagens, apresentaram valores ligeiramente inferiores, sugerindo que a inclusão de mais atributos pode ajudar o modelo a se ajustar melhor a certos conjuntos de imagens. No entanto, em termos de MAE, PPD e SSIM, a diferença entre os modelos com todos os atributos e os com atributos selecionados é mínima. Esses resultados indicam que a utilização dos melhores atributos não gera uma melhoria significativa no desempenho do modelo. Em geral, a diferença entre os resultados do treinamento com todos os atributos e os atributos selecionados, da perspectiva de todas as métricas de avaliação discutidas aqui, é mínima. A seguir, a Figura 4.10 apresenta os resultados das imagens preditas para

as duas abordagens no 1º conjunto de imagens e no 2º conjunto de imagens.

Figura 4.10 – *Na primeira linha, está o 1º conjunto de imagens, na segunda linha, o segundo conjunto. Em ambas as linhas, da esquerda para a direita, é apresentado a imagem de entrada, a imagem desejada, a imagem predita com o treinamento utilizando todos os atributos, e a imagem predita com o treinamento utilizando apenas os atributos selecionados.*



(a) *Imagem de entrada* **(b)** *Imagem desejada* **(c)** *Imagem reconstruída, todos atributos* **(d)** *Imagem reconstruída, atributos selecionados*

(e) *Imagem entrada* **(f)** *Imagem desejada* **(g)** *Imagem reconstruída, todos atributos* **(h)** *Imagem reconstruída, atributos selecionados*

Fonte: Elaborado pelo autor (2024).

Na Figura 4.10, as imagens reconstruídas com as duas abordagens são praticamente imperceptíveis.

Em complemento, o objetivo deste experimento foi investigar a aprendizagem dos filtros conexos utilizando a seleção dos melhores atributos para o treinamento dos modelos para remoção de objetos. Tal abordagem pode ser considerada um aprimoramento, pois permite o treinamento de modelos mais eficientes, sem a sobrecarga de atributos desnecessários. Embora a seleção de atributos seja uma prática comum no aprendizado de máquina e em redes neurais, com o intuito de reduzir a dimensionalidade dos dados e potencialmente melhorar o desempenho do modelo, os resultados deste experimento demonstraram que a utilização de todos os atributos disponíveis obteve resultados comparáveis, e, em alguns casos, até ligeiramente melhores. Pode-se concluir que o uso de todos os atributos não resulta, necessariamente, em desempenho superior em todas as métricas no processo

de aprendizagem dos filtros conexos, especialmente quando a seleção de atributos oferece uma representação mais compacta e relevante dos dados. Dessa forma, pode haver uma alta correlação entre os atributos, tornando alguns deles redundantes e, portanto, sua remoção não impacta significativamente o desempenho da aprendizagem. Por outro lado, é possível observar que os modelos treinados com todos os atributos mostraram-se eficientes na realização das predições dos nós. Por fim, em relação ao comparativo visual da aprendizagem, embora os resultados tenham sido considerados satisfatórios, dependendo da aplicação, é necessário realizar um pós-processamento para aproximar-se ao máximo da imagem desejada.

5 CONCLUSÃO

Nesta dissertação, foi apresentada uma nova abordagem para a aprendizagem de filtros conexos através de redes neurais, com o intuito de otimizar a filtragem de árvores morfológicas de forma integrada e diferenciável. Os resultados experimentais demonstraram que a escolha criteriosa dos atributos de filtragem, como área e inércia, bem como a seleção entre as estruturas *max-tree* e *min-tree*, impacta significativamente o desempenho dos modelos, tanto em termos quantitativos quanto qualitativos.

No primeiro experimento, a combinação do atributo área com a utilização da *max-tree* evidenciou um desempenho superior, refletido por métricas robustas e pela qualidade visual das imagens reconstruídas. Esse resultado valida a proposta de desenvolver um método que otimize a filtragem de árvores morfológicas, atendendo ao objetivo de aprender critérios baseados em filtros conexos, indo ao encontro do primeiro objetivo específico. Além disso, a análise revelou que a variação dos parâmetros na estrutura da árvore morfológica pode influenciar, ainda que de forma sutil, os resultados obtidos, enfatizando a necessidade de uma investigação aprofundada sobre o impacto desses parâmetros na performance da rede neural de acordo com o segundo objetivo específico.

O segundo experimento concentrou-se na seleção de atributos para a remoção de objetos, demonstrando que essa estratégia pode reduzir a complexidade do modelo sem comprometer o desempenho. A comparação entre modelos que utilizam todos os atributos e aqueles que empregam uma seleção criteriosa evidenciou que a redução de atributos não apenas mantém a eficácia na reconstrução das imagens, mas também simplifica o processo de treinamento. Esse achado reforça os objetivos de avaliar o desempenho do método proposto em conjuntos de dados diversificados, terceiro objetivo específico, e de analisar os efeitos da escolha dos atributos para a tarefa de filtragem.

Adicionalmente, a integração de uma função contínua e diferenciável no lugar da função booleana tradicional para a seleção de nós na árvore morfológica representou um avanço. Essa modificação possibilita a otimização via gradiente descendente, alinhando o método aos requisitos fundamentais do aprendizado em redes neurais e proporcionando uma adaptação mais flexível durante o treinamento. Essa abordagem não só enriquece a metodologia proposta, mas também amplia o leque de aplicações potenciais em sistemas de processamento e análise de imagens.

De forma geral, os resultados obtidos demonstram que a aprendizagem dos filtros conexos com redes neurais pode melhorar a qualidade da reconstrução de imagens, ao mesmo tempo em que reduz a complexidade computacional e oferece uma abordagem mais adaptativa para a filtragem. Essa integração contribui para o avanço ou a integração com outras técnicas de aprendizado profundo.

Portanto, esta dissertação atendeu aos objetivos propostos, desenvolvendo um método otimizado, investigando a influência dos parâmetros das árvores morfológicas, ava-

liando o desempenho em diferentes cenários e analisando a escolha dos atributos, como também oferece uma base para trabalhos futuros que busquem expandir e aplicar essas técnicas em contextos mais amplos do processamento de imagens.

5.1 Trabalho Futuros

Embora os resultados de ambos os experimentos deste trabalho tenham sido satisfatórios na aprendizagem dos filtros conexos com redes neurais, alguns tópicos merecem destaque como possíveis direções para estudos futuros:

- Estudar novas formas de uma função de custo personalizada para penalizar a remoção ou não remoção de nós. De maneira mais específica, projetar uma função de custo que penalize erros de forma mais intensa, por exemplo, usando um MSE, mas multiplicado por um fator de penalização que varia dependendo de algum critério adicional, podendo ser o nível de cinza do nó.
- Avaliar o impacto das variações da função de ativação sigmoide por outras versões. A sigmoide tradicional pode ser ajustada por meio de modificações em seus parâmetros, como a inclinação ou o deslocamento, o que pode alterar a resposta do modelo a diferentes entradas.
- Construir novas estratégias de aprendizagem dos filtros conexos com redes neurais para segmentação de imagens com técnicas de aprendizagem que exploram de forma mais eficaz as relações e dependências nas imagens. O objetivo é melhorar a precisão, a generalização e a eficiência da segmentação.

REFERÊNCIAS BIBLIOGRÁFICAS

AGATONOVIC-KUSTRIN, S.; BERESFORD, R. Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. *Journal of pharmaceutical and biomedical analysis*, Elsevier, v. 22, n. 5, p. 717–727, 2000. Citado na pág. 49.

ALVES, W. A. L. *Últimos Levelings: Conceitos, Propriedades, Algoritmos e Aplicações em Processamento e Análise de Imagens*. Tese (Tese (Doutorado)) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015. 138 f. Citado na pág. 27, 30, 32, 36, 37, 39.

ALZUBAIDI, L. et al. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, Springer, v. 8, p. 1–74, 2021. Citado na pág. 23.

AOUAD, T.; TALBOT, H. Binary morphological neural network. In: IEEE. *2022 IEEE International Conference on Image Processing (ICIP)*. [S.l.], 2022. p. 3276–3280. Citado na pág. 22.

BREEN, E. J.; JONES, R. Attribute openings, thinnings, and granulometries. *Computer vision and image understanding*, Elsevier, v. 64, n. 3, p. 377–389, 1996. Citado na pág. 27.

BURGER, W.; BURGE, M. J. *Digital Image Processing: An Algorithmic Introduction*. [S.l.]: Springer Nature, 2022. Citado na pág. 40.

CABRERA, D. F. et al. Segmentation of axillary and supraclavicular tumoral lymph nodes in pet/ct: A hybrid cnn/component-tree approach. In: IEEE. *2020 25th International Conference on Pattern Recognition (ICPR)*. [S.l.], 2021. p. 6672–6679. Citado na pág. 24.

CHANDRASHEKAR, G.; SAHIN, F. A survey on feature selection methods. *Computers & electrical engineering*, Elsevier, v. 40, n. 1, p. 16–28, 2014. Citado na pág. 55.

COLLIOT, O. Machine learning for brain disorders. Springer Nature, 2023. Citado na pág. 49.

CORMEN, T. H. et al. *Introduction to algorithms*. [S.l.]: MIT press, 2022. Citado na pág. 35.

DUCH, W.; JANKOWSKI, N. Survey of neural transfer functions. *Neural computing surveys*, v. 2, n. 1, p. 163–212, 1999. Citado na pág. 52.

FAEZI, M. H.; PELETIER, R.; WILKINSON, M. H. Multi-spectral source-segmentation using semantically-informed max-trees. *IEEE Access*, IEEE, 2024. Citado na pág. 23.

GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. [S.l.]: "O'Reilly Media, Inc.", 2022. Citado na pág. 54.

GOBBER, C. F. *Residual spaces based on component trees: theory and applications*. Tese (Doutorado) — Universidade Nove de Julho - UNINOVE, São Paulo, 2021. Tese (Doutorado) – Universidade Nove de Julho - UNINOVE, 94 f. Citado na pág. 27.

GONZALEZ, R. R. C.; WOODS, R. E. *Digital image processing*. [S.l.]: Pearson Education Ltd., 2008. Citado na pág. 30.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016. Citado na pág. 49.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *Journal of machine learning research*, v. 3, n. Mar, p. 1157–1182, 2003. Citado na pág. 55.

HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 770–778. Citado na pág. 22.

HEIJMANS, H. J. Connected morphological operators for binary images. *Computer Vision and Image Understanding*, Elsevier, v. 73, n. 1, p. 99–120, 1999. Citado na pág. 27.

HERMARY, R. et al. Learning grayscale mathematical morphology with smooth morphological layers. *Journal of Mathematical Imaging and Vision*, Springer, v. 64, n. 7, p. 736–753, 2022. Citado na pág. 22.

HUGHES-HALLETT, D.; GLEASON, A. M.; MCCALLUM, W. G. *Calculus: Single and multivariable*. [S.l.]: John Wiley & Sons, 2020. Citado na pág. 60.

JAIN, A. K.; MAO, J.; MOHIUDDIN, K. M. Artificial neural networks: A tutorial. *Computer*, IEEE, v. 29, n. 3, p. 31–44, 1996. Citado na pág. 49.

JONES, R. Component trees for image filtering and segmentation. In: MACKINAC ISLAND. *IEEE Workshop on Nonlinear Signal and Image Processing*. [S.l.], 1997. v. 9. Citado na pág. 23, 35.

JONES, R. Connected filtering and segmentation using component trees. *Computer Vision and Image Understanding*, Elsevier, v. 75, n. 3, p. 215–228, 1999. Citado na pág. 27.

JORDAN, M. I.; MITCHELL, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, American Association for the Advancement of Science, v. 349, n. 6245, p. 255–260, 2015. Citado na pág. 22.

JOVIĆ, A.; BRKIĆ, K.; BOGUNOVIĆ, N. A review of feature selection methods with applications. In: IEEE. *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. [S.l.], 2015. p. 1200–1205. Citado na pág. 55.

KHANAM, R. et al. A comprehensive review of convolutional neural networks for defect detection in industrial applications. *IEEE Access*, IEEE, 2024. Citado na pág. 22.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na pág. 57.

KORTLI, Y. et al. Face recognition systems: A survey. *Sensors*, MDPI, v. 20, n. 2, p. 342, 2020. Citado na pág. 22.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015. Citado na pág. 52, 54.

MANAKITSA, N. et al. A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision. *Technologies*, MDPI, v. 12, n. 2, p. 15, 2024. Citado na pág. 22.

MEYER, C. et al. Combining max-tree and cnn for segmentation of cellular fib-sem images. In: *Discrete Geometry and Mathematical Morphology (DGMM) Reproducible Research in Pattern Recognition (RRPR), Strasbourg, France, octobre 2022*. [S.l.: s.n.], 2022. Citado na pág. 24.

MONASSE, P.; GUICHARD, F. Scale-space from a level lines tree. *Journal of Visual Communication and Image Representation*, Elsevier, v. 11, n. 2, p. 224–236, 2000. Citado na pág. 27.

MONDAL, R. et al. Morphological network: How far can we go with morphological neurons? *arXiv preprint arXiv:1901.00109*, 2019. Citado na pág. 22.

MORIMITSU, A. *Computação Incremental e Eficiente de Sequências de Árvores de Componentes*. Dissertação (Dissertação (Mestrado)) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2015. Citado na pág. 27.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 807–814. Citado na pág. 52.

NAJMAN, L.; TALBOT, H. *Mathematical morphology: from theory to applications*. [S.l.]: John Wiley & Sons, 2013. Citado na pág. 27.

NWANKPA, C. et al. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018. Citado na pág. 52.

PASSAT, N. et al. Interactive segmentation based on component-trees. *Pattern Recognition*, Elsevier, v. 44, n. 10-11, p. 2539–2554, 2011. Citado na pág. 23.

PASZKE, A. et al. Automatic differentiation in pytorch. 2017. Citado na pág. 66.

PERRET, B.; COUSTY, J. Component tree loss function: Definition and optimization. In: SPRINGER. *Discrete Geometry and Mathematical Morphology: Second International Joint Conference, DGMM 2022, Strasbourg, France, October 24–27, 2022, Proceedings*. [S.l.], 2022. p. 248–260. Citado na pág. 24, 62, 64.

PERRET, B. et al. Directed connected operators: Asymmetric hierarchies for image filtering and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 37, n. 6, p. 1162–1176, 2014. Citado na pág. 23, 27, 28.

ROADTEXT, O. *Occluded RoadText: Robust Scene Text Detection and Recognition in the Presence of Occlusions*. 2024. <https://rrc.cvc.uab.es/ch=29&com=downloads>. Acessado: 2024-11-01. Citado na pág. 66.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na pág. 49.

RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. Citado na pág. 57.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *nature*, Nature Publishing Group UK London, v. 323, n. 6088, p. 533–536, 1986. Citado na pág. 54.

RYU, J.; TRAGER, S. C.; WILKINSON, M. H. A fast alpha-tree algorithm for extreme dynamic range pixel dissimilarities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2023. Citado na pág. 23.

SALEH, A. et al. Forest fire surveillance systems: A review of deep learning methods. *Heliyon*, Elsevier, 2024. Citado na pág. 22.

SALEMBIER, P.; GARRIDO, L. Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval. *IEEE transactions on Image Processing*, IEEE, v. 9, n. 4, p. 561–576, 2000. Citado na pág. 27.

SALEMBIER, P.; OLIVERAS, A. Practical extensions of connected operators. In: *Mathematical Morphology and its applications to image and signal processing*. [S.l.]: Springer, 1996. p. 97–110. Citado na pág. 23, 27.

SALEMBIER, P.; OLIVERAS, A.; GARRIDO, L. Antiextensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, IEEE, v. 7, n. 4, p. 555–570, 1998. Citado na pág. 23, 27, 28, 37, 40, 44, 45.

SALEMBIER, P.; SERRA, J. Flat zones filtering, connected operators, and filters by reconstruction. *IEEE Transactions on image processing*, IEEE, v. 4, n. 8, p. 1153–1160, 1995. Citado na pág. 27.

SALEMBIER, P.; WILKINSON, M. H. Connected operators. *IEEE Signal Processing Magazine*, IEEE, v. 26, n. 6, p. 136–157, 2009. Citado na pág. 27, 40, 44, 45.

SHAFIEE, S. et al. Sequential forward selection and support vector regression in comparison to lasso regression for spring wheat yield prediction based on uav imagery. *Computers and Electronics in Agriculture*, Elsevier, v. 183, p. 106036, 2021. Citado na pág. 55.

SHEN, Y.; ZHONG, X.; SHIH, F. Y. Deep morphological neural networks. *arXiv preprint arXiv:1909.01532*, 2019. Citado na pág. 23.

SILVA, D. J. da. *Contagem incremental de padrões locais em árvores de componentes para cálculo de atributos*. Dissertação (Dissertação (Mestrado)) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017. Citado na pág. 27.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado na pág. 22.

TAN, K. et al. Integrating advanced computer vision and ai algorithms for autonomous driving systems. *Journal of Theory and Practice of Engineering Science*, v. 4, n. 01, p. 41–48, 2024. Citado na pág. 22.

URBACH, E. R.; WILKINSON, M. H. Shape-only granulometries and grey-scale shape filters. In: *Proc. Int. Symp. Math. Morphology (ISMM)*. [S.l.: s.n.], 2002. v. 2002, p. 305–314. Citado na pág. 46.

WAH, Y. B. et al. Feature selection methods: Case of filter and wrapper approaches for maximising classification accuracy. *Pertanika Journal of Science & Technology*, v. 26, n. 1, 2018. Citado na pág. 55.

WANG, J. et al. Interactive image manipulation using morphological trees and spline-based skeletons. *Computers & Graphics*, Elsevier, v. 108, p. 61–73, 2022. Citado na pág. 23.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, IEEE, v. 13, n. 4, p. 600–612, 2004. Citado na pág. 58.

WILKINSON, M. et al. Connected operators: A review of region-based morphological image processing techniques. *IEEE Signal Processing Magazine*, v. 26, n. 6, p. 136–157, 2009. Citado na pág. 23.

WILSON, S. S. Morphological networks. In: SPIE. *Visual Communications and Image Processing IV*. [S.l.], 1989. v. 1199, p. 483–495. Citado na pág. 22.

XU, Y. *Tree-based shape spaces: Definition and applications in image processing and computer vision*. Tese (Doutorado) — Université Paris-Est, 2013. Citado na pág. 39, 44, 45.

XU, Y.; GÉRAUD, T.; NAJMAN, L. Morphological filtering in shape spaces: Applications using tree-based image representations. In: IEEE. *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. [S.l.], 2012. p. 485–488. Citado na pág. 23.

ZHANG, W.; SHI, D.; YANG, X. An improved edge detection algorithm based on mathematical morphology and directional wavelet transform. In: IEEE. *2015 8th International Congress on Image and Signal Processing (CISP)*. [S.l.], 2015. p. 335–339. Citado na pág. 22.