UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

CRISTHIAN XAVIER HERNÁNDEZ RODRÍGUEZ

Um modelo computacional para o diagnóstico de leucemia linfoblástica aguda baseado em aprendizado de máquina e técnicas de processamento de imagens

Campinas

2021

Cristhian Xavier Hernández Rodríguez

Um modelo computacional para o diagnóstico de leucemia linfoblástica aguda baseado em aprendizado de máquina e técnicas de processamento de imagens

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Matemática Aplicada.

Orientador: Marcos Eduardo Ribeiro do Valle Mesquita

Este trabalho corresponde à versão final da Dissertação defendida pelo aluno Cristhian Xavier Hernández Rodríguez e orientada pelo Prof. Dr. Marcos Eduardo Ribeiro do Valle Mesquita.

Campinas

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

H43m

Hernández Rodríguez, Cristhian Xavier, 1991-
Um modelo computacional para o diagnóstico de leucemia linfoblástica aguda baseado em aprendizado de máquina e técnicas de processamento de imagens / Cristhian Xavier Hernández Rodríguez. – Campinas, SP : [s.n.], 2021.

Orientador: Marcos Eduardo Ribeiro do Valle Mesquita.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Processamento imagens - Técnicas digitais. 2. Algoritmos de computador. 3. Aprendizado de máquina. 4. Leucemia linfóide aguda. 5. Segmentação de imagens. I. Mesquita, Marcos Eduardo Ribeiro do Valle, 1979-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Un modelo computacional para el diagnóstico de leucemia linfoblástica aguda basado en aprendizaje de máquina y técnicas de procesamiento de imagenes
Palavras-chave em inglês:
Images processing - Digital techniques
Computer algorithms
Machine learning
Acute lymphoblastic leukemia
Image segmentation
Área de concentração: Matemática Aplicada
Titulação: Mestre em Matemática Aplicada
Banca examinadora:
Marcos Eduardo Ribeiro do Valle Mesquita [Orientador]
Nina Sumiko Tomita Hirata
João Batista Florindo
Data de defesa: 15-06-2021
Programa de Pós-Graduação: Matemática Aplicada

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-8530-9880
- Currículo Lattes do autor: http://lattes.cnpq.br/7353101199768097

Dissertação de Mestrado defendida em 15 de junho de 2021 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). MARCOS EDUARDO RIBEIRO DO VALLE MESQUITA

Prof(a). Dr(a). JOÃO BATISTA FLORINDO

Prof(a). Dr(a). NINA SUMIKO TOMITA HIRATA

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Este trabalho é dedicado à minha mãe,
porque sem seu esforço e amor, eu nunca
teria chegado a escrever estas linhas.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Eu agradeço ao Brasil por me receber com tanto carinho e me ensinar tudo o que aprendi durante meus estudos de mestrado. Quero agradecer também ao meu orientador, Marcos Eduardo Valle, pela paciência, suporte, ajuda e compreensão infinita.

Agradeço a Deus que jamais me abandonou e cuidou dos meus seres queridos na pandemia que atingiu o mundo no ano 2020. Aos meus pais e irmãos que sempre estiveram me dando o seu apoio e se preocuparam comigo durante todo o tempo que estive fora de casa. Finalmente, agradeço à minha noiva Nereida Troya, que me ensinou que o amor e a distância são diretamente proporcionais se existe vontade, paciência e dois corações dispostos a tudo.

"Não temas, porque eu estou contigo;
não te assombres, porque eu sou teu Deus;
eu te fortaleço, e te ajudo, e te sustento
com a destra da minha justiça".
(Bíblia Sagrada, Isaías 41, 10)

Resumo

Neste trabalho faremos uma descrição prática de um algoritmo computacional que utiliza diversas técnicas de processamento digital de imagens e um modelo de aprendizado de máquinas para contar o número de linfoblastos presentes em uma imagem de esfregaço sanguíneo obtida por microscópio com a finalidade de contribuir com o diagnóstico da leucemia linfoblástica aguda. Em concreto, o algoritmo recebe uma imagem colorida, faz uma eliminação de ruído e posteriormente uma binarização ótima que permite usar morfologia matemática para estimar o diâmetro das células presentes na imagem. Mediante técnicas de segmentação, o algoritmo localiza possíveis células brancas na imagem e, com ajuda do diâmetro estimado, gera automaticamente sub-imagens enquadrando cada célula. As sub-imagens são analisadas por uma rede neural convolucional que classifica a célula como um linfoblasto, se for o caso, ou uma célula saudável. Finalmente, o algoritmo retorna uma imagem igual à inicial mas com os linfoblastos marcados e a quantidade deles. Para avaliar seu desempenho, o modelo foi executado em 108 imagens de esfregaço de sangue nas duas formas de avaliação da base ALL-IDB disponibilizada pela "Università Degli Studi di Milano": uma focada na classificação da célula como linfoblasto e outra no diagnóstico do paciente com base na imagem do esfregaço de sangue. Obtivemos uma especificidade de 98.04% e 100% respectivamente.

Palavras-chave: Processamento digital de imagens. Aprendizado de máquinas. Linfoblastos. Leucemia linfoblástica aguda. Morfologia matemática. Rede neural convolucional.

Abstract

In this work, we will provide a practical description of a computational algorithm that uses several digital image processing techniques and a machine learning model to count the number of lymphoblasts present in a blood smear image obtained by microscope as a way to contribute to the diagnosis of acute lymphoblastic leukemia. Specifically, the algorithm receives a color image, the elimination of noise is performed to subsequently achieve an optimal binarization, which allows for the use of mathematical morphology to estimate the diameter of the cells present in the image. Through segmentation techniques, the algorithm locates possible white cells in the image and, with the estimated diameter, automatically generates sub-images for each one of them, digitally cutting a square segment where they appear. These sub-images are analyzed by a convolutional neural network that classify each white cell as a lymphoblast or a healthy cell. Finally, the algorithm returns an image equal to the initial image but with the lymphoblasts annotated and their quantity. To assess its performance, the model was executed on 108 blood smear images in the two forms of assessment of the ALL-IDB base provided by the "Università Degli Studi di Milano": one focused on the classification of the cell as a lymphoblast and the other on the diagnosis of the patient based on the blood smear image. We obtained a recall about 98.04% and 100% respectively.

Keywords: Digital image processing. Machine learning. Lymphoblast. Acute lymphoblastic leukemia. Mathematical morphology. Convolutional neural network.

Lista de ilustrações

Lista de tabelas

Sumário

Introdução

Um dos procedimentos de diagnóstico disponíveis para avaliar a presença da Leucemia Linfoblástica Aguda (LLA) é o uso da técnica de esfregaço de sangue periférico, para que operadores especializados (hematologistas) determinem a quantidade de linfoblastos presentes no sangue pelo microscópio (MSD, 2020).

O processo de contagem dos linfoblastos em uma imagem é uma tarefa que demanda muito tempo de um profissional que poderia ser mais produtivo se ocupando em outros assuntos. Além disso, os resultados dependem da experiência subjetiva do operador e do seu estado de cansaço e, por causa disso, dois hematologistas podem fornecer resultados diferentes para um mesmo caso.

Fazer a contagem de linfoblastos de forma automática com um modelo computacional pode ajudar a realizar o processo de diagnóstico da LLA de uma forma mais eficiente, pois diminuiria o tempo requerido do especialista. Além disso, também não seria necessário um equipamento custoso, pois uma ferramenta como esta pode ser executada na nuvem atendendo várias solicitações de forma simultânea.

Trabalhos recentes que abordam este assunto têm sido realizados utilizando diversas técnicas. O artigo "Computer-Aided Diagnosis of Acute Lymphoblastic Leukaemia" (Shafique and Tehsin, 2018) apresenta uma revisão de várias metodologias para diagnóstico automático de LLA. Especificamente, a Tabela 2 do artigo mostra uma comparação sistemática de vários métodos de diagnóstico de LLA. Cada método com suas próprias técnicas de: pré-processamento, segmentação, extração de características e classificação. Nesta dissertação, apresentamos um modelo computacional cujas etapas de pré-processamento e segmentação foram baseadas no artigo "Robust Segmentation and Measurements Techniques of White Cells in Blood Microscope Images" (Scotti, 2006). A extração de características e classificação foram realizadas mediante uma rede neural convolucional clássica.

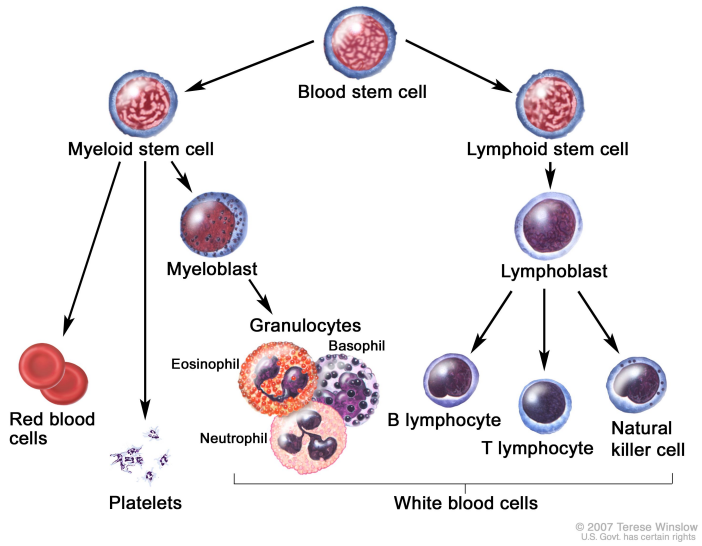
Este trabalho começa com uma explicação e justificação do problema a resolver, além de definir os objetivos e descrever brevemente a abordagem considerada para tratar o problema. Posteriormente, fazemos uma revisão dos fundamentos teóricos utilizados ao longo da dissertação. Em seguida, abordamos de forma detalhada a metodologia proposta, explicando passo por passo cada etapa do processo. Depois, definimos as métricas de avaliação a utilizar, mostramos o desempenho do modelo e apresentamos alguns exemplos das suas saídas. Finalmente, apresentamos os aspectos positivos e negativos do modelo e a conclusão final do trabalho.

1 Preliminares

1.1 Leucemia Linfoblástica Aguda (LLA)

Segundo o "National Cancer Institute" (NCI, 2021a) dos Estados Unidos, a LLA é um tipo de leucemia (câncer no sangue) que aparece e cresce rapidamente, e é caracterizada pela presença de muitos linfoblastos no sangue e também na medula óssea. Um linfoblasto neste contexto é uma célula imatura que pode se converter em um linfócito (NCI, 2021b). Na Figura 1 podemos ver em que parte do desenvolvimento das células sanguíneas aparecem este tipo de célula.

Figura 1 – Desenvolvimento de células sanguíneas. Uma célula tronco do sangue passa por várias etapas para se tornar um glóbulo vermelho, plaqueta ou glóbulo branco.



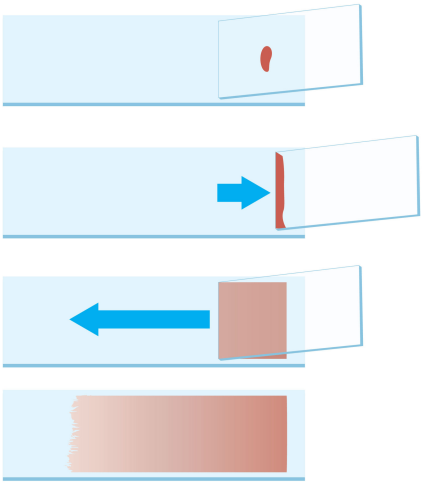
Fonte: NCI (2021b).

O "National Health Service" do Reino Unido diz que todas as células sanguíneas do corpo são produzidas pela medula óssea que produz células tronco com a capacidade de virar células vermelhas (que transportam oxigênio pelo corpo), células brancas (que ajudam a combater infecções) e as plaquetas (que ajudam na coagulação do sangue) como mostra a Figura 1. A medula óssea geralmente não libera células tronco no sangue até elas se encontrarem completamente desenvolvidas mas, na LLA, muitas células brancas são liberadas antes de estarem maduras (linfoblastos) (NHS, 2021a). Por causa disso, é importante a observação da quantidade de linfoblastos presentes no sangue como parte do processo de diagnóstico da doença.

1.2 Diagnóstico da LLA

Exitem diversos métodos utilizados para o diagnóstico da LLA que podem ser encontrados na literatura (NHS, 2021b). Neste trabalho vamos focar em um deles: a técnica de esfregaço de sangue periférico. O esfregaço de sangue consiste em colocar uma gota de sangue sobre uma lâmina de vidro espalhando-a em uma camada fina pela superfície com a ajuda de uma lâmina histológica, com o objetivo de produzir uma monocamada de células (ver Figura 2). Posteriormente realiza-se um processo de coloração para tingir as células de interesse (no nosso caso, as células brancas) com uma cor azulada como vemos na Figura 3. Kasvi (2021) descreve bem todos os detalhes da técnica do esfregaço.

Figura 2 – Técnica de esfregaço de sangue periférico.



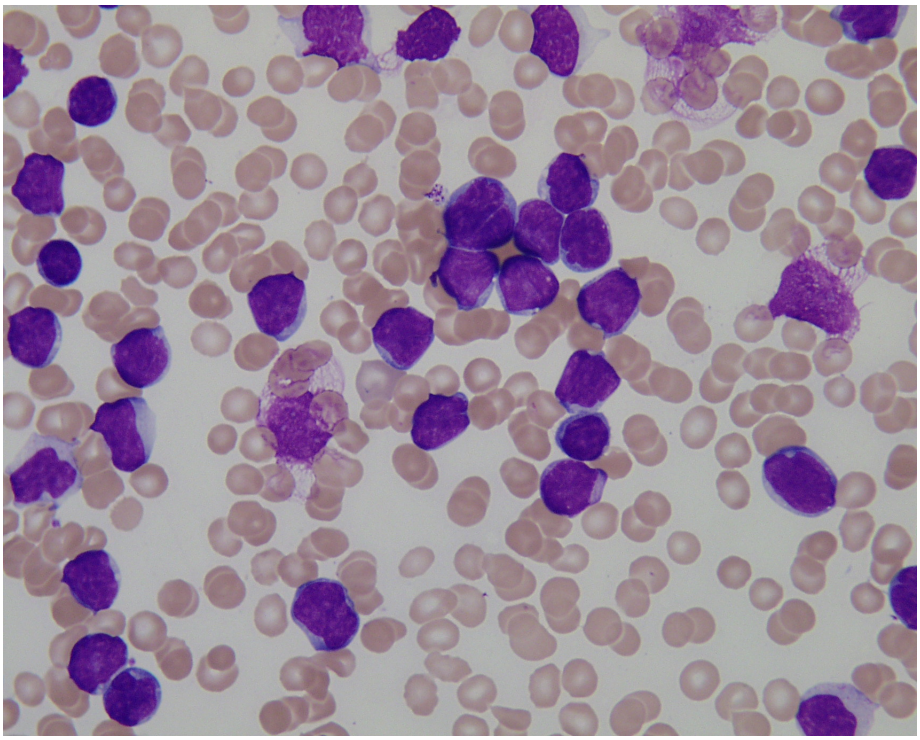
Fonte: Kasvi (2021).

A lâmina com o esfregaço de sangue pode ser analisada mediante um microscópio pelo qual o hematologista poderá visualizar uma imagem parecida com a Figura 3 que lhe permitirá realizar a contagem de linfoblastos. Esse tipo de imagem é aquele que nosso modelo computacional analisará. Segundo o artigo "Acute lymphoblastic leukemia: a comprehensive review and 2017 update", o diagnóstico de LLA pode ser estabelecido pela presença do 20% ou mais linfoblastos na medula óssea ou no sangue periférico (T Terwilliger, 2017).

1.3 Desvantagem da contagem manual de linfoblastos

A contagem manual dos linfoblastos no microscópio é uma tarefa que demanda tempo de um profissional que poderia ser mais produtivo se utilizado em outros assuntos. Com efeito, esse tempo implica um custo econômico, pois um especialista tem um valor

Figura 3 – Imagem por microscópio da lâmina resultante da aplicação da técnica de esfregaço de sangue periférico.



Fonte: Ruggero Donida Labati (2011).

significante no mercado de trabalho. Além disso, os resultados da análise podem ser afetados por fatores humanos como o cansaço, o estresse, etc. A experiência do operador também tem um papel importante e, portanto, existe uma componente de subjetividade afetando os resultados da contagem (Shafique and Tehsin, 2018).

1.4 Objetivos

Com o explicado até o momento, o problema a resolver é a contagem automática de linfoblastos em uma imagem de esfregaço de sangue. Para tratar o problema, foram definidos os seguintes objetivos:

- **Objetivo geral:** Propor um algoritmo computacional para realizar diagnósticos de LLA mais rapidamente.
- **Objetivo específico:** Desenvolver um programa computacional com software livre que, utilizando diversas técnicas de processamento digital de imagens e um modelo de aprendizado de máquinas, recebe uma imagem colorida obtida por microscópio como entrada e devolve a porcentagem, o número de linfoblastos encontrados e a mesma imagem com anotações nessas células.

1.5 Abordagem do problema

O problema da contagem de linfoblastos foi dividido em duas etapas principais: uma de segmentação, que consiste em conhecer exatamente quais pixels da imagem correspondem a células brancas; e outra de classificação, cuja tarefa é nos dizer se cada célula identificada é um linfoblasto ou uma célula saudável. Mais detalhes podem ser vistos na seção 3.1.

A maior parte das ideias da primeira etapa do problema foram tomadas do artigo "Robust Segmentation and Measurements Techniques of White Cells in Blood Microscope Images" (Scotti, 2006). Porém, o artigo não aborda algumas problemáticas que são discutidas no Capítulo 5, e não é muito claro nos detalhes de algumas das técnicas utilizadas. Portanto, apesar de nos ter fornecido uma base para o nosso desenvolvimento, foram necessárias algumas outras implementações, como o uso de morfologia matemática e o algoritmo watershed, para conseguir melhores resultados. Os detalhes específicos desta etapa serão tratados na seção 3.2.

Para o desenvolvimento da segunda etapa, a ideia inicial foi reproduzir o artigo "Morphological Classification of Blood Leucocytes by Microscope Images", onde os autores propõem fazer a classificação das células brancas mediante uma extração de características baseada principalmente na geometria das células, calculando variáveis como: área, perímetro, convexidade, solidez, etc; usando *Matlab Image Processing Toolbox*. Com as variáveis medidas para o núcleo e citoplasma, fizeram uma rede neural progressiva para obter o classificador desejado (Scotti, 2004). É importante dizer que, embora esta tenha sido a ideia a ser implementada no começo, decidiu-se que esta etapa seria executada por uma rede neural convolucional clássica que possui suas camadas de extração de características (camadas convolucionais) e suas camadas de classificação (perceptron multicamadas) e cujos detalhes serão apresentados na subseção 3.3.1.

2 Fundamentação Teórica

Neste capítulo revisaremos brevemente algumas definições e resultados fundamentais que foram utilizados no presente trabalho.

2.1 Processamento Digital de Imagens

2.1.1 Representações matemáticas das imagens digitais

Definição 1 (Imagem Digital)**.** *Sejam* $X = \{0, 1, 2, ..., M-1\}$ *e* $Y = \{0, 1, 2, ..., N-1\}$ *com* $M, N \in \mathbb{N}$*, dizemos que* $I : X \times Y \to Z$ *é uma imagem digital de dimensões* $M \times N$*:*

- *Binária, se* $Z = \{0, 255\}$.
- *Em tons de cinza, se* $Z = \{0, 1, ..., 255\}$ *ou* $Z = [0, 1]$.
- *Colorida, se* $Z \subset \mathbb{R}^3$.

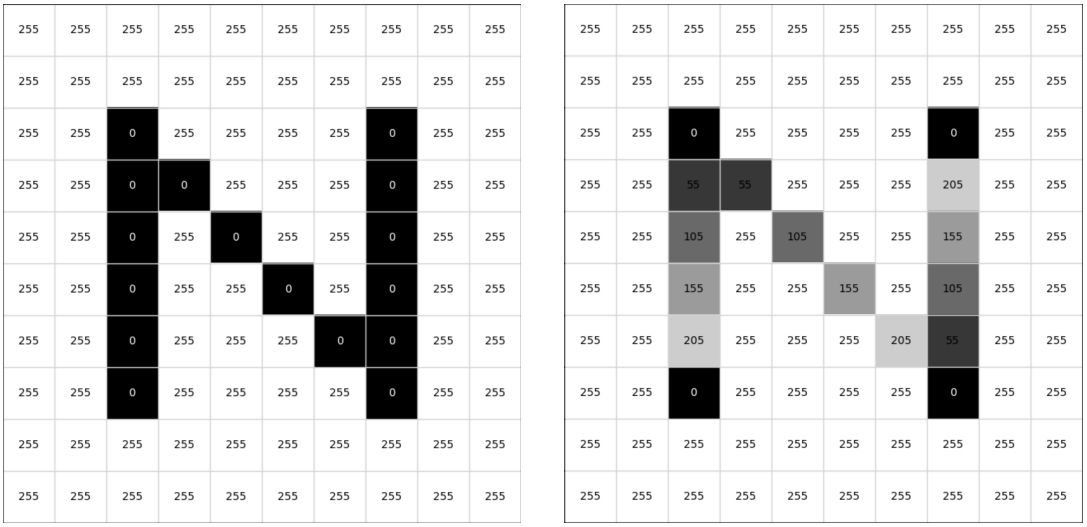
Se $(x, y) \in X \times Y$*, a terna* $(x, y, I(x, y))$ *é denominada* ***pixel****, na qual* (x, y) *e* $I(x, y)$ *são a posição e o valor do pixel respectivamente.*

Note que uma imagem digital pode ser expressa da seguinte forma:

$$Imagem = \begin{pmatrix} \text{I(0,0)} & \text{I(0,1)} & ... & \text{I(0,N-1)} \\ \text{I(1,0)} & \text{I(1,1)} & ... & \text{I(1,N-1)} \\ . & . & ... & . \\ . & . & ... & . \\ \text{I(M-1,0)} & \text{I(M-1,1)} & ... & \text{I(M-1,N-1)} \end{pmatrix}. \tag{2.1}$$

Se a imagem é colorida, ela pode ser expressa por três matrizes chamadas **canais da imagem**. Se a imagem é binária ou em tons de cinza dizemos que possui apenas um canal. A Figura 4 mostra exemplos da composição matricial de duas imagens, uma binaria e outra em tons de cinza. Por outro lado, a Figura 5 mostra a forma em que está composta uma imagem colorida no espaço RGB, onde podemos ver como cada canal em tons de cinza intervém para gerar as cores da imagem. Vale dizer que apesar de ter usado valores entre 0 e 255 para a intensidade dos pixels nesta definição, eles também podem ter valores entre 0 e 1.

Figura 4 – Exemplos de imagens digitais de um canal. De esquerda a direita vemos: imagem binária, imagem em tons de cinza.



Fontes: o autor.

2.1.2 Espaços de cores

Um espaço de cores é um mapeamento que atribui a cada cor uma coordenada. Dado que os humanos têm visão de cores tricromática (células cônicas que têm sensibilidades máximas nos comprimentos de onda de vermelho, verde e azul), essas coordenadas são geralmente de três dimensões. Existem vários espaços de cores, mas apenas falaremos dos que foram utilizados neste trabalho, os espaços RGB e CIELAB. As ideias explicadas nesta subseção foram tomadas de Weller (2018).

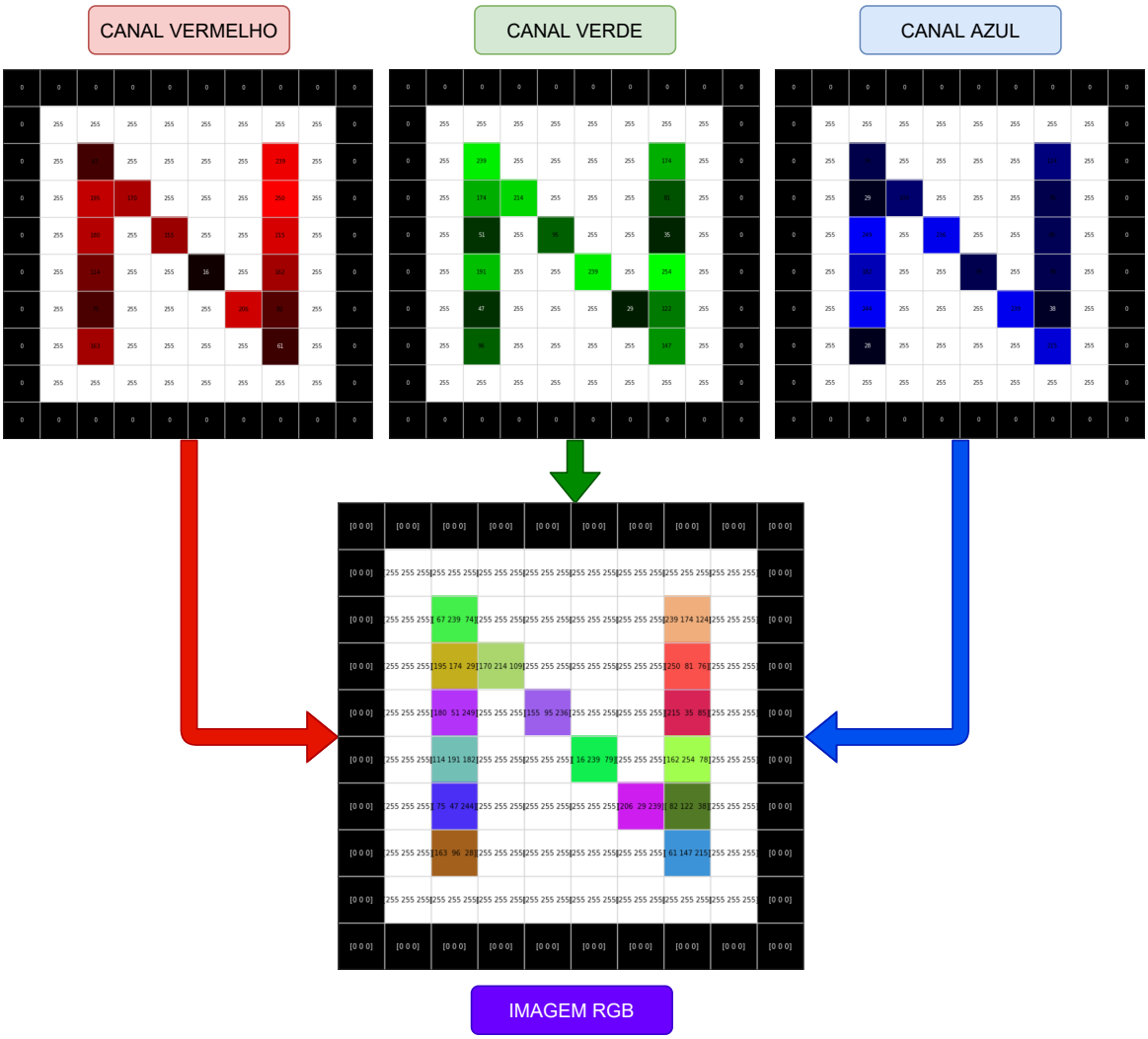
2.1.2.1 Espaço RGB

O espaço RGB é um dos espaços de cores mais utilizados em processamento de imagens coloridas (Rafael C. Gonzalez, 2008). No espaço RGB, cada cor é representada por três números reais correspondendo às intensidades de vermelho (R), verde (G), e azul (B). Este é o sistema que a maioria dos computadores usam para guardar e mostrar imagens coloridas. Por conveniência, supomos que todos os valores de cor foram quantizados entre 0 e 1 de modo que o cubo mostrado na Figura 6 é o cubo unitário, ou seja, todos os valores de R, G e B são considerados no intervalo $[0, 1]$. Note que as ternas $(1, 0, 0), (0, 1, 0)$ e $(0, 0, 1)$ representam as cores vermelho, verde e azul, respectivamente.

2.1.2.2 Espaço Lab

Lab é uma abreviação para os espaços CIELAB ou Hunter Lab. O espaço CIELAB foi definido pela Comissão Internacional de Iluminação ou CIE. Este é o espaço mais robusto e complexo para realizar comparações numéricas. O espaço Lab é percep-

Figura 5 – Exemplo de imagem digital de três canais. Esta é a composição de uma imagem digital no espaço RGB.



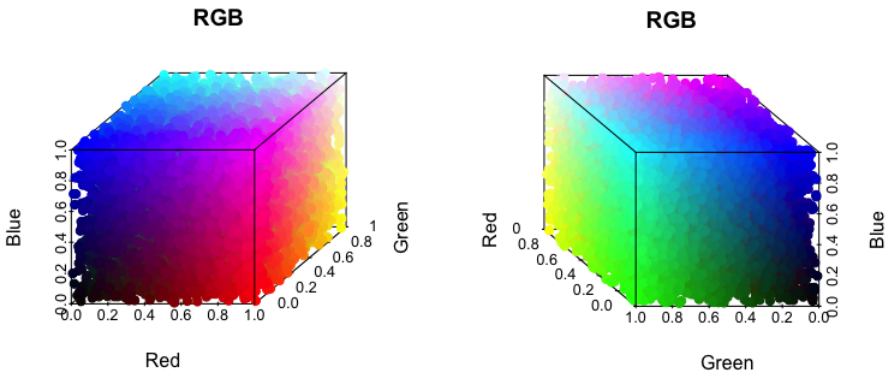
Fontes: o autor.

tualmente uniforme (Rafael C. Gonzalez, 2008), isto é, a distância euclideana no espaço CIELAB corresponde, empiricamente, à distância percebida das cores por humanos. O espaço CIELAB possui três canais: L (Iluminação, de preto a branco), a (de verde a vermelho) e b (de azul a amarelo). A Figura 7 mostra os pontos deste espaço colocados num cubo de $100 \times 100 \times 100$ unidades. Note que uma imagem colorida no espaço CIELAB é representada como uma função $I : X \times Y \longrightarrow Z$, em que $Z = [0, 100] \times \mathbb{R} \times \mathbb{R}$.

2.1.3 Convolução discreta bidimensional

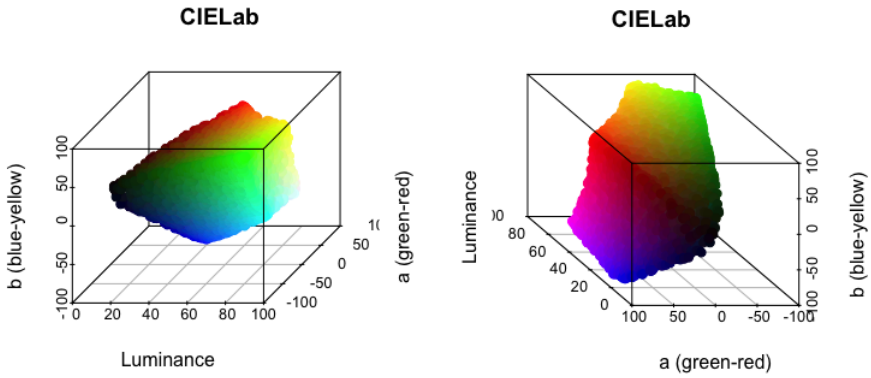
O livro "Digital Image Processing - third edition" [pag. 249,250] (Rafael C. Gonzalez, 2008) apresenta a definição da convolução discreta bidimensional de uma função f

Figura 6 – Espaço RGB representado em um cubo de volume 1.



Fonte: Weller (2018).

Figura 7 – Representação gráfica do espaço CIELAB.



Fonte: Weller (2018).

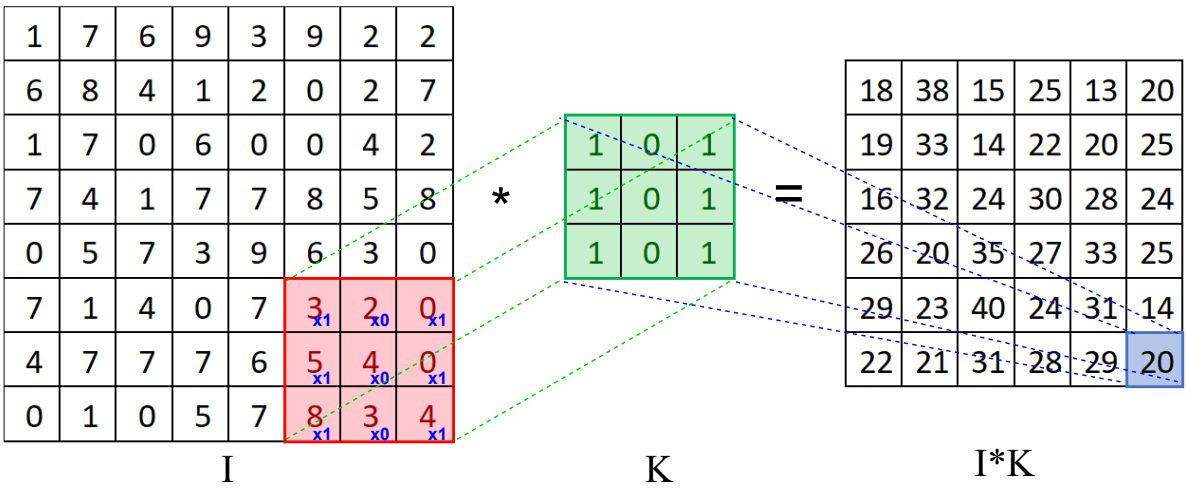
com outra função K como:

$$(I * K)(x, y) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I(m, n)K(x - m, y - n), \tag{2.2}$$

para $x = 0, 1, 2, ..., M - 1$ e $y = 0, 1, 2, ..., N - 1$. Em termos práticos, esta operação matemática é utilizada para aplicar **filtros** a um canal de uma imagem digital. Na Equação 2.2, I representa um canal da imagem digital e K é chamada de **máscara ou kernel** de convolução. Graficamente, a operação de convolução percorre o kernel sobre o canal da imagem implementando somas ponderadas como se mostra na Figura 8. O tipo de filtro que será aplicado sobre a imagem é determinado pela máscara. Assim, existem máscaras para detectar bordas, desfocar, remover ruídos, entre outras aplicações.

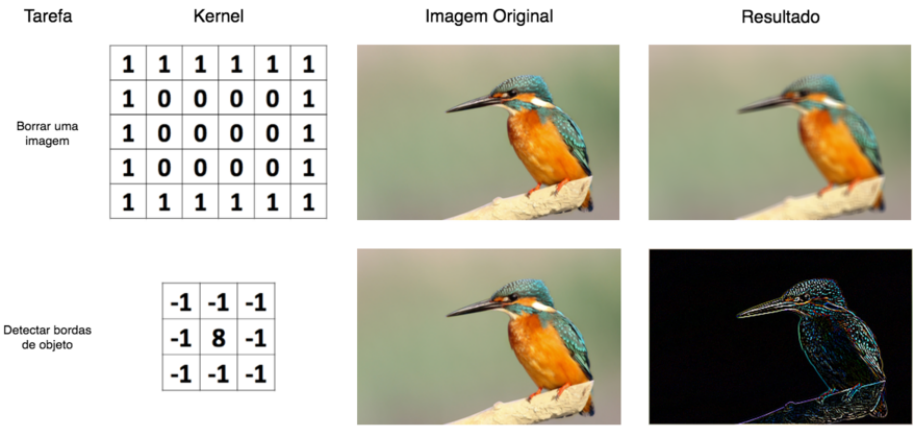
Na Figura 9 podemos ver o efeito que produz a convolução com dois kernels diferentes aplicados em todos os canais de uma imagem colorida.

Figura 8 – Operação de convolução sobre um canal de uma imagem digital. I é o canal da imagem e K é o Kernel de convolução.



Fonte: o autor.

Figura 9 – Efeitos de filtragens feitas sobre uma imagem colorida (em seus três canais).



Fonte: Elo7 (2018).

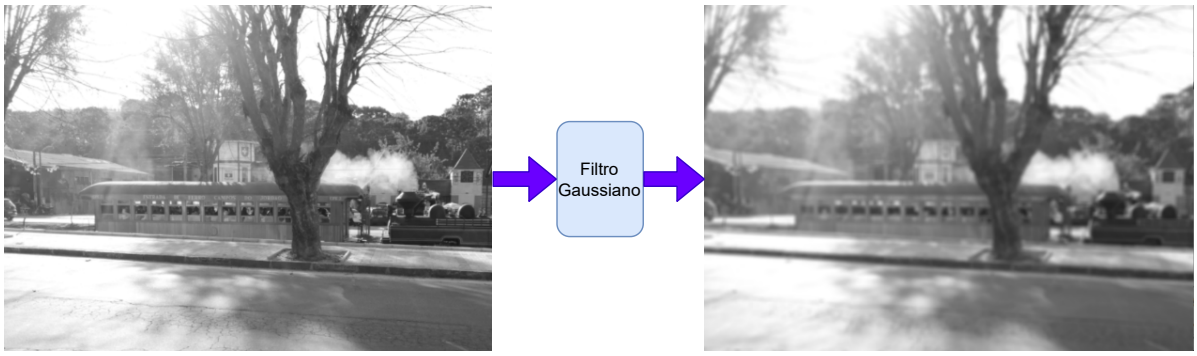
2.1.4 Filtro Gaussiano Bidimensional

O filtro Gaussiano Bidimensional é um filtro cuja máscara de convolução está dada pela discretização da função:

$$K(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}.$$

Este filtro é utilizado para borrar imagens, tal como ilustrado na Figura 10. A escolha do tamanho do kernel e o valor do parâmetro σ dependem do nível de borrão que desejamos ter e da resolução da imagem.

Figura 10 – Efeito do filtro Gaussiano em uma imagem.



Fonte: o autor.

2.1.5 Histograma de uma imagem

Quando falamos de histograma de uma imagem, referimos-nos à frequência das intensidades de pixels presentes em uma imagem em tons de cinza. Uma imagem colorida no espaço RGB terá três histogramas, um para cada canal. O histograma nos fornece principalmente informação de iluminação e contraste, como ilustrado nas Figuras 11 e 12. Esta informação pode ser utilizada para melhorar a imagem mediante um ajuste de contraste (subseção 2.1.6), escolher um limiar de binarização (subseção 2.1.7), entre outras coisas.

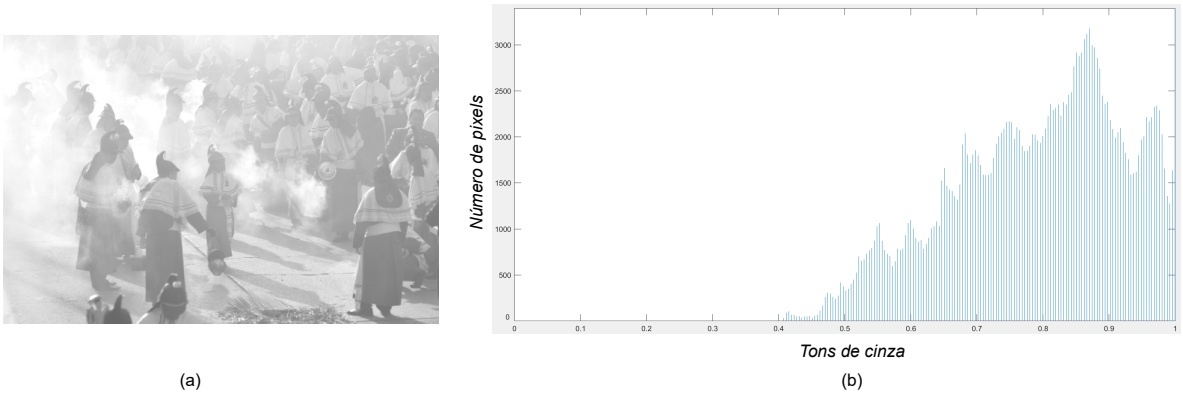
2.1.6 Ajuste de contraste

Na Figura 12 vemos que um alto contraste em uma imagem em tons de cinza é evidenciado em um histograma de imagem com uma distribuição com tendencia uniforme. Se quisermos realçar o contraste de uma imagem como a que vemos na Figura 11, então podemos uniformizar o histograma da imagem em questão (Figura 13). Existem diferentes métodos para realizar esta tarefa, mas neste trabalho utilizamos o ajuste por máximos e mínimos que matematicamente se define como:

$$I_{ajustada}(x,y) = 255\left(\frac{I(x,y) - \min_{(u,v)\in X\times Y} I(u,v)}{\max_{(u,v)\in X\times Y} I(u,v) - \min_{(u,v)\in X\times Y} I(u,v)}\right),\ \forall (x,y)\in X\times Y, \tag{2.3}$$

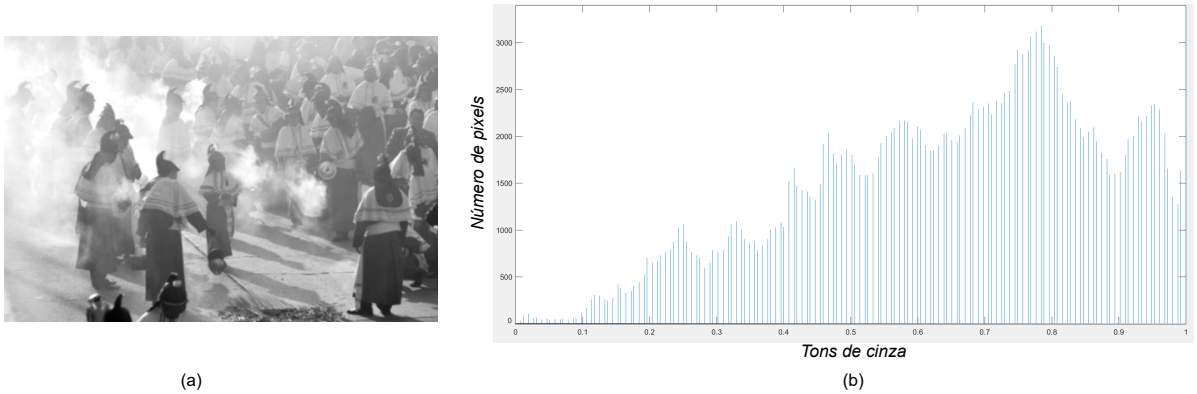
onde I é a imagem a tratar e $I_{ajustada}$ é o resultado do ajuste. Note que a imagem mostrada na Figura 12 é a versão com contraste ajustado da imagem da Figura 11 mediante a Equação 2.3.

Figura 11 – No histograma (b), vemos uma distribuição de pixels concentrada em uma faixa de intensidades. Isso nos diz que trata-se de uma imagem de baixo contraste (a).



Fonte: o autor

Figura 12 – No histograma (b), vemos uma distribuição de pixels que tende a ser uniforme ao longo do eixo das intensidades. Isso nos diz que trata-se de uma imagem de alto contraste (a).



Fonte: o autor.

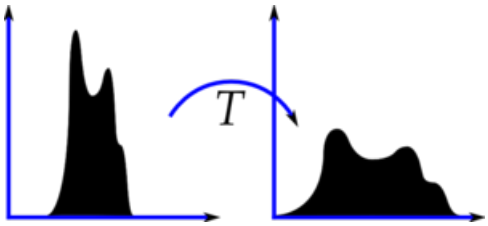
2.1.7 Binarização

A binarização é um método utilizado para converter uma imagem em tons de cinza em uma binária. De uma forma simples, podemos binarizar uma imagem em tons de cinza I através da equação:

$$I_{binária}(x,y) = \begin{cases} 1, & I(x,y) \geqslant T, \\ 0, & I(x,y) < T, \end{cases} \quad \forall (x,y) \in X \times Y,$$

onde I é a imagem original, T é um parâmetro denominado **limiar** (conhecido em inglês como *threshold*) e $I_{binária}$ é o resultado da binarização. A Figura 14 (a) mostra uma imagem em tons de cinza que posteriormente é binarizada, resultando na imagem Figura 14 (c). Na

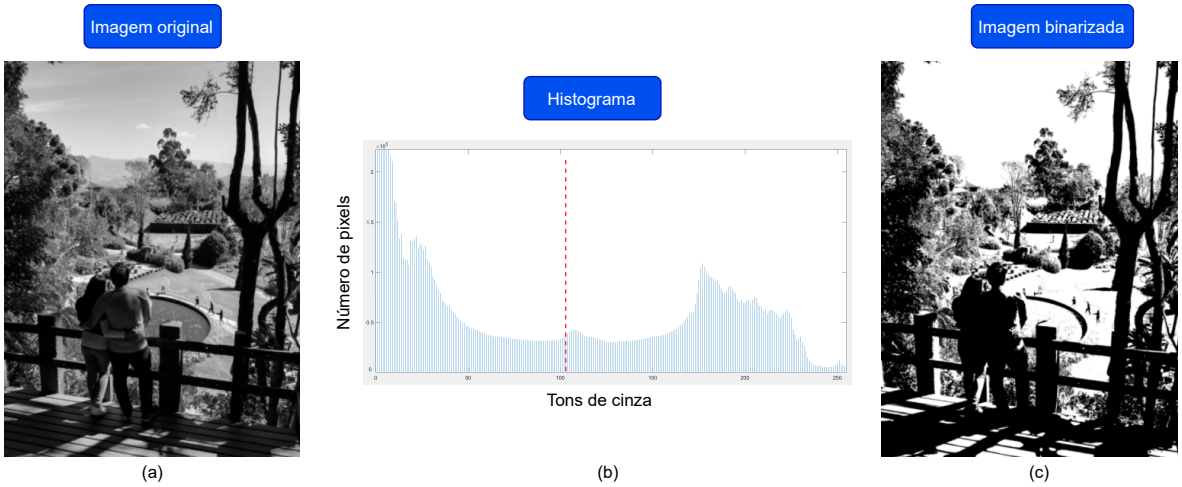
Figura 13 – Ilustração do objetivo do ajuste de contraste em uma imagem (uniformizar o histograma).



Fonte: OpenCV (2015).

Figura 14 (b), vemos marcado com uma linha vermelha o limiar de binarização escolhido no histograma da imagem. A escolha do limiar é o mais sensível da binarização, pois uma má escolha vai nos fornecer uma má binarização, como ilustrado na Figura 15, onde obtivemos uma imagem escura e com pouca informação da imagem original. Por causa disso, na literatura, foram desenvolvidas algumas técnicas para encontrar o limiar ótimo. Neste trabalho, foram usadas apenas duas técnicas de binarização: a binarização de Otsu, útil para imagens bimodais (OTSU, 1979); e o algoritmo do triângulo, útil para imagens unimodais (G. W. Zack, 1977).

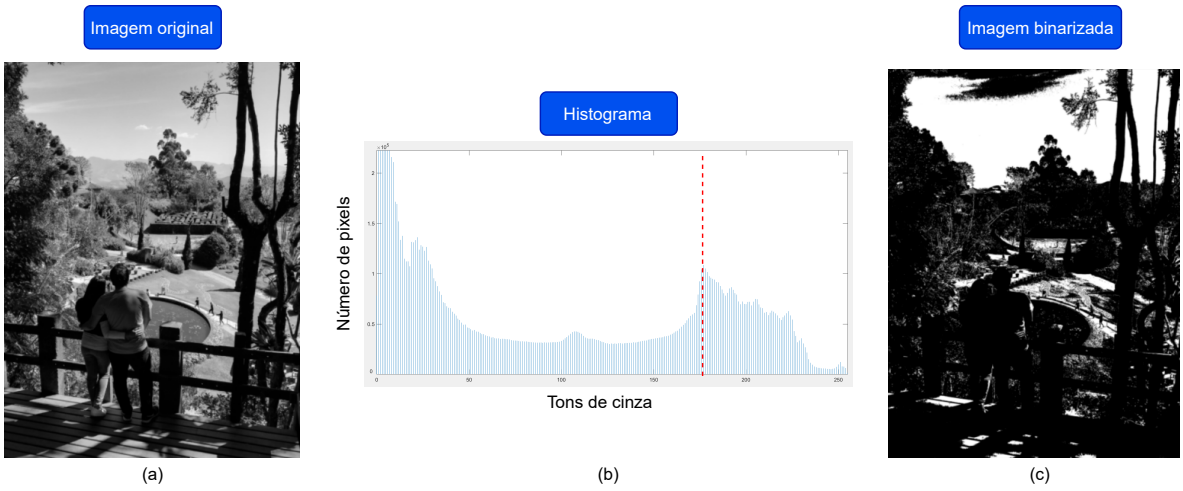
Figura 14 – Efeito da binarização. O limiar (linha vermelha) apresentado no histograma da imagem original foi calculado com o método de Otsu.



Fonte: o autor.

Na Figura 16 (a) vemos uma imagem de abacates sobrepostos. Note que na imagem podemos visualizar apenas um tipo de objeto (abacates) e não existe fundo. Este fato faz que, empiricamente, binarizar a imagem da Figura 16 (a) não seja uma tarefa fácil, pois apenas temos um objeto claramente identificável (os abacates). O explicado

Figura 15 – Efeito da binarização. O limiar (linha vermelha) apresentado no histograma da imagem original foi escolhido aleatoriamente.



Fonte: o autor.

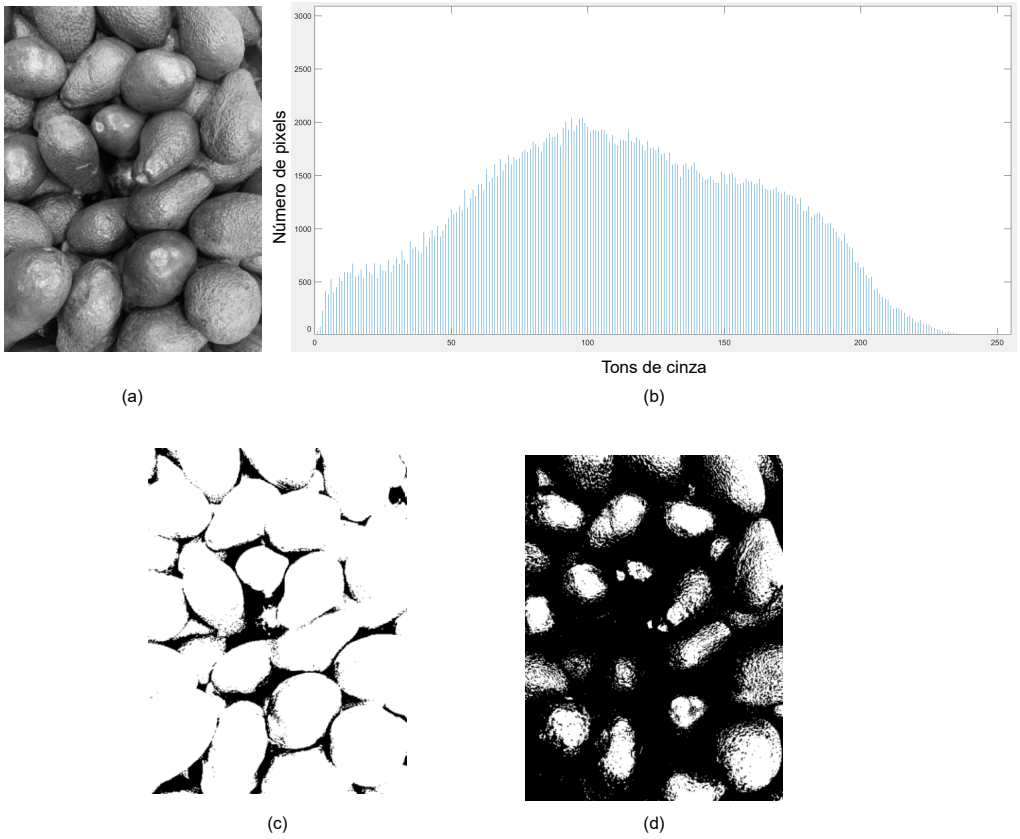
anteriormente pode ser visto na Figura 16 (b), onde vemos que as intensidades estão quase uniformemente distribuídas e, portanto, não é possível encontrar um bom limiar de binarização. As Figuras 16 (c) e 16 (d) mostram os maus resultados obtidos com dois limiares distintos ao tentar binarizar a Figura 16 (a). Um exemplo de uma imagem simples de binarizar é a mostrada na Figura 17, onde empiricamente é fácil distinguir o abacate do fundo. Ao ter um componente facilmente identificável, seu histograma é unimodal (um pico) e, portanto, não é uma tarefa difícil encontrar um limiar que nos permita separar o abacate do fundo. Vale dizer que, às vezes uma imagem pode conter ruído que provoque uma binarização também ruidosa, mesmo quando trata-se de uma imagem unimodal. Nesses casos é necessário preprocessar a imagem. Neste trabalho utilizamos o filtro Gaussiano e o ajuste de contraste como técnicas de preprocessamento.

2.1.8 Morfologia Matemática

A morfologia matemática é uma teoria não linear de transformações de imagens baseada na teoria de reticulados completos, geometria e conceitos topológicos. É particularmente útil para a análise de estruturas geométricas em uma imagem (Heijmans, 1995). Existem diversos operadores morfológicos que nos permitem realizar transformações em uma imagem em tons de cinza ou binária. Porém, neste trabalho, iremos apresentar apenas alguns operadores básicos utilizadas sobre imagens binárias.

Na morfologia matemática binária usamos a teoria de conjuntos para descrever os operadores. Uma imagem binária está composta pelos pixels dos objetos e o fundo. Chamaremos X ao conjunto formado pelas coordenadas dos pixels dos objetos da imagem

Figura 16 – Binarização complicada por causa de um histograma muito uniforme.



Fonte: o autor

binária. Note que X é uma completa descrição morfológica da imagem. Cada elemento de X é um par ordenado $(x, y) \in \mathbb{Z}^2$ cuja origem do sistema de coordenadas está localizado no canto superior esquerda da imagem (Rafael C. Gonzalez, 2008)[pag. 628].

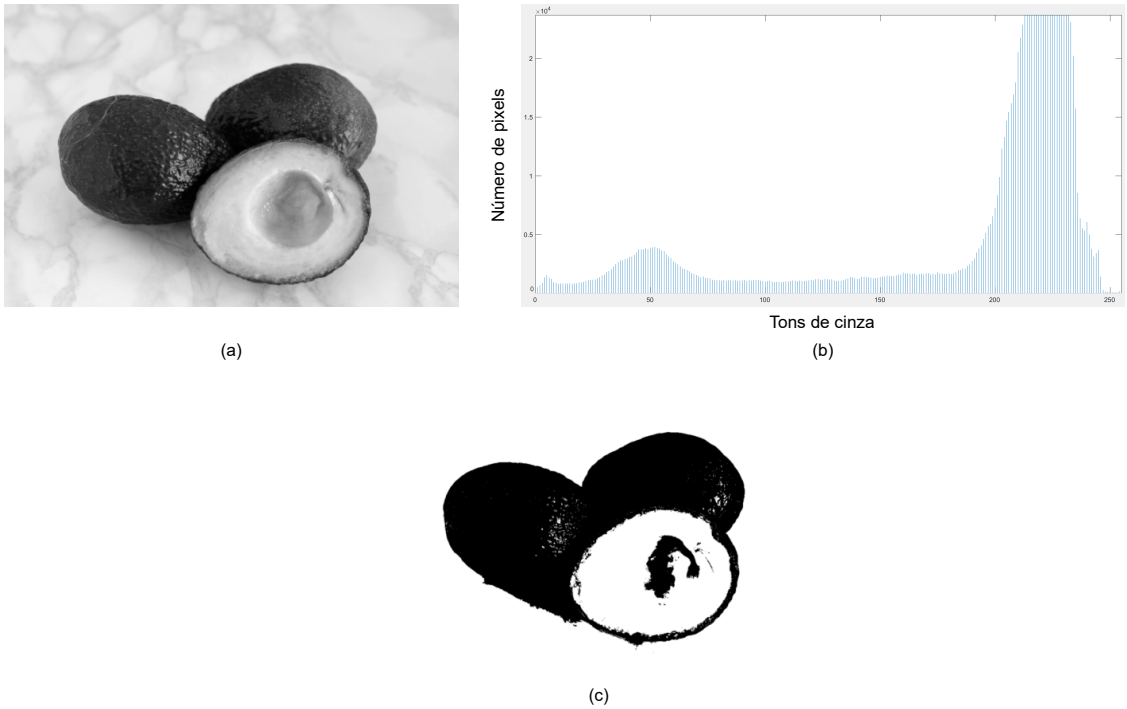
Os operadores morfológicos geralmente utilizam um subconjunto de $\mathbb{Z}^2$ chamado **elemento estruturante** (B), que é usado para extrair informações relevantes da imagem. Na Figura 18 vemos exemplos de elementos estruturantes com um ponto preto bem marcado indicando um sistema de referência, isto é, a localização da coordenada $(0, 0)$. Neste trabalho faremos referência a elementos estruturantes como os que aparecem na Figura 18. Vale dizer que, nos sistemas de coordenadas de X e B, as abscisas positivas estão na direita da origem e as ordenadas positivas para abaixo dele.

2.1.8.1 Deslocamento

O deslocamento de um conjunto $X \subset \mathbb{Z}^2$ pelo ponto $h = (h_1, h_2)$, denotado por X_h, é definido como:

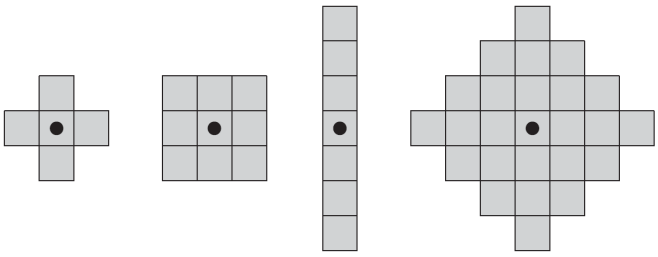
$$X_h = \{c \mid c = x + h, \ para \ x \in X\}.$$

Figura 17 – Binarização fácil por causa de um histograma unimodal.



Fonte: o autor.

Figura 18 – Exemplos de elementos estruturantes.



Fonte: Rafael C. Gonzalez (2008)[pag. 629].

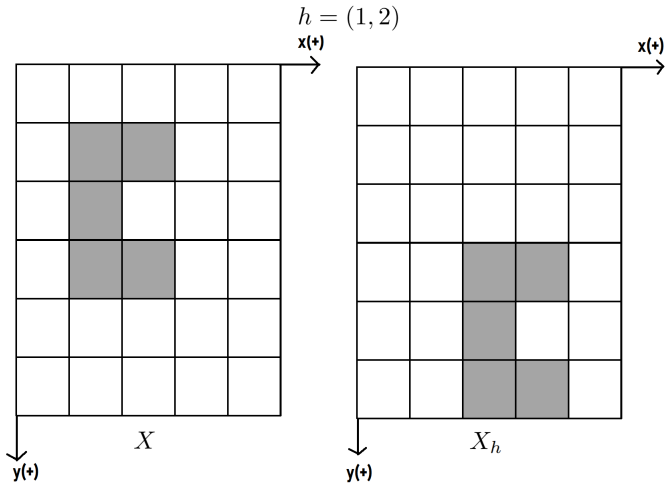
Na Figura 19 vemos um exemplo deste operador. Note que a direção de deslocamento está dada pela mesma convenção do plano cartesiano, isto é, h_1 positivo implica deslocamento à direita e h_2 positivo significa deslocamento para abaixo.

2.1.8.2 Erosão

A erosão de $X \subset \mathbb{Z}^2$ por $B \subset \mathbb{Z}^2$, denotada por $X \ominus B$, é definida como:

$$X \ominus B = \{z \mid B_z \subset X\}.$$

Figura 19 – Exemplo do deslocamento em imagens binárias.



Fonte: o autor.

Equivalentemente tem-se (Soille, 1999):

$$X \ominus B = \bigcap_{b \in B} X_{-b} \ .$$

Na Figura 20 vemos um exemplo da aplicação de uma das definições matemáticas deste operador, enquanto na Figura 21 podemos observar o efeito prático da erosão em uma imagem binária.

2.1.8.3 Dilatação

A dilatação de $X \subset \mathbb{Z}^2$ por $B \subset \mathbb{Z}^2$, denotada por $X \oplus B$, é definida como:

$$X \oplus B = \bigcup_{b \in B} X_b \ .$$

Na Figura 22 vemos um exemplo da aplicação de uma das definições matemáticas deste operador, enquanto na Figura 23 podemos observar o efeito prático da dilatação.

2.1.8.4 Abertura e Fechamento

A abertura de um conjunto X por um elemento estruturante B, denotada por $X \circ B$, é definida como:

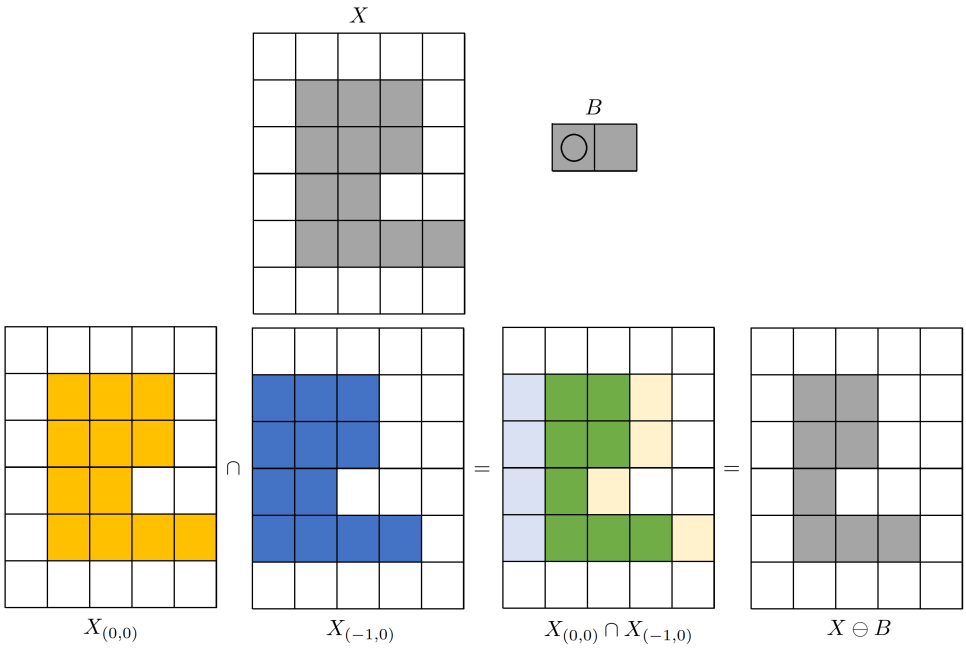
$$X \circ B = (X \ominus B) \oplus B.$$

O fechamento de um conjunto X por um elemento estruturante B, denotado por $X \bullet B$, é definido como:

$$X \bullet B = (X \oplus B) \ominus B.$$

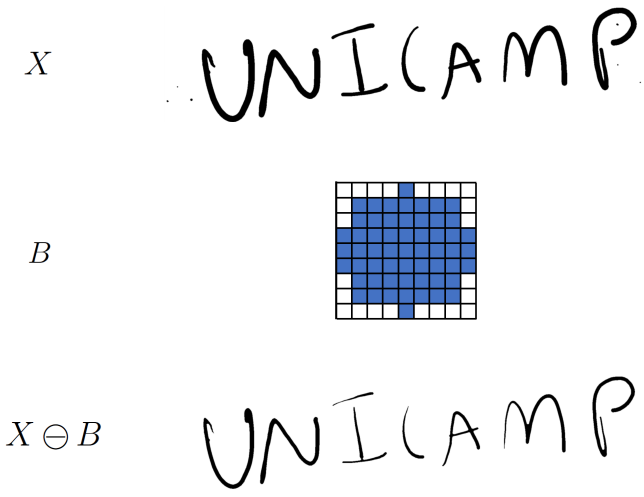
Na Figura 24, podemos ver que a abertura tem um efeito de eliminação de ruído e suavização dos contornos da imagem mediante a remoção de detalhes que fiquem dentro

Figura 20 – Exemplo do operador erosão em imagens binárias.



Fonte: o autor.

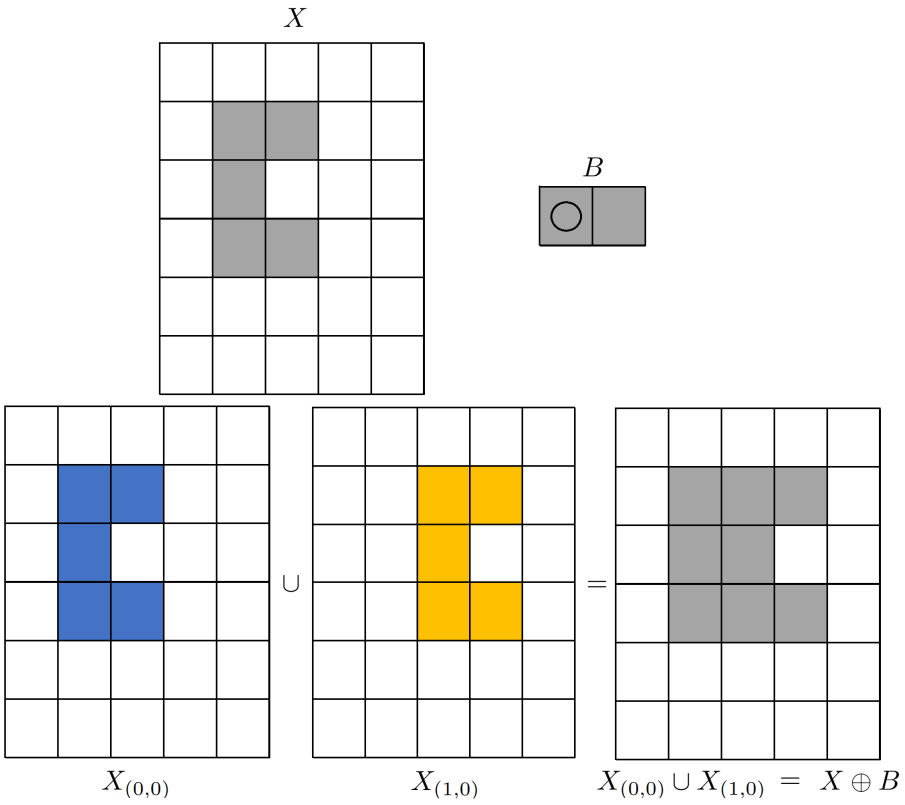
Figura 21 – Efeito da erosão em uma imagem binária.



Fonte: o autor.

do elemento estruturante. Por outro lado, o fechamento permite definir melhor os objetos presentes (no exemplo, os objetos são as letras e os pontos) o "preenchimento" dos espaços do fundo que fiquem dentro do elemento estruturante.

Figura 22 – Exemplo do operador dilatação em imagens binárias.



Fonte: o autor.

2.1.8.5 Abertura por área

A abertura por área é uma operação morfológica que permite eliminar objetos de uma imagem, que possuam uma área (medida em pixels) inferior a um limiar estabelecido. Sejam $(X_i)_{i \in M}$ os componentes conexos da imagem X, e seja X' a imagem que resulta ao se aplicar a abertura por área na imagem X com um limiar λ, segundo Vincent (1993) tem-se:

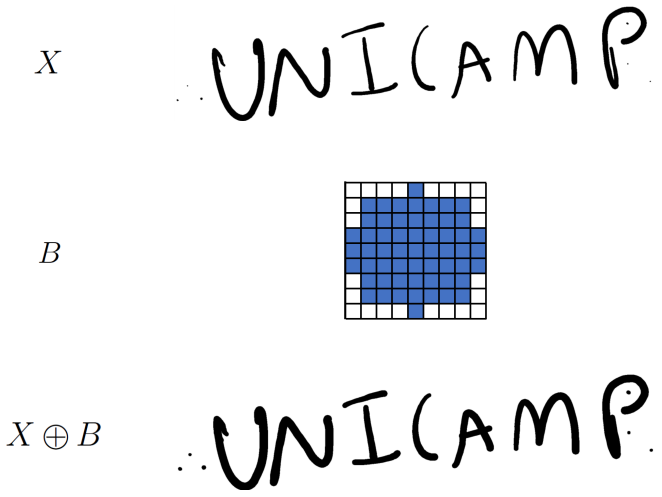
$$X' = \bigcup \{X_i \mid i \in M,\ Area(X_i) \geqslant \lambda\}$$

Na Figura 25 vemos um exemplo da aplicação da abertura por área em uma imagem binária. Note o efeito de eliminação de objetos pequenos na imagem.

2.1.9 Contorno de objetos em imagens binárias

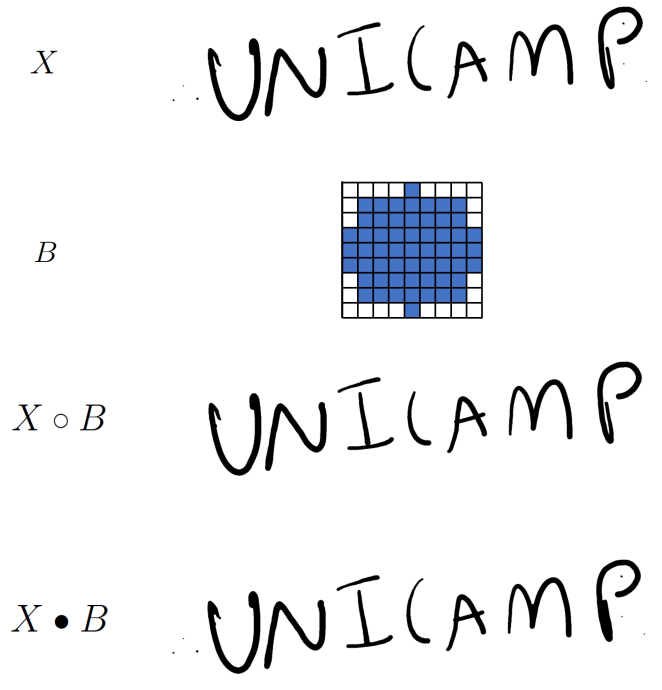
Encontrar o contorno dos objetos em uma imagem binária é uma tarefa muito útil porque fornece informação da localização desses objetos dentro da imagem. Neste trabalho, utilizamos a função findContours da biblioteca OpenCV em python para encontrar os contornos das células brancas. Os detalhes do algoritmo dessa implementação podem ser encontrados no artigo “Topological Structural Analysis of Digitized Binary Images by

Figura 23 – Efeito da dilatação em uma imagem binária.



Fonte: o autor.

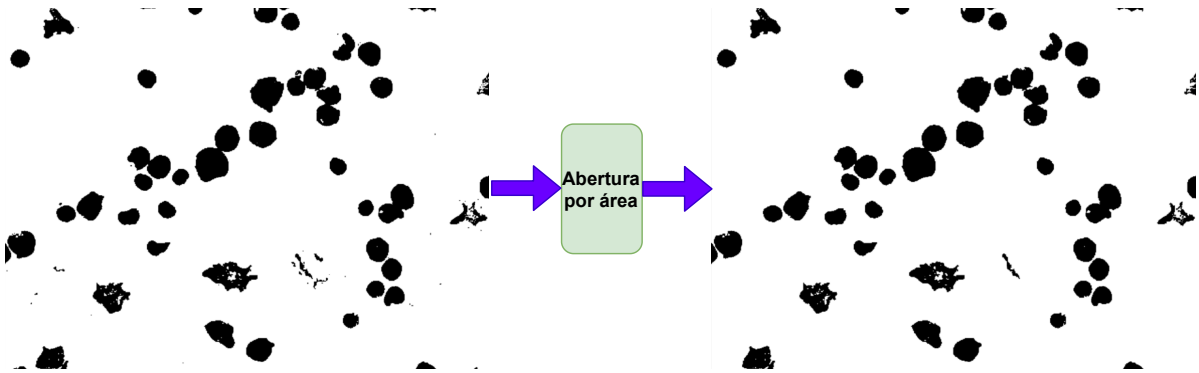
Figura 24 – Efeito das operações de abertura e fechamento em uma imagem binária.



Fonte: o autor.

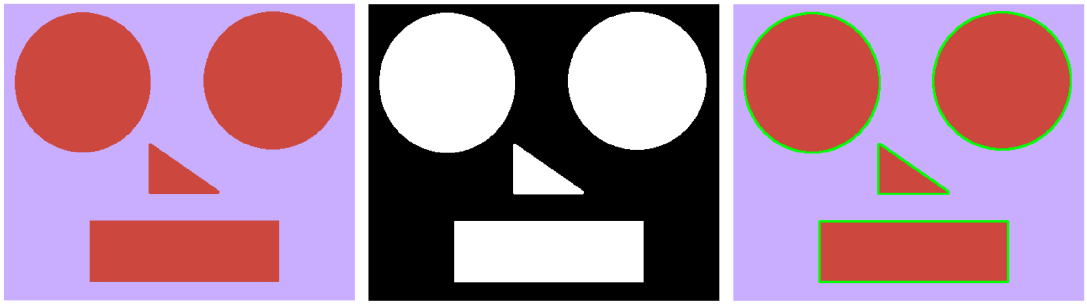
Border Following" (Satoshi Suzuki, 1985). Na Figura 26 vemos um exemplo da aplicação da função mencionada usada para encontrar os contornos dos objetos na imagem.

Figura 25 – (a) Imagem original; (b) Imagem resultante após uma abertura por área.



Fonte: o autor.

Figura 26 – Aplicação da função findContours. Na esquerda: imagem original; no centro: imagem original binarizada; na direita: imagem original com contornos encontrados pelo algoritmo.

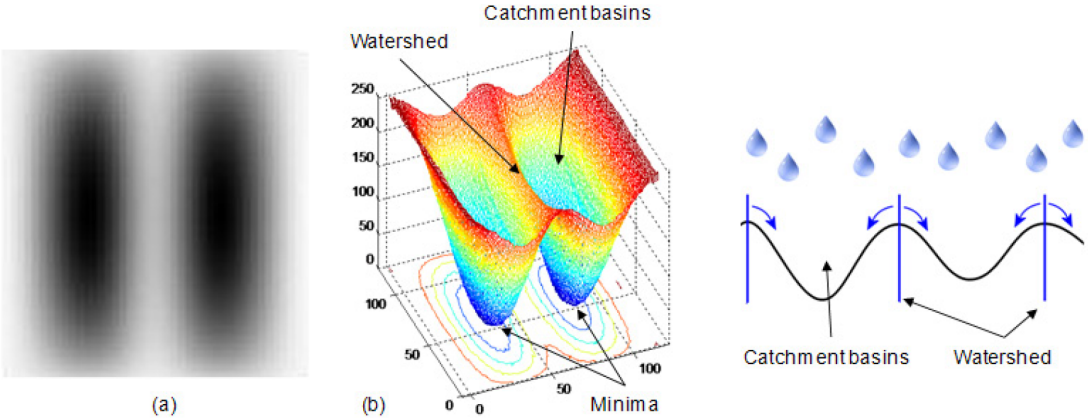


Fonte: o autor.

2.1.10 Algoritmo Watershed

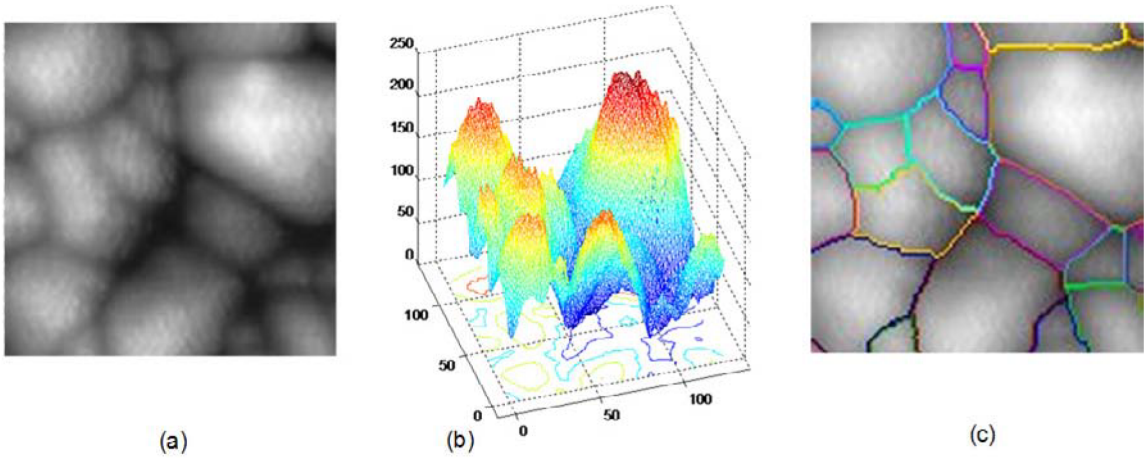
O algoritmo watershed é aplicado em imagens em tons de cinza e sua principal função é a segmentação da imagem (Soille, 1999). Uma forma para entender o método é pensar na imagem (Figura 27 (a)) como se fosse uma superficie (Figura 27 (b)) com buracos que gradualmente serão preenchidos com água (Huiyu Zhou, 2010). Primeiro são identificados os segmentos da superficie asociados aos pontos mínimos locais dela, conhecidos como bacias de captação, ou *catchment basins* em inglês. A partir dos mínimos locais inicia-se o processo de preenchimento das bacias de captação, todas ao mesmo tempo. O processo pára quando a água das bacias de captação vizinhas se encontram, criando linhas divisórias conhecidas na literatura como *watershed lines* (Figura 28 (c)). A informação das *watershed lines*, junto com a imagem original, permitem conseguir a segmentação desejada (Figura 30 (e)). Se nós quisermos usar este algoritmo com uma imagem binária, primeiro é necessário criar uma representação topográfica da imagem (Figura 30 (c)). Neste trabalho usamos a **transformada de distância** para criar a

Figura 27 – Algoritmo Watershed. (a) Imagem em tons de cinza; (b) Interpretação da imagem (a) como uma superfície topográfica onde vemos as bases de captação e as linhas divisórias.



Fonte: Huiyu Zhou (2010).

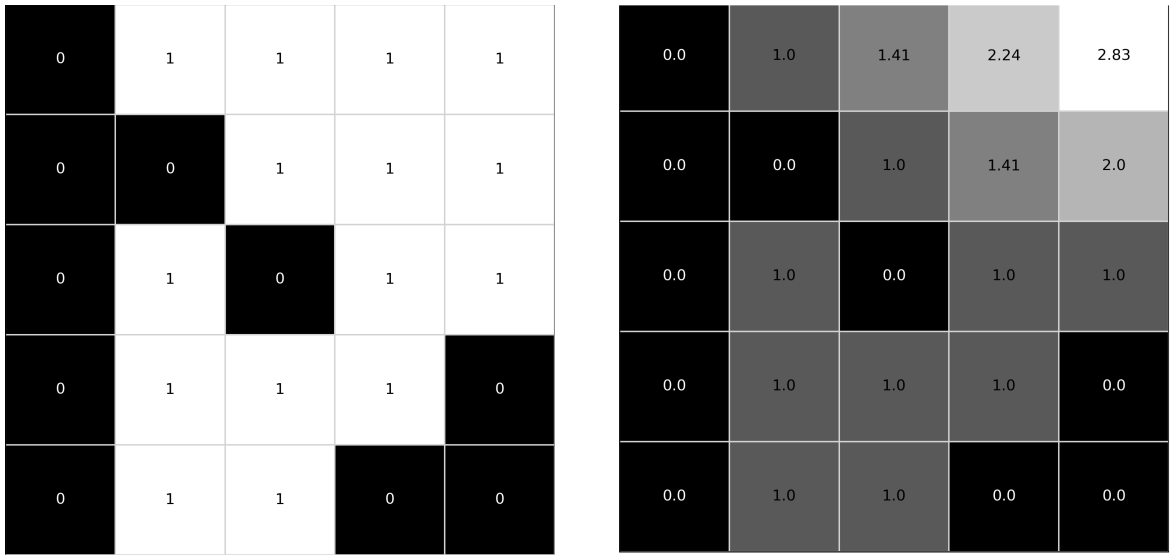
Figura 28 – Segmentação watershed. (a) Imagem em tons de cinza; (b) Representação topográfica da imagem em (a); (c) Linhas divisórias obtidas pelo algoritmo watershed.



Fonte: Huiyu Zhou (2010).

representação topográfica de uma imagem binária. Esta transformação consiste em atribuir a cada pixel p da imagem binária, a distância entre p e o pixel com valor igual a zero mais próximo do p, como se mostra na Figura 29. Vale dizer que neste trabalho, utilizamos a distância Euclideana. Na Figura 30 vemos o uso da transformada de distância junto com o algoritmo watershed para resolver um problema de separação de objetos conectados em uma imagem binária.

Figura 29 – Transformada de distância. Na esquerda, uma imagem binária; na direita: transformada de distância da imagem da esquerda.



Fonte: o autor.

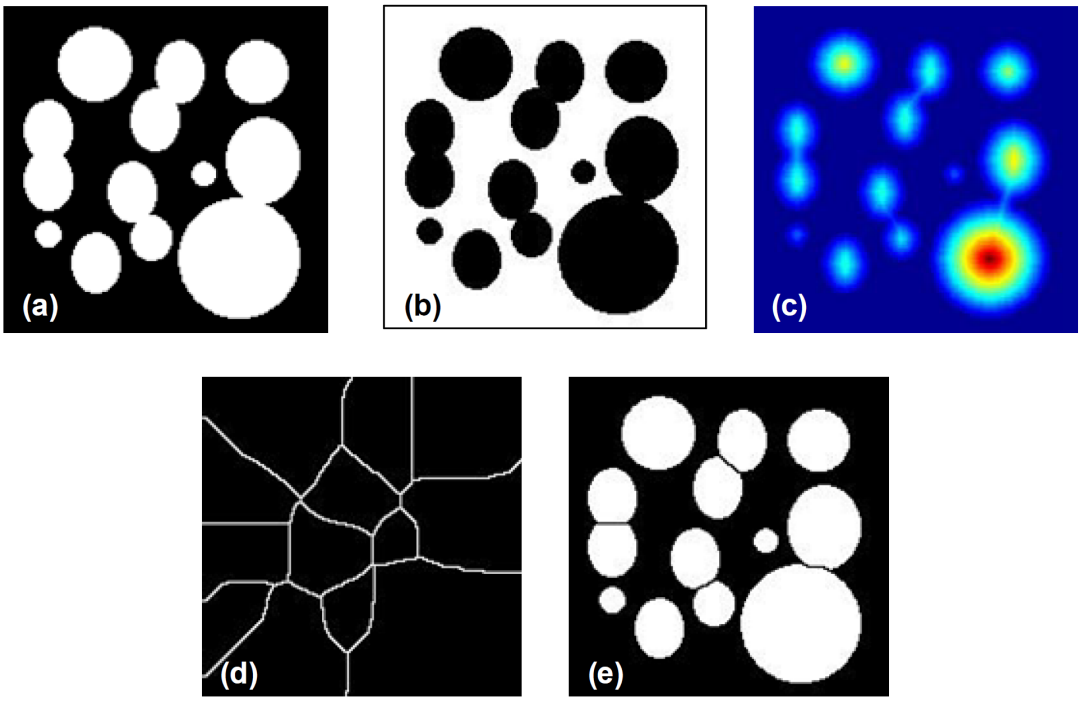
2.2 Aprendizado de Máquinas

O aprendizado de máquina é um ramo da Inteligência Artificial (IA) focado na construção de algoritmos que aprendem a partir dos dados e/ou melhoram sua precisão ao longo do tempo (IBM, 2020). Segundo a forma em que esses algoritmos aprendem, classificamos eles como: não supervisionado, supervisionado, entre outros (Haykin, 2009). Cada um com vários algoritmos existentes. Nesta seção, faremos uma breve descrição apenas dos algoritmos utilizados no nosso modelo computacional.

2.2.1 Aprendizado não supervisionado

O aprendizado não supervisionado trata de algoritmos cujo treinamento é realizado com dados não rotulados. Portanto, são capazes de encontrar padrões por si mesmos para posteriormente, após o treinamento, fazer tarefas de classificação/agrupamento. Explicaremos a ideia principal deste tipo de aprendizado com um exemplo. Imaginemos que somos donos de uma loja e para cada cliente medimos dois atributos: o número de pagamentos atrasados anuais e a sua compra média mensal. Logo, cada cliente pode ser representado como um ponto em $\mathbb{R}^2$. Se colocarmos todos os nossos clientes no plano cartesiano (Figura 31 esquerda), é muito provável observar que existem pontos (clientes) que estarão se agrupando por terem características similares. Por exemplo, clientes que compram muito e nunca se atrasam em seus pagamentos estarão geometricamente próximos. Isso permite estabelecer grupos que vão ajudar a fazer uma tarefa de classificação (Figura 31 direita). Podemos definir, por exemplo, dois grupos: clientes com opção a

Figura 30 – Segmentação watershed. (a) Imagem binária com bolhas circulares sobrepostas; (b) complemento da imagem em (a); (c) Transformada de distância da imagem em (b). (d) Linhas divisórias obtidas pelo algoritmo watershed aplicado na imagem em (c); (e) Uso das linhas divisórias para separar as bolhas circulares na imagem em (a).



Fonte: Huiyu Zhou (2010).

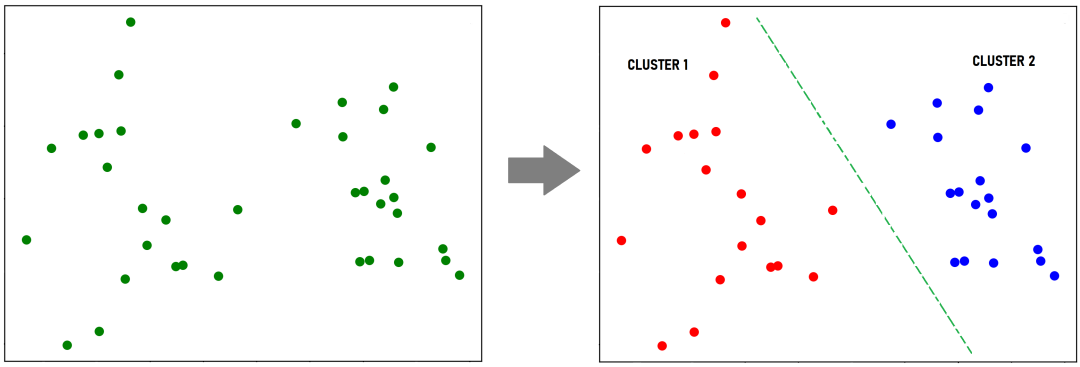
crédito, e clientes sem opção a crédito na loja. Este raciocínio bem simples permite ter uma noção básica do comportamento dos clientes da loja que permitirá tomar decisões. Concluindo, o aprendizado não supervisionado agrupa os dados sem precisar de rótulos (no exemplo, jamais foi fornecida informação de quem era um cliente candidato a receber crédito na loja e quem não). O único algoritmo não supervisionado utilizado no presente trabalho é o conhecido K-Means, que será explicado na seguinte subseção.

2.2.1.1 Algoritmo K-Means

O algoritmo K-Means é um algoritmo de aprendizado de máquinas não supervisionado cuja tarefa é criar k grupos $G_1, G_2, ..., G_k$ em uma nuvem de pontos $X \subset \mathbb{R}^n$ fornecidos para o treinamento. Cada grupo G_i é caracterizado pelo seu ponto médio c_i, chamado centroide do grupo. A ideia central deste agrupamento é que todos os pontos de G_i estejam mais próximos do centroide c_i que qualquer outro centroide.

Vamos descrever o algoritmo. Na iteração $t = 0$, escolhemos k pontos aleatórios do conjunto X que funcionam como nossos centroides iniciais: $\{c_1^{(0)}, c_2^{(0)}, ..., c_k^{(0)}\} \subset X$. Cada

Figura 31 – Aprendizado não supervisionado. Esquerda: Nuvem de pontos; Direita: agrupamento de pontos próximos.



Fonte: o autor.

grupo $G_i^{(t)}$, com $i \in 1, 2, ..., k$ e $t \geqslant 0$, estará dado matematicamente como:

$$G_i^{(t)} = \{x \ \in \ X \mid ||x - c_i^{(t)}|| < ||x - c_j^{(t)}||; \ j = 1, 2, ..., k\},$$

onde $||.||$ é a norma Euclideana. O seguinte passo consiste em calcular os centros dos grupos obtidos, os quais serão os novos centroides. Matematicamente:

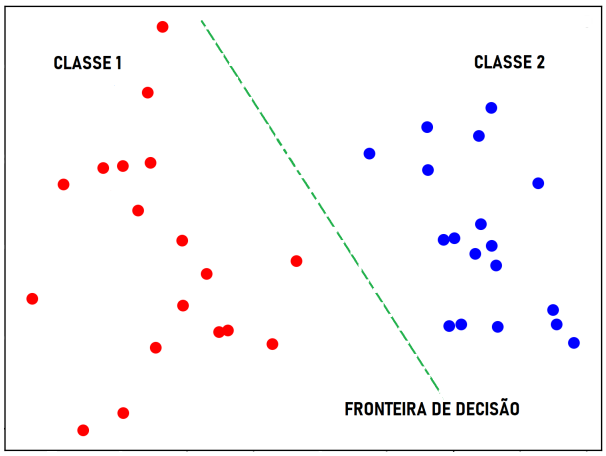
$$c_i^{(t+1)} = \frac{1}{|G_i^{(t)}|} \sum_{x_j \in G_i^{(t)}} x_j, \ \text{para} \ \ i = 1, 2, ..., k,$$

onde $|.|$ é a cardinalidade de um conjunto. Com os novos centroides, atualizam-se os grupos (iteração $t + 1$), depois calculam-se outros centroides e o processo se repete até quando estes não mudem mais. A Figura 31 mostra a tarefa de agrupamento que realiza este algoritmo com $k = 2$. As ideias desta explicação foram tomadas de Shafique and Tehsin (2018).

2.2.2 Aprendizado supervisionado

Este tipo de aprendizado é utilizado para realizar tarefas de classificação e regressão. O aprendizado supervisionado para problemas de classificação trata-se de algoritmos cujo treinamento é realizado com dados rotulados, portanto, são capazes de encontrar padrões mantendo a relação entre entrada e rótulo. Para explicar a ideia principal deste tipo de aprendizado, tomaremos o mesmo exemplo da subseção 2.2.1, mas considerando que agora os dados dos clientes estão rotulados, ou seja, que um ser humano analisou cada cliente e atribuiu a ele um rótulo de cliente com opção a crédito e cliente sem opção a crédito. Depois do treinamento, o algoritmo criará uma fronteira de decisão (linha verde da Figura 32) que permitirá realizar uma tarefa de classificação (o ponto que está acima da fronteira pertence a uma classe e aquele que está abaixo dela à outra).

Figura 32 – Aprendizado supervisionado. Dados de treinamento e fronteira de decisão.



Fonte: o autor.

Matematicamente, se temos m dados de treinamento, podemos representá-los como o conjunto:

$$\{(x^{(i)}, y^{(i)})\}_{i=1}^{m},$$

onde $x^{(i)}$ é o dado i, e y_i seu respectivo rótulo. Treinar um modelo de aprendizado supervisionado significa encontrar uma função h_w tal que $h_w(x^{(i)})$ esteja o mais próximo possível de $y^{(i)}$ para todo i. Aqui, w é um vetor de parâmetros e a função h_w fica completamente definida por ele. A proximidade é medida mediante uma função de perda (*loss function* em inglês) que, segundo o objetivo do modelo, pode ser calculada com diversas equações presentes na literatura. Como neste trabalho abordamos um problema de classificação binária, apenas definiremos a função de perda chamada **entropia cruzada binária**, conhecida em inglês como *binary CrossEntropy*:

$$L = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log(h_w(x^{(i)})) + (1 - y^{(i)})(1 - \log(h_w(x^{(i)}))).$$

O que um modelo de aprendizado supervisado busca no treinamento é o vetor de parâmetros w tal que L seja mínimo. Este problema de optimização é abordado mediante o **algoritmo do gradiente descendente** que opera da seguinte forma (Zhang, 2019): primeiro, o vetor w é inicializado de forma aleatória (embora existem modelos com métodos para inicializar w). Logo, é calculado o gradiente de L com respeito a w e, aproveitando o fato de que o negativo do gradiente de L nos diz a direção na qual L diminui mais rapidamente, atualizamos w como segue:

$$w \longleftarrow w - \alpha\frac{\partial L}{\partial w},$$

sendo α um hiper-parâmetro denominado taxa de aprendizado ou *learning rate* em inglês. Este é o princípio básico de funcionamento do gradiente descendente, mas ao longo do

tempo foram desenvolvidas algumas variantes que podem ser consultadas na literatura (Zhang, 2019). Nesta dissertação usamos especificamente o algoritmo Adam.

Todos os algoritmos de aprendizado supervisado usam os princípios explicados. Agora vamos descrever brevemente alguns detalhes de um dos modelos mais utilizados e famosos atualmente, as redes neurais artificiais.

2.2.3 Redes neurais artificiais

As redes neurais artificiais são modelos matemáticos que permitem ser treinados para fazer tarefas de classificação, regressão, entre outros. Elas são compostas por unidades básicas denominadas neurônios, que estão conectados entre si. Cada neurônio pode receber como entrada um vetor e devolver um número real na sua saída. A arquitetura de uma rede neural é a forma em que estão conectados seus neurônios; a escolha de uma topologia específica depende da necessidade da aplicação. Nesta subseção iremos falar brevemente do modelo de um neurônio e de duas arquiteturas de redes neurais artificias: o perceptron de múltiplas camadas e a rede neural convolucional.

2.2.3.0.1 Modelo de um neurônio

Um neurônio é uma unidade de processamento de informação fundamental para o funcionamento de um rede neural (Haykin, 2009). A Figura 33 mostra o modelo matemático de um neurônio, onde podemos identificar três elementos básicos:

1. Um **conjunto de ligações**, cada uma caracterizada por um peso chamado **peso sináptico**. De forma específica, para $i = 1, 2, ..., n$, um sinal x_i na entrada da ligação i, é multiplicado pelo peso sináptico w_i.

2. Um **somador** cuja tarefa é somar os sinais de entrada ponderados pelos respectivos pesos sinápticos do neurônio.

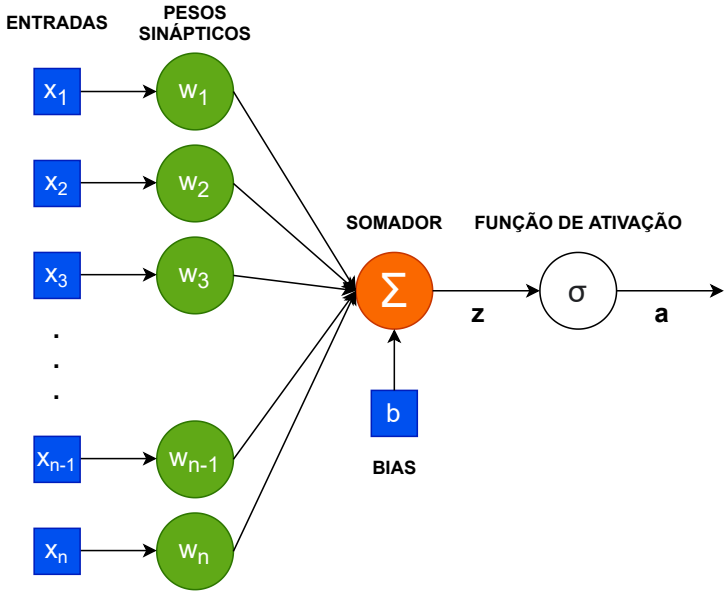
3. Uma **função de ativação** para limitar a amplitude da saída do neurônio a um intervalo. É comum que o intervalo seja $[0, 1]$ ou $[-1, 1]$. Haykin (2009) mostra alguns exemplos de funções de ativação tais como: a função sigmoide, a função de Heaviside, a função ReLU, a tangente hiperbólica, entre outros.

O modelo ilustrado na Figura 33 possui também um valor b, chamado *bias*, que afeta o valor da entrada à função de ativação. Em termos matemáticos, a saída de um neurônio com bias b, pesos $w = (w_1, w_2, ..., w_n)^T$ e função de ativação σ, quando recebe na entrada um vetor $x = (x_1, x_2, ..., x_n)^T$, é:

$$a = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{2.4}$$

O modelo descrito de um neurônio é conhecido como **neurônio artificial**. Porém, por simplicidade, neste trabalho usaremos apenas o termo "neurônio", para nos referir a um neurônio artificial.

Figura 33 – Modelo de um neurônio.

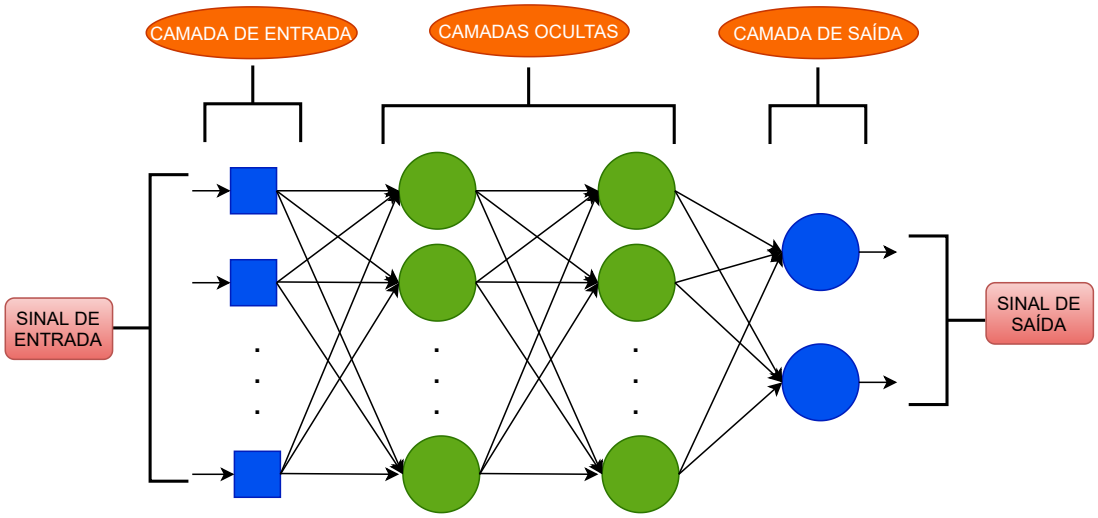


Fonte: o autor.

2.2.3.1 Perceptron de Múltiplas Camadas

O perceptron multicamada, conhecido como *Multilayer Perceptron* ou *MLP*, é uma arquitetura de rede neural artificial, composta por uma camada de entrada, pelo menos uma camada oculta e uma camada de saída (Haykin, 2009). Cada camada possui pelo menos um neurônio. A forma geral de um perceptron de múltiplas camadas consiste em um esquema totalmente conectado, isto é, cada neurônio em qualquer camada da rede, está conectado com todos os neurônios da camada anterior. Na Figura 34 vemos um perceptron de múltiplas camadas com duas camadas ocultas. Na Equação 2.4 mostram-se as operações matemáticas envolvidas em cada neurônio. Os cálculos necessários para obter a saída nesta arquitetura são realizados da esquerda para direta, isto é conhecido como *forward propagation*. O cálculo do gradiente da função de perda para esta arquitetura é realizado mediante o algoritmo chamado *backpropagation*, cujos detalhes podem ser encontrados no artigo "Learning representations by back-propagating errors" (Rumelhart David E., Hinton Geoffrey E., Williams Ronald J, 1986).

Figura 34 – Perceptron multicamada.



Fonte: o autor.

2.2.3.1.1 Rede neural convolucional

Imaginemos que desejamos criar um algoritmo que receba uma imagem (em tons de cinza para simplificar a explicação) de um cachorro ou um gato e que nos diga a qual animal corresponde. Poderíamos pensar que o modelo MLP é uma boa opção para abordar este problema; dado que as imagens são matrizes que podem ser vetorizadas e recebidas pela rede neural para o treinamento, mas essa ideia é pouco eficiente. Vetorizar uma imagem de, por exemplo, 100x100 pixels, implica ter 10000 neurônios na camada de entrada, o que significa que vamos precisar de muito mais que 10000 pesos sinápticos (mais cálculos) para treinar imagens dessas dimensões. Por outo lado, quando uma imagem é vetorizada, estamos perdendo sua informação espacial, como por exemplo, o fato de que a boca do gato está perto do nariz. Então é aqui que nasce a ideia das redes neurais convolucionais, cujo foco é extrair um vetor de dimensão reduzida denominado "mapa de características" que possui informação importante da imagem para depois utilizar uma rede neural MLP e fazer a classificação. Sabemos que a operação matemática capaz de extrair informação importante de uma imagem mediante a filtragem dela é chamada convolução. Como desejamos que seja a rede neural quem aprenda das imagens, ela deverá aprender quais filtros (máscaras de convolução) capturam as características da imagem. Portanto, os kernels dos filtros de convolução serão matrizes de pesos sinápticos que devem ser determinados durante o treinamento.

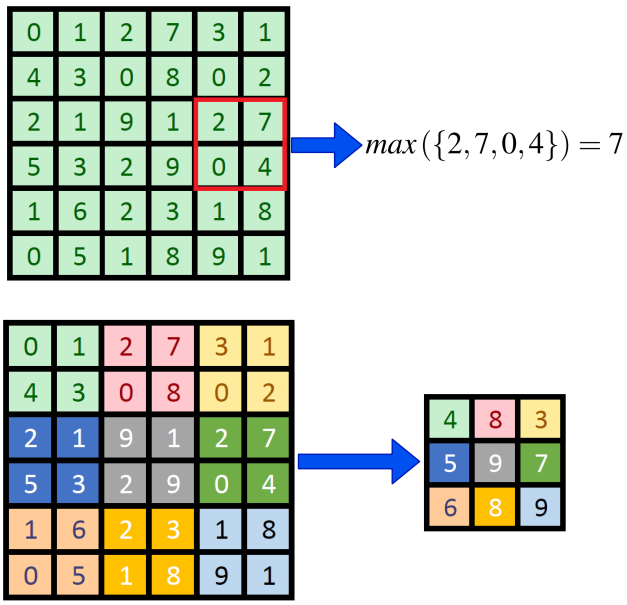
As operações de convolução têm parâmetros configuráveis como o tamanho do kernel, o **stride** (numero de pixels que pula o kernel enquanto percorre a imagem na convolução) e o **padding** (tratamento dos limites da imagem durante a convolução), cujos

detalhes estão muito bem explicados em Vidhya (2020). Além disso, dado que desejamos capturar a maior quantidade de informação da imagem com poucos valores numéricos, este tipo de rede utiliza uma técnica chamada **pooling**, que reduz o tamanho de uma imagem sem perder muita informação. Na Figura 35, vemos um exemplo de pooling, que consiste em percorrer a imagem com uma janela (de 2x2 neste caso) aplicando operações de máximo (max pooling) ou média (average pooling). Por outro lado, vale dizer que a função de ativação recomendada para aplicações com imagens é a ReLU. Além disso, vale dizer que se trabalhamos com imagens coloridas, as convoluções devem ser aplicadas sobre todos os canais da imagem.

Na etapa de extração de características, podem existir várias camadas de convolução-pooling, e isso depende de quão profunda e larga precisamos que seja a rede neural convolucional. Vale comentar que aumentar a profundidade ou largura da rede, ou de forma mais geral o número de parâmetros, não implica necessariamente uma melhora no desempenho dela.

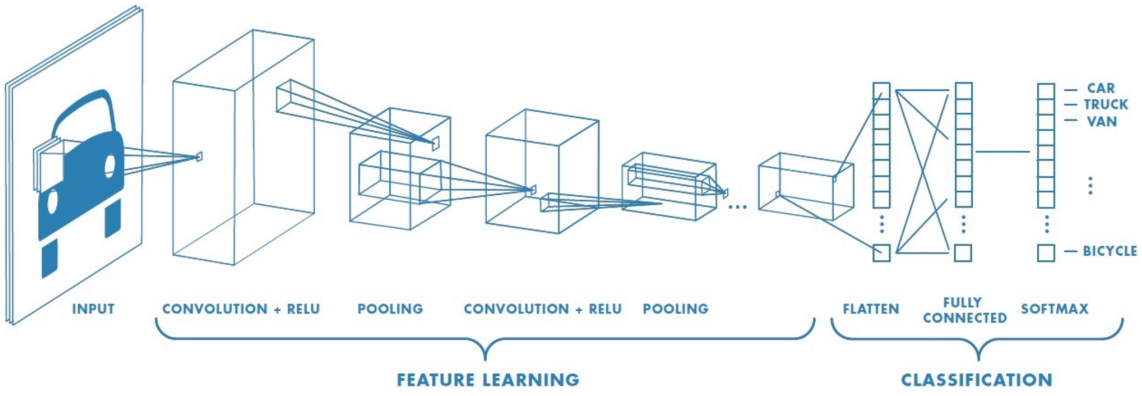
Com a intuição do modelo e as operações envolvidas entendidas, podemos compreender o esquema mostrado na Figura 36 onde, mediante operações de convolução e pooling, é determinado um vetor de características que representa a imagem de entrada (o carro). O vetor de características é posteriormente processado por uma rede MLP que realiza a tarefa de classificação. Finalmente, Solai (2018) explica de forma detalhada como funciona o *backpropagation* para este tipo de rede neural.

Figura 35 – Exemplo de max pooling.



Fonte: o autor

Figura 36 – Ilustração da arquitetura de uma rede neural convolucional.



Fonte: Towards Machine Learning (2018).

3 Metodologia

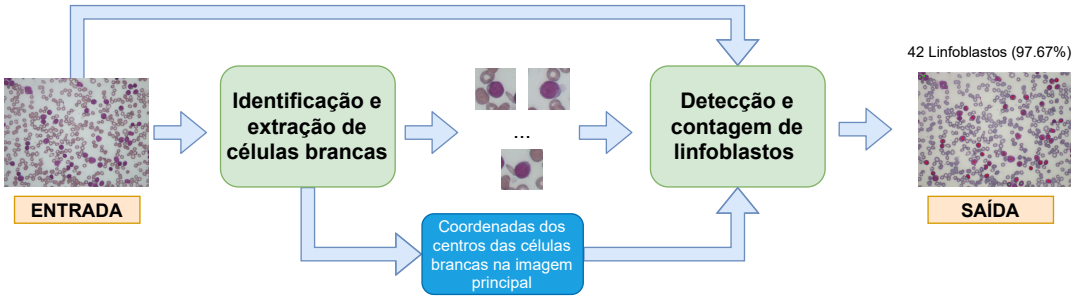
Neste capítulo explicaremos os métodos utilizados para desenvolver o modelo computacional que propomos como ferramenta para o diagnóstico de LLA.

A estrutura dos sistemas que compõem o modelo computacional será mostrada em forma de diagramas de blocos. Primeiro, apresentamos um esquema geral e, enquanto avançamos, aprofundaremos em cada um dos blocos revisando os processos que eles realizam.

3.1 Esquema Geral

Em termos gerais, o modelo é composto por dois sistemas principais como se mostra na Figura 37. O primeiro é responsável por identificar quais são as possíveis células brancas na imagem de entrada com a finalidade de gerar uma lista das coordenadas dos centros delas (ao longo do texto chamados de **centroides**) e uma sub imagem para cada célula branca. O segundo tem como objetivo analisar se cada sub imagem gerada pelo bloco anterior corresponde a um linfoblasto ou não, fazer a contagem deles e gerar os resultados esperados na saída do modelo.

Figura 37 – Esquema geral do modelo computacional.



Fonte: o autor.

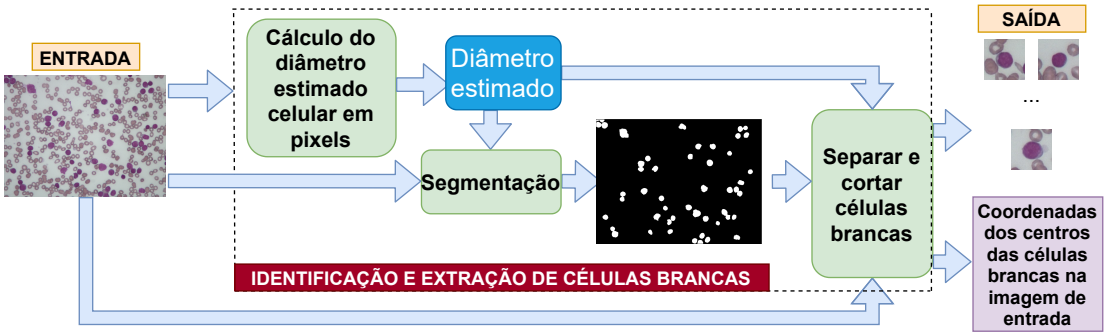
Após entender o funcionamento principal do modelo, vamos revisar a composição dos dois sistemas.

3.2 Identificação e extração de células brancas

Para estimar os centroides é necessário ter uma imagem binária apenas com a informação das células brancas (subseção 3.2.2). Por esse motivo, precisamos fazer um

processo de segmentação da imagem de entrada. Com os centroides obtidos, as células brancas ficam localizadas e podemos recortá-las para a classificação. Para saber o tamanho adequado das sub-imagens das células brancas, precisamos de uma medida que nos forneça informação das dimensões das células. Neste trabalho, essa medida será chamada **diâmetro estimado**. Estas ideias foram colocadas no esquema da Figura 38.

Figura 38 – Diagrama de blocos usado para a identificação e extração de células brancas.



Fonte: o autor.

O diagrama está composto por três sub sistemas, o primeiro se ocupa de calcular o diâmetro estimado, o segundo é responsável por identificar os pixels correspondentes às células brancas (segmentação), e o terceiro tem como objetivo encontrar as saídas esperadas (centroides e sub imagens). Cada uma destas três etapas serão discutidas a seguir.

3.2.1 Cálculo do diâmetro estimado

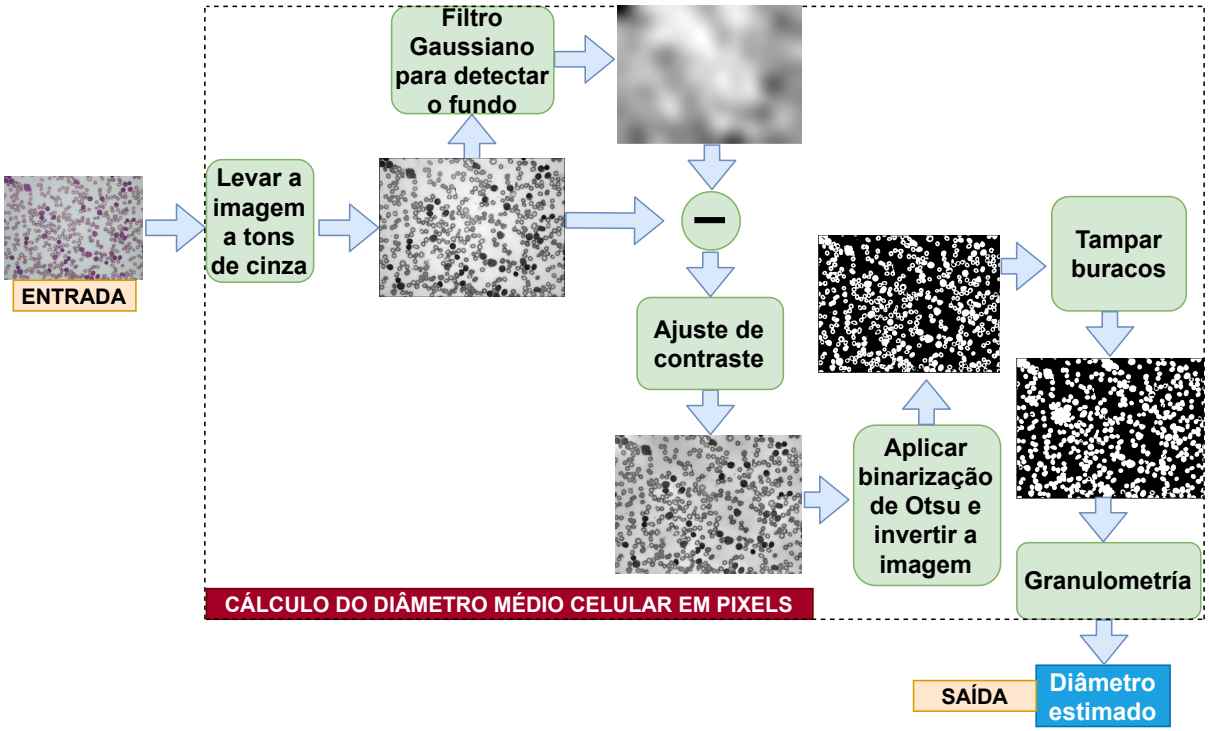
O diâmetro estimado nos fornece uma ideia do tamanho das células. Esta medida vai nos ajudar a definir o tamanho das sub imagens e também a fazer um ajuste mostrado posteriormente na subseção 3.2.3. O diâmetro estimado é medido em pixels.

Para encontrá-lo, fazemos uma análise granulométrica (subseção 3.2.1.3) baseada nas áreas da imagem onde estejam as células brancas e vermelhas; então é necessário realizar uma separação entre elas e o fundo, ou seja, binarizar a imagem. Para que o processo de binarização seja robusto, a imagem de entrada (em tons de cinza) é preprocessada mediante uma eliminação do ruído presente no fundo, que é provocado pelo microscópio (Figura 40). Além disso, é feito um ajuste de contraste na imagem. Esses processos estão descritos no diagrama da Figura 39 e serão analisados posteriormente.

3.2.1.1 Filtragem e ajuste de contraste

Dado que estamos processando imagens adquiridas por meio de microscópios, existe um ruído inerente na área onde não há células (o fundo), produto das luminosidades

Figura 39 – Diagrama de blocos usado para o cálculo do diâmetro estimado.

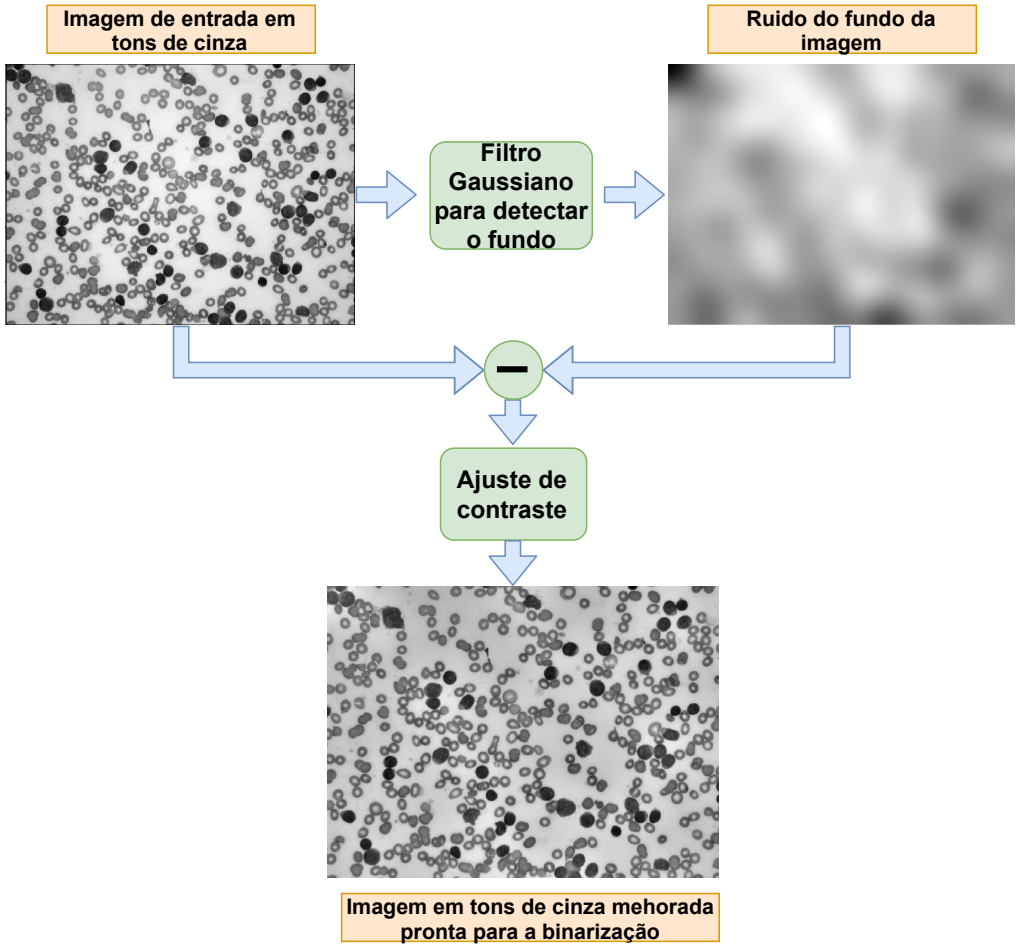


Fonte: o autor.

desse tipo de aparelho; esse ruído pode provocar erros na binarização da imagem e, portanto, no cálculo do diâmetro estimado. Este problema pode ser resolvido subtraindo-se o ruído do fundo da imagem de entrada. Dada a natureza do tipo de imagens que estamos processando, podemos supor que as irregularidades do fundo são mais suaves que nas células presentes. Então o fundo possui menores frequências espaciais que as células e, portanto, sua informação pode ser extraída com um filtro passa-baixas (Scotti, 2006). Sugere-se fazer experimentos para determinar as dimensões do kernel de convolução do filtro porque ele depende do tamanho da imagem que está sendo processada; neste trabalho, obtivemos bons resultados com um kernel Gaussiano de dimensão 401x401.

Após a eliminação do ruído do fundo, aplicamos um ajuste de contraste antes de fazer a binarização. Estas operações provocam efeitos importantes no histograma da imagem, onde se percebe uma distribuição de pixels mais suave com uma bimodalidade mais evidente (picos mais separados), como mostra a Figura 41. Na prática, isso significa que os pixels que pertencem ao fundo estão melhor agrupados entre eles (isso é um pico do histograma), e os pixels das células estão também melhor agrupados entre eles (o outro pico do histograma). Dessa forma, as informações do fundo e das células estão melhor diferenciadas (bimodalidade mais evidente) e, portanto, teremos uma binarização mais robusta.

Figura 40 – Eliminação de ruído do fundo da imagem usando filtro Gaussiano.

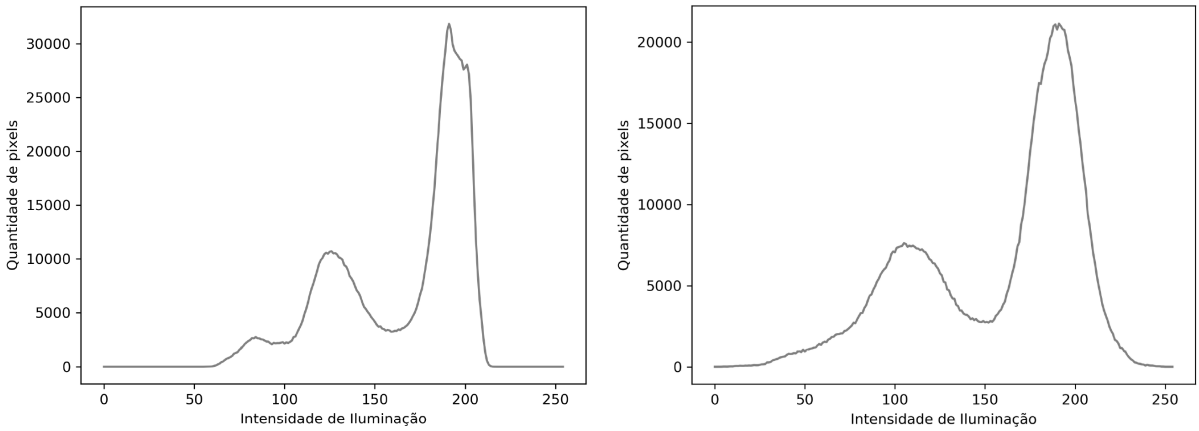


Fonte: o autor.

3.2.1.2 Separação de células e fundo

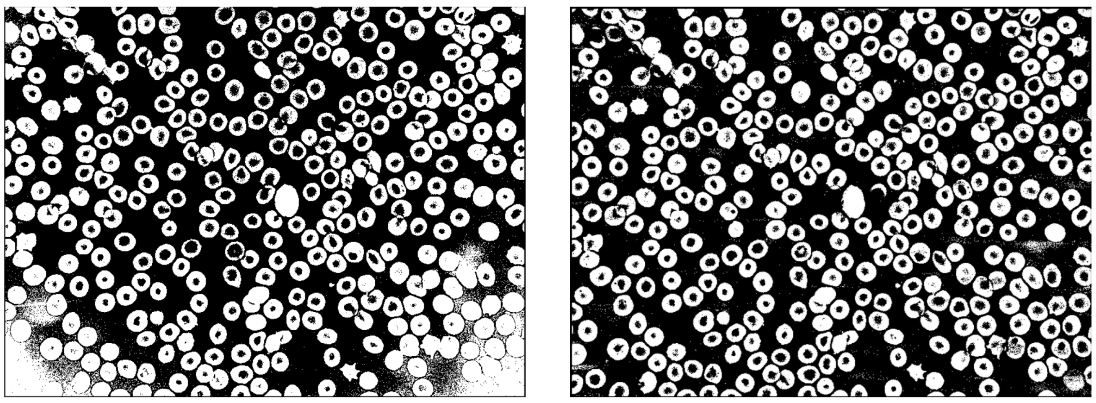
Melhorado o histograma da imagem e, dada sua característica bimodal, o método selecionado para fazer o processo de binarização foi o método de Otsu. No método de Otsu, a escolha do limiar é automática e, portanto, não será um parâmetro que devemos levar em conta. Vale dizer que, para realizar a binarização, usamos o comando *threshold* da biblioteca *cv2* em Python. Este método devolve uma imagem binária com o fundo branco e as células pretas. Como os objetos de interesse são as células, elas devem ter cor branca, então fizemos uma inversão da imagem. Note que, após a binarização, existem buracos dentro das células vermelhas, como ilustrado na Figura 42. Como os pixels dentro dos buracos são parte da geometria das células, eles devem ser considerados no cálculo do diâmetro estimado e, portanto, foram preenchidos. O preenchimento dos buracos é feito mediante operações de morfologia matemática implementadas na função *ndimage.binary_fill_holes* da biblioteca de código aberto *scipy* em Python. Além disso, foi aplicada uma operação de abertura com um elemento estruturante em forma de disco de

Figura 41 – Esquerda: histograma da imagem de entrada em tons de cinza; Direita: histograma após a filtragem e o ajuste de contraste.



Fonte: o autor.

Figura 42 – Esquerda: Binarização sem filtragem nem ajuste de contraste; Direita: Binarização com filtragem e ajuste de contraste.



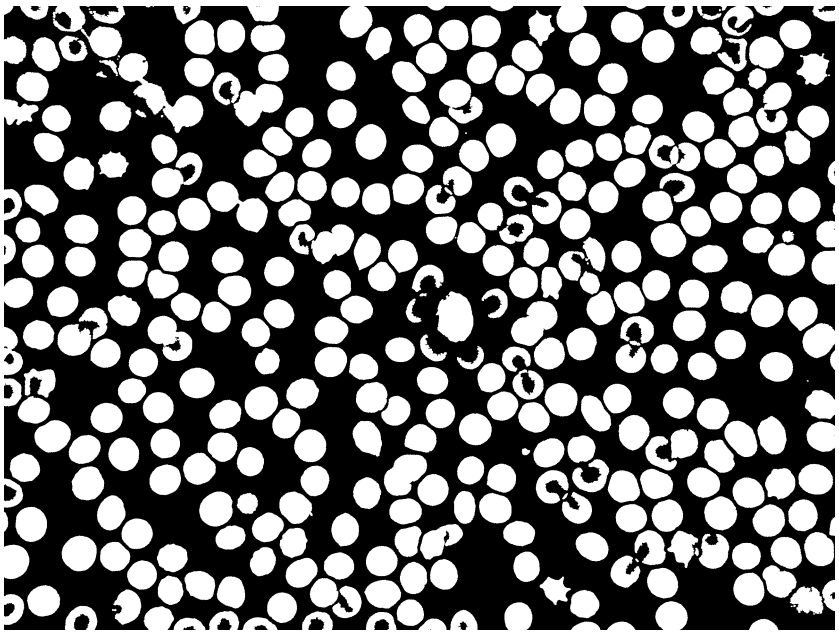
Fonte: o autor.

diâmetro 5 para eliminar o ruído do tipo sal e pimenta. A Figura 43 mostra um exemplo do resultado de fazer o preenchimento dos buracos e a operação de abertura.

3.2.1.3 Obtenção do diâmetro

Neste ponto do processo, temos uma imagem binária I onde os pixels do fundo estão em cor preta e os pixels das células em cor branca, e estamos prontos para calcular o diâmetro estimado. A técnica usada para esse objetivo, chamada granulometria e introduzida por Georges Matheron na década de 1960, usa operações de morfologia matemática para gerar uma distribuição de tamanhos. O algoritmo utilizado para implementar a granulometria segue um processo iterativo. Em cada iteração i, aplicamos uma operação

Figura 43 – Imagem binarizada após preencher buracos e eliminar ruído de sal e pimenta.



Fonte: o autor.

de abertura sobre I com elementos estruturantes em forma de disco (ver à direita da Figura 18) de diâmetro $(2i + 1)$, para $i \in \mathbb{N}$. Após a abertura, calculamos A_i, que é a área (número de pixels brancos) que ficou na imagem após a abertura. Com esta técnica, se definimos $D_i = A_i - A_{i+1}$, então D_i representa a distribuição dos tamanhos das células. O valor de i que fornece o máximo valor de D_i, será o diâmetro estimado. Para visualizar o comportamento dos resultados da granulometria, apresentamos na Figura 44 dois gráficos, de A_i e D_i, que para uma melhor interpretação foram desenhados com valores porcentuais (em relação à área inicial da imagem).

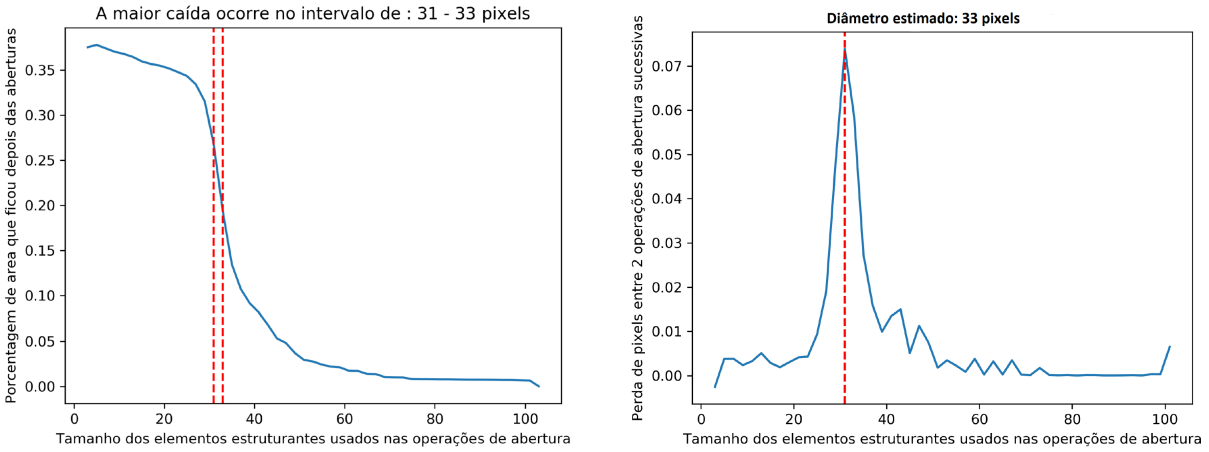
3.2.2 Segmentação

Segundo o diagrama visto na Figura 38, após o cálculo do diâmetro estimado, devemos analisar o funcionamento do bloco "segmentação", cuja função é identificar os pixels que pertencem às células brancas, as quais estão pintadas com tonalidades azuis. O funcionamento deste sub sistema pode ser visto no esquema mostrado na Figura 45.

3.2.2.1 Mudança do espaço de cor

Para conseguir o objetivo, usaremos uma segmentação baseada em cor sobre a imagem de entrada no espaço Lab, que reproduz com maior precisão a possível gama de cores que o olho humano é capaz de perceber. Além disso, como a informação das cores está nos canais a e b (o canal L tem informação só de luminosidade), podemos trabalhar

Figura 44 – Esquerda: Gráfico de A_i; Direita: Gráfico de D_i.



Fonte: o autor.

apenas com duas dimensões. Portanto, a cor de cada pixel pode ser vista como um ponto em $\mathbb{R}^2$, gerando uma nuvem de pontos no plano tal como ilustrado na imagem da esquerda na Figura 46.

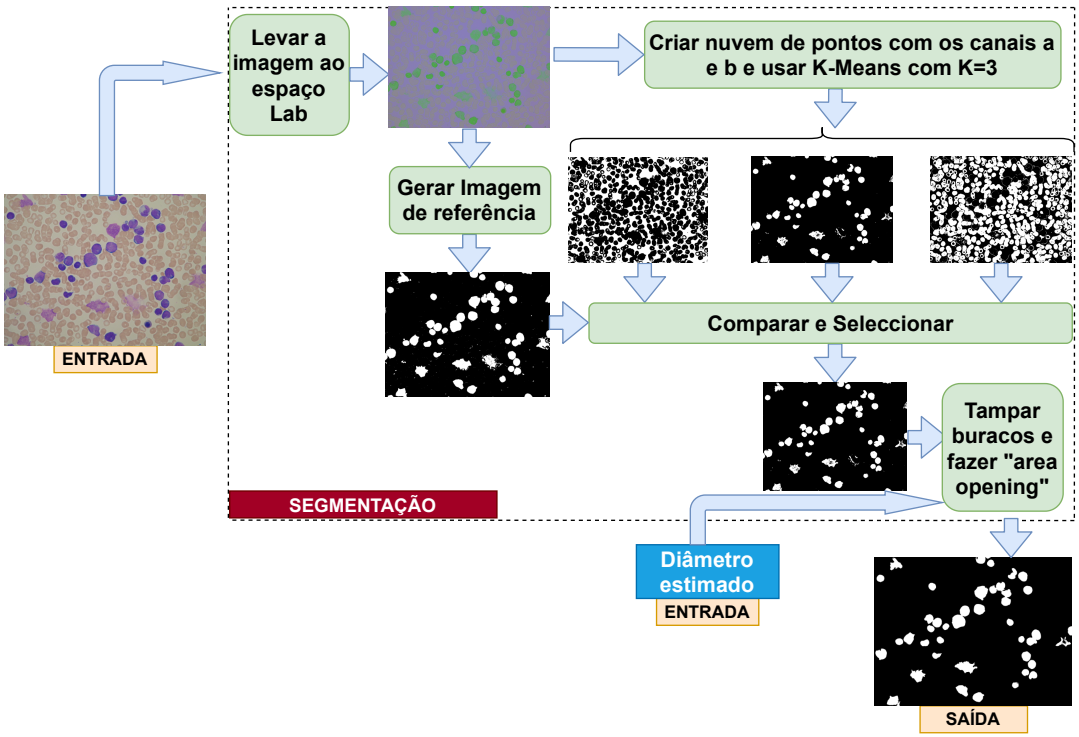
3.2.2.2 Identificação dos pixels das células brancas

Como as nossas imagens possuem três tipos de tonalidades de cor bem diferenciadas devido às suas componentes (células vermelhas, células brancas e fundo), usamos o algoritmo de classificação não supervisionado K-Means com $K = 3$ para fazer a segmentação dos pixels segundo sua cor. O resultado do algoritmo K-Means nos devolve a que grupo pertence cada pixel da imagem, como mostra a imagem da direita na Figura 46. Logo, colocando na imagem i ($i = 1, 2, 3$) o valor de 255 nas posições dos pixels do grupo i e 0 nas outras posições, podemos formar três imagens binárias, como mostra a Figura 47. Com a segmentação feita, devemos lembrar que nosso foco atual está nas células brancas e, portanto, devemos selecionar a imagem binária que contém essa informação. Para que este processo seja automático, a estratégia usada consiste em gerar uma imagem que vamos chamar de **imagem de referência**, cujo objetivo será nos fornecer uma guia de como deve ser a imagem que o sistema deve escolher.

3.2.2.2.1 Geração da imagem de referência

Nossa tarefa é detectar os pixels que compõem as possíveis células brancas da imagem. Para essa tarefa, usaremos o fato de que elas estão pintadas de tonalidades azuis e, portanto, a maior parte da sua informação deve estar contida no canal b da imagem de entrada no espaço Lab (Figura 48 esquerda).

Figura 45 – Diagrama de blocos usado para a segmentação da imagem de entrada com a finalidade de identificar as células brancas presentes nela.

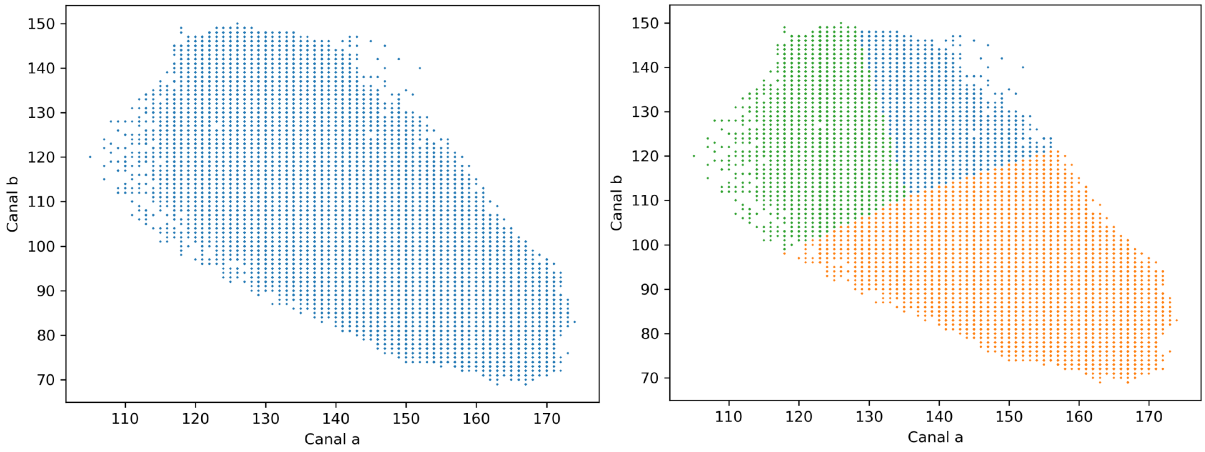


Fonte: o autor.

Pelo dito no parágrafo anterior, admitimos que o histograma do canal b é unimodal, posto que vamos ter uma alta quantidade de pixels com tons pouco azuis (fundo e células vermelhas) e uma baixa quantidade de pixels com tons azuis (células brancas). Este comportamento nos leva a pensar que podemos fazer uma binarização do canal b, onde separemos as células brancas do resto da imagem e, dada a uni-modalidade, o algoritmo selecionado para esse objetivo é o algoritmo do triângulo, que toma como limiar de binarização o ponto do histograma cuja distância até a reta formada pelo pico dele e a origem seja a maior. A Figura 49 mostra a obtenção de um limiar de binarização adequado em um histograma unimodal mediante o algoritmo do triângulo.

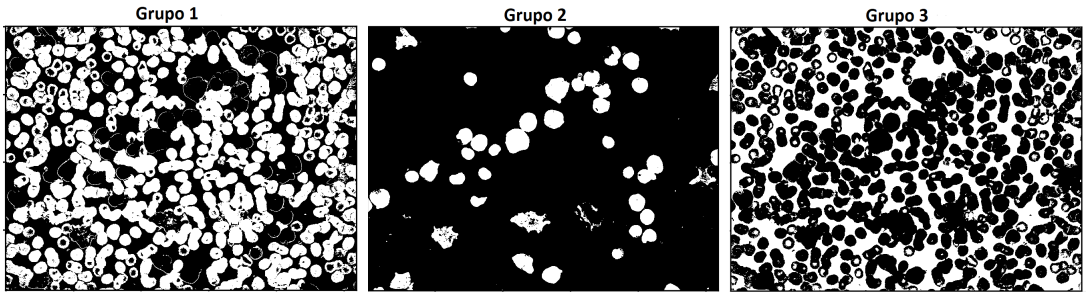
Na Figura 48 (direita) vemos que este método nos permite coletar muita informação das células brancas e, apesar de não ser uma segmentação perfeita, é suficiente para que o modelo saiba a imagem que deve selecionar entre as imagens binárias geradas com o resultado do algoritmo K-Means.

Figura 46 – Esquerda: nuvem de pontos gerada pela imagem no espaço Lab; Direita: resultado do algoritmo K-Means com K=3.



Fonte: o autor.

Figura 47 – Imagens geradas com cada um dos grupos de pixels classificados pelo algoritmo K-Means com K=3.



Fonte: o autor.

3.2.2.2.2 Seleção de imagem segmentada

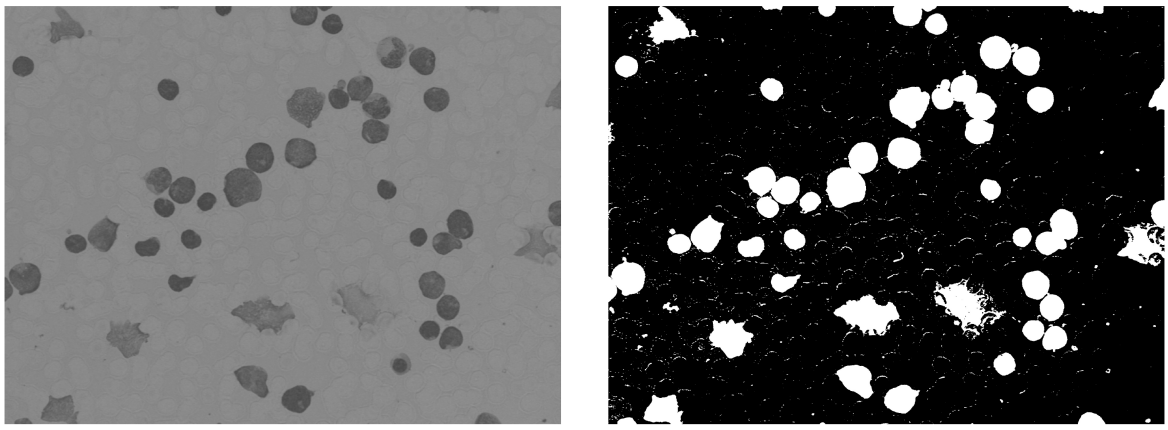
Sejam I_1, I_2, I_3, as imagens geradas pelo algoritmo K-Means, e I_r a imagem de referência, definimos:

$$e = \underset{i \in \{1,2,3\}}{\arg\min} ||I_i - I_r||,$$

onde $||.||$ é a norma de Frobenius. Dizemos que I_e é a imagem escolhida. Entenda-se que I_1, I_2, I_3, I_r, I_e são as representações matriciais das respectivas imagens. Em termos simples, a imagem escolhida será aquela que esteja mais próxima da imagem de referência.

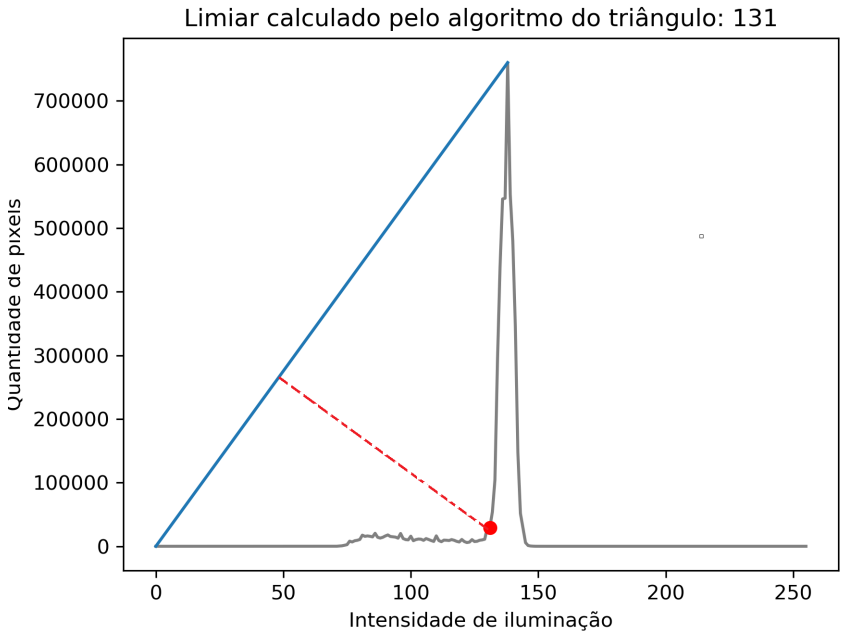
A imagem selecionada I_e pode ter células brancas com buracos dentro delas que podem ser preenchidos com a mesma técnica usada na subseção 3.2.1.2. Além disso, para eliminar objetos não desejados, fazemos uma operação de abertura por área, onde eliminamos os objetos que possuam uma área menor do que $\frac{d^2+1}{2}$ pixels, sendo d o

Figura 48 – Esquerda: Canal b da imagem de entrada no espaço Lab; Direita: Imagem de referência gerada pelo algoritmo do triângulo.



Fonte: o autor.

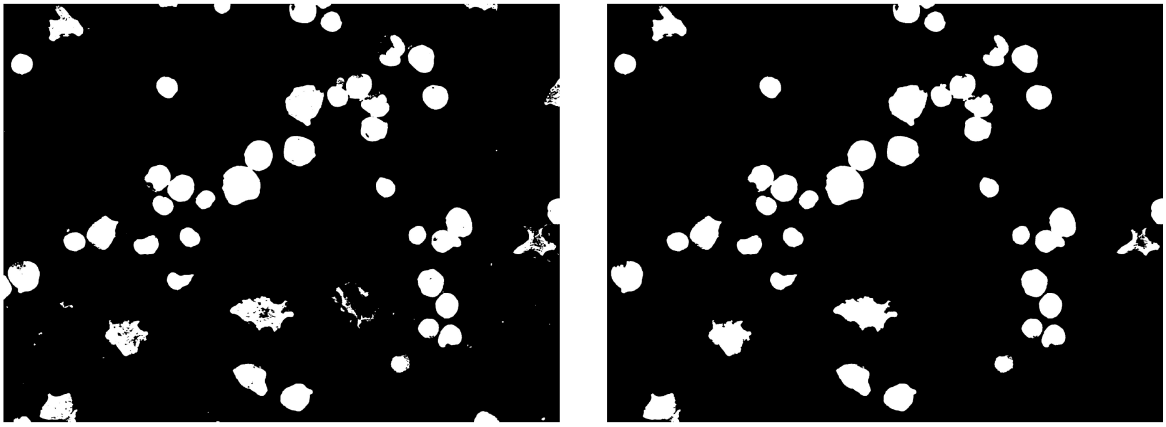
Figura 49 – Ilustração da escolha do limiar de binarização no método do triângulo.



Fonte: o autor.

diâmetro estimado calculado em etapas prévias. Depois dessas últimas operações, obtemos a saída esperada no esquema indicado na Figura 45. A Figura 50 mostra à esquerda a imagem I_e, e à direita, a imagem I_e depois de fazer uma abertura por área sobre ela.

Figura 50 – Esquerda: imagem escolhida dos resultados obtidos pelo algoritmo K-Means; Direita: imagem escolhida depois da abertura por área.



Fonte: o autor.

3.2.3 Ajustes na segmentação

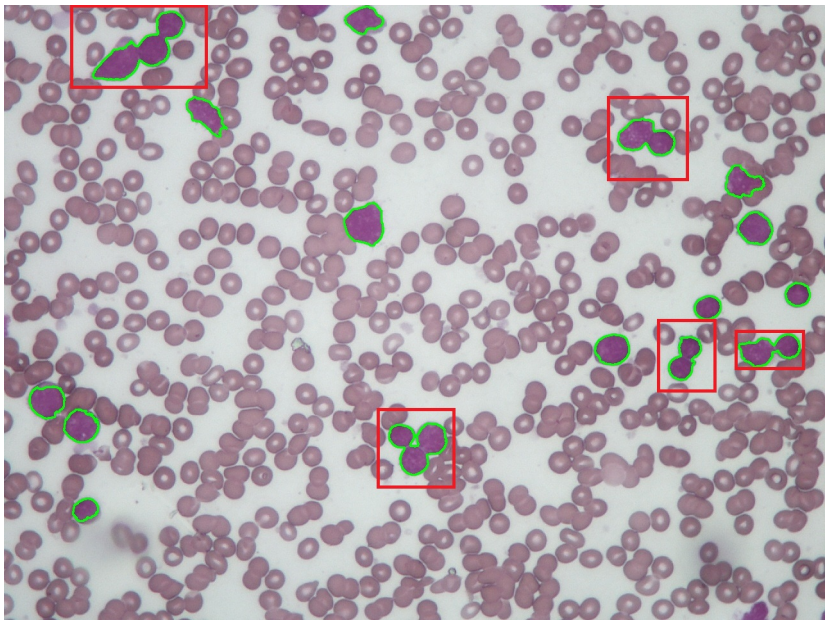
Na etapa anterior, poderíamos pensar que as células brancas ficaram totalmente identificadas, mas não é assim. Se nessas condições desenharmos os contornos de cada objeto na imagem, obtêm-se células juntas como mostram os retângulos vermelhos na Figura 51. Vemos que quando as células brancas estão juntas em um grupo, o algoritmo detecta apenas o contorno desse grupo e não de cada célula. Isso representa um problema porque o cálculo dos centros das células que precisamos obter na saída do diagrama da Figura 38 depende dos contornos delas.

Na subseção 3.2.3.1 mostra-se a forma como foi abordada esta problemática para conseguir realizar a separação das células.

3.2.3.1 Separação de células juntas

Para conseguir resolver o problema das células juntas, foi implementado o conhecido algoritmo de segmentação chamado de watershed, usando a transformada de distância da imagem de saída mostrada no esquema da Figura 45. O resultado da implementação desse algoritmo nos fornece a informação suficiente para obter uma imagem para cada célula que foi separada por ele (com fundo preto e apenas a célula). Podemos calcular o contorno para cada e, desenhando na imagem colorida de entrada, obtém-se um resultado como o que se mostra na Figura 53. Note que as células que estavam juntas na Figura 51 agora ficaram separadas na Figura 53. O contorno de cada célula é um conjunto de pontos $B \subset \mathbb{R}^2$. Para calcular o centro de uma célula dado seu contorno, usaremos o conceito estatístico de *momentos*. Considerando que a imagem que estamos processando é

Figura 51 – Problema de detecção devido a células juntas.



Fonte: o autor.

$I(x, y)$, definimos o momento M_{ij} do conjunto B como segue para $i, j \in \{0, 1\}$:

$$M_{ij} = \sum_{(x,y)\in B} x^i y^j I(x, y).$$

Logo, o centro (c_x, c_y) que desejamos saber é:

$$(c_x, c_y) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right) = \left(\frac{\sum\limits_{(x,y)\in B} xI(x, y)}{\sum\limits_{(x,y)\in B} I(x, y)}, \frac{\sum\limits_{(x,y)\in B} yI(x, y)}{\sum\limits_{(x,y)\in B} I(x, y)}\right).$$

Fazendo uma analogia com a física, pode-se pensar no contorno como um sistema de massas puntuais no plano localizadas em cada $(x, y) \in B$ e de magnitude $I(x, y)$, onde o centro de massa é (c_x, c_y).

Neste ponto, aparentemente, a segmentação foi ajustada com sucesso e até conhecemos os centroides mas, por conta do algoritmo watershed, existe a possibilidade de que uma única célula fique dividida em várias. Ilustramos essa observação no retângulo vermelho da Figura 53 que surge pela aplicação da transformada de distância quando a célula não é totalmente convexa. Para lidar com esse problema, o diâmetro estimado d foi utilizado de modo que, para cada centroide c_i, é aplicada a seguinte regra:

$$\bar{c_i} = \begin{cases} \dfrac{c_i + c_j}{2} & , \quad ||c_i - c_j|| < kd \text{ e } i \neq j \\ c_i & , \quad c.c. \end{cases}$$

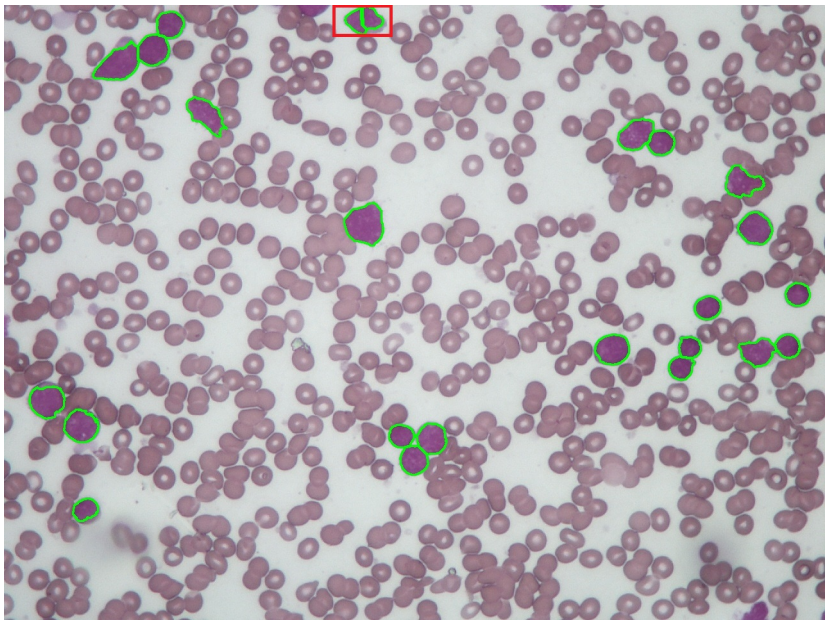
Figura 52 – Separação de células juntas usando o algoritmo watershed.

Fonte: o autor.

sendo $\bar{c}_i$ o conjunto de centroides final que usaremos, $||.||$ a norma euclideana, e k uma constante que neste trabalho é 0.9. Fazendo isso, o problema mencionado fica corrigido. Na Figura 54 podemos ver como a célula que mostramos no retângulo vermelho da Figura 53 ficou apenas com um centroide apesar de ter sido dividida em dois na aplicação do algoritmo watershed.

Após a obtenção dos centroides, geramos uma lista de sub-imagens de células brancas. Para obter cada sub-imagem, cortamos quadrados de dimensões $3d \times 3d$ com centro em $\bar{c}_i$ da imagem de entrada do sistema. Logo, a saída esperada do sistema mostrado na Figura 38 é obtida com sucesso. Na Figura 55 pode-se observar o tipo de imagens obtidas depois do processo descrito, as quais são candidatas a ser linfoblastos.

Figura 53 – Resultado Separação das células juntas.



Fonte: o autor

3.3 Detecção e contagem de linfoblastos

Esta etapa é a parte final do nosso sistema, que corresponde ao último bloco mostrado no diagrama da Figura 37. A tarefa deste subsistema é determinar se as sub imagens obtidas na seção anterior contém no seu centro um linfoblasto ou não, com a finalidade de fazer uma contagem do total de células brancas analisadas e o total de linfoblastos identificados. A abordagem deste problema será descrita nas próximas subseções.

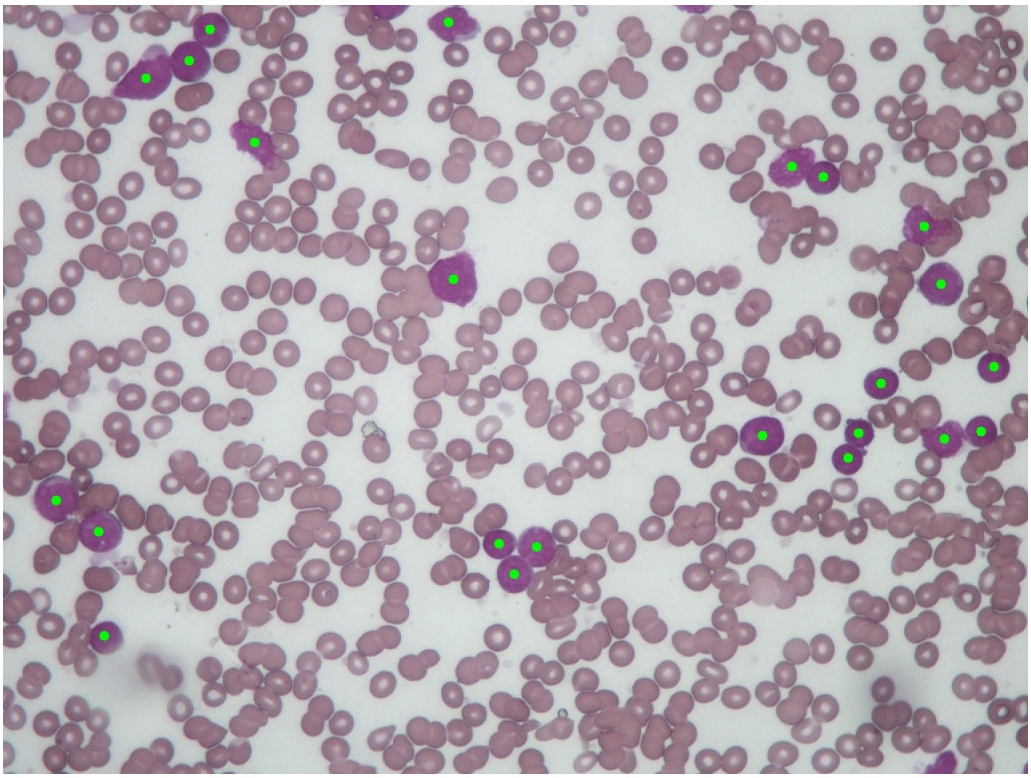
3.3.1 Modelo de classificação

Como estamos frente a um problema de classificação binária de imagens coloridas que relativamente não possuem muita complexidade quanto às formas geométricas e quantidade de objetos presentes nela, o problema pode ser resolvido mediante uma rede neural convolucional clássica cujos parâmetros podem ser vistos na Figura 56.

Dado que estamos falando de um modelo de aprendizado de máquina supervisionado, precisamos de dados de treinamento para sua implementação, os quais serão detalhados na subseção 3.3.2.

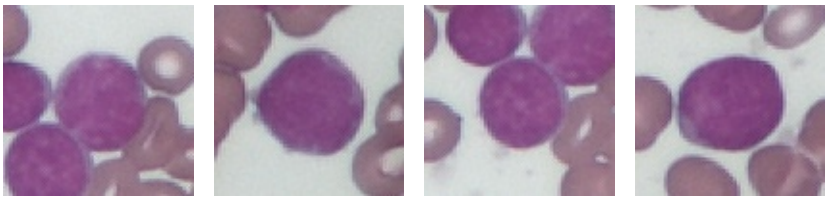
Os dados de treinamento foram aumentados mediante a técnica chamada de *data augmentation*, que consiste em fazer transformações nas imagens com o objetivo de gerar mais imagens que sirvam para treinamento. Neste caso, usamos reflexões horizontais e verticais que foram implementadas em python com o ImageDataGenerator da biblioteca

Figura 54 – Resultado final da segmentação das células brancas.



Fonte: o autor.

Figura 55 – Exemplo da lista de sub imagens de células brancas cortadas pelo sistema.



Fonte: o autor.

Keras.

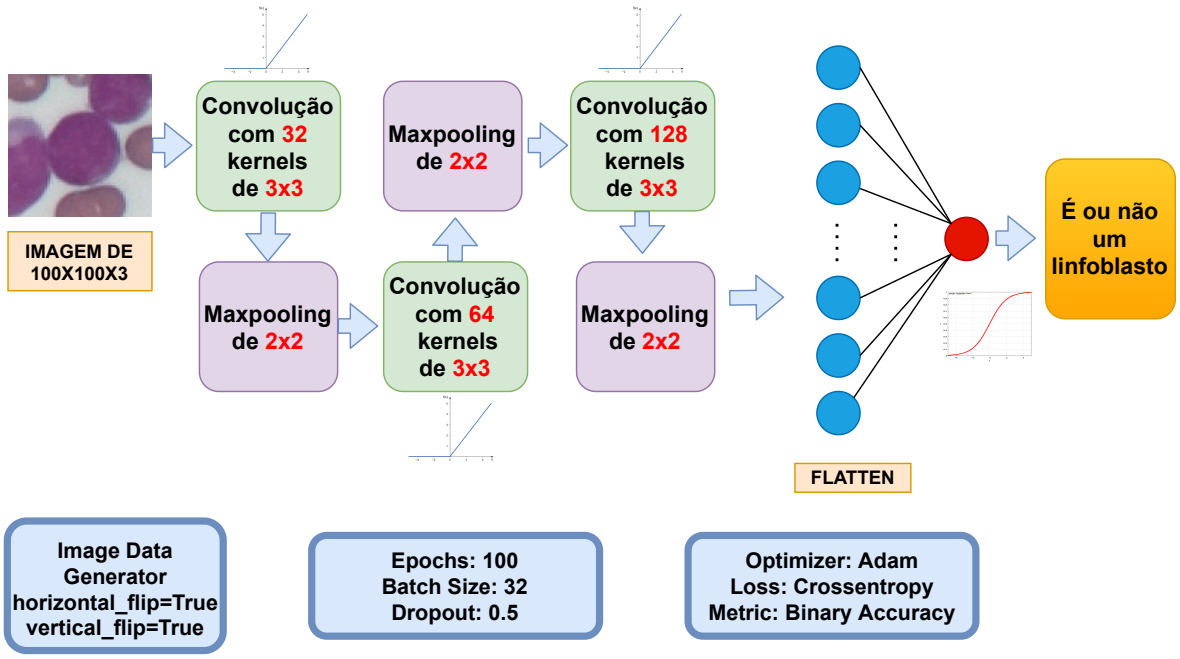
3.3.2 Bancos de dados

Para o nosso estudo, foram usados dois bancos de dados, cujos conteúdos e fontes serão detalhados nas seguintes subseções.

3.3.2.1 Banco de dados 1

O artigo “ALL-IDB: THE ACUTE LYMPHOBLASTIC LEUKEMIA IMAGE DATABASE FOR IMAGE PROCESSING” (Ruggero Donida Labati, 2011), possui infor-

Figura 56 – Modelo de classificação binária utilizado para a detecção de linfoblastos.



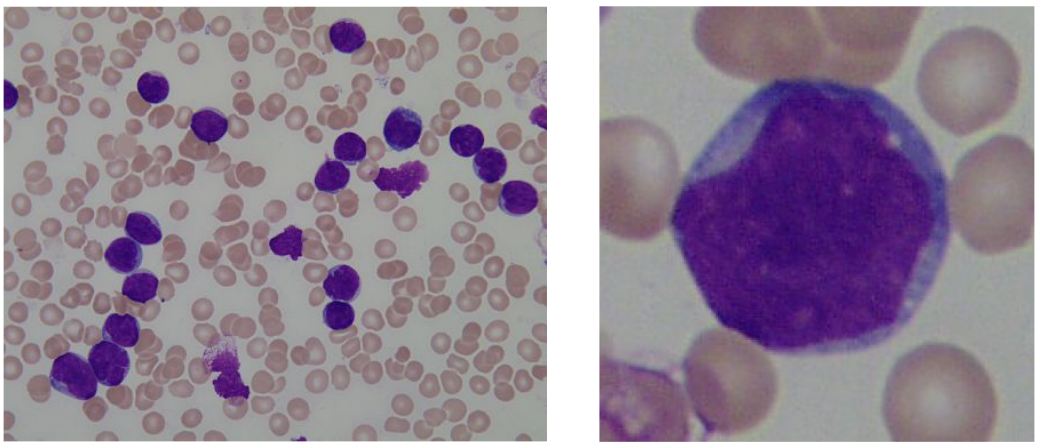
Fonte: o autor.

mação do conteúdo de um banco de dados que contém o tipo de imagens que precisamos. Mediante solicitação, os dados foram fornecidos pela "Università Degli Studi di Milano" e são compostos por:

- 108 imagens de esfregaço de sangue em formato JPG com resolução de 2592 × 1944. Além disso, cada uma dessas imagens têm associado um arquivo de texto com extensão ".xyc" com as coordenadas das células identificadas como linfoblastos marcadas por especialistas. A Figura 57 (esquerda) mostra uma destas imagens.
- 260 imagens de células brancas em formato TIF com resolução de 257 × 257 onde 130 estão rotuladas com 1, se é linfoblasto, e 130 com 0, se não é linfoblasto. Como exemplo deste banco, veja a Figura 57 (direita).

As imagens têm uma profundidade de cor de 24 bits e foram tiradas com uma câmera Canon PowerShot G5 com zoom entre 300 e 500.

Figura 57 – Exemplo de imagens do banco de dados 1.



Fonte: Ruggero Donida Labati (2011).

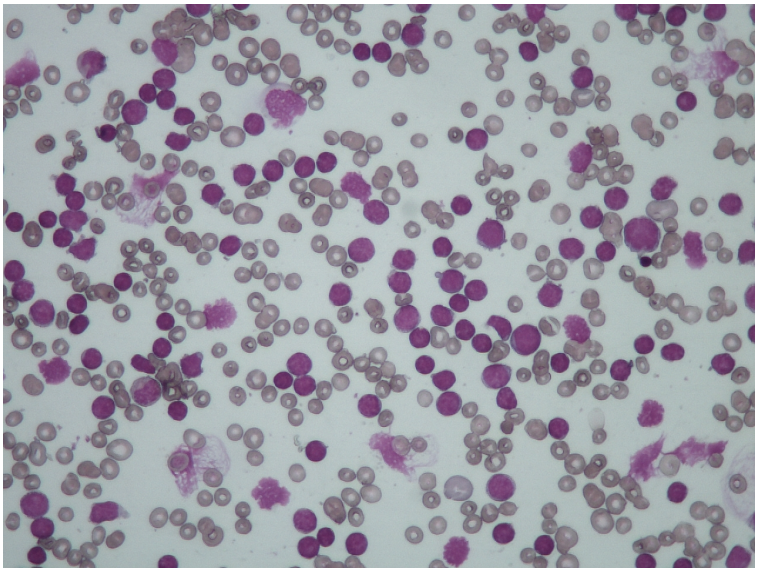
3.3.2.2 Banco de dados 2

O Centro de Pesquisa Boldrini, localizado em Campinas, São Paulo, Brasil, que na data da escrita deste trabalho é o maior centro de pesquisa da América Latina com foco no câncer pediátrico, contribuiu neste trabalho nos fornecendo acesso a seus laboratórios e a um banco de 102 imagens de esfregaço de sangue com uma resolução de 1280×960 em formato JPG com 24 bits de profundidade de cor e um zoom de 400. Estas imagens foram processadas pelo nosso sistema com a finalidade de obter sub-imagens que sirvam para o classificador binário, as quais foram rotuladas por cientistas do Boldrini, gerando 102 imagens de células brancas em formato JPG com uma resolução média de 99×99 onde 46 foram rotuladas com 1 (é linfoblasto) e 56 com 0 (não é). A Figura 58 mostra uma das 102 imagens deste banco.

3.3.3 Avaliação do modelo de classificação

Na subseção 3.3.2, vimos que contamos com 362 imagens de células brancas, 260 do banco de dados 1 e 102 do banco de dados 2, onde 186 estão rotuladas com 0 e 176 com 1. Por este motivo podemos dizer que os dados estão balanceados e, neste caso, a acurácia é uma boa medida para avaliar o desempenho do classificador. O método utilizado para a avaliação da rede neural é o Holdout estratificado com $K = 10$ iterações, que consiste em separar o conjunto de dados aleatoriamente e de forma balanceada em dois grupos, um para treinamento e um para teste, medindo a acurácia em cada um e para cada iteração. No presente trabalho a separação dos dados foi feita com 67% e 33% dos dados para treinamento e teste, respectivamente. A acurácia média obtida no conjunto de teste foi de 96.67% (Tabela 1).

Figura 58 – Imagem Im182_04_2.jpg do banco de dados 2



Fonte: Centro de Pesquisa Boldrini.

Tabela 1 – Resultados obtidos na avaliação do classificador pelo método Holdout com estratificação usando K=10 iterações.

K	Acurácia no Treino	Acurácia no Teste
1	98.35%	97.50%
2	98.76%	94.17%
3	100.00%	95.00%
4	98.76%	97.50%
5	96.28%	95.83%
6	97.11%	99.17%
7	97.93%	96.67%
8	100.00%	96.67%
9	99.17%	95.00%
10	99.17%	99.17%
Acurácia média	98.55%	96.67%

Fonte: o autor.

4 Resultados

Neste capítulo, iremos avaliar o desempenho do modelo computacional trabalhado nesta dissertação e mostraremos alguns exemplos da saída do nosso sistema.

4.1 Avaliação do modelo

Como foi descrito na subseção 3.3.2.1, no banco de dados 1, contamos com a informação das localizações dos linfoblastos marcadas por especialistas para cada imagem de esfregaço de sangue. É por esse motivo que levaremos em conta apenas essas 108 imagens para avaliar o desempenho do nosso modelo computacional.

No esquema geral visto na Figura 37, é claro que o modelo computacional apresentado tem duas etapas: extração das células brancas e a classificação delas. Logo, a avaliação do sistema não pode ser realizada somente olhando para os resultados do classificador mostrados na subseção 3.3.3. Para melhor medir o desempenho do modelo, este foi aplicado em cada uma das 108 imagens e se determinou quantas células brancas foram classificadas como linfoblastos sendo efetivamente linfoblastos (verdadeiros positivos ou VP), quantas foram classificadas como células saudáveis sendo linfoblastos (falsos negativos ou FN), quantas foram classificadas como células saudáveis sendo células saudáveis (verdadeiros negativos ou VN) e quantas o modelo classificou como linfoblastos sendo células saudáveis (falso positivo ou FP). Dado que contamos com as coordenadas exatas do conjunto dos centroides dos linfoblastos marcados pelos especialistas ($\bar{C}$), que não necessariamente devem coincidir com as coordenadas do conjunto de centroides das células brancas estimados pelo modelo (C), usamos a distância euclideana ($\hat{d}(.,.)$) e o diâmetro estimado (d) para fazer a seguinte contagem: para cada $c \in C$, calculamos $\bar{c} = \underset{\bar{c} \in \bar{C}}{\arg\min}\, \hat{d}(c, \bar{c})$ e $\Phi(c)$, onde $\Phi : C \to \{0, 1\}$ é a função que nos diz se a célula associada com um centroide calculado c é classificada como um linfoblasto ($\Phi(c) = 1$) ou como uma célula saudável ($\Phi(c) = 0$). Logo, sendo $r = \frac{d}{2}$, seguimos a seguinte regra:

$$\begin{cases} \text{Contar VP} & , \Phi(c) = 1 \ e \ \hat{d}(c, \bar{c}) \leqslant r \\ \text{Contar FN} & , \Phi(c) = 0 \ e \ \hat{d}(c, \bar{c}) \leqslant r \\ \text{Contar FP} & , \Phi(c) = 1 \ e \ \hat{d}(c, \bar{c}) > r \\ \text{Contar VN} & , \Phi(c) = 0 \ e \ \hat{d}(c, \bar{c}) > r \end{cases}$$

Na Tabela 2 podemos visualizar os resultados obtidos nesta contagem com os quais podemos avaliar quantitativamente o desempenho do modelo.

As métricas de avaliação para o modelo são:

Tabela 2 – Detecções realizadas com o modelo proposto.

		Valores Reais		
		0	1	Total
Valores Preditos	0	656(VN)	9(FN)	665
	1	138(FP)	451(VP)	589
	Total	794	460	

Fonte: o autor.

Tabela 3 – Métricas usadas para avaliar o modelo no banco de dados 1.

Métrica	Porcentagem
Precisão	76.57 %
Especificidade	98.04 %
Valor F1	85.99 %
Acurácia	88.28 %

Fonte: o autor.

- A **precisão**, que nos fornece a resposta à pergunta: Dentre todas as células que o modelo previu como linfoblastos, qual porcentagem realmente eram linfoblastos?
- O **recall ou especificidade**, que nos responde a pergunta: Qual é a porcentagem de linfoblastos que o modelo é capaz de detectar?
- O **valor F1**, que é a média harmônica entre a precisão e o recall.
- A **acurácia**, que mede a porcentagem de casos em que o modelo acertou.

Os valores obtidos para estas métricas podem ser observados na Tabela 3. Por um lado, o alto valor para a especificidade tem a ver com o baixo número de falsos negativos. Por outro lado, a precisão ficou baixa por causa da alta quantidade de falsos positivos. Porém, dado o contexto do problema (detecção de LLA), é melhor minimizar os falsos negativos, ou seja, queremos que a especificidade seja alta, pois uma especificidade baixa significaria uma alta probabilidade de atribuir como saudável a uma pessoa que precisa de atenção médica porque está doente. Por esse motivo, podemos dizer que o modelo teve um bom desempenho ao ser analisado com o banco de dados 1.

Por outro lado, também podemos avaliar o modelo segundo os resultados que "por paciente ou esfregaço de sangue". Para isto, admitimos que as imagens que contêm pelo menos um linfoblasto são "de pessoas doentes [1]", e aquelas imagens que não contêm nenhum linfoblasto são consideradas "de pessoas saudáveis (0)". Esta avaliação é possível dado que o banco de dados 1 está rotulado seguindo essas regras (Ruggero Donida Labati,

[1] colocamos as aspas porque desde o ponto de vista médico não é assim.

Tabela 4 – Detecções realizadas com o modelo “por paciente”.

		Valores Reais		
		0	1	Total
Valores Preditos	0	56(VN)	0(FN)	56
	1	3(FP)	49(VP)	52
	Total	59	49	

Fonte: o autor.

Tabela 5 – Métricas usadas para avaliar o modelo “por paciente”.

Métrica	Porcentagem
Precisão	94.23 %
Especificidade	100 %
Valor F1	97.03 %
Acurácia	97.22 %

Fonte: o autor.

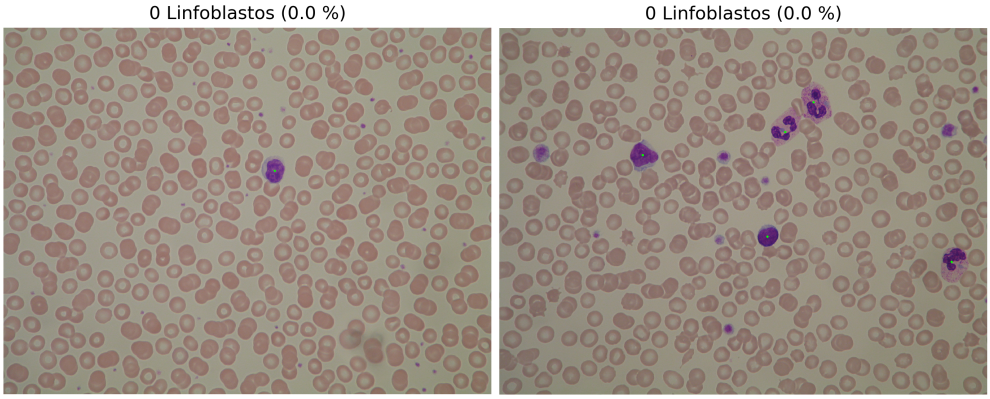
2011). Executando o modelo para as 108 imagens têm-se os resultados mostrados na Tabela 4, enquanto que as métricas calculadas com eles podem ser visualizadas na Tabela 5.

4.2 Saídas do sistema

Nesta seção vamos colocar alguns exemplos dos resultados obtidos pelo modelo com os bancos de dados mencionados na subseção 3.3.2. Os linfoblastos foram marcados com pontos vermelhos sobre as células e os pontos verdes correspondem a células que o sistema detectou como saudáveis.

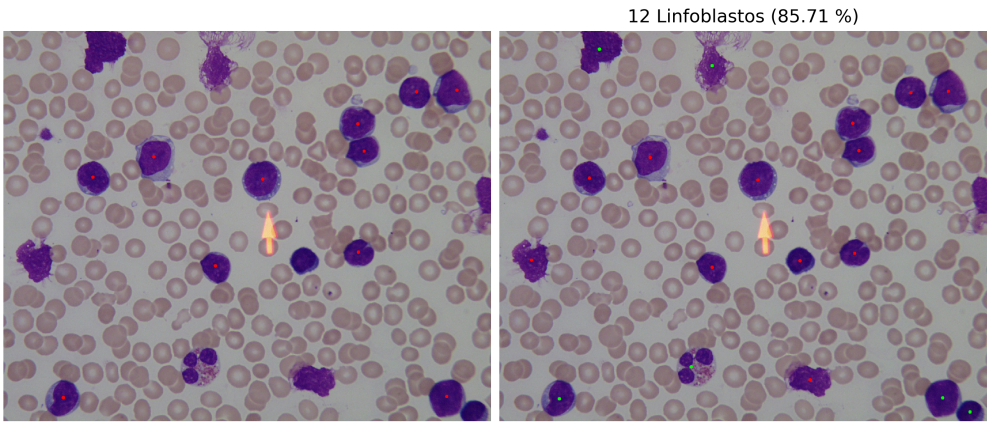
- **Banco de dados 1**: dado que contamos com as coordenadas dos linfoblastos marcadas pelos especialistas nas imagens de esfregaço de sangue, mostraremos para cada imagem a posição dos linfoblastos marcada pelo especialista e a predição do modelo. Ver Figura 59, Figura 60 e Figura 61.
- **Banco de dados 2**: dado que neste caso não contamos com as coordenadas dos linfoblastos marcadas pelos especialistas nas imagens de esfregaço de sangue, mostramos apenas a imagem original de entrada e a posição dos linfoblastos marcados pelo modelo. Ver Figura 62, Figura 63, e Figura 64.

Figura 59 – Imagens Im035_0.jpg e Im079_0.jpg do banco de dados 1. As células marcadas com um ponto verde foram avaliadas pelo modelo mas descartadas como linfoblastos.



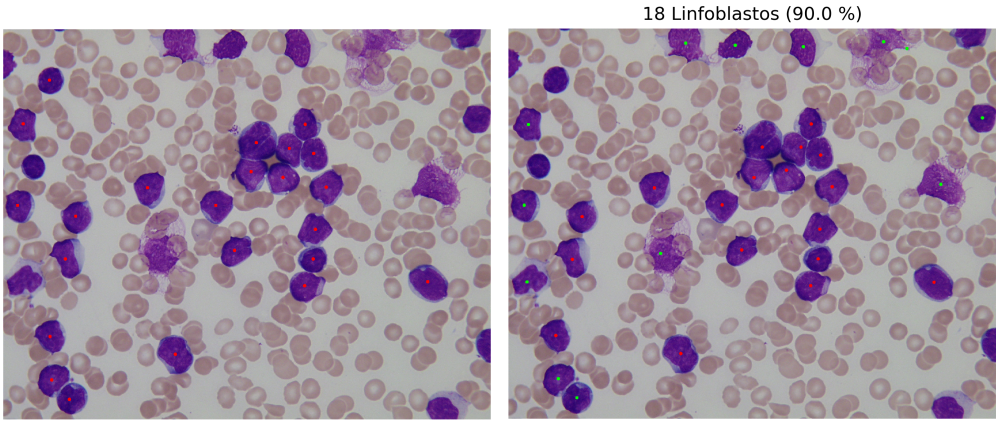
Fonte: Ruggero Donida Labati (2011). Editada pelo autor.

Figura 60 – Im010_1.jpg do banco de dados 1. Do lado esquerdo: linfoblastos marcados em vermelho por especialistas. Do lado direito: linfoblastos marcados em vermelho pelo modelo, enquanto as células analisadas mas não classificadas como linfoblastos, foram marcadas com verde.



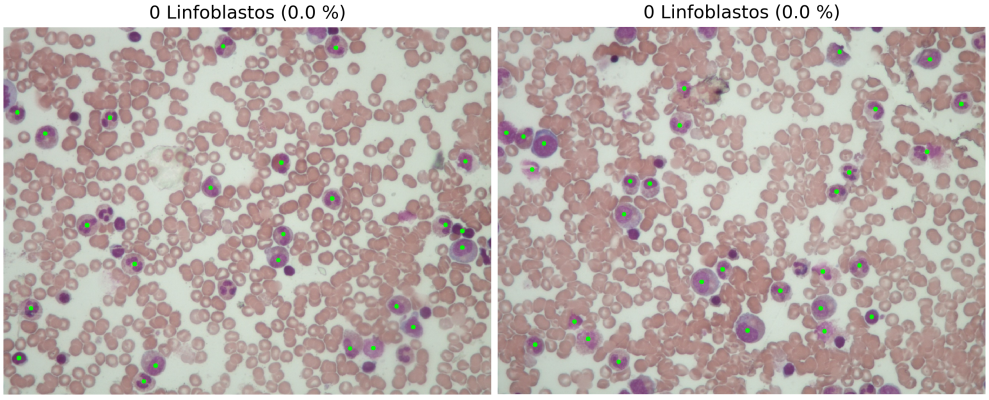
Fonte: Ruggero Donida Labati (2011). Editada pelo autor.

Figura 61 – Im005_1.jpg do banco de dados 1. Do lado esquerdo: linfoblastos marcados em vermelho por especialistas. Do lado direito: linfoblastos marcados em vermelho pelo modelo, enquanto as células analisadas mas não classificadas como linfoblastos, foram marcadas com verde.



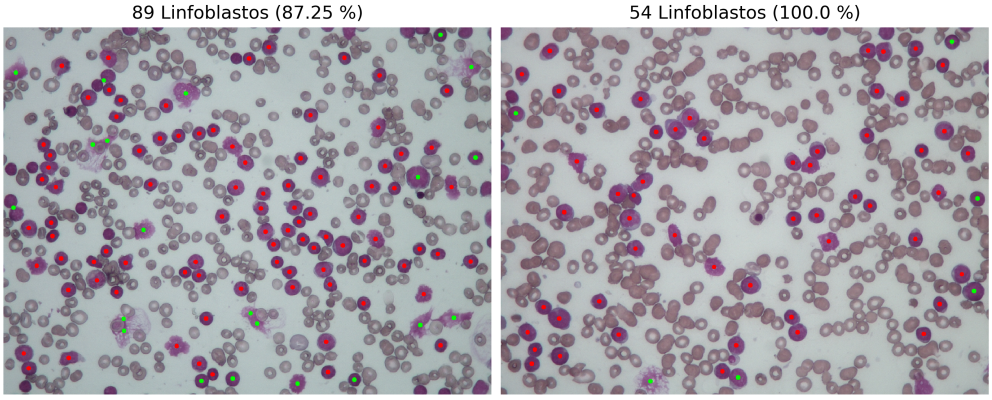
Fonte: Ruggero Donida Labati (2011). Editada pelo autor.

Figura 62 – Imagens tomadas do banco de dados 2. As células marcadas com um ponto verde foram avaliadas pelo modelo mas descartadas como linfoblastos.



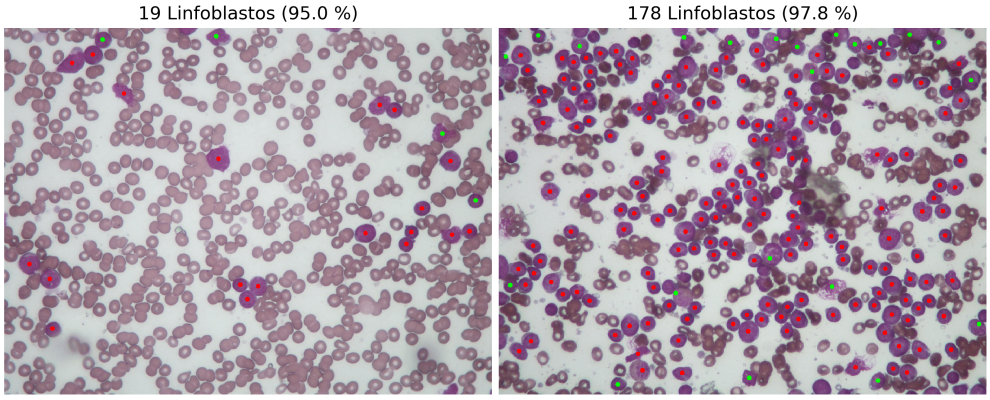
Fonte: Centro de Pesquisa Boldrini. Editada pelo autor.

Figura 63 – Imagens tomadas do banco de dados 2. Linfoblastos marcados em vermelho pelo modelo, enquanto as células analisadas mas não classificadas como linfoblastos, foram marcadas com verde.



Fonte: Centro de Pesquisa Boldrini. Editada pelo autor.

Figura 64 – Imagens tomadas do banco de dados 2. Linfoblastos marcados em vermelho pelo modelo, enquanto as células analisadas mas não classificadas como linfoblastos, foram marcadas com verde.



Fonte: Centro de Pesquisa Boldrini. Editada pelo autor.

5 Considerações finais

Neste capítulo, iremos comentar sobre os aspectos positivos e negativos do modelo, e revisaremos brevemente outra abordagem deste problema presente na literatura.

5.1 Aspectos Positivos

Na seção 4.1 comentamos sobre as métricas utilizadas para avaliar o desempenho do modelo. O banco de dados 1 foi utilizado para esta tarefa. A métrica que decidimos levar em conta com maior ênfase foi a especificidade, pois em um diagnóstico médico é desejável minimizar os casos de falsos negativos. Na avaliação das predições feitas para cada célula em todo o banco de dados obtivemos uma especifidade de 98.04%. Na avaliação dos resultados por cada imagem de esfregaço de sangue do banco de dado 1 (por paciente), a especificidade foi de 100%. Além disso, a binarização, o cálculo do diâmetro estimado, e o algoritmo watershed, que funcionam sem intervenção de um humano e os parâmetros especificados neste trabalho tais como o tamanho dos elemento estruturantes usados nas operações de morfologia matemática, o valor de K utilizado no algoritmo K-Means, entre outros, fazem que uma vez programado um software com este modelo, o usuário deva apenas fazer um upload da imagem de esfregaço de sangue para obter resultados, sem precisar de fazer ajustes na imagem de entrada. Por outro lado, existem todas as ferramentas (funções e bibliotecas) para implementar o modelo em Python e, portanto, não precisamos de licencias de software custosas. Uma vantagem deste modelo é o uso da filosofia "dividir e conquistar", pois enquanto mais blocos e sub-blocos possua um sistema, a identificação de erros e a implementação de melhoras é mais simples, esta é principal causa pela qual não foi utilizado um modelo de deep learning de ponta a ponta. Em conclusão o modelo mostra um bom desempenho, uma vez implementado permite um uso amigável com os usuários, é escalável e sua estrutura permite uma fácil detecção de erros.

5.2 Aspectos Negativos

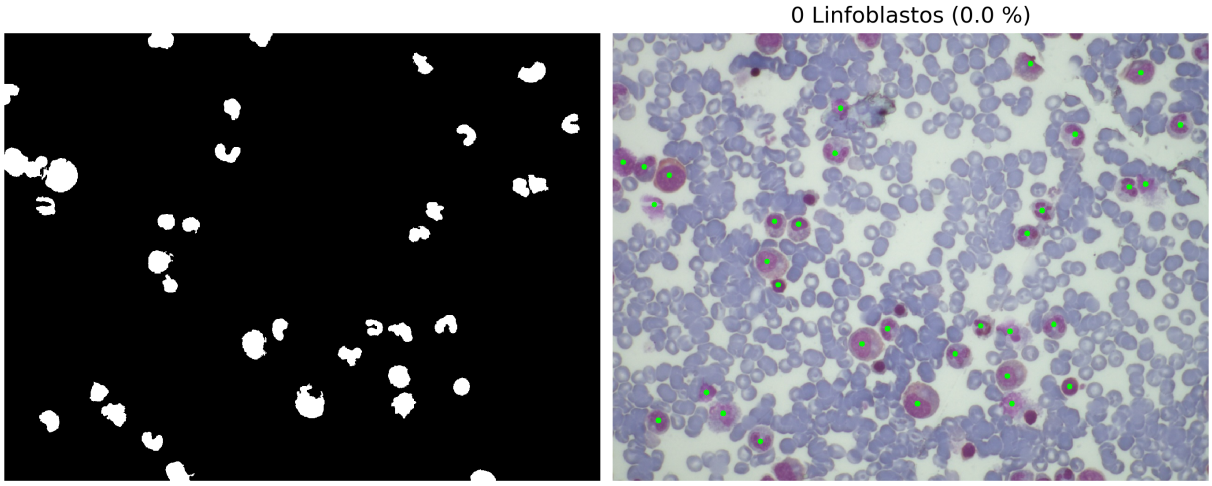
Na implementação do modelo computacional em Python, ficaram evidentes alguns aspectos importantes que devem ser levados em conta e serão comentados nesta seção.

5.2.1 Sobre a segmentação e binarização

5.2.1.1 Citoplasma celular

Experimentalmente, percebeu-se que os pixels que conformam o citoplasma celular nem sempre são considerados parte da célula na hora de aplicar o algoritmo K-Means para fazer a segmentação por cores explicada na subseção 3.2.2.2. Na imagem à esquerda na Figura 65, podemos ver como o citoplasma acabou sendo parte do fundo e os pixels dos núcleos celulares finalmente foram considerados como células.

Figura 65 – Imagem tomada do banco de dados 2. Do lado esquerdo: imagem binarizada pelo algoritmo K-Means. Do lado direito: efeito da segmentação do núcleo nos centroides.



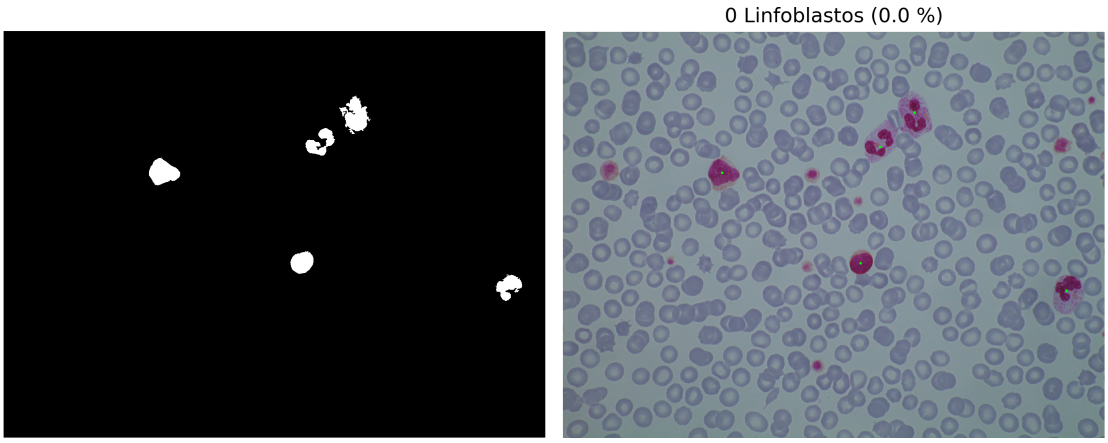
Fonte: Centro de pesquisa Boldrini. Editada pelo autor.

Se durante o processo de segmentação o modelo apenas considera o núcleo celular, então, ao calcular os centroides, o que realmente estaremos calculando será o centro dos núcleos celulares. Como o núcleo não necessariamente está no centro da célula, o modelo interpreta o centro do núcleo como o centro da célula. Veja, por exemplo, a imagem à direita na Figura 65. Este fato pode ocasionar erros na geração das sub-imagens, pois as células não ficariam no centro da sub-imagem, resultando em um possível erro na classificação. Sempre preferimos que o objeto a classificar esteja no centro da imagem para evitar que outros elementos em torno dele afetem a decisão do classificador. É importante dizer também que, apesar de notar este fato, nos experimentos houver poucos erros provocados por causa dele.

Outro fato que pode ocasionar algum problema é quando uma célula apresenta divisão no núcleo. Nesse caso, na segmentação, poderiam ser identificados dois ou mais centroides dentro de apenas uma célula. Veja, por exemplo, a imagem à esquerda na Figura 66. Porém, com a informação do diâmetro estimado, isso é resolvido da mesma

forma que o problema indicado na caixa vermelha da Figura 53 e cuja explicação detalhada está na subseção 3.2.1.2. Na imagem à direita na Figura 66 vemos como, apesar de existir divisão do núcleo em algumas células brancas, elas ficam com um único centroide.

Figura 66 – Im079_0.jpg do banco de dados 1. Do lado esquerdo: imagem binarizada pelo algoritmo K-Means com célula de núcleo dividido. Do lado direito: problema de núcleo dividido corregido.



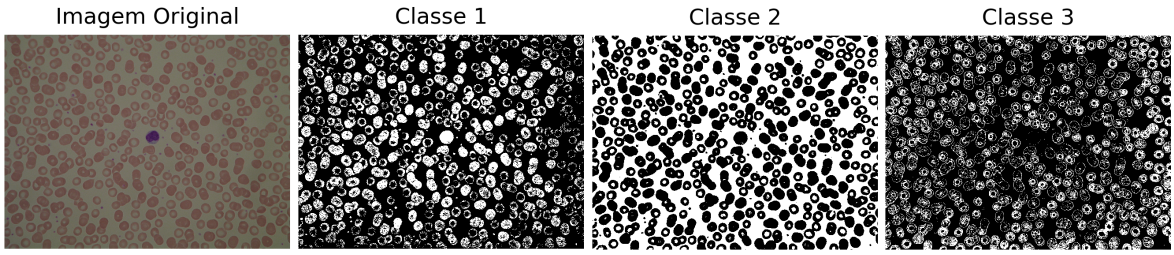
Fonte: Ruggero Donida Labati (2011). Editada pelo autor.

5.2.1.2 Tonalidades de cores

As cores possuem um papel muito importante na hora de processar a imagem. É provável que algumas imagens possuam tonalidades de cores que não permitam fazer uma correta segmentação. Por exemplo, no presente trabalho, isso aconteceu apenas com três imagens do banco de dados 1: as imagens Im100_0.jpg, Im101_0.jpg e Im102_0.jpg. Na Figura 67 se observa como o algoritmo de agrupamento K-Means não conseguiu separar corretamente o fundo, as células vermelhas e as células brancas, deixando como resultado imagens com muito ruído e sem a informação que precisamos. Vale dizer que o problema de tonalidade teve um impacto direto na avaliação dos resultados do modelo, provocando uma quantidade de verdadeiros negativos muito grande que pode ser confirmada na Tabela 2. As três imagens que possuem este problema geraram 431 verdadeiros negativos, número que impacta apenas na métrica da acurácia, que muda de 88.28 % para 82.14 % se deixamos de considerar as imagens mencionadas.

Neste trabalho não se considerou a implementação de nenhuma solução para este problema. Portanto, esse processo pode ser melhorado.

Figura 67 – Im100_0.jpg do banco de dados 1. Efeitos do problema de tonalidade na segmentação baseada em cores com o algoritmo K-Means.

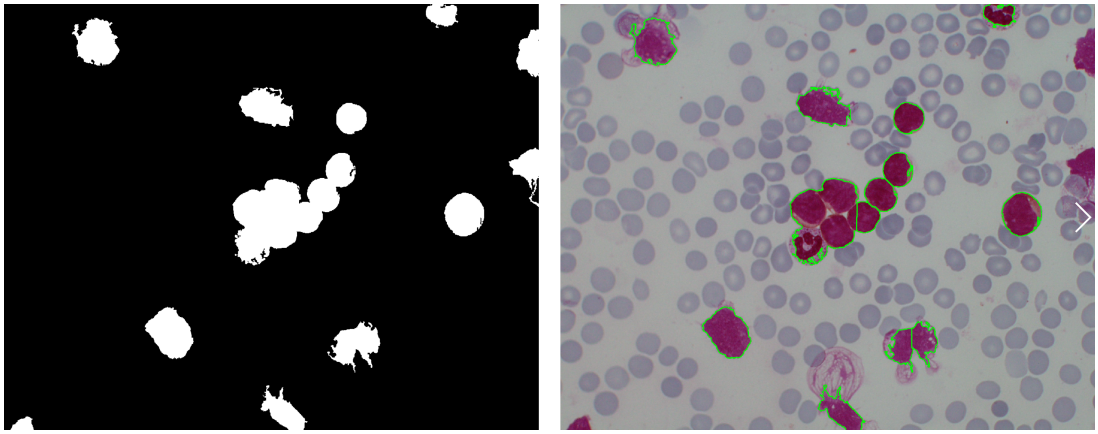


Fonte: Ruggero Donida Labati (2011). Editada pelo autor

5.2.1.3 Células juntas não separáveis

Apesar de termos apresentado uma solução para o problema das células juntas na subseção 3.2.3.1, mediante experimentação, percebeu-se que existem ocasiões em que não é possível fazer essa separação. Por exemplo, o algoritmo watershed não conseguiu separar as células no grupo ilustrado na Figura 68.

Figura 68 – Im001_1.jpg do banco de dados 1. Mostra-se um conjunto de células que não conseguiram se separar.



Fonte: Ruggero Donida Labati (2011). Editada pelo autor.

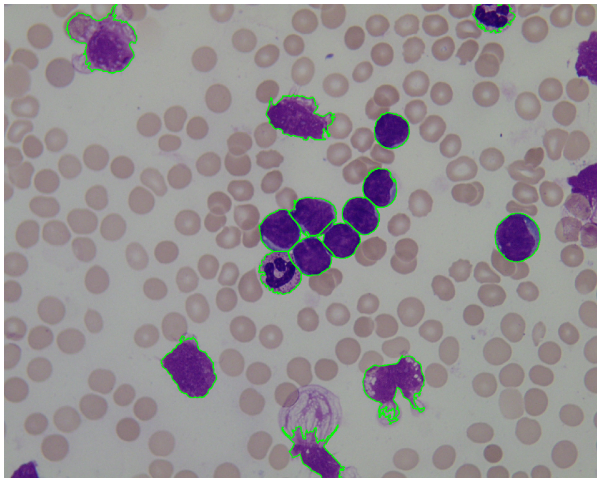
Vale dizer que neste trabalho não se considerou a implementação de nenhuma solução para este problema, portanto, é algo que pode ser melhorado.

5.2.1.4 Outras técnicas de segmentação

No artigo "Blob Detection and Deep Learning for Leukemic Blood Image Analysis" são mencionadas algumas técnicas de segmentação que vale a pena revisar. Além disso, os autores propõem uma segmentação que utiliza uma filtragem baseada em cores no

espaço HSV junto com o algoritmo "Blob detection" baseado no laplaciano da Gaussiana, e o algoritmo watershed (Di Ruberto et al., 2020). Este procedimento foi aplicado em experimentos obtendo resultados bons para algumas imagens tal como aquela ilustrada na Figura 69. Porém, ao executar o modelo com essa técnica em todas as imagens do banco de dados 1 e calcular as métricas descritas na seção 4.1 com esta nova segmentação, os resultados não demonstraram ser melhores em termos gerais. Porém, vale comentar que os parâmetros que envolvem o método podem ser ajustados de modo que os resultados tenham uma melhoria, mas isto não foi realizado neste trabalho.

Figura 69 – Im001_1.jpg do banco de dados 1. Resultado da segmentação descrita por Di Ruberto et al. (2020).



Fonte: Ruggero Donida Labati (2011). Editada pelo autor.

5.3 Conclusões

Este trabalho apresentou um modelo computacional para ajudar a realizar diagnósticos de LLA mais rapidamente, o que significa que pode ser usado como uma ferramenta para o especialista mas não como um sistema de diagnóstico sem intervenção humana. Na avaliação das predições feitas para cada célula em todo o banco de dados 1 obtivemos uma especifidade de 98.04 %. Na avaliação dos resultados por cada imagem de esfregaço de sangue do banco de dados 1 (por paciente), a especificidade foi de 100 %. Em conclusão, o modelo mostra um bom desempenho, pois o desejável em um diagnóstico médico é minimizar os falsos negativos (especificidade alta). Destacamos que, apesar de contar com dois bancos de dados, apenas um deles (banco de dados 1) está totalmente rotulado. Por causa disso, os resultados mostrados somente foram medidos com essas imagens. É provável que existam problemas com outro tipo de imagens que tenham sido tiradas com câmeras de baixa qualidade. Além disso, os

aspectos negativos mostrados na seção 5.2 também devem ser pontos a considerar na hora de utilizar o modelo. Finalmente, destacamos que o modelo implementado em Python está disponível gratuitamente no github através do link https://github.com/cxhernan/Modelo-Computacional-para-Diagnosticar-LLA/tree/master.

Referências

C. Di Ruberto, A. Loddo, and G. Puglisi. Blob detection and deep learning for leukemic blood image analysis. *Applied Sciences*, 10(3), 2020. ISSN 2076-3417. doi: 10.3390/app10031176. URL https://www.mdpi.com/2076-3417/10/3/1176. Citado 2 vezes nas páginas 13 e 76.

Elo7. Princípios de processamento de imagens: Uma introdução à convolução. https://elo7.dev/convolucao/, 2018. Acessado: 2021-02-25. Citado na página 26.

S. A. L. G. W. Zack, W. E. Rogers. Automatic measurement of sister chromatid exchange frequency. *The Journal of Histochemistry and Cytochemistry*, 1977. doi: Vol.25,No.7,pp. 741-753,1977. Citado na página 29.

S. Haykin. *Neural Networks and Learning Machines: Third Edition*. Pearson, Upper Saddle River, New Jersey 07458, 2009. ISBN 978-0-13-147139-9. Citado 3 vezes nas páginas 39, 43 e 44.

H. J. A. M. Heijmans. Mathematical morphology: A modern approach in image processing based on algebra and geometry. *SIAM Review*, 37(1):1–36, 1995. doi: 10.1137/1037001. URL https://doi.org/10.1137/1037001. Citado na página 30.

J. Z. Huiyu Zhou, Jiahua Wu. *Digital Image Processing: Part II*. Ventus Publishing ApS, 2010. ISBN 978-87-7681-542-4. Citado 3 vezes nas páginas 37, 38 e 40.

IBM. Machine learning. https://www.ibm.com/cloud/learn/machine-learning, 2020. Acessado: 2021-03-04. Citado na página 39.

Kasvi. Hematologia: Como é realizada a técnica de esfregaço de sangue? https://kasvi.com.br/esfregaco-de-sangue-hematologia/, 2021. Acessado: 2021-02-18. Citado na página 19.

MSD. Diagnóstico da leucemia linfoblástica aguda. https://www.msdmanuals.com/professional/hematology-and-oncology/leukemias/acute-lymphoblastic-leukemia-all, 2020. Acessado: 2020-10-28. Citado na página 17.

NCI. Leucemia linfoblástica aguda. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/acute-lymphoblastic-leukemia, 2021a. Acessado: 2021-02-18. Citado na página 18.

NCI. Linfoblasto. https://www.cancer.gov/publications/dictionaries/cancer-terms/def/lymphoblast, 2021b. Acessado: 2021-02-18. Citado na página 18.

NHS. ALL - Acute Lymphoblastic Leukemia. https://www.nhs.uk/conditions/acute-lymphoblastic-leukaemia/, 2021a. Acessado: 2021-02-18. Citado na página 18.

NHS. Diagnóstico da leucemia linfoblástica aguda. https://www.nhs.uk/conditions/acute-lymphoblastic-leukaemia/diagnosis/, 2021b. Acessado: 2021-02-18. Citado na página 19.

OpenCV. Histograms - 2: Histogram equalization. https://docs.opencv.org/3.1.0/d5/daf/tutorial_py_histogram_equalization.html, 2015. Acessado: 2021-02-25. Citado na página 29.

N. OTSU. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, man, and Cybernetics, Vol. SMC-9, NO. 1, January 1979*, 1979. doi: 0018-9472/79/0100-0062. Citado na página 29.

R. E. W. Rafael C. Gonzalez. *Digital Image Processing third edition*. Pearson, Upper Saddle River, New Jersey 07458, 2008. ISBN 0-13-168728-x978-0-13-168728-8. Citado 4 vezes nas páginas 23, 24, 31 e 32.

F. S. Ruggero Donida Labati, Vincenzo Piuri. All-idb: The acute lymphoblastic leukemia image database for image processing. *2011 18th IEEE International Conference on Image Processing*, 2011. doi: 978-1-4577-1303-3. Citado 9 vezes nas páginas 20, 62, 64, 67, 69, 70, 74, 75 e 76.

Rumelhart David E., Hinton Geoffrey E., Williams Ronald J. Learning representations by back-propagating errors. *Nature*, 1986. doi: 10.1038/323533a0. Citado na página 44.

K. A. Satoshi Suzuki. Topological structural analysis of digitized binary images by border following. *CVGIP*, 1985. doi: 0734-189X/85. Citado na página 36.

F. Scotti. Morphological classification of blood leucocytes by microscope images. *CIMSA 2004 - IEEE lnternational Conference on Computational Intelligence for Measurement Systems and Applications*, 2004. doi: 0-7803-9360-0. Citado na página 21.

F. Scotti. Robust segmentation and measurements techniques of white cells in blood microscope images. *IMTC 2006 - Instrumentation and Measurement Technology Conference*, 2006. doi: 0-7803-9360-0. Citado 3 vezes nas páginas 17, 21 e 50.

S. Shafique and S. Tehsin. Computer-aided diagnosis of acute lymphoblastic leukaemia. *Computational and Mathematical Methods in Medicine*, 2018:6125289, Feb 2018.

ISSN 1748-670X. doi: 10.1155/2018/6125289. URL https://doi.org/10.1155/2018/6125289. Citado 3 vezes nas páginas 17, 20 e 41.

P. Soille. *Morphological Image Analysis, Principles and Applications*. Springer, Silsoe Research Institute, Wrest Park, Silsoe, bedfordshire mk45 4hs, United Kingdom, 1999. ISBN 3-540-65671-5. Citado 2 vezes nas páginas 33 e 37.

P. Solai. Convolutions and backpropagations. https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c, 2018. Acessado: 2021-03-05. Citado na página 46.

M. A.-H. T Terwilliger. Acute lymphoblastic leukemia: a comprehensive review and 2017 update. *Blood Cancer J.*, 2017. doi: 10.1038/bcj.2017.53. Citado na página 19.

Towards Machine Learning. Deep learning series, p2: Understanding convolutional neural networks. https://medium.com/analytics-vidhya/convolution-padding-stride-and-pooling-in-cnn-13dc1f3ada26, 2018. Acessado: 2021-03-05. Citado na página 47.

A. Vidhya. Convolution, padding, stride, and pooling in cnn. https://medium.com/analytics-vidhya/convolution-padding-stride-and-pooling-in-cnn-13dc1f3ada26, 2020. Acessado: 2021-03-05. Citado na página 46.

L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. *EURASIP*, 1993. Citado na página 35.

H. Weller. Color spaces. https://hiweller.github.io/colordistance/color-spaces.html/, 2018. Acessado: 2021-02-25. Citado 2 vezes nas páginas 23 e 25.

J. Zhang. Gradient Descent based Optimization Algorithms for Deep Learning Models Training. *arXiv e-prints*, art. arXiv:1903.03614, Mar. 2019. Citado 2 vezes nas páginas 42 e 43.