UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Validação de Redes Generativas Condicionadas para Segmentação de Escavação Óptica em Imagens de Fundo de Retina.

Tales Henrique Carvalho

Itajubá, 9 de fevereiro de 2022

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Tales Henrique Carvalho

Validação de Redes Generativas Condicionadas para Segmentação de Escavação Óptica em Imagens de Fundo de Retina.

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Microeletrônica

Orientador: Prof. Dr. Danilo Henrique Spadoti

9 de fevereiro de 2022
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Validação de Redes Generativas Condicionadas para Segmentação de Escavação Óptica em Imagens de Fundo de Retina.

Tales Henrique Carvalho

Dissertação aprovada por banca examinadora em 09 de dezembro de 2021, conferindo ao autor o título de **Mestre em Ciências em Engenharia Elétrica.**

Banca Examinadora:
Prof. Dr. João Paulo Reus Rodrigues Leite
Prof. Dr. Carlos Alberto Ynoguti
MSc. Rafael Corrêa de Almeida

Itajubá
2021

Tales Henrique Carvalho

Validação de Redes Generativas Condicionadas para Segmentação de Escavação Óptica em Imagens de Fundo de Retina

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Trabalho aprovado. Itajubá, 09 de dezembro de 2021:

Prof. Dr. Danilo Henrique Spadoti
Orientador

Prof. Dr. João Paulo Reus Rodrigues Leite

Prof. Dr. Carlos Alberto Ynoguti

MSc. Rafael Corrêa de Almeida

Itajubá
9 de fevereiro de 2022

Agradecimentos

Primeiramente, gostaria de agradecer à empresa EyeConnect - Soluções em Telemedicina LTDA., em especial ao CEO MSc. Rafael C. Almeida, pela proposta geral de projeto em que a presente dissertação se baseia e pelo embasamento médico relevante para seu desenvolvimento. Também gostaria de agradecer ao professor Dr. Carlos H. V. Moraes pelo apoio técnico durante o desenvolvimento do modelo principal proposto neste projeto, que foi imprescindível para a finalização do mesmo. Agradeço também à colega Érika M. S. Tagima pelo auxílio durante o início do projeto, e ao Dr. Danilo H. Spadoti por ceder o espaço do LabTel (Laboratório de Telecomunicações), localizado na Universidade Federal de Itajubá, como ambiente de trabalho para o desenvolvimento de minha pesquisa. Por último, e não menos importante, gostaria de agradecer a todos os amigos e familiares que me apoiaram especialmente em momentos de dificuldade.

Resumo

A avaliação de imagens de fundo de retina é uma tarefa importante na oftalmologia, sendo um dos principais indicadores de condições oculares adversas. Dentre elas, o glaucoma se destaca pela necessidade de um diagnóstico durante seus estágios iniciais, para que o tratamento possa evitar sintomas graves na visão do paciente. Devido à escassez e ao alto custo de especialistas de retina, processos automáticos que identifiquem estruturas adversas em imagens de retina podem beneficiar a obtenção de diagnósticos. Para o glaucoma, é importante identificar a dimensão da escavação do nervo óptico, já que uma relação entre a escavação óptica e o disco óptico acima de 0,5 é um forte indicador da condição. Um processo automático depende da segmentação automática do disco e escavação óptica nas imagens de retina, que provê as dimensões para o cálculo da relação. O presente trabalho propõe o uso de modelos de redes neurais generativas adversárias condicionadas para a tarefa de segmentação da retina, baseado na arquitetura *Pix2Pix*. Para validar o modelo, comparou-se o modelo generativo proposto com os modelos convolucionais U-Net e M-Net, que representam os melhores resultados da literatura. Os resultados indicam que o modelo generativo é capaz de realizar a segmentação de retina com precisão comparável aos modelos do estado-da-arte, e é capaz de generalizar a tarefa com maior robustez.

Palavras-chaves: Aprendizado de Máquina. Glaucoma. Modelo Generativo. Redes Neurais. Retinografia. Segmentação de Imagens.

Abstract

The evaluation of retinal fundus images represents an important task in ophtalmology, as it provides indication of eye-related pathologies. Among them, glaucoma stands out due to the need of an early diagnosis and early treatment, so severe vision symptoms can be avoided. Due to the high cost and low availability of retina specialists, automatic processes that identify adverse structures in retinal fundus images can aid the process of obtaining diagnoses. For glaucoma diagnosis, it is important to identify the dimension of the optic cup in the retina nerve head, as a ratio between optic cup and optic disc greater than 0.5 is a strong indicator of the condition. An automatic process depends on the segmentation between optic disc and optic cup in retinal fundus images, which provides the structures dimensions for calculating such ratio. This work proposes the usage of conditional generative adversarial neural networks for the retina segmentation task, based on Pix2Pix architecture. To validate the model, the proposed model was compared to U-Net and M-Net convolutional models, that represent the best results in literature. Results indicate that the generative model is capable of providing retina segmentation with precision comparable with state-of-art models, and it is capable of doing such task with higher robustness.

Key-words: Machine Learning. Glaucoma. Generative Model. Neural Networks. Retinography. Image Segmentation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ACM	*Active Contour Modeling*, do inglês, Modelagem Ativa de Contorno	18
ANN	*Artificial Neural Network*, do inglês, Rede Neural Artificial	15, 18–27, 30, 32, 35, 42
BCE	*Binary Cross-Entropy*, do inglês, Entropia Cruzada Binária	24, 27, 38
CBO	Conselho Brasileiro de Oftalmologia	16
CDR	*Cup to Disc Ratio*, do inglês, Razão da Escavação sobre o Disco	17–19, 32, 38–40, 47–49
CGAN	*Conditional Generative Adversarial Networks*, do inglês, Redes Adversárias Generativas Condicionadas	28–30, 32, 49
CHT	*Circular Hough Transform*, do inglês, Transformada Hough Circular	18
CNN	*Convolutional Neural Network*, do inglês, Rede Neural Convolucional	18, 19, 26, 27, 30, 32
DL	*Dice Loss*, do inglês, Perda de Dice	24, 38, 39, 42–45
DSC	*Dice Similarity Coefficient*, do inglês, Coeficiente de Similaridade Dice	19, 39, 43, 44, 46, 47, 49
GAN	*Generative Adversarial Networks*, do inglês, Redes Adversárias Generativas	13, 27, 28, 30
IoU	*Intersection over Union*, do inglês, Interseção sobre União	18, 19
IQR	*Interquartile Range*, do inglês, Intervalo Interquartil	46, 48
LReLU	*Leaky Rectified Linear Unit*, do inglês, Unidade Linear Retificada com Vazamento	22, 23
M-LD	Modelo de Localização de Disco	32–37, 39, 42, 45
M-SR	Modelo de Segmentação de Retina	32–39, 42, 45–47
MAE	*Mean Absolute Error*, do inglês, Erro Absoluto Médio	40, 47, 49
ML	*Machine Learning*, do inglês, Aprendizado de Máquina	15, 16, 18
MLP	*Multi Layer Perceptron*, do inglês, *Perceptron* de Múltiplas Camadas	23
MSE	*Mean Squared Error*, do inglês, Erro Quadrático Médio	24
OMS	Organização Mundial da Saúde	16
PyPI	*Python Package Index*, do inglês, Índice de Pacotes *Python*	40
ReLU	*Rectified Linear Unit*, do inglês, Unidade Linear Retificada	22, 23
SGD	*Stochastic Gradient Descent*, do inglês, Descida do Gradiente Estocástico	38

Lista de símbolos

K	Matriz *Kernel* da operação de convolução	24
Θ	Conjunto de parâmetros treináveis de uma rede neural	24
$\Theta_{\mathcal{D}}$	Conjunto de parâmetros treináveis do módulo discriminador	28
$\Theta_{\mathcal{G}}$	Conjunto de parâmetros treináveis do módulo gerador	28
η	Taxa de aprendizado do processo de treinamento de uma rede neural	25
$\hat{y}$	Previsão da saída gerada pela rede neural	24
$\mathbb{E}$	Esperança da saída de uma rede neural	27
$\mathcal{D}$	Módulo discriminador de uma GAN	28
$\mathcal{G}$	Módulo gerador de uma GAN	28
$\mathcal{L}$	Função de perda	24
$\mathcal{X}$	Conjunto de dados de entrada	27
$\mathcal{Z}$	Conjunto de dados de distribuição aleatória	27
σ	Função de ativação de um neurônio artificial	21
d_x	Comprimento da matriz que representa um dado de entrada arbitrário	26
d_y	Altura da matriz que representa um dado de entrada arbitrário	26
k_x	Comprimento da matriz *Kernel*	25
k_y	Altura da matriz *Kernel*	25
n_c	Número de canais de uma matriz de dados	25
n_f	Número de filtros da operação de convolução	26
s_x	Comprimento do *Stride* da operação de convolução	26
s_y	Altura do *Stride* da operação de convolução	26
x	Dado de entrada em uma rede neural	24
y	Saída esperada da rede neural	28
z	Variável latente do módulo gerador	27

Sumário

1 Introdução

Com o progressivo aumento de poder computacional e redução de custo de processamento, a análise de imagens digitais a partir de sistemas computacionais vem se tornando mais popular a cada ano. Também conhecida como processamento de imagem, este processo consiste em descrever e reconhecer conteúdos simbólicos a partir de uma imagem digital, simulando funções da visão humana [1]. Atualmente, a técnica descrita é aplicada em diversas áreas, como tomografia computacional, exploração geofísica, testes não destrutivos, sensoriamento remoto e robótica industrial [2]. O processamento de imagem, de forma geral, pode ser descrito por seis tarefas [3]:

1. **Pré-processamento**: consiste em aplicar filtros na imagem original como um tratamento inicial, como supressão de ruídos, e realce e detecção de contornos;
2. **Redução de dados**: a imagem é comprimida e suas partes relevantes são extraídas;
3. **Segmentação**: a partir das partes extraídas da imagem, aspectos como texturas, cores e iluminação são segmentadas;
4. **Reconhecimento de objetos**: a segmentação dos aspectos da imagem são comparadas com modelos pré-definidos para se identificar objetos conhecidos;
5. **Interpretação de imagem**: a partir do conjunto dos objetos reconhecidos, a cena apresentada pela imagem é interpretada;
6. **Otimização**: parâmetros envolvidos nos outros passos são alterados para se obter melhor eficiência do algoritmo como um todo.

A etapa de otimização usualmente envolve a aplicação de algoritmos de ML (*Machine Learning*, do inglês, Aprendizado de Máquina), que melhora a eficiência do processamento de imagem recursivamente, a partir do ajuste dos parâmetros usados nas outras etapas do processo. Por sua vez, o ML explora o uso de algoritmos que podem gerar um modelo de predição de dados a partir da retroalimentação do modelo existente, que pode condicionar sua saída comparando-a com um resultado esperado. Dentre as abordagens que aplicam o conceito de ML, uma das mais relevantes é a ANN (*Artificial Neural Network*, do inglês, Rede Neural Artificial), cujo modelo se baseia na conexão dos neurônios de um cérebro humano [4]. Neste contexto, os neurônios humanos são traduzidos em equações matemáticas que transportam e modificam os dados ao longo da rede, a fim de aproximar os resultados obtidos aos resultados esperados.

Técnicas de processamento de imagem, em conjunto com o ML, contribuem significativamente para aplicações na medicina que envolvem a interpretação de imagens. A partir de exames médicos, o procedimento pode auxiliar na geração de modelos, diagnósticos e prognósticos [5]. Em específico, a oftalmologia se beneficia do processamento de imagens com a análise de exames de retinografias. Dentre as patologias que podem ser identificadas por meio deste exame, o glaucoma se destaca pela necessidade de um diagnóstico rápido para que o paciente possa ser tratado. Dentre outras características, o diagnóstico do glaucoma pode ser feito pela avaliação do nervo óptico, obtido pelo exame de retinografia, considerando o formato e tamanho do disco óptico e da escavação do disco óptico [6].

1.1 Motivação

Segundo uma pesquisa realizada pela OMS (Organização Mundial da Saúde) em 2002, estimou-se que existiam 161 milhões de pessoas no mundo com deficiências visuais, sendo 37 milhões consideradas cegas [7]. Dentre as causas que levam a essa condição, destacam-se, em ordem de importância, a catarata, o glaucoma, a degeneração macular relacionada à idade, a retinopatia diabética e a conjuntivite granulomatosa. Enquanto a catarata, primeira causa de deficiências visuais, pode ser tratada a partir de cirurgia, o glaucoma, posicionado em segundo na lista, é uma condição que, se não diagnosticada e tratada a partir de seu estágio inicial, pode causar cegueira irreversível [8]. Segundo o CBO (Conselho Brasileiro de Oftalmologia), a incidência do glaucoma no Brasil é estimada de 1 a 2% na população geral, e de até 7% na população acima de 70 anos de idade [9].

O glaucoma é definido como um grupo de neuropatias caracterizado pela degeneração progressiva das células ganglionares retinianas [10]. Estas estruturas são localizadas na retina humana, que compõe a camada interna ou sensorial do globo ocular, e são responsáveis por transmitir para o cérebro os impulsos elétricos estimulados pela luz recebida no olho [11]. A Figura 1 apresenta as estruturas que compõem o globo ocular humano, incluindo a localização da retina. A degeneração resulta em alterações no disco óptico localizado no fundo da retina, observadas pelo formato e crescimento da escavação óptica, surgimento de hemorragias na região peripapilar, e perda de camadas de fibras dos nervos retinianos [12]. A partir de uma imagem do fundo da retina, que pode ser obtida por exames de retinografia, estereofoto de papila ou tomografia de coerência óptica, pode-se observar as alterações no disco óptico que indicam o surgimento do glaucoma. A Figura 2 exemplifica, a partir de um exame de retinografia, as alterações observadas no disco óptico causadas pelo glaucoma.

Tendo em vista as alterações típicas causada pelo glaucoma, o diagnóstico da condição pode ser dado pela, dentre outras características, relação entre o diâmetro da

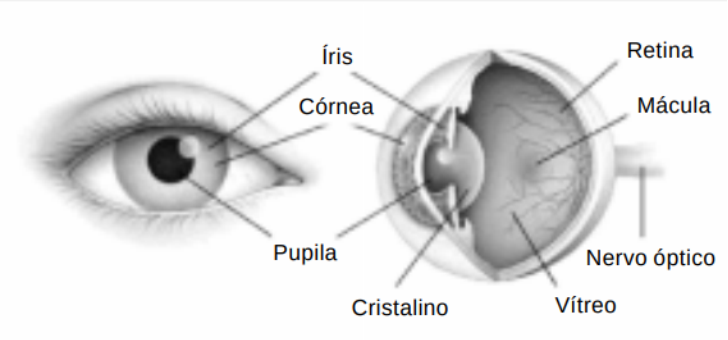


Figura 1 – Corte esquemático da anatomia de um olho humano. Adaptado de R. Graziano e C. Leone [11].

(a) Escavação no disco óptico

(b) Hemorragias peripapilares

(c) Imperfeições nos nervos retinianos

Figura 2 – Alterações típicas no disco óptico causadas pelo glaucoma observadas em exames de retinografia. Adaptado de J. Orlando et al. [13]

escavação óptica e do disco óptico, definido na literatura como CDR (*Cup to Disc Ratio*, do inglês, Razão da Escavação sobre o Disco). O Ministério da Saúde, a partir da Portaria Conjunta nº 11, de 02 de Abril de 2018, define que uma relação CDR vertical maior do que 0,4 pode ser indicativo de glaucoma. A Portaria também define que a escavação deve ser identificada pela deflexão dos vasos sanguíneos e pela visibilidade da lâmina cribriforme, tanto na horizontal quanto na vertical [6].

Como observado, o diagnóstico do glaucoma depende de análise técnica em que apenas médicos especializados em condições relacionadas à retina são capazes de realizar, muitas vezes de forma manual e aproximada. Por se tratar de uma especialidade da área médica, estes profissionais são escassos para essa demanda, composta, também, por pacientes que podem vir a desenvolver o glaucoma, sobretudo em países menos desenvolvidos. No SUS (Sistema Único de Saúde) o tempo de espera entre a realização de exame de retinografia e uma consulta a um oftalmologia para tratamento de glaucoma pode superar 800 dias no Rio de Janeiro, tempo que pode ser crucial para o agravamento da condição[1]. Portanto, médicos e clínicas médicas que não têm a especialidade necessária podem se beneficiar de métodos automáticos que indicam possíveis alterações na estrutura retiniana do paciente, permitindo aos profissionais um diagnóstico mais assertivo e rápido.

[1] Segundo notícia publicada no G1, disponível em <https://g1.globo.com/rj/rio-de-janeiro/noticia/2021/06/16/fila-de-espera-do-sisreg-no-rio-tem-mais-249-mil-pedidos-de-consultas-exames-e-cirurgias-paciente-com-glaucoma-espera-mais-de-800-dias.ghtml>

1.2 Trabalhos Relacionados

Os primeiros esforços relacionados à identificação automática de estruturas retinianas em imagens do fundo da retina se deram com o processo de localização autônoma do disco óptico. R. Chrástek et al. [14] abordaram a tarefa usando o método de detecção de contornos Canny em conjunto com CHT (*Circular Hough Transform*, do inglês, Transformada Hough Circular) em imagens filtradas de retinografia, obtendo assim 97,3% de taxa de sucesso na localização do disco óptico e 81,7% na definição do contorno do disco óptico, avaliado por inspeção visual profissional. Uma abordagem diferente surgiu com A. Hoover e M. Goldbaum [15], em que consideram que o disco óptico representa o ponto focal formado pela rede dos vasos sanguíneos da retina. Ao identificar a convergência dos vasos sanguíneos, os autores obtiveram 89% de taxa de sucesso na localização do disco óptico, considerando uma margem de erro aceitável de 60 *pixels* em torno da localização esperada, em imagens de resolução 605×700 *pixels*.

Com o desenvolvimento de métodos que determinam o contorno do disco óptico na retina, surgiu-se a necessidade de adotar métricas de segmentação que avaliam a precisão das regiões encontradas. Para isso, pesquisas passam a adotar a métrica IoU (*Intersection over Union*, do inglês, Interseção sobre União) que, como o nome sugere, compara o quanto a interseção entre a região segmentada e a região esperada representa sobre a união entre estas regiões, em uma escala de 0 (nenhuma interseção entre as regiões) a 1 (regiões idênticas). Desta forma, A. Aquino et al. [16] otimiza o método baseado na aplicação de CHT para segmentação de disco óptico e obtém IoU médio de 0,82 ao avaliar usando a base de dados aberta MESSIDOR [17]. J. Cheng et al. [18], por outro lado, se baseiam no mesmo método e validam usando a base de dados ORIGA-light [19], obtendo IoU médio de 0,90.

Visando avaliar métodos iterativos para resolver o problema de segmentação do disco óptico, o trabalho de C. Muramatsu et al. [20] apresenta três métodos baseados em ML: determinação de contorno usando ACM (*Active Contour Modeling*, do inglês, Modelagem Ativa de Contorno), segmentação usando agrupamento difuso, e segmentação usando ANN. Os resultados indicam, a partir da avaliação do IoU das segmentações obtidas, que os métodos que aplicam ACM e ANN obtêm levemente maior acurácia do que o método baseado em agrupamento difuso. Apesar das métricas de IoU obtidas (entre 0,86 e 0,89) não representem melhora significativa em relação aos métodos adotados por outras pesquisas, os autores observaram que os modelos eram capazes de generalizar a segmentação de modo efetivo, podendo ser utilizado para determinar CDR, caso um método de detecção de escavação óptica fosse desenvolvido.

Enquanto alguns trabalhos adotaram métodos baseados em ACM e agrupamento difuso para segmentar a escavação do disco óptico em imagens de retinografia [21, 22, 23], observa-se que os melhores resultados se dão pelo uso de CNN (*Convolutional Neural*

Network, do inglês, Rede Neural Convolucional), que expande o conceito de ANN aplicando camadas de convolução. A. Sevastopolsky [24] adota um modelo baseado na arquitetura U-Net [25] para realizar a segmentação entre a escavação e o disco óptico. Como métricas de avaliação, além de se utilizar a métrica IoU para comparar resultados obtidos de resultados esperados, o trabalho também considera a métrica DSC (*Dice Similarity Coefficient*, do inglês, Coeficiente de Similaridade Dice) [26], que também representa um índice de acurácia de interseção espacial em uma escala de 0 (nenhuma interseção entre as regiões) a 1 (regiões idênticas). Ao aplicar o método nas bases de dados abertas RIM-ONE e DRISHTI-GS, o autor obtém IoU de 0,89 e DSC de 0,94 na segmentação de disco óptico, e IoU de 0,72 e DSC de 0,83 na segmentação de escavação óptica.

Outra implementação de modelos baseados em CNN é observada no trabalho de H. Fu et al. [27] Os autores propõem uma arquitetura, nomeada M-Net, capaz de segmentar o disco e a escavação óptica em imagens de retina. Ao se comparar com a U-Net tradicional, a M-Net separa a imagem de entrada em múltiplas matrizes que são usadas ao longo da arquitetura convolucional. Ao aplicar nas bases de dados ORIGA e SCES, os autores obtêm como resultado IoU de 0,93 e DSC de 0,98 na segmentação de disco óptico, e IoU de 0,77 e DSC de 0,93 na segmentação de escavação óptica, representando o estado-da-arte na tarefa de segmentação de retina.

1.3 Objetivos

O objetivo deste trabalho é propor e validar técnicas de redes neurais generativas condicionadas no processamento de imagens de retinografias, visando indicar alterações na estrutura retiniana do paciente que podem caracterizar a condição de glaucoma.

Os objetivos específicos deste trabalho são:

- Desenvolver modelos generativos condicionados capazes de segmentar imagens de retinografia;
- Desenvolver algoritmo que calcula a CDR horizontal e vertical a partir dos resultados de segmentação do modelo generativo;
- Comparar resultados dos modelos desenvolvidos com modelos convolucionais da literatura.

1.4 Estrutura do trabalho

A presente dissertação está estruturada em cinco capítulos. O Capítulo 1 introduz a motivação do tema e os trabalhos relacionados ao objetivo do projeto. O Capítulo 2

explora a teoria e os modelos de ANNs que são relevantes para o desenvolvimento do projeto, cujo modelo é descrito no Capítulo 3. Os resultados originados da aplicação do modelo proposto são apresentados no Capítulo 4, e as conclusões relacionadas, no Capítulo 5.

1.5 Publicações

Durante o desenvolvimento do presente trabalho, resultados prévios foram publicados no *17th International Symposium on Medical Information Processing and Analysis*, pertencente à conferência SIPAIM 2021, em artigo de título "*Application of Conditional GAN Models in Optic Disc/Optic Cup Segmentation of Retinal Fundus Images*" e autores Tales H. Carvalho, Carlos H. V. Moraes, Rafael C. Almeida e Danilo H. Spadoti. O artigo foi publicado em modalidade pôster e será apresentado dia 19 de novembro de 2021.

Outras publicações foram realizadas durante o período de Mestrado, relacionados a projetos em execução no Laboratório de Telecomunicações, listadas em ordem cronológica abaixo:

- CARVALHO, Tales H.; HIROTA, Felipe H.; ESPER, Miguel M. P.; FRÉ, Gabriel L. S.; SPADOTI, Danilo H. Análise de Consumo de Energia de Dispositivo Classe A em Rede LoRaWAN. *MOMAG 2020, 19º SBMO – Simpósio Brasileiro de Micro-ondas e Optoeletrônica e 14º CBMag – Congresso Brasileiro de Eletromagnetismo*. 2020;
- ESPER, Miguel M. P.; CARRANZA, Pedro B.; POSSATTO, Henrique M.; FRÉ, Gabriel L. S.; CARVALHO, Tales H.; SPADOTI, Danilo H. Sistema de medição automática em redes sem fio para Internet das Coisas (IoT). *MOMAG 2020, 19º SBMO – Simpósio Brasileiro de Micro-ondas e Optoeletrônica e 14º CBMag – Congresso Brasileiro de Eletromagnetismo*. 2020;
- PEREIRA, Vitor H. B.; CLEMENTE, Danilo L. T.; CARVALHO, Tales H.; FRÉ, Gabriel L. S.; ABREU, Reinaldo L.; RIBEIRO, Mikaelle O.; SPADOTI, Danilo H. Energy consumption analysis of Sub-1GHz IoT device. *International Microwave and Optoelectronics Conference* (IMOC) 2021. 2021;
- CLEMENTE, Danilo L. T.; PEREIRA, Vitor H. B.; CARVALHO, Tales H.; ABREU, Reinaldo L.; SPADOTI, Danilo H. Comparing communication protocols for interfacing IoT hardware. *6th Workshop on Communication Networks and Power Systems* (WCNPS) 2021. 2021.

2 Redes Neurais Artificiais

Uma ANN (*Artificial Neural Network*, do inglês, Rede Neural Artificial) consiste em um modelo computacional baseado no funcionamento de um cérebro humano, com o objetivo de modelar a forma que este realiza uma certa atividade ou função de interesse [28]. Desde sua criação e popularização, o conceito foi aplicado para resolver problemas de reconhecimento de fala e escrita, visão computacional, previsão de mercado, identificação de anomalias, entre outros [29].

A arquitetura básica de uma ANN, responsável por tratar as entradas do modelo e obter uma saída, é baseada na conexão entre neurônios artificiais, expressos por equações matemáticas que definem suas conexões sinápticas [30]. A saída y_j do neurônio arbitrário j relaciona-se com os valores de ativação x_i, $i \in [1, n]$, dos n dados de entrada ou neurônios conectados à sua entrada a partir dos pesos $w_{j,i}$ e do limiar de ativação b_j [31], segundo o diagrama da Figura 3.

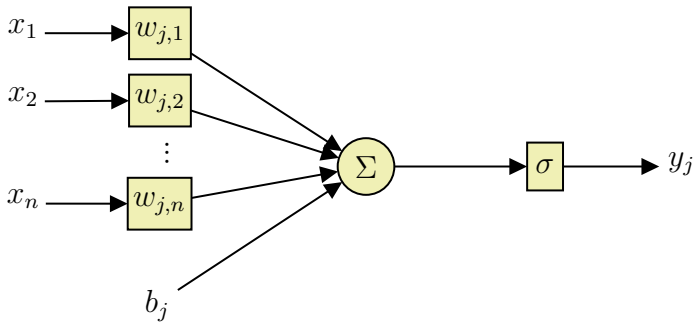


Figura 3 – Diagrama que representa as conexões sinápticas que compõem a saída de um neurônio artificial.

Nota-se que a combinação das entradas do neurônio artificial é aplicada no bloco σ, que representa uma função de ativação a qual ajusta a saída gerada dentro de um intervalo definido, seguindo o comportamento de um neurônio biológico [32].

Dentre as funções de ativação usadas na literatura, destacam-se:

1. **Sigmoide**: Também conhecida por função logística, a função sigmoide traduz uma entrada real para o intervalo $(0, 1)$, e é representada pela equação

$$\sigma_{\text{sigm}}(x) = \frac{1}{1 + \exp\left(-x/\beta\right)}, \tag{2.1}$$

em que $\beta \in (0, \infty)$ representa a elevação da função sigmoide em seu ponto de inflexão.

2. **Tangente hiperbólica**: É uma versão redimensionada da função sigmoide, com imagem em $(-1, 1)$, que pode ser representada por

$$\sigma_{\tanh}(x) = \frac{1 - \exp(-\beta\ x)}{1 + \exp(-\beta\ x)}, \tag{2.2}$$

   em que $\beta \in (0, \infty)$ também é associada com a elevação da função no ponto de inflexão [29];

3. **ReLU**: A ReLU (*Rectified Linear Unit*, do inglês, Unidade Linear Retificada) é uma função monotônica que anula valores negativos do domínio, representada por

$$\sigma_{\text{ReLU}}(x) = \begin{cases} 0, & \text{se } x < 0 \\ x, & \text{caso contrário;} \end{cases} \tag{2.3}$$

4. **LReLU**: A LReLU (*Leaky Rectified Linear Unit*, do inglês, Unidade Linear Retificada com Vazamento) estende o conceito da função ReLU, mas permite que ativações negativas se propaguem pela rede, a partir da equação

$$\sigma_{\text{LReLU}}(x) = \begin{cases} \alpha\ x, & \text{se } x < 0 \\ x, & \text{caso contrário,} \end{cases} \tag{2.4}$$

   em que $\alpha \in (0, 1)$ representa a constante de vazamento da função.

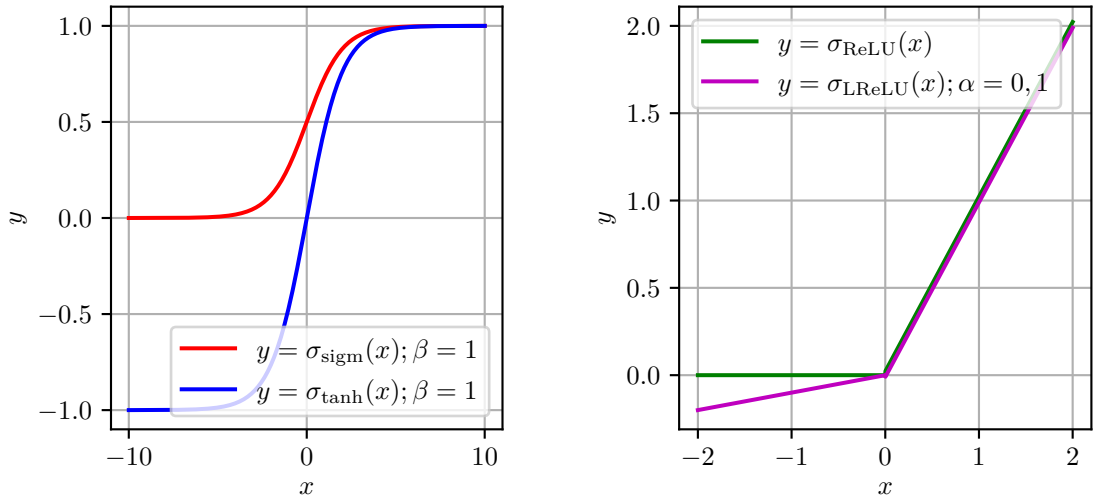
A Figura 4 compara as famílias de funções de ativação mencionadas. Usualmente, a família de funções baseadas em ReLU são adotadas como escolhas padrões para a arquitetura interna da rede, mas é praticamente impossível prever quais serão as melhores opções para uma rede específica [33].

2.1 Camadas de redes neurais

A composição dos neurônios de uma ANN pode ser dividida em três partes, também denominadas por camadas:

1. ***Input layer* (do inglês, camada de entradas)**: responsável por receber informações de entrada por meio de dados, sinais e medições do ambiente externo. As entradas são normalizadas levando em conta os limites das funções de ativação adotadas no modelo, para se obter melhor precisão numérica para as operações matemáticas realizadas pela rede;

2. ***Hidden layers* (do inglês, camadas ocultas)**: são compostas por neurônios responsáveis por extrair padrões associados com o processo ou sistema analisado. Estas camadas representam a maior parte do processamento de uma ANN;



(a) (b)

Figura 4 – Comparação entre gráficos de famílias de funções de ativação: (a) Funções sigmoide e tangente hiperbólica; (b) Funções ReLU e LReLU.

3. ***Output layer* (do inglês, camada de saída)**: também composta de neurônios, esta camada é responsável por produzir e expor as saídas da rede, que combina o resultado do processamento das camadas anteriores [30].

Enquanto que a composição das camadas *Input layer* e *Output layer* dependem da aplicação na qual a ANN está inserida, o tamanho e as conexões das *Hidden layers* dependem da arquitetura de modelo a ser adotada. Como exemplo, a Figura 5 apresenta uma MLP (*Multi Layer Perceptron*, do inglês, *Perceptron* de Múltiplas Camadas) totalmente conectada.

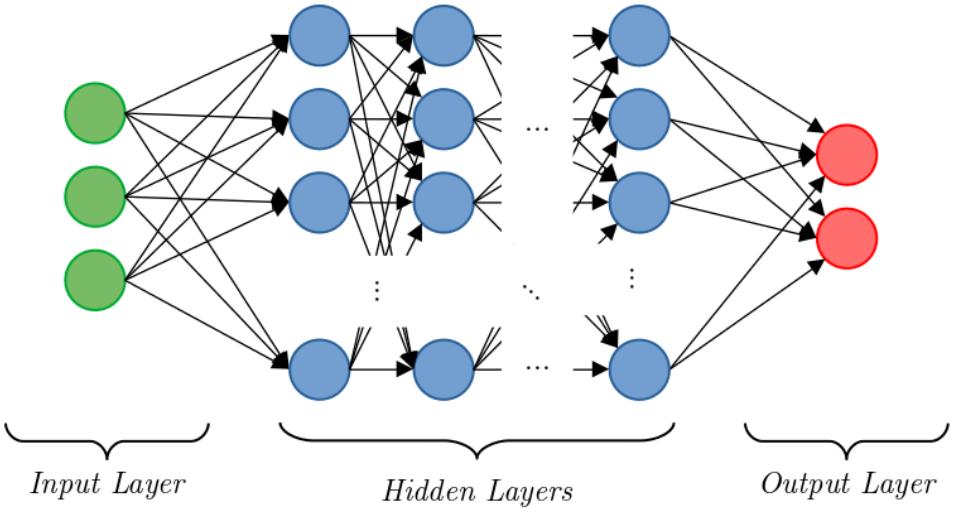


Figura 5 – Representação das conexões das camadas de neurônios artificiais em um modelo MLP totalmente conectado.

Cada um dos neurônios artificiais das *Hidden layers* computa as conexões sinápticas apresentadas na Figura 3 de forma sequencial até a *Output layer*, que traduz as conexões em saídas do modelo. Considerando o conjunto de neurônios que compõem uma camada do modelo e a conexão sináptica que representa suas ativações, pode-se representar as ativações de uma camada arbitrária $Y^{(j)}$, composta por n neurônios, relacionando com a camada anterior $Y^{(j-1)}$, de m neurônios, em forma matricial por

$$\underset{n\times 1}{Y^{(j)}} = f^{(j)}\left(\underset{m\times 1}{Y^{(j-1)}}\right) = \sigma\left(\underset{n\times m}{W^{(j)}} \cdot \underset{m\times 1}{Y^{(j-1)}} + \underset{n\times 1}{B^{(j)}}\right), \tag{2.5}$$

em que $W^{(j)}$ representa a matriz de pesos aplicados entre a camada $j-1$ e j, e $B^{(j)}$ representa a matriz de limiares de ativação da camada j.

A Equação (2.5) é aplicada na conexão entre todas as l camadas ocultas que compõem a rede neural, partindo da conexão da *Input Layer*, $Y^{(0)}$, com a primeira *Hidden Layer*, $Y^{(1)}$, até a conexão da última *Hidden Layer*, $Y^{(l-1)}$, com a *Output Layer*, $Y^{(l)}$. Desta forma, observa-se que os parâmetros ajustáveis são compostos pelas matrizes $W^{(j)}$ e $B^{(j)}$ para cada camada $j \in [1, l]$. As matrizes compõem o conjunto Θ, que representam os parâmetros treináveis de uma ANN. Com isso, pode-se representar uma ANN como uma função $\mathcal{R}(x;\Theta)$, onde $\mathcal{R}(\cdot) = f^{(l)}(f^{(l-1)}(...f^{(2)}(f^{(1)}(\cdot))...))$, que gera uma saída $\hat{y}$ a partir da entrada x e do conjunto de parâmetros Θ.

2.2 Treinamento da rede neural

A otimização de uma ANN $\mathcal{R}(x;\Theta)$ consiste no processo de treinamento da rede, que envolve aplicar uma sequência de passos para ajustar o conjunto Θ e generalizar a solução baseado em valores desejados [30]. O objetivo do processo é minimizar a função de custo $J(\Theta)$ composta pela soma dos erros entre a saída da rede $\mathcal{R}$ e os valores esperados y_k para todas as amostras de um lote de treinamento K, representada por

$$J(\Theta) = \sum_{k} \mathcal{L}\left(\mathcal{R}(x_k;\Theta), y_k\right), \tag{2.6}$$

em que $\mathcal{L}$ representa a função de perda que compara a saída $\mathcal{R}(x_k;\Theta)$, gerada pela entrada de treinamento x_k, à saída esperada y_k. A escolha da função de perda na equação (2.6) depende do comportamento desejado para a rede neural e do tipo de dado de entrada e saída da rede. Dentre as funções mais utilizadas como funções de perda, pode-se destacar MSE (*Mean Squared Error*, do inglês, Erro Quadrático Médio), ideal para modelos de regressão, BCE (*Binary Cross-Entropy*, do inglês, Entropia Cruzada Binária) e DL (*Dice Loss*, do inglês, Perda de Dice) [34], ideais para modelos de segmentação binária.

A minimização da função de custo $J(\Theta)$ usualmente se dá pelo algoritmo *Gradient Descent* (do inglês, Descida do Gradiente), ou por variações do mesmo [35]. O algoritmo

original se baseia em atualizar o conjunto Θ a partir do negativo do gradiente da função de custo em relação aos parâmetros, ou seja,

$$\Theta \leftarrow \Theta - \eta \nabla_{\Theta} J(\Theta), \tag{2.7}$$

em que η representa a taxa de aprendizado do treinamento que determina o tamanho do passo tomado para atingir um mínimo local da função de custo, que deve ser ajustado para evitar comportamento divergente no processo de treinamento. A equação (2.7) é aplicada em cada lote que compõe o conjunto de amostras para treinamento, para um ajuste gradativo do conjunto Θ. O procedimento é denominado como época de treinamento, e representa uma iteração do processo cíclico do treinamento de uma ANN [30].

Para verificar se a rede é capaz de generalizar soluções em um nível aceitável, um segundo conjunto de amostras, denominado conjunto de validação, é aplicado na rede neural. A eficiência da rede neural aplicada ao conjunto de validação pode ser verificada pela função de custo da equação (2.6), comparando a seu custo quando aplicada ao conjunto de treinamento. Caso a rede neural apresente alto valor de custo para ambos conjuntos, a rede não foi capaz de construir um modelo adequado para resolver o problema, que pode indicar *Underfitting* (do inglês, Sub-Ajuste) dos parâmetros. Caso a rede apresente baixo valor de custo para o conjunto de treinamento, mas alto valor de custo para o conjunto de validação, a rede neural generalizou aspectos do conjunto de treinamento que não são inerentes ao problema analisado, então falha ao generalizar o conjunto de validação. Esta situação é denominada *Overfitting* (do inglês, Super-Ajuste) [36].

A configuração desejada para o conjunto de parâmetros de uma ANN se dá quando o processo de treinamento evita *Underfitting* e *Overfitting*, ou seja, quando a rede apresenta baixo valor de custo para ambos conjuntos de amostras. Entre outros aspectos relacionados à arquitetura da rede, este ponto pode ser visto ao acompanhar a função de custo de ambos conjuntos em relação ao número de épocas de treinamento da rede, como visto na Figura 6. Observa-se que o modelo causa *Underfitting* nas primeiras épocas de treinamento, causa *Overfitting* após um certo número de épocas, e atinge comportamento ótimo entre as duas situações [36].

2.3 Camadas convolucionais

Ao considerar o processamento de dados com topologia multicanal com duas dimensões espaciais, como é o caso de imagens cujos *pixels* em cada canal de cor são representados por matrizes de duas dimensões, o uso de operações convolucionais se mostra efetivo [33]. Considerando uma matriz de entrada I de n_c canais e uma matriz *Kernel* K, de dimensões $k_x \times k_y \times n_c$, os elementos da saída da operação de convolução discreta

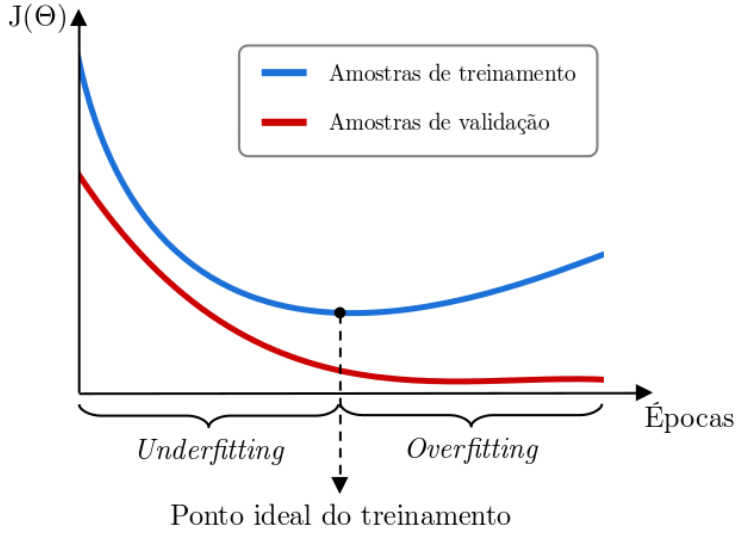


Figura 6 – Representação do comportamento de *Underfitting* e *Overfitting* em gráfico de função de custo em relação ao número de épocas de treinamento de uma ANN.

bidimensional entre I e K são definidos por

$$(I * K)_{i,j} = \sum_{c=1}^{n_c} \sum_{m=0}^{k_x-1} \sum_{n=0}^{k_y-1} K_{m,n,c} I_{i-m,j-n,c}. \tag{2.8}$$

A matriz *Kernel* é composta por parâmetros escalares que podem mapear padrões da imagem I.

A operação de convolução discreta pode ser combinada com o conceito de neurônio artificial para construir uma camada de convolução. Neste caso, define-se o número de operações convolucionais presentes na camada pelo parâmetro número de filtros (n_f), que também define o número de canais da matriz de saída. Define-se o canal i da matriz de saída da camada de convolução arbitrária j, $Y_i^{(j)}$, relacionando com a camada anterior, $Y^{(j-1)}$, de dimensão $d_x \times d_y$ e n_c canais, a matriz *Kernel* K, de dimensão $k_x \times k_y \times n_c$, e a matriz de limiares de ativação $B^{(j)}$, a partir da equação

$$\underset{d_x \times d_y}{Y_i^{(j)}} = \sigma \left(\underset{d_x \times d_y \times n_c}{Y^{(j-1)}} * \underset{k_x \times k_y \times n_c}{K^{(i,j)}} + \underset{d_x \times d_y \times n_c}{B^{(i,j)}} \right), \forall\ i \in [1, n_f]. \tag{2.9}$$

Desta forma, a matriz de saída é composta por $Y^{(j)} = [Y_1^{(j)}, Y_2^{(j)}, ..., Y_{n_f}^{(j)}]$. Redes neurais compostas por pelo menos uma camada de convolução são denominadas CNN (*Convolutional Neural Network*, do inglês, Rede Neural Convolucional).

Para reduzir custos computacionais, camadas convolucionais podem ser adaptadas para que a matriz *Kernel* não use todas as posições da convolução, e que a matriz resultante tenha dimensões menores. O processo é realizado com a operação *Stride*, caracterizada pelo vetor $[s_x, s_y]$, que indica quantas posições da matriz de convolução serão ignoradas em cada direção [33]. Em uma operação convolucional com *Strides* s_x e s_y aplicada em uma matriz de entrada de dimensões $d_x \times d_y$ e *Kernel* de dimensões $k_x \times k_y$,

a matriz resultante tem dimensões $\left\lfloor \frac{d_x - k_x}{s_x} \right\rfloor \times \left\lfloor \frac{d_y - k_y}{s_y} \right\rfloor$ [37]. Uma convolução bidimensional com *Strides* também é conhecida por convolução codificadora, e pode ser totalmente descrita pelos parâmetros n_f, k_x, k_y, s_x e s_y.

Por outro lado, para que seja possível obter como saída uma matriz de mesmas dimensões em relação à entrada do modelo, é considerada a convolução decodificadora, composta pela operação de convolução transposta. A operação consiste em aplicar o processo inverso da convolução, construindo uma matriz de tamanho maior à matriz de entrada. Uma convolução transposta entre uma matriz de entrada de dimensões $d_x \times d_y$ e uma matriz *Kernel* de dimensões $k_x \times k_y$, com *Strides* s_x e s_y, gera uma matriz de saída de dimensões $s_x(d_x - 1) + k_x \times s_y(d_y - 1) + k_y$ [37]. Da mesma forma que a convolução codificadora, a convolução decodificadora pode ser totalmente descrita pelos parâmetros n_f, k_x, k_y, s_x e s_y.

Em camadas convolucionais, os elementos das matrizes *Kernels* a das matrizes de limiares de ativação representam os parâmetros treináveis da camada. Desta forma, o conjunto Θ de uma CNN é composto por todos os elementos das matrizes $K^{(l_c)}$ e $B^{(l_c)}$ que compõem camadas convolucionais l_c e por todos os elementos das matrizes $W^{(l_n)}$ e $B^{(l_n)}$ das camadas não convolucionais l_n. Assim, CNNs são treinadas pelo mesmo algoritmo de *Back Propagation* descrito na seção 2.2.

2.4 Modelos generativos adversários

GAN (*Generative Adversarial Networks*, do inglês, Redes Adversárias Generativas) são modelos híbridos compostos por duas ANNs: um módulo gerador, treinado para gerar dados sintéticos que sejam indistinguíveis de dados reais, e um módulo discriminador, treinado para distinguir dados reais de dados gerados [38]. Dependendo da aplicação, GAN podem ser modeladas para geração de dados não-condicionada ou condicionada.

2.4.1 Geração de dados não-condicionada

Em GAN não-condicionadas, o módulo gerador $\mathcal{G}(z; \Theta_\mathcal{G}) \rightarrow \hat{y}$ gera dados sintéticos $\hat{y}$ a partir da variável latente z, vinda de uma distribuição aleatória $\mathcal{Z}$. Por outro lado, o módulo discriminador $\mathcal{D}(y; \Theta_\mathcal{D}) \rightarrow p$ retorna um escalar p que indica a probabilidade de que certo dado $\hat{y}$ pertença ao conjunto de dados reais $\mathcal{X}$. Desta forma, pode-se definir a função de perda de modelos GAN, baseada na BCE entre $\mathcal{D}(x)$ e $\mathcal{D}(\mathcal{G}(z))$, como

$$\mathcal{L}_{\text{GAN}}(\mathcal{G}, \mathcal{D}) = \mathbb{E}_x[\log \mathcal{D}(x)] + \mathbb{E}_z[\log(1 - \mathcal{D}(\mathcal{G}(z)))], \tag{2.10}$$

em que $\mathbb{E}$ representa a esperança da saída do modelo e a variável x representa uma amostra do conjunto $\mathcal{X}$ [39].

O treinamento de GAN não-condicionadas consiste em gerar os conjuntos de parâmetros $\Theta_{\mathcal{G}}$ e $\Theta_{\mathcal{D}}$ para os módulos gerador e discriminador que atinja o objetivo

$$G^* = \arg\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}_{\text{GAN}}(\mathcal{G}, \mathcal{D}). \tag{2.11}$$

É importante notar que, apesar da equação (2.11) relacionar ambos módulos do modelo a um único objetivo, os parâmetros do gerador e do discriminador, $\Theta_{\mathcal{G}}$ e $\Theta_{\mathcal{D}}$, são ajustados de forma isoladas e concorrentes. Enquanto o ajuste de parâmetros de $\mathcal{G}$ tem como objetivo minimizar a equação (2.10), o ajuste de $\mathcal{D}$ maximiza a equação. O comportamento de ambos modelos durante o treinamento é representado no diagrama da Figura 7.

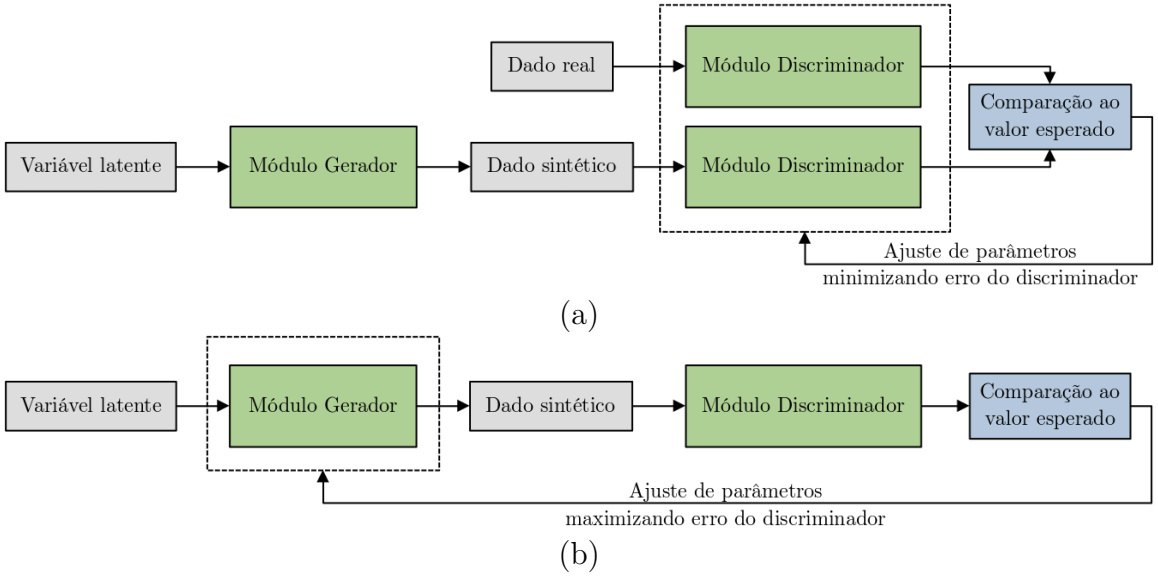


Figura 7 – Diagramas representando o ajustes de parâmetros do módulo discriminador (a) e do módulo gerador (b) durante o processo de treinamento de uma GAN não-condicionada.

O treinamento também pode ser considerado ótimo quando os dados gerados por $\mathcal{G}$ sejam equivalentes aos dados reais, ou seja, quando $\mathcal{D}$ não é mais capaz de distinguir dados reais de dados sintéticos [38].

2.4.2 Geração de dados condicionada

CGAN (*Conditional Generative Adversarial Networks*, do inglês, Redes Adversárias Generativas Condicionadas) são extensões de GAN para geração de dados condicionada. Para esses modelos, a variável y é interpretada como uma informação de condicionamento, que pode representar classes de dados ou outras modalidades de identificação [40], e representa a saída esperada do módulo gerador. Além disso, tanto o gerador quanto o discriminador são modificados para observar os dados reais x, sendo representado por

$\mathcal{G}(x, z; \Theta_\mathcal{G}) \rightarrow \hat{y}$ e $\mathcal{D}(x, y; \Theta_\mathcal{D}) \rightarrow p$, respectivamente. Com isso, a função de perda de modelos CGAN pode ser representada por

$$\mathcal{L}_{\text{CGAN}}(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{x,y}[\log \mathcal{D}(x, y)] + \mathbb{E}_{x,z}[\log (1 - \mathcal{D}(x, \mathcal{G}(x, z)))]. \tag{2.12}$$

Por sua vez, o objetivo do modelo segue a equação

$$G^* = \arg \min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}_{\text{CGAN}}(\mathcal{G}, \mathcal{D}) \tag{2.13}$$

e o ajuste de parâmetros dos módulos durante o processo de treinamento é representado pela Figura 8.

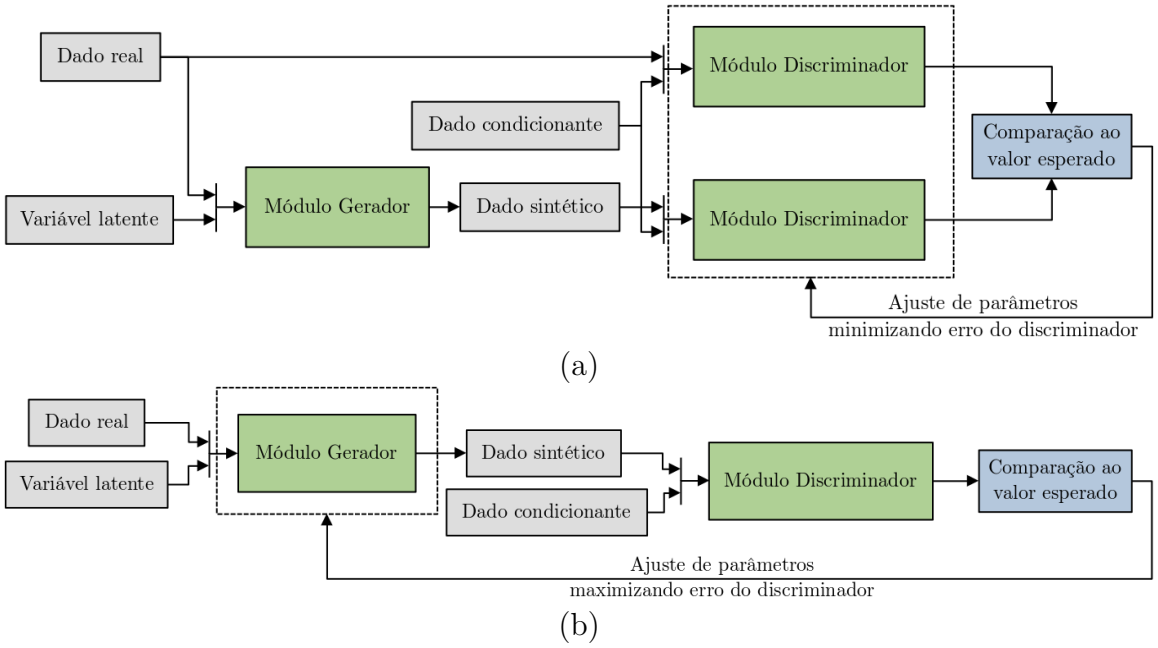


Figura 8 – Diagramas representando o ajustes de parâmetros do módulo discriminador (a) e do módulo gerador (b) durante o processo de treinamento de uma CGAN.

2.4.3 Tradução imagem-a-imagem

Com base em CGAN, o projeto *Pix2Pix* foi criado como um modelo de tradução imagem-a-imagem de propósito geral [41]. O projeto tem como objetivo mapear imagens como dados de entrada a uma saída traduzida, também no formato de imagem, seguindo um dado condicionante. O modelo obtém resultados efetivos para diversos problemas de computação visual, incluindo segmentação semântica, geração de mapas a partir de fotos aéreas e coloração de imagens [39].

Para conduzir o treinamento do modelo, o *Pix2Pix* adota uma função de perda adicional aplicada ao módulo gerador, que considera a norma L1 (representada por $\|\cdot\|_1$) entre a saída do módulo e o dado condicionante, ou seja,

$$\mathcal{L}_{\text{L1}}(\mathcal{G}) = \mathbb{E}_{x,y,z}[\|y - \mathcal{G}(x, z)\|_1]. \tag{2.14}$$

Desta forma, o modelo segue como objetivo uma combinação ponderada das funções de perda $\mathcal{L}_{\text{CGAN}}$ e $\mathcal{L}_{\text{L1}}$, a partir da equação

$$G^* = \arg\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}_{\text{CGAN}}(\mathcal{G}, \mathcal{D}) + \lambda \mathcal{L}_{L1}(\mathcal{G}), \tag{2.15}$$

em que λ representa uma constante de ponderação. Os autores do projeto *Pix2Pix* recomendam, a partir de análise empírica, adotar $\lambda = 100$ para resultados ideais [41].

Para garantir eficiência na tradução de imagens, os autores do projeto *Pix2Pix* definem os modelos de ANNs, baseados em CNNs, para os módulos gerador e discriminador:

- **Módulo Gerador**: Baseado no modelo de segmentação de imagens biomédicas U-Net [25], este módulo é construído a partir de uma estrutura composta por um conjunto de blocos de convolução codificadora seguido de um conjunto de blocos de convolução decodificadora, incluindo blocos de concatenação entre ambos conjuntos;
- **Módulo Discriminador**: O módulo discriminador do projeto GAN Markovianas [42] foi adotado já que resultados empíricos mostram que este modelo favorece o processo de tradução de imagens [41].

O modelo descrito é representado na Figura 9. Nota-se que, durante o treinamento do modelo a variável latente z não é usada explicitamente. Enquanto que modelos CGAN usam esta variável como ruído de entrada do gerador, os autores do projeto *Pix2Pix* notaram que o modelo simplesmente ignora o ruído após o treinamento condicionado. Desta forma, a variável latente é substituída pela inicialização aleatória dos blocos de convolução usados no modelo [41].

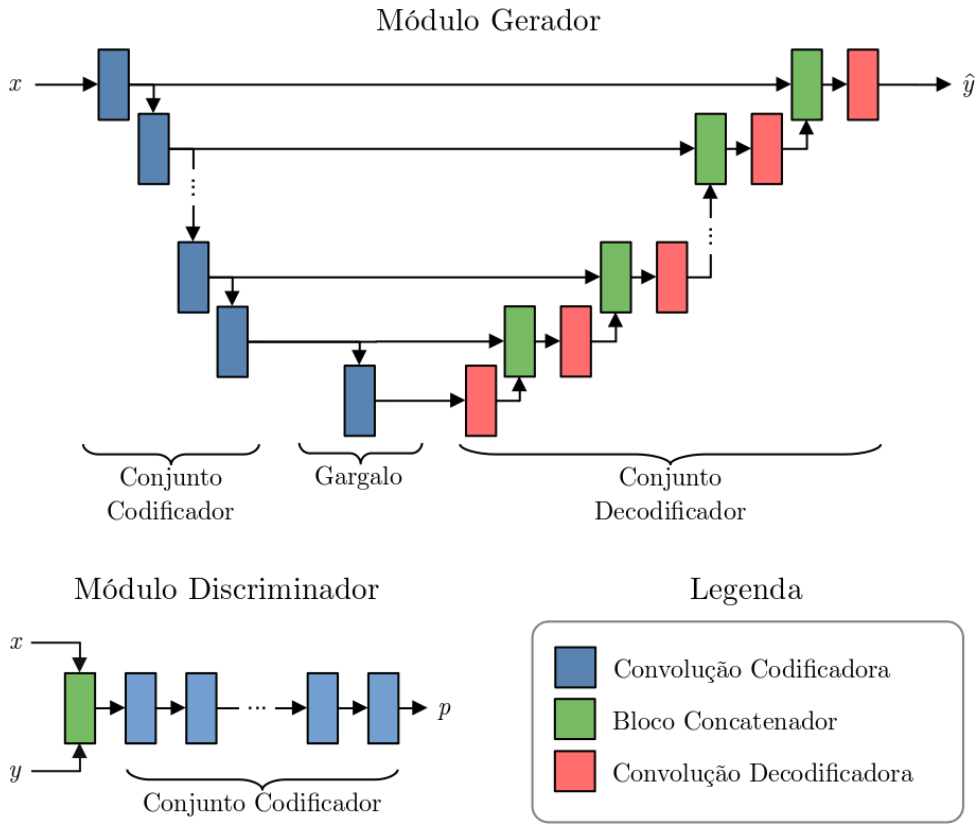


Figura 9 – Representação dos blocos de convolução envolvidos nos módulos gerador e discriminador de um modelo *Pix2Pix*.

3 Modelo Proposto

Este Capítulo descreve o modelo proposto para retornar, de forma autônoma, dados relevantes para diagnósticos de glaucoma a partir de imagens de fundo de retina, originadas de exames de retinografia. Para isso, deseja-se que a saída do modelo se aproxime da conclusão advinda da análise de especialistas de retina.

O processo de análise das imagens de fundo de retina foi dividido em três partes para o desenvolvimento deste trabalho:

1. **Localização do disco óptico**: consiste em segmentar a imagem original do fundo de retina para identificar o centro e o tamanho aproximado do disco óptico;
2. **Segmentação entre escavação óptica e disco óptico**: uma vez identificado o disco óptico, é necessário segmentar a região definida pelo disco óptico e pela escavação do disco óptico;
3. **Cálculo do CDR**: as regiões segmentadas são aproximadas para regiões elípticas e seus eixos horizontal e vertical são medidos. A CDR é calculada a partir da razão entre o eixo da escavação e o eixo do disco óptico, tanto horizontal quanto vertical. As relações CDR horizontal e vertical serão apresentadas como saída do programa.

Devido à complexidade de se otimizar os processos envolvidos nas etapas 1 e 2 por métodos experimentais, a otimização dessas etapas será feita pelo treinamento de modelos de ANN. Para a tarefa da etapa 1, considerou-se uma CNN padrão, baseada no modelo U-Net, devido a pesquisas anteriores já mostrarem resultados aceitáveis com a arquitetura [25]. Para a tarefa da etapa 2, adotou-se três modelos diferentes para se comparar: uma CNN baseada na arquitetura U-Net, uma CNN baseada na arquitetura M-Net, e CGAN baseadas no projeto *Pix2Pix*, como apresentado na Subseção 2.4.3. Os modelos das etapas 1 e 2 foram denominados, respectivamente, M-LD (Modelo de Localização de Disco) e M-SR (Modelo de Segmentação de Retina), para o escopo deste trabalho. Para o desenvolvimento desses modelos, são considerados os seguintes procedimentos, aplicados paralelamente para ambos:

1. **Coleta de dados**: bases de dados abertas que fornecem imagens de fundo de retina com demarcações do disco óptico e da escavação óptica são consideradas para alimentar os modelos;

2. **Tratamento de dados**: como as imagens e segmentações são originadas de fontes diferentes, elas devem ser triadas e padronizadas para treinamento adequado dos modelos;

3. **Desenvolvimento do modelo**: neste procedimento são desenvolvidos algoritmos que descrevem a entrada e a saída de dados, a arquitetura das redes, e os meta-parâmetros de treinamento para otimização do modelo;

4. **Treinamento do modelo**: o modelo desenvolvido é treinado pelo método de *Gradient Descent*, como visto na Seção 2.2, e o conjunto de parâmetros otimizados do módulo gerador é obtido;

5. **Avaliação do modelo**: usando a arquitetura definida para o módulo gerador e o conjunto de parâmetros obtido, o modelo é aplicado em imagens de validação para se obter predições, que são comparadas com os resultados esperados com métricas de avaliação.

3.1 Bases de Dados

Para a entrada de dados de treinamento e validação dos modelos M-LD e M-SR, foram levantados requisitos em relação às imagens e aos *labels* para se aprovar bases de dados pesquisadas, levando em conta as limitações técnicas para treinar e validar os modelos em computador pessoal. Os requisitos relativos aos modelos M-LD e M-SR são apresentados nas Tabelas 1 e 2, respectivamente.

Tabela 1 – Requisitos para se considerar uma base de dados para treinamento e validação do Modelo de Localização de Disco.

Ind.	**Requisito**
M-LD-R1	As imagens devem conter a retina completa, sem obstruções ou cortes
M-LD-R2	As imagens devem ter resolução de no mínimo 512×512 *pixels*
M-LD-R3	Os *labels* devem conter segmentação do disco óptico

Aplicando os requisitos descritos em bases de dados abertas contendo imagens de fundo de retina, relacionou-se as bases que são adequadas para os modelos M-LD e M-SR na Tabela 3. A Tabela 4 apresenta informações adicionais sobre as bases de dados selecionadas.

3.2 Tratamento de Dados

Como forma de padronizar o formato dos dados, foram definidos os seguintes padrões para transformação das imagens e dos *labels*:

Tabela 2 – Requisitos para se considerar uma base de dados para treinamento e validação do Modelo de Segmentação de Retina.

Ind.	**Requisito**
M-SR-R1	As imagens devem conter a região do nervo óptico com definição clara
M-SR-R2	A região em torno do nervo óptico deve ter resolução de no mínimo 256×256 *pixels*
M-SR-R3	Os *labels* devem conter segmentação do disco óptico e da escavação do disco óptico

Tabela 3 – Relação das bases de dados aprovadas e reprovadas para treinamento e validação dos modelos M-LD e M-SR.

Base de dados	**Aprovado para M-LD?**	**Aprovado para M-SR?**
REFUGE [13]	Sim	Sim
RIGA [43]	Sim	Sim
RIM-ONE [44]	Não (critério M-LD-R1)	Sim
Drishti-GS [45]	Sim	Sim
IDRiD [46]	Sim	Não (critério M-SR-R3)
DRIONS-DB [47]	Não (critério M-LD-R2)	Não (critério M-SR-R3)
MESSIDOR [17]	Não (critério M-LD-R3)	Não (critério M-SR-R3)
DRIVE [48]	Não (critério M-LD-R3)	Não (critério M-SR-R3)

- Imagens de retina devem ser cortadas a um formato quadrado que contorna a borda da retina e devem ser armazenadas com três canais de cor, de forma a preservar as informações contidas em cada canal, em formato JPEG;
- A segmentação dos *labels* deve ser codificada para uma única imagem, em que regiões que não pertencem ao disco óptico são marcadas como pretas, regiões que pertencem ao disco óptico mas não à escavação são marcadas como cinzas e regiões que representam a escavação do disco óptico são marcadas como brancas. Tais imagens devem seguir o mesmo formato da imagem de retina, e devem ser armazenadas usando um canal de cor no formato PNG.

A base de dados REFUGE possui seus *labels* codificados da forma desejada, bastando apenas um corte quadrado em torno da imagem da retina para adequar seus dados ao formato. Já na base de dados RIGA, os *labels* são apresentados como contornos em volta do disco óptico e em volta da escavação óptica, feitos por seis profissionais diferentes. Desta forma, foi necessário re-codificar as imagens de forma que as regiões de disco óptico e de escavação do disco óptico representassem a média das regiões detectadas pelos especialistas. Para a base de dados IDRiD, como esta foi usada apenas para o modelo M-LD, apenas a codificação de disco óptico em cinza foi necessária para a geração de *la-*

Tabela 4 – Descrição das imagens contidas nas bases de dados consideradas para treinamento e validação dos modelos M-LD e M-SR.

Base de dados	**Total de imagens**	**Resolução original das imagens**	**Resolução aproximada do corte em torno do nervo óptico***
REFUGE	1200	2124×2056 (400 imagens); 1634×1634 (800 imagens)	486×486
RIGA**	750	2240×1488 (460 imagens); 2376×1584 (195 imagens); 2743×1936 (95 imagens)	397×397
RIM-ONE	159	1072×1424***	518×518
Drishti-GS	101	2047×1760, aproximadamente****	690×690
IDRiD	81	4288×2848	N/A

* O corte em torno do nervo óptico considera o raio da segmentação do disco óptico contido nos *labels*, com uma margem de 75%.
** A base de dados RIGA é composta por três bases de dados diferentes, denominadas Messidor, Magrabia e BinRushed, que foram segmentadas por especialistas em retina.
*** Como as imagens foram capturadas em pares e são apresentadas lado a lado, a primeira dimensão da resolução original de 2144×1424 foi cortada pela metade
**** Os autores da base de dados cortaram as margens externas das imagens originais, de resolução 2896×1944, portanto as imagens finais têm resoluções diferentes

bel. Finalmente, em ambas bases de dados RIM-ONE e Drishti-GS, os *labels* referentes ao disco óptico e à escavação do disco óptico são apresentados em arquivos diferentes. Neste caso, foi preciso unir ambas imagens em apenas uma, alterando a cor quando necessário.

Para utilizar as bases de dados para treinamento de ANNs, é necessário separar o conjunto de amostras de treinamento do conjunto de amostras de validação. Adotou-se que o conjunto de treinamento seja composto por 90% das imagens, e o conjunto de validação pelo restante. A quantidade absoluta de imagens, separado por base de dados de origem, pode ser vista na Tabela 5.

O processo de treinamento também se beneficia com técnicas de *Data Augmentation*. Ao transformar os dados de treinamento dinamicamente e aleatoriamente na entrada do modelo, o modelo é capaz de generalizar os dados de validação com mais precisão. Foram adotadas as seguintes técnicas de *Data Augmentation* para a entrada dos modelos:

- **Espelhamento Aleatório**: as imagens são aleatoriamente espelhadas horizontalmente e/ou verticalmente;
- **Corte Aleatório**: em seguida, as imagens são cortadas com uma margem aleatória entre 0% a 25% da resolução original da imagem;

Tabela 5 – Relação de número de imagens utilizadas como conjunto de amostras de treinamento e conjunto de amostras de validação.

Base de dados	**Número de imagens do conjunto de treinamento**	**Número de imagens do conjunto de validação**
REFUGE	1080	120
RIGA	675	75
RIM-ONE*	145	14
Drishti-GS	91	10
IDRiD**	72	9

* Aplicável apenas para o modelo M-SR
** Aplicável apenas para o modelo M-LD

- **Deslocamento Aleatório**: finalmente, as imagens são deslocadas aleatoriamente em até 64 *pixels* em ambos sentidos na direção vertical e horizontal.

3.3 Desenvolvimento dos Modelos

Inicialmente, foram definidas quais são as entradas e saídas de cada um dos modelos:

- **M-LD**: imagem de retina completa em formato $256 \times 256 \times 3$ como entrada, e segmentação da região do disco, codificada em branco, em formato $256 \times 256 \times 1$ como saída;
- **M-SR**: imagem de retina cortada em volta do disco óptico com uma margem de 50%, interpolada pelo método LANCZOS3 [49], em formato $256 \times 256 \times 3$ como entrada, e segmentação da região da escavação e do disco, codificadas em branco e cinza respectivamente, em formato $256 \times 256 \times 1$ como saída.

A conexão entre a saída do modelo M-LD e a entrada do modelo M-SR é representada no diagrama da Figura 10, relacionando com as tarefas 1 e 2 apresentadas no início deste Capítulo.

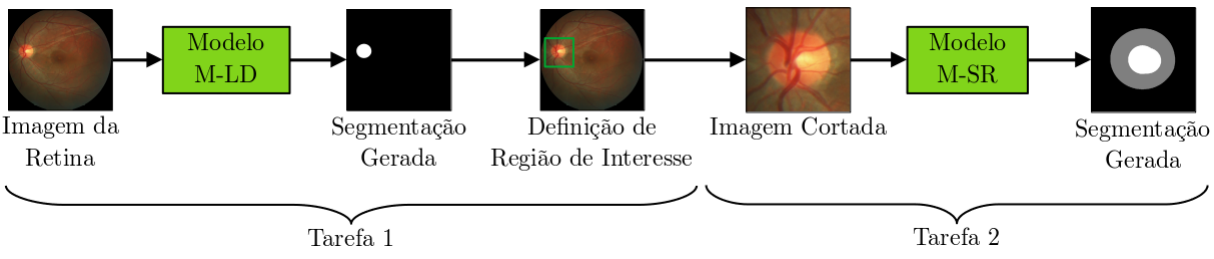


Figura 10 – Diagrama relacionando as tarefas 1 e 2 do modelo proposto com as entradas e saídas das redes neurais utilizadas.

Para o modelo M-LD, considerou-se a implementação da arquitetura U-Net modificada[1] feita pela equipe do projeto *Pix2Pix*, adaptando os formatos de entrada e saída para a aplicação. Já para o M-SR, um modelo generativo baseado na implementação do projeto *Pix2Pix*[2] foi construído e adaptado para a aplicação de segmentação de retina. Os módulos gerador e discriminador do modelo generativo proposto são apresentados nas Figuras 11 e 12, respectivamente. Para validar o modelo proposto, adotou-se as arquiteturas puramente convolucionais de estado-da-arte U-Net e M-Net[3] como modelos de comparação.

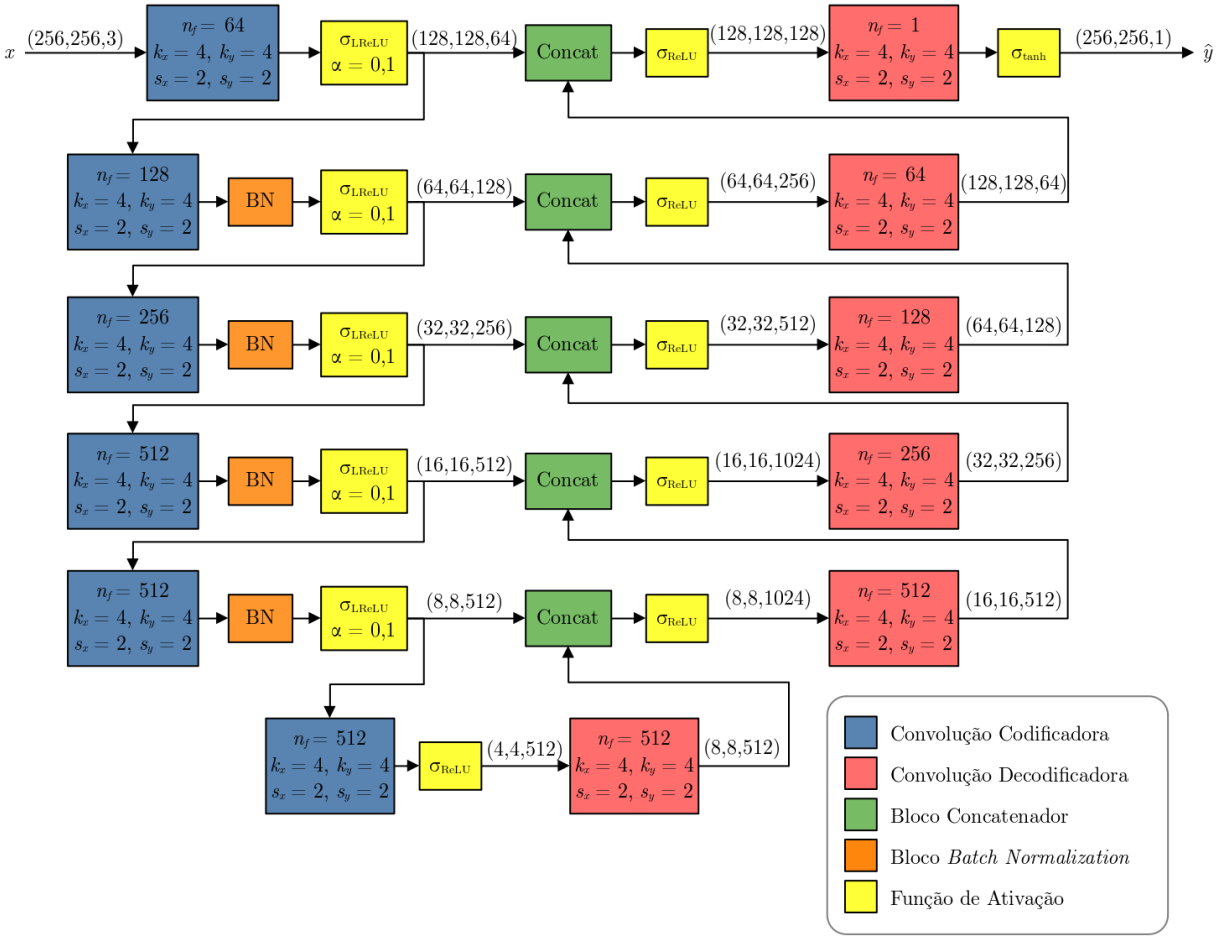


Figura 11 – Representação da arquitetura do módulo gerador do modelo generativo proposto.

Os parâmetros de treinamento adotados para os quatro modelos descritos foram ajustados empiricamente e são apresentados na Tabela 6. Com esses parâmetros, os modelos podem ser treinados paralelamente e suas métricas de avaliação comparadas.

[1] Disponível nos repositórios do TensorFlow, em <https://github.com/tensorflow/docs-l10n/blob/master/site/pt-br/tutorials/images/segmentation.ipynb>

[2] Disponível em <https://github.com/tensorflow/examples/blob/master/tensorflow_examples/models/pix2pix/pix2pix.py>

[3] Repositório disponível em <https://github.com/HzFu/MNet_DeepCDR>

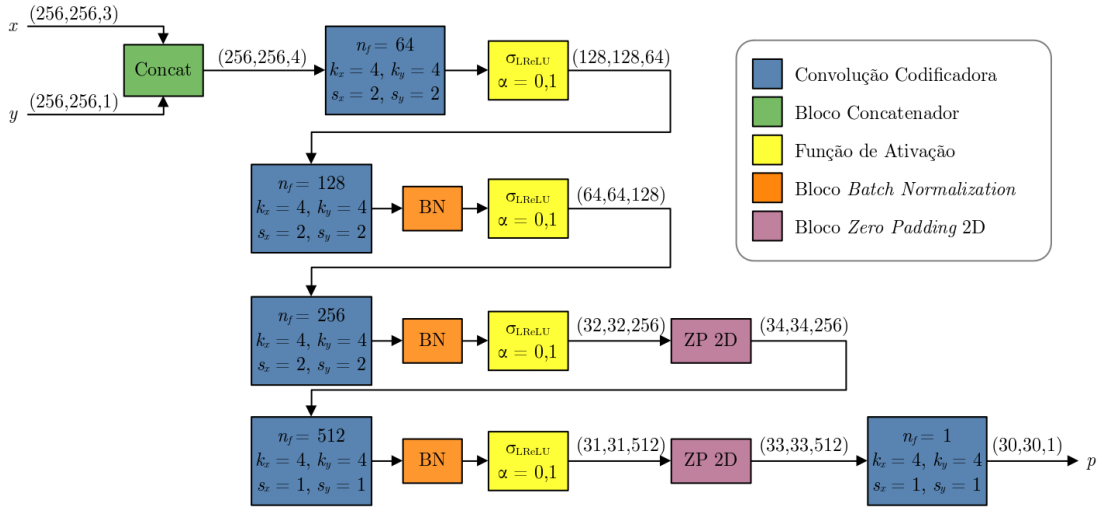


Figura 12 – Representação da arquitetura do módulo discriminador do modelo generativo proposto.

Tabela 6 – Relação dos parâmetros de treinamento dos modelos de redes neurais propostos.

	U-Net (M-LD)	**U-Net (M-SR)**	**M-Net (M-SR)**	**Modelo Generativo (M-SR)**
Método de Otimização	SGD	SGD	SGD	Adam [50]
Taxa de Aprendizado	0,0001	0,0001	0,0001	0,0002
Função de Perda	DL	DL	DL	BCE
Número de Épocas	50	300	300	150

3.4 Cálculo da CDR

Para realizar o cálculo da CDR a partir da segmentação gerada pelo modelo M-SR, são considerados os seguintes procedimentos:

1. **Ajuste a Elipse**: usando a função *fitEllipse* da biblioteca OpenCV[4], que corresponde a uma implementação do primeiro algoritmo apresentado por Fitzgibbon A. e Fisher R. [51], os contornos gerados pela segmentação são aproximados por elipses;

2. **Medição das Dimensões das Elipses**: em seguida, a altura e largura, em relação à orientação da imagem, das elipses correspondentes ao disco e à escavação óptica são medidas;

[4] Documentação disponível em <https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#gaf259efaad93098103d6c27b9e4900ffa>

3. **Cálculo das CDRs**: finalmente, as CDRs são calculadas pela relação das dimensões da escavação óptica ao disco óptico.

A Figura 13 apresenta um diagrama descrevendo os procedimentos descritos.

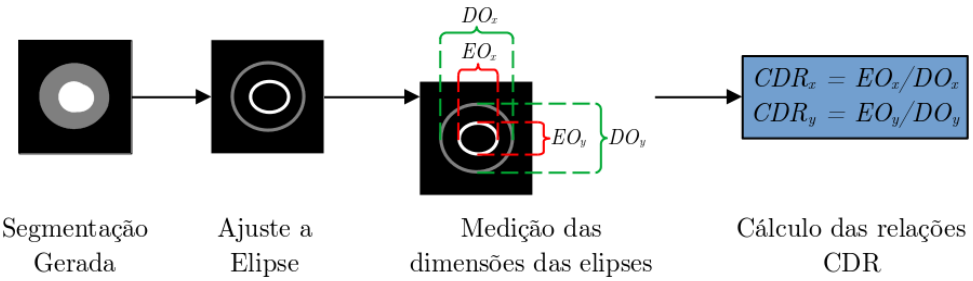


Figura 13 – Representação dos procedimentos envolvidos no cálculo da CDR a partir de uma imagem de segmentação.

3.5 Métricas de Avaliação

Para tarefas de segmentação, como é o caso da saída dos modelos M-LD e M-SR, considera-se a métrica DSC (*Dice Similarity Coefficient*, do inglês, Coeficiente de Similaridade Dice), que representa um índice de similaridade espacial [26], para comparar a saída do modelo com a saída esperada. O índice DSC é definido por

$$\mathrm{DSC}(A, B) = \frac{2|A \cap B|}{|A| + |B|}, \tag{3.1}$$

em que A e B são matrizes binárias, com mesmas dimensões, que representam as imagens de segmentação. A e B são codificadas de forma que os *pixels* dentro das regiões de interesse são representados por 1, e *pixels* fora dessa região, por 0. Com tal codificação, DSC indica o quanto as matrizes A e B são similares baseado na interseção entre elas, variando entre 0 (nenhuma interseção entre as matrizes) e 1 (as matrizes são iguais). Por sua vez, a função de perda DL (*Dice Loss*, do inglês, Perda de Dice) é definida pelo complemento do índice DSC, ou seja,

$$\mathrm{DL}(A, B) = 1 - \mathrm{DSC}(A, B). \tag{3.2}$$

Em específico para o modelo M-SR, surge a necessidade de separar a segmentação da escavação óptica da segmentação do disco óptico para avaliar suas métricas de similaridade individualmente. Considerando y como a matriz de saída esperada e $\hat{y}$ como a matriz de saída do modelo, ambas codificadas com valores no intervalo $[0, 1]$ (em que 0 representa *pixels* na cor preta, e 1 representa *pixels* na cor branca), as medições de DSC podem ser separadas para a segmentação de disco (que considera ambas regiões de disco e escavação) e de escavação óptica, respectivamente, pelas expressões

$$\mathrm{DSC}_{\mathrm{disco}}(\hat{y}, y) = \frac{2|(\hat{y} : \hat{y} \geq 0,25) \cap (y : y \geq 0,25)|}{|(\hat{y} : \hat{y} \geq 0,25)| + |(y : y \geq 0,25)|}, \tag{3.3}$$

$$\mathrm{DSC}_{\text{escavação}}(\hat{y}, y) = \frac{2|(\hat{y} : \hat{y} \geq 0,75) \cap (y : y \geq 0,75)|}{|(\hat{y} : \hat{y} \geq 0,75)| + |(y : y \geq 0,75)|}. \tag{3.4}$$

Para avaliar as predições de CDR, por se tratarem de valores escalares, foi considerada a métrica MAE (*Mean Absolute Error*, do inglês, Erro Absoluto Médio), definida por

$$\mathrm{MAE}(A, B) = \frac{\sum_{i=1}^{n} |A_i - B_i|}{n}, \tag{3.5}$$

em que A e B representam os conjuntos de escalares a serem comparados. No caso da avaliação de CDR, A e B representam, respectivamente, o conjunto de valores de CDR previstos e o conjunto de valores de CDR esperados.

3.6 Ambiente de Desenvolvimento

As arquiteturas apresentadas foram implementadas usando a linguagem Python e o *framework* TensorFlow[5]. Trechos relevantes da implementação do modelo estão presentes no Apêndice A.

O ambiente de desenvolvimento do projeto foi criado a partir de uma imagem Docker[6] disponibilizada pelos desenvolvedores do TensorFlow, com o objetivo de garantir estabilidade de versionamento das bibliotecas. A imagem[7] contém um ambiente com a biblioteca tensorflow-gpu, versão 2.5.0, instalada a partir do repositório PyPI (*Python Package Index*, do inglês, Índice de Pacotes *Python*), em *Python* versão 3.6.9. A partir desta imagem, foram adicionadas as bibliotecas PyPI adicionais:

- **opencv-python, versão 4.5.2.54**[8]: utilizado para desenvolver os algoritmos de tratamento descritos nas Seções 3.2 e 3.4;
- **matplotlib, versão 3.3.4**[9]: utilizado para traçar os gráficos contidos nesta Dissertação;
- **tdqm, versão 4.61.0**[10]: provê uma interface de barra de progresso com baixo *overhead* em *Python*, útil para verificar o tempo decorrido das iterações de um *loop*;
- **pandas, versão 1.3.3**[11]: biblioteca de análise de dados, usada para prover as análises estatísticas desta Dissertação.

[5] Mais informações disponíveis em <https://www.tensorflow.org/>.
[6] Mais informações disponíveis em <https://www.docker.com/>.
[7] Disponível no Docker Hub em <https://hub.docker.com/r/tensorflow/tensorflow>.
[8] Disponível em <https://pypi.org/project/opencv-python/>
[9] Disponível em <https://pypi.org/project/matplotlib/>
[10] Disponível em <https://pypi.org/project/tqdm/>
[11] Disponível em <https://pypi.org/project/pandas/>

Os processos de treinamento e validação dos modelos descritos foram aplicados no ambiente de desenvolvimento em computador pessoal (processador Intel i7-7700HQ, 16GB de RAM DDR4 e placa de vídeo NVidia GTX1050Ti, com 4GB de VRAM GDDR5 e 768 núcelos CUDA), e seus resultados são apresentados no próximo Capítulo.

4 Resultados e Análises

Neste Capítulo são apresentados os resultados dos processos de treinamento descritos anteriormente para os modelos M-LD e M-SR, assim como o processo de validação e comparação dos mesmos.

4.1 Modelo de Localização de Disco

Durante o processo de treinamento da U-Net modificada que compõe o modelo M-LD, registrou-se a função de perda, DL, aplicando-a nas amostras de treinamento e de validação em cada época de treinamento. A Figura 14 representa um gráfico de linhas comparando a saída da função para o conjunto de amostras de treinamento e para o conjunto de amostras de validação.

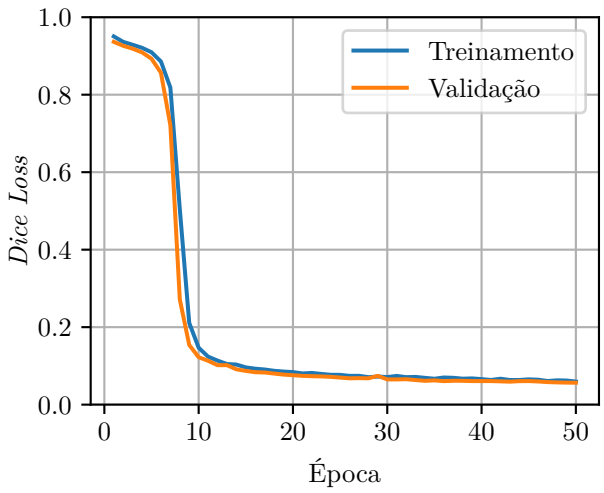


Figura 14 – Representações em gráficos de linhas da função DL aplicados nas amostras de treinamento e validação do Modelo de Localização de Disco durante suas épocas de treinamento.

A partir do gráfico, nota-se que os valores de perda do conjunto de treinamento estão levemente acima da curva de perda do conjunto de validação. Enquanto que isso não é um comportamento esperado do processo de treinamento de uma ANN, pode ser justificado pelo uso de camadas reguladoras, como *Dropout*, na arquitetura da rede. Além disso, o uso de técnicas de *Data Augmentation* durante o processo de treinamento prejudica a saída da função de perda do conjunto de amostras de treinamento.

Apesar do comportamento observado, pode-se notar que a rede atingiu um local mínimo na função de custo e, portanto, foi capaz de generalizar a tarefa de localização

de disco. Na época 50, a rede atingiu DL de 0,0566 no conjunto de validação, que indica um DSC médio de 0,9434 na localização do disco óptico na imagem de fundo de retina. Conclui-se, portanto, que o modelo é capaz de segmentar a região do disco óptico.

4.2 Modelos de Segmentação de Retina

Nesta Seção são apresentados os resultados de treinamento do modelo generativo proposto e dos modelos convolucionais adotados como comparação, que compõem a tarefa de segmentação de retina.

4.2.1 Modelo Generativo Proposto

O modelo generativo proposto, que compõe a tarefa de segmentação de retina, foi treinado seguindo os procedimentos descritos na Seção 2.4. A relação da função de perda dos módulos discriminador e gerador com as épocas de treinamento é representada por gráficos de linhas na Figura 15.

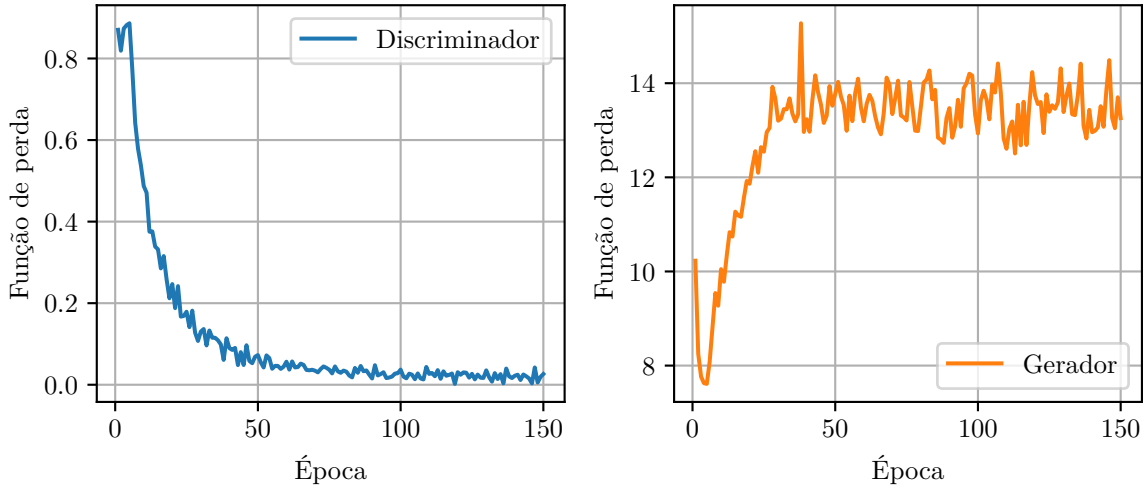


Figura 15 – Representações em gráficos de linhas das funções de perda dos módulos gerador e discriminador do modelo generativo proposto durante suas épocas de treinamento.

Nota-se que o modelo começa a generalizar a solução a partir da época 50, evidenciado pelo comportamento estável da função de perda do módulo discriminador. Este valor, no entanto, nunca será nulo devido ao comportamento adversário do módulo gerador, que força o discriminador a falhar em sua predição. A generalização do modelo também pode ser observada pelo comportamento oscilante da função de perda do módulo gerador, a partir da época 40. Como neste momento o módulo discriminador já é capaz de diferenciar dados do modelo de dados reais, o módulo gerador deve aumentar sua função de perda para forçar o módulo discriminador a falhar em sua predição.

O módulo gerador obtido após a 150ª época de treinamento foi aplicado no conjunto de amostras de validação. Este foi adotado já que apresenta melhores resultados dentre os modelos obtidos de épocas anteriores. As Tabelas 7 e 8 apresentam levantamentos estatísticos da métrica DSC aplicada para avaliar, respectivamente, segmentação de disco óptico e de escavação óptica, separadas por base de dados analisada. Com uma DSC média de 0,9445 (desvio padrão de 0,0432) para segmentação de disco e 0,8659 (desvio padrão de 0,0886) para segmentação de escavação óptica, conclui-se o modelo realiza a tarefa de segmentação de retina. Nota-se que há uma discrepância dos valores de DSC obtidos da base de dados RIM-ONE em relação às outras, que se deve pela base conter imagens capturadas com equipamentos e técnicas diferentes das demais.

Tabela 7 – Levantamento estatístico dos valores de DSC_{disco} das predições do modelo generativo de Segmentação de Retina por base de dados analisado.

	REFUGE	**RIGA**	**RIM-ONE**	**Drishti-GS**
Média	0,9398	0,9641	0,8838	0,9542
Desvio Padrão	0,0430	0,0209	0,0637	0,0393
Mínimo	0,6649	0,8659	0,7431	0,8666
25%	0,9309	0,9602	0,8442	0,9608
50%	0,9513	0,9693	0,8936	0,9652
75%	0,9623	0,9770	0,9262	0,9745
Máximo	0,9814	0,9871	0,9590	0,9774

Tabela 8 – Levantamento estatístico dos valores de $DSC_{escavação}$ das predições do modelo generativo de Segmentação de Retina por base de dados analisado.

	REFUGE	**RIGA**	**RIM-ONE**	**Drishti-GS**
Média	0,8716	0,8842	0,7514	0,7998
Desvio Padrão	0,0836	0,0739	0,1213	0,0587
Mínimo	0,2978	0,5559	0,4450	0,7460
25%	0,8328	0,8721	0,6916	0,7766
50%	0,8978	0,9060	0,7914	0,7881
75%	0,9271	0,9260	0,8382	0,7921
Máximo	0,9636	0,9654	0,8795	0,9272

4.2.2 Modelos Convolucionais

Como comparação ao modelo generativo proposto, foram treinadas as redes convolucionais que realizam segmentação de retina. A Figura 16 apresenta o comportamento da função DL, separada por disco e escavação óptica, durante o treinamento das redes U-Net e M-Net. Observa-se que ambos modelos convolucionais estabilizaram após 100 épocas de treinamento, atingindo um mínimo local nos conjuntos de parâmetros.

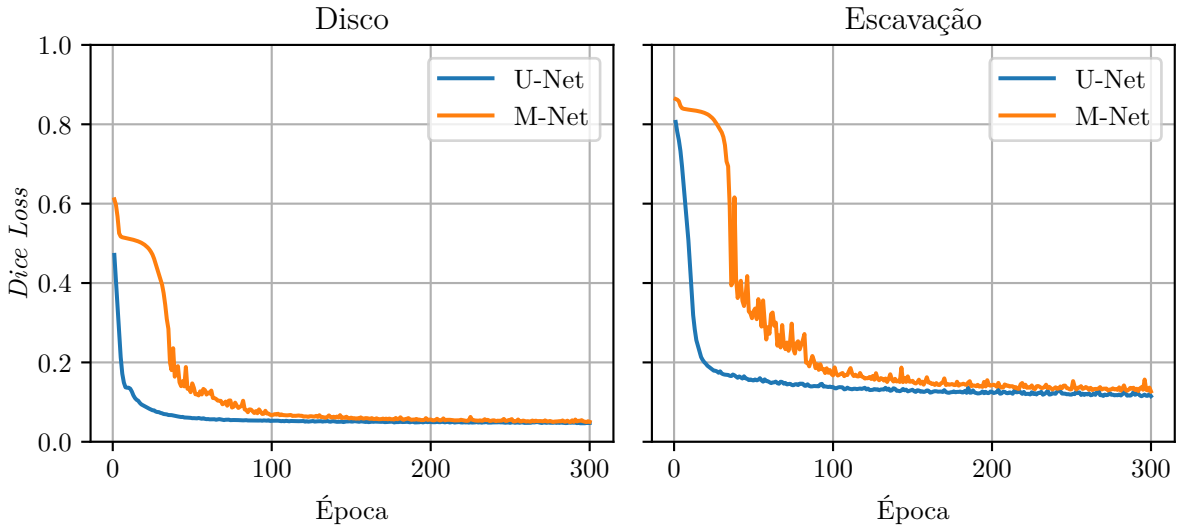


Figura 16 – Representações em gráficos de linhas da função DL durante as épocas de treinamento dos modelos convolucionais comparados.

4.3 Validação dos Modelos

Para validar os modelos M-LD e M-SR treinados, considera-se aplicar o diagrama descrito na Figura 10 no conjunto de dados de validação. A Figura 17 apresenta amostras de saídas do procedimento para cada modelo de M-SR considerado.

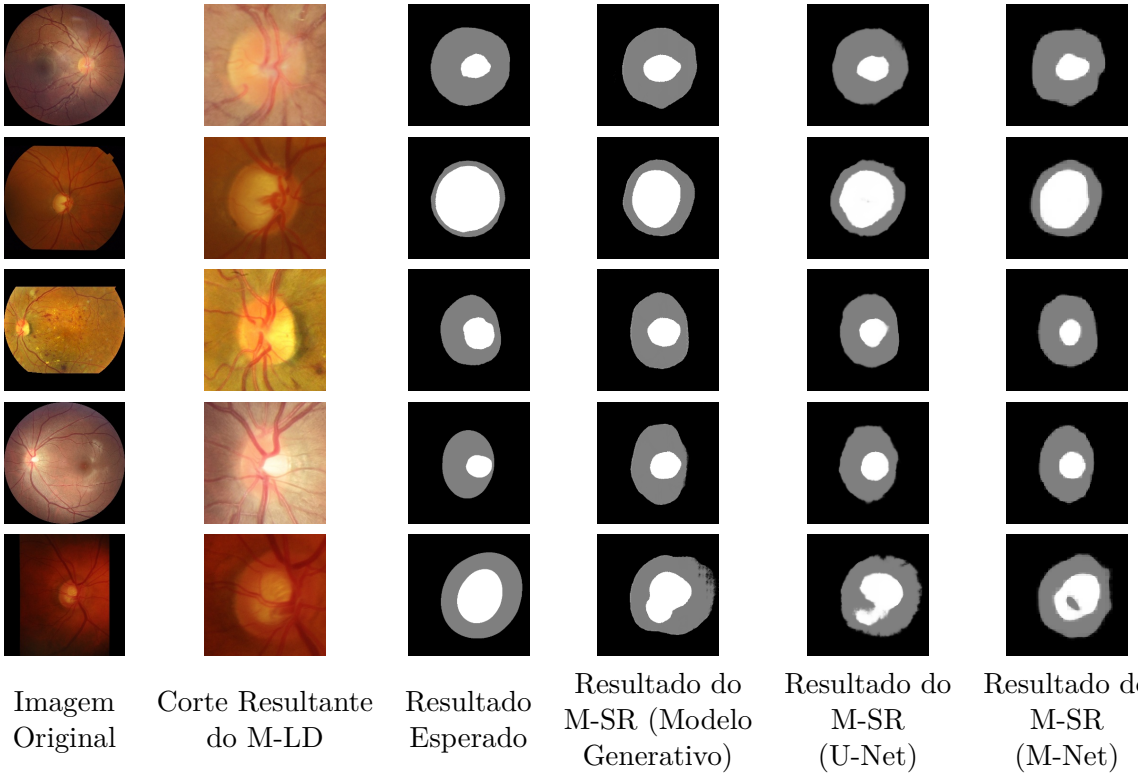


Figura 17 – Amostras de predições de segmentação de retina usando imagens de validação das bases de dados.

A fim de comparar as escolhas de modelo de M-SR, realiza-se uma análise estatística da saída do procedimento em todo o conjunto de amostras de validação. A Figura 18 compara a saída do procedimento com os diferentes modelos de M-SR a partir de representação em diagrama de caixas do valor da métrica DSC para a segmentação de disco e de escavação óptica. As Tabelas 9 e 10 apresentam levantamentos estatísticos dos dados apresentados no gráfico. Além disso, compara-se as saídas obtidas dos modelos com o estado-da-arte na Tabela 11.

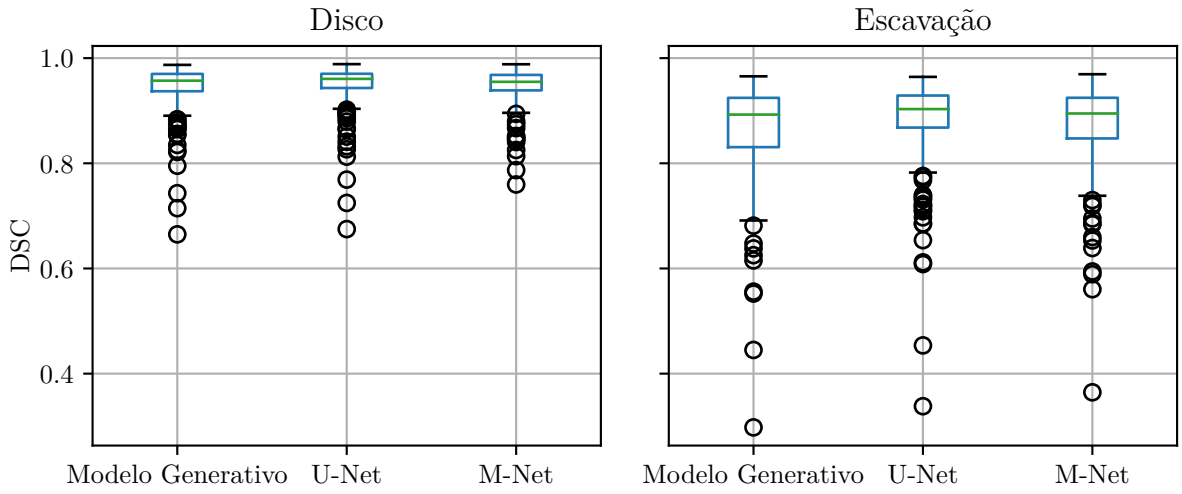


Figura 18 – Representação em diagrama de caixas da variação das métricas de DSC dos Modelos de Segmentação de Retina aplicados ao conjunto de dados de validação, considerando os limiares de valores esperados como 1,5 IQR.

Tabela 9 – Levantamento estatístico das métricas de DSC_{disco} dos resultados dos Modelos de Segmentação de Retina aplicadas nas amostras de validação.

	Modelo Generativo	**U-Net**	**M-Net**
Média	0,9445	0,9492	0,9472
Desvio Padrão	0,0432	0,0396	0,0339
Mínimo	0,6649	0,6749	0,7597
25%	0,9370	0,9432	0,9387
50%	0,9571	0,9605	0,9550
75%	0,9700	0,9703	0,9681
Máximo	0,9871	0,9886	0,9883

A partir dos resultados de segmentação obtidos, foram também calculados os valores de CDR_x e CDR_y pelo procedimento da Figura 13. Com o erro absoluto entre os valores obtidos e os valores esperados, pode-se comparar estatisticamente a precisão de cada modelo. A Figura 19 apresenta esta comparação com representações em diagramas de caixa, e as Tabelas 12 e 13 evidenciam os dados estatísticos dos erros absolutos de CDR_x e CDR_y, respectivamente.

Tabela 10 – Levantamento estatístico das métricas de $DSC_{escavação}$ dos resultados dos Modelos de Segmentação de Retina aplicadas nas amostras de validação.

	Modelo Generativo	**U-Net**	**M-Net**
Média	0,8659	0,8817	0,8713
Desvio Padrão	0,0886	0,0813	0,0814
Mínimo	0,2978	0,3381	0,3645
25%	0,8307	0,8679	0,8473
50%	0,8926	0,9031	0,8947
75%	0,9245	0,9289	0,9246
Máximo	0,9654	0,9644	0,9694

Tabela 11 – Comparação das métricas DSC_{disco} e $DSC_{escavação}$ obtidas dos modelos M-SR estudados com resultados do estado-da-arte.

	DSC_{disco} Médio	**$DSC_{escavação}$ Médio**
Modelo Generativo	0,9445	0,8659
U-Net Proposta	0,9492	0,8817
M-Net Proposta	0,9472	0,8713
U-Net Original [24]	0,94	0,83
M-Net Original [27]	0,98	0,93

Tabela 12 – Levantamento estatístico dos erros absolutos do cálculo de CDR_x a partir dos resultados de cada Modelo de Segmentação de Retina.

	Modelo Generativo	**U-Net**	**M-Net**
Média	0,0518	0,0487	0,0559
Desvio Padrão	0,0476	0,0448	0,0495
Mínimo	0,0000	0,0003	0,0000
25%	0,0185	0,0173	0,0181
50%	0,0383	0,0371	0,0442
75%	0,0690	0,0675	0,0815
Máximo	0,3092	0,3156	0,3020

Com os gráficos estatísticos e tabelas levantados, observa-se que os três modelos de M-SR apresentam resultados satisfatórios para a segmentação de retina e para o cálculo de CDR, evidenciados, respectivamente, pelas altas médias de DSC de segmentação e baixas MAE no cálculo de CDR. Nota-se, também, que os resultados obtidos são comparáveis com o estado-da-arte, apresentando uma pequena diferença devido ao uso de bases de dados e processo de treinamento diferentes. Finalmente, percebe-se que ao adotar o modelo generativo como M-SR, os resultados apresentam menor erro máximo absoluto, em relação a adotar modelos convolucionais. Esta robustez é inerente ao processo de treinamento de modelos generativos, que continuamente corrige as predições de erros maiores do conjunto de amostras de treinamento.

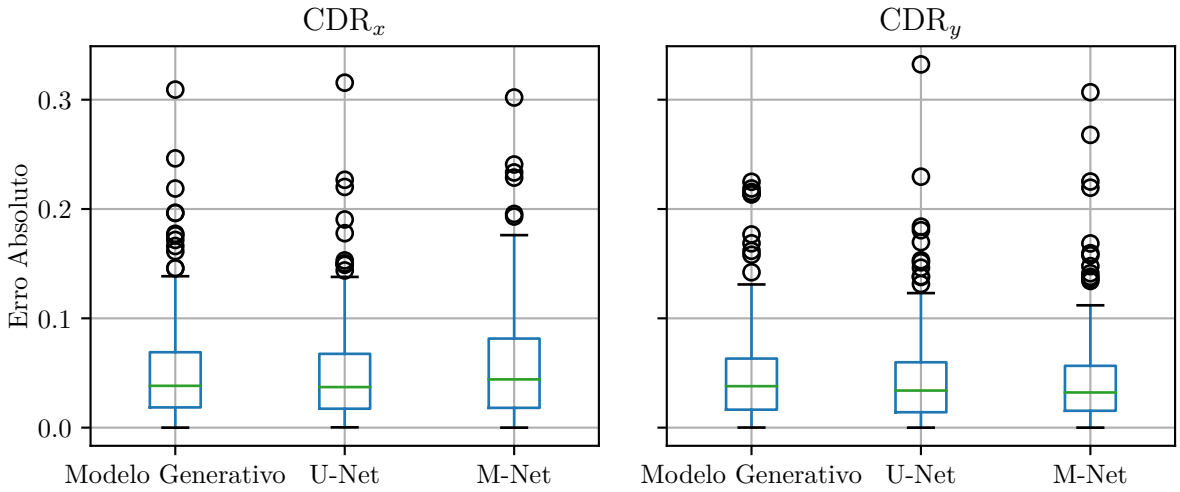


Figura 19 – Representação em diagrama de caixas da variação do erro absoluto do cálculo de CDR a partir dos resultados dos Modelos de Segmentação de Retina aplicados ao conjunto de dados de validação, considerando os limiares de valores esperados como 1,5 IQR.

Tabela 13 – Levantamento estatístico dos erros absolutos do cálculo de CDR$_y$ a partir dos resultados de cada Modelo de Segmentação de Retina.

	Modelo Generativo	**U-Net**	**M-Net**
Média	0,0479	0,0434	0,0433
Desvio Padrão	0,0432	0,0425	0,0447
Mínimo	0,0001	0,0000	0,0000
25%	0,0164	0,0141	0,0154
50%	0,0379	0,0339	0,0322
75%	0,0632	0,0598	0,0565
Máximo	0,2249	0,3323	0,3068

5 Conclusões e Trabalhos Futuros

O presente trabalho propôs o uso de modelos baseados em CGAN para a segmentação de escavação e disco óptico em imagens de fundo de retina. As dimensões das segmentações obtidas provêm dados relevantes para o cálculo de CDR, que corresponde a uma métrica importante para o diagnóstico de glaucoma.

Os resultados apontam que o modelo generativo proposto é capaz de generalizar a tarefa de segmentação de retina com precisão comparável a modelos convolucionais do estado-da-arte, obtendo DSC médio de 0,9445 para a segmentação de disco e 0,8659 para a segmentação de escavação. Ao validar o processo completo de cálculo de CDR a partir dos resultados de segmentação, os MAE observados são de 5,18% e 4,79% para CDR horizontal e vertical, respectivamente. Apesar de se tratar de modelos com processo de treinamento instável devido à sua natureza adversária, CGAN são capazes de adquirir robustez maior em predições de dados, resultado em menor número de resultados discrepantes.

Para trabalhos futuros, recomenda-se adotar variações no processo de treinamento dos módulos gerador e discriminador do modelo generativo, considerando arquiteturas com maior número de camadas convolucionais, diferentes métodos de otimização, diferentes funções de perda e diferentes funções de ativação. Além disso, o uso de uma base de dados com maior número de imagens pode auxiliar a obtenção de um modelo mais preciso, ao custo de um processo de treinamento mais longo.

Finalmente, é importante notar que a arquitetura generativa é um modelo de redes neurais flexível. Desta forma, o modelo generativo proposto pode ser utilizado em outras aplicações com poucas modificações. Por exemplo, é possível utilizar o modelo apresentado para identificar outras estruturas adversas na retina, tais como hemorragias, exsudatos e papiledema, auxiliando o diagnóstico de mais patologias oculares.

APÊNDICE A – Lista de Algoritmos

Algoritmo A.1 – Definição de classe que descreve o processamento de dados necessário para o treinamento dos modelos

```
1  import os, glob
2  import tensorflow as tf
3
4  class DataParser(object):
5
6    def __init__(self, image_path, label_path, resolution=(256,256),
         data_augmentation=False):
7      self.image_paths = glob.glob(os.path.join(image_path, '*.jpg'))
8      self.label_paths = glob.glob(os.path.join(label_path, '*.png'))
9      self.resolution = resolution
10     self.data_augmentation = data_augmentation
11     self.rng = tf.random.Generator.from_seed(42, alg='philox')
12
13   def _load_data(self, image_path, label_path):
14     input_image = tf.io.read_file(image_path)
15     input_image = tf.io.decode_jpeg(input_image)
16     input_image = tf.cast(input_image, tf.float32)
17     label_image = tf.io.read_file(label_path)
18     label_image = tf.io.decode_png(label_image)
19     label_image = tf.cast(label_image, tf.float32)
20
21     if self.data_augmentation:
22       flip_seed_h = self.rng.make_seeds(2)[0]
23       input_image = tf.image.stateless_random_flip_left_right(input_image,
             flip_seed_h)
24       label_image = tf.image.stateless_random_flip_left_right(label_image,
             flip_seed_h)
25       flip_seed_v = self.rng.make_seeds(2)[0]
26       input_image = tf.image.stateless_random_flip_up_down(input_image,
             flip_seed_v)
27       label_image = tf.image.stateless_random_flip_up_down(label_image,
             flip_seed_v)
28       zoom_seed = self.rng.make_seeds(2)[0]
29       zoom = tf.random.stateless_uniform([1], zoom_seed, minval=0, maxval
             =64, dtype=tf.dtypes.int32)[0]
30       input_image = tf.image.resize(input_image, (self.resolution[0] + zoom
             , self.resolution[1] + zoom), method=tf.image.ResizeMethod.
             LANCZOS3)
31       label_image = tf.image.resize(label_image, (self.resolution[0] + zoom
             , self.resolution[1] + zoom), method=tf.image.ResizeMethod.
```

```
            LANCZOS3)
32        crop_seed = self.rng.make_seeds(2)[0]
33        input_image = tf.image.stateless_random_crop(input_image, (self.
              resolution[0], self.resolution[1], 3), seed=crop_seed)
34        label_image = tf.image.stateless_random_crop(label_image, (self.
              resolution[0], self.resolution[1], 1), seed=crop_seed)
35      else:
36        input_image = tf.image.resize(input_image, self.resolution, method=tf
              .image.ResizeMethod.LANCZOS3)
37        label_image = tf.image.resize(label_image, self.resolution, method=tf
              .image.ResizeMethod.LANCZOS3)
38
39      input_image = (input_image / 127.5) - 1
40      label_image = (label_image / 127.5) - 1
41      return input_image, label_image
42
43    def data_batch(self, batch_size):
44      data = tf.data.Dataset.from_tensor_slices((self.image_paths, self.
            label_paths))
45      data = data.map(self._load_data, num_parallel_calls=tf.data.AUTOTUNE)
46      shuffle_seed = self.rng.make_seeds(1)[0][0]
47      return data.prefetch(tf.data.AUTOTUNE).shuffle(len(self.image_paths),
            seed=shuffle_seed).batch(batch_size)
```

Algoritmo A.2 – Implementação dos módulos gerador e discriminador do modelo generativo proposto

```
1  import tensorflow as tf
2  import tqdm
3
4  def encoder_block(layer_in, n_filters, batchnorm=True):
5    init = tf.keras.initializers.RandomNormal(stddev=0.02)
6    g = tf.keras.layers.Conv2D(
7      n_filters, (4,4), strides=(2,2),
8      padding='same',
9      kernel_initializer=init
10   )(layer_in)
11   if batchnorm:
12     g = tf.keras.layers.BatchNormalization()(g, training=True)
13   g = tf.keras.layers.LeakyReLU(alpha=0.2)(g)
14   return g
15
16 def decoder_block(layer_in, skip_in, n_filters, dropout=True):
17   init = tf.keras.initializers.RandomNormal(stddev=0.02)
18   g = tf.keras.layers.Conv2DTranspose(
19     n_filters, (4,4), strides=(2,2),
20     padding='same',
21     kernel_initializer=init
```

```
22    )(layer_in)
23    g = tf.keras.layers.BatchNormalization()(g, training=True)
24    if dropout:
25      g = tf.keras.layers.Dropout(0.5)(g, training=True)
26    g = tf.keras.layers.Concatenate()([g, skip_in])
27    g = tf.keras.layers.Activation('relu')(g)
28    return g
29
30  def discriminator_model(image_shape=(256,256,3), label_shape=(256,256,1)):
31    init = tf.keras.initializers.RandomNormal(stddev=0.02)
32    in_src_image = tf.keras.layers.Input(shape=image_shape)
33    in_target_image = tf.keras.layers.Input(shape=label_shape)
34    merged = tf.keras.layers.Concatenate()([in_src_image, in_target_image])
35    e1 = encoder_block(merged, 64, batchnorm=False)
36    e2 = encoder_block(e1, 128)
37    e3 = encoder_block(e2, 256)
38    zero_pad1 = tf.keras.layers.ZeroPadding2D()(e3)
39    conv = tf.keras.layers.Conv2D(
40      512, 4, strides=1,
41      kernel_initializer=init,
42      use_bias=False
43    )(zero_pad1)
44    batchnorm1 = tf.keras.layers.BatchNormalization()(conv)
45    leaky_relu = tf.keras.layers.LeakyReLU()(batchnorm1)
46    zero_pad2 = tf.keras.layers.ZeroPadding2D()(leaky_relu)
47    patch_out = tf.keras.layers.Conv2D(
48      1, (4,4), strides=1,
49      kernel_initializer=init
50    )(zero_pad2)
51    return tf.keras.models.Model([in_src_image, in_target_image], patch_out)
52
53  def generator_model(image_shape=(256,256,3), output_channels=3,
        output_activation='tanh'):
54    init = tf.keras.initializers.RandomNormal(stddev=0.02)
55    in_image = tf.keras.layers.Input(shape=image_shape)
56    e1 = encoder_block(in_image, 64, batchnorm=False)
57    e2 = encoder_block(e1, 128)
58    e3 = encoder_block(e2, 256)
59    e4 = encoder_block(e3, 512)
60    e5 = encoder_block(e4, 512)
61    b = tf.keras.layers.Conv2D(
62      512, (4,4), strides=(2,2),
63      padding='same',
64      kernel_initializer=init,
65      activation='relu'
66    )(e5)
67    d3 = decoder_block(b, e5, 512)
```

```
68    d4 = decoder_block(d3, e4, 512, dropout=False)
69    d5 = decoder_block(d4, e3, 256, dropout=False)
70    d6 = decoder_block(d5, e2, 128, dropout=False)
71    d7 = decoder_block(d6, e1, 64, dropout=False)
72    out_image = tf.keras.layers.Conv2DTranspose(
73      output_channels, (4,4), strides=(2,2),
74      padding='same',
75      kernel_initializer=init,
76      activation=output_activation
77    )(d7)
78    return tf.keras.models.Model(in_image, out_image)
79
80  class Pix2Pix(object):
81    def __init__(self,
82      discriminator = discriminator_model(),
83      generator = generator_model(),
84      d_optimizer = tf.keras.optimizers.Adam(learning_rate=0.0002, beta_1
          =0.5),
85      g_optimizer = tf.keras.optimizers.Adam(learning_rate=0.0002, beta_1
          =0.5),
86      loss_fn = tf.keras.losses.BinaryCrossentropy(from_logits=True)
87    ):
88      self.d_model = discriminator
89      self.g_model = generator
90      self.d_optimizer = d_optimizer
91      self.g_optimizer = g_optimizer
92      self.loss_fn = loss_fn
93
94    @tf.function
95    def train_step(self, data):
96      input_image, label_image = data
97      with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
98        gen_output = self.g_model(input_image, training=True)
99        disc_real_output = self.d_model([input_image, label_image], training=
            True)
100       disc_fake_output = self.d_model([input_image, gen_output], training=
            True)
101       gen_gan_loss = self.loss_fn(tf.ones_like(disc_fake_output),
            disc_fake_output)
102       gen_l1_loss = tf.reduce_mean(tf.abs(label_image - gen_output))
103       gen_total_loss = gen_gan_loss + (100 * gen_l1_loss)
104       disc_real_loss = self.loss_fn(tf.ones_like(disc_real_output),
            disc_real_output)
105       disc_fake_loss = self.loss_fn(tf.zeros_like(disc_fake_output),
            disc_fake_output)
106       disc_total_loss = disc_real_loss + disc_fake_loss
107
```

```
108       g_gradients = gen_tape.gradient(gen_total_loss, self.g_model.
              trainable_variables)
109       d_gradients = disc_tape.gradient(disc_total_loss, self.d_model.
              trainable_variables)
110       self.g_optimizer.apply_gradients(zip(g_gradients, self.g_model.
              trainable_variables))
111       self.d_optimizer.apply_gradients(zip(d_gradients, self.d_model.
              trainable_variables))
112       return gen_total_loss, disc_total_loss
113
114     def save_models(self):
115       self.g_model.save('g_model.h5')
116       self.d_model.save('d_model.h5')
117
118     def fit(self, train_ds, epochs):
119       # Epochs iteration
120       for epoch in range(epochs):
121         # Steps iteration
122         with tqdm.tqdm(
123           desc=f'Epoch {epoch+1:02d}/{epochs:02d}',
124           total=tf.data.experimental.cardinality(train_ds).numpy()
125         ) as pbar:
126           for data in train_ds:
127             gtl, dtl = self.train_step(data)
128             pbar.set_postfix_str(f'g_loss={gtl.numpy():6.3f}, d_loss={dtl.
                  numpy():6.4f}')
129             pbar.update(1)
130
131       # Save models after training
132       self.save_models(epochs)
```

Algoritmo A.3 – Implementação do algoritmo de cálculo de CDR a partir de imagem de segmentação de disco e escavação ópticas

```
1  import cv2
2  import numpy as np
3
4  def CDR(img_path):
5
6      img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
7
8      _, thresh_disc = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY)
9      _, thresh_cup  = cv2.threshold(img, 200, 255, cv2.THRESH_BINARY)
10     contour_disc, _ = cv2.findContours(thresh_disc, cv2.RETR_TREE, cv2.
           CHAIN_APPROX_NONE)
11     contour_cup,  _ = cv2.findContours(thresh_cup,  cv2.RETR_TREE, cv2.
           CHAIN_APPROX_NONE)
12
```

```
13      contour_disc = sorted(contour_disc, key=lambda x: cv2.contourArea(x),
            reverse=True)
14      contour_cup  = sorted(contour_cup,  key=lambda x: cv2.contourArea(x),
            reverse=True)
15
16      ellipse_disc = cv2.fitEllipse(contour_disc[0])
17      ellipse_cup = cv2.fitEllipse(contour_cup[0])
18
19      img_ellipse_disc = np.zeros(img.shape[0:2])
20      cv2.ellipse(img_ellipse_disc, ellipse_disc, 1, 1)
21      disc_intersections_x = [0, 0]
22      for i in range(int(ellipse_disc[0][0] - np.max(ellipse_disc[1])), int(
            ellipse_disc[0][0])):
23          if cv2.pointPolygonTest(contourDisc[0], (i, int(ellipse_disc[0][1])
                ), False) >= 0:
24              disc_intersections_x[0] = i
25              break
26
27      for i in reversed(range(int(ellipse_disc[0][0]), int(ellipse_disc[0][0]
            + np.max(ellipse_disc[1])))):
28          if cv2.pointPolygonTest(contourDisc[0], (i, int(ellipse_disc[0][1])
                ), False) >= 0:
29              disc_intersections_x[1] = i
30              break
31
32      disc_intersections_y = [0, 0]
33      for i in range(int(ellipse_disc[0][1] - np.max(ellipse_disc[1])), int(
            ellipse_disc[0][1])):
34          if cv2.pointPolygonTest(contourDisc[0], (int(ellipse_disc[0][0]), i
                ), False) >= 0:
35              disc_intersections_y[0] = i
36              break
37
38      for i in reversed(range(int(ellipse_disc[0][1]), int(ellipse_disc[0][1]
            + np.max(ellipse_disc[1])))):
39          if cv2.pointPolygonTest(contourDisc[0], (int(ellipse_disc[0][0]), i
                ), False) >= 0:
40              disc_intersections_y[1] = i
41              break
42
43      cv2.line(img_ellipse_disc, (disc_intersections_x[0], int(ellipse_disc
            [0][1])), (disc_intersections_x[1], int(ellipse_disc[0][1])), 1, 1)
44      cv2.line(img_ellipse_disc, (int(ellipse_disc[0][0]),
            disc_intersections_y[0]), (int(ellipse_disc[0][0]),
            disc_intersections_y[1]), 1, 1)
45
46      img_ellipse_cup = np.zeros(maskImg.shape[0:2])
```

```
47      cv2.ellipse(img_ellipse_cup, ellipse_cup, 1, 1)
48      cup_intersections_x = [0, 0]
49      for i in range(int(ellipse_cup[0][0] - np.max(ellipse_cup[1])), int(
            ellipse_cup[0][0])):
50          if cv2.pointPolygonTest(contourCup[0], (i, int(ellipse_cup[0][1])),
                False) >= 0:
51              cup_intersections_x[0] = i
52              break
53
54      for i in reversed(range(int(ellipse_cup[0][0]), int(ellipse_cup[0][0] +
            np.max(ellipse_cup[1])))):
55          if cv2.pointPolygonTest(contourCup[0], (i, int(ellipse_cup[0][1])),
                False) >= 0:
56              cup_intersections_x[1] = i
57              break
58
59      cup_intersections_y = [0, 0]
60      for i in range(int(ellipse_cup[0][1] - np.max(ellipse_cup[1])), int(
            ellipse_cup[0][1])):
61          if cv2.pointPolygonTest(contourCup[0], (int(ellipse_cup[0][0]), i),
                False) >= 0:
62              cup_intersections_y[0] = i
63              break
64
65      for i in reversed(range(int(ellipse_cup[0][1]), int(ellipse_cup[0][1] +
            np.max(ellipse_cup[1])))):
66          if cv2.pointPolygonTest(contourCup[0], (int(ellipse_cup[0][0]), i),
                False) >= 0:
67              cup_intersections_y[1] = i
68              break
69
70      cv2.line(img_ellipse_cup, (cup_intersections_x[0], int(ellipse_cup
            [0][1])),
71               (cup_intersections_x[1], int(ellipse_cup[0][1])), 1, 1)
72      cv2.line(img_ellipse_cup, (int(ellipse_cup[0][0]), cup_intersections_y
            [0]),
73               (int(ellipse_cup[0][0]), cup_intersections_y[1]), 1, 1)
74
75      cdr_x = (cup_intersections_x[1] - cup_intersections_x[0]) / (
            disc_intersections_x[1] - disc_intersections_x[0])
76      cdr_y = (cup_intersections_y[1] - cup_intersections_y[0]) / (
            disc_intersections_y[1] - disc_intersections_y[0])
77
78      return cdr_x, cdr_y
```

Referências

1 PITAS, I. *Digital image processing algorithms and applications*. [S.l.]: John Wiley & Sons, 2000. 15

2 EKSTROM, M. P. *Digital image processing techniques*. [S.l.]: Academic Press, 2012. v. 2. 15

3 EGMONT-PETERSEN, M.; RIDDER, D. de; HANDELS, H. Image processing with neural networks—a review. *Pattern recognition*, Elsevier, v. 35, n. 10, p. 2279–2301, 2002. 15

4 CHEN, Y.-Y. et al. Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 9, p. 2047, 2019. 15

5 PAPIK, K. et al. Application of neural networks in medicine-a review. *Medical Science Monitor*, International Scientific Information, Inc., v. 4, n. 3, p. MT538–MT546, 1998. 16

6 BRASIL, M. da S. *Portaria nº 11, de 02 de abril de 2018. Aprova o Protocolo Clínico e Diretrizes Terapêuticas do Glaucoma*. 2018. 100 p. 16, 17

7 RESNIKOFF, S. et al. Global data on visual impairment in the year 2002. *Bulletin of the world health organization*, SciELO Public Health, v. 82, p. 844–851, 2004. 16

8 LEE, D. A.; HIGGINBOTHAM, E. J. Glaucoma and its treatment: a review. *American journal of health-system pharmacy*, Oxford University Press, v. 62, n. 7, p. 691–699, 2005. 16

9 OTTAIANO, J. A. A. et al. Cegueira e baixa visão no brasil. In: *As condições de saúde ocular no Brasil: 2019*. [S.l.]: Conselho Brasileiro de Oftalmologia, 2019. v. 1, p. 35–62. 16

10 WEINREB, R. N.; AUNG, T.; MEDEIROS, F. A. The pathophysiology and treatment of glaucoma: a review. *Jama*, American Medical Association, v. 311, n. 18, p. 1901–1911, 2014. 16

11 GRAZIANO, R. M.; LEONE, C. R. Problemas oftalmológicos mais freqüentes e desenvolvimento visual do pré-termo extremo. *Jornal de Pediatria*, SciELO Brasil, v. 81, n. 1, p. S95–S100, 2005. 16, 17

12 WEINREB, R. N.; KHAW, P. T. Primary open-angle glaucoma. *The lancet*, Elsevier, v. 363, n. 9422, p. 1711–1720, 2004. 16

13 ORLANDO, J. I. et al. Refuge challenge: A unified framework for evaluating automated methods for glaucoma assessment from fundus photographs. *Medical image analysis*, Elsevier, v. 59, p. 21, 2020. 17, 34

14 CHRÁSTEK, R. et al. Optic disc segmentation in retinal images. In: *Bildverarbeitung für die Medizin 2002*. [S.l.]: Springer, 2002. p. 263–266. 18

15 HOOVER, A.; GOLDBAUM, M. Locating the optic nerve in a retinal image using the fuzzy convergence of the blood vessels. *IEEE transactions on medical imaging*, IEEE, v. 22, n. 8, p. 951–958, 2003. 18

16 AQUINO, A.; GEGÚNDEZ-ARIAS, M. E.; MARÍN, D. Detecting the optic disc boundary in digital fundus images using morphological, edge detection, and feature extraction techniques. *IEEE transactions on medical imaging*, IEEE, v. 29, n. 11, p. 1860–1869, 2010. 18

17 DECENCIÈRE, E. et al. Feedback on a publicly distributed image database: the messidor database. *Image Analysis & Stereology*, v. 33, n. 3, p. 231–234, 2014. 18, 34

18 CHENG, J. et al. Automatic optic disc segmentation with peripapillary atrophy elimination. In: IEEE. *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. [S.l.], 2011. p. 6224–6227. 18

19 ZHANG, Z. et al. Origa-light: An online retinal fundus image database for glaucoma analysis and research. In: IEEE. *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*. [S.l.], 2010. p. 3065–3068. 18

20 MURAMATSU, C. et al. Automated segmentation of optic disc region on retinal fundus photographs: Comparison of contour modeling and pixel classification methods. *Computer methods and programs in biomedicine*, Elsevier, v. 101, n. 1, p. 23–32, 2011. 18

21 YIN, F. et al. Automated segmentation of optic disc and optic cup in fundus images for glaucoma diagnosis. In: IEEE. *2012 25th IEEE international symposium on computer-based medical systems (CBMS)*. [S.l.], 2012. p. 1–6. 18

22 CHENG, J. et al. Superpixel classification based optic disc and optic cup segmentation for glaucoma screening. *IEEE transactions on medical imaging*, IEEE, v. 32, n. 6, p. 1019–1032, 2013. 18

23 XU, Y. et al. Optic cup segmentation for glaucoma detection using low-rank superpixel representation. In: SPRINGER. *International conference on medical image computing and computer-assisted intervention*. [S.l.], 2014. p. 788–795. 18

24 SEVASTOPOLSKY, A. Optic disc and cup segmentation methods for glaucoma detection with modification of u-net convolutional neural network. *Pattern Recognition and Image Analysis*, Springer, v. 27, n. 3, p. 618–624, 2017. 19, 47

25 RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. In: SPRINGER. *International Conference on Medical image computing and computer-assisted intervention*. [S.l.], 2015. p. 234–241. 19, 30, 32

26 ZOU, K. H. et al. Statistical validation of image segmentation quality based on a spatial overlap index1: scientific reports. *Academic radiology*, Elsevier, v. 11, n. 2, p. 178–189, 2004. 19, 39

27 FU, H. et al. Joint optic disc and cup segmentation based on multi-label deep network and polar transformation. *IEEE transactions on medical imaging*, IEEE, v. 37, n. 7, p. 1597–1605, 2018. 19, 47

28 HAYKIN, S. *Neural networks and learning machines, 3/E*. [S.l.]: Pearson Education India, 2010. 21

29 SILVA, I. N. da et al. Introduction. In: _____. *Artificial Neural Networks : A Practical Course*. Cham: Springer International Publishing, 2017. p. 3–19. ISBN 978-3-319-43162-8. Disponível em: <https://doi.org/10.1007/978-3-319-43162-8_1>. 21, 22

30 SILVA, I. N. D. et al. Artificial neural network architectures and training processes. In: *Artificial neural networks*. [S.l.]: Springer, 2017. p. 21–28. 21, 23, 24, 25

31 WANG, S.-C. Artificial neural network. In: *Interdisciplinary computing in java programming*. [S.l.]: Springer, 2003. p. 81–100. 21

32 GRAUPE, D. *Principles of artificial neural networks*. [S.l.]: World Scientific, 2013. v. 7. 21

33 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. 22, 25, 26

34 MILLETARI, F.; NAVAB, N.; AHMADI, S.-A. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: IEEE. *2016 fourth international conference on 3D vision (3DV)*. [S.l.], 2016. p. 565–571. 24

35 RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 24

36 SILVA, I. N. da et al. Multilayer perceptron networks. In: _____. *Artificial Neural Networks : A Practical Course*. Cham: Springer International Publishing, 2017. p. 55–115. ISBN 978-3-319-43162-8. Disponível em: <https://doi.org/10.1007/978-3-319-43162-8_5>. 25

37 DUMOULIN, V.; VISIN, F. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. 27

38 GOODFELLOW, I. J. et al. *Generative Adversarial Networks*. 2014. 27, 28

39 CRESWELL, A. et al. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, IEEE, v. 35, n. 1, p. 53–65, 2018. 27, 29

40 MIRZA, M.; OSINDERO, S. *Conditional Generative Adversarial Nets*. 2014. 28

41 ISOLA, P. et al. Image-to-image translation with conditional adversarial networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2017. p. 1125–1134. 29, 30

42 LI, C.; WAND, M. Precomputed real-time texture synthesis with markovian generative adversarial networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 702–716. 30

43 ALMAZROA, A. et al. Retinal fundus images for glaucoma analysis: the riga dataset. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Medical Imaging 2018: Imaging Informatics for Healthcare, Research, and Applications*. [S.l.], 2018. v. 10579, p. 105790B. 34

44 FUMERO, F. et al. Rim-one: An open retinal image database for optic nerve evaluation. In: IEEE. *2011 24th international symposium on computer-based medical systems (CBMS)*. [S.l.], 2011. p. 1–6. 34

45 SIVASWAMY, J. et al. Drishti-gs: Retinal image dataset for optic nerve head (onh) segmentation. In: IEEE. *2014 IEEE 11th international symposium on biomedical imaging (ISBI)*. [S.l.], 2014. p. 53–56. 34

46 PORWAL, P. et al. Indian diabetic retinopathy image dataset (idrid): A database for diabetic retinopathy screening research. *Data*, v. 3, n. 3, 2018. ISSN 2306-5729. Disponível em: <https://www.mdpi.com/2306-5729/3/3/25>. 34

47 CARMONA, E. J. et al. Identification of the optic nerve head with genetic algorithms. *Artificial Intelligence in Medicine*, Elsevier, v. 43, n. 3, p. 243–259, 2008. 34

48 STAAL, J. et al. Ridge-based vessel segmentation in color images of the retina. *IEEE transactions on medical imaging*, IEEE, v. 23, n. 4, p. 501–509, 2004. 34

49 DUCHON, C. E. Lanczos filtering in one and two dimensions. *Journal of Applied Meteorology and Climatology*, v. 18, n. 8, p. 1016–1022, 1979. 36

50 KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 38

51 FITZGIBBON, A. W.; FISHER, R. B. et al. *A buyer's guide to conic fitting*. [S.l.]: Citeseer, 1996. 38