Redes Neurais Convolucionais Aplicadas ao Projeto de Operadores de Imagens

André Vinícius Lopes

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS DA COMPUTAÇÃO

Programa: Ciência da Computação

Orientador: Prof. Dr. Roberto Hirata Jr

Durante parte do desenvolvimento deste trabalho o autor recebeu auxílio financeiro da CNPq

São Paulo, Dezembro de 2017

Resumo

LOPES, André Vinícius. **Redes Neurais Convolucionais Aplicadas ao Projeto de Operadores de Imagens.** Dissertação - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

O projeto de W-operadores de imagens requer a estimação de um operador local a partir de exemplos de treinamento e da indução de um classificador baseado em aprendizado de máquina para a classificação de exemplos pouco, ou nunca, observados no treinamento.

Nos últimos anos, a área de aprendizado de máquina passou por um avanço muito grande devido às redes neurais convolucionais (CNN). Esse avanço é principalmente devido ao poder de representação das redes neurais e pelo fato das redes convolucionais serem efetivas na extração de características locais. Devido a isso, elas estão presentes em muitas soluções do estado da arte de diversos problemas de visão computacional [MPGC17, HGC+17, FTM+17, MZY+17, CGW+17].

Neste trabalho, estudamos e exploramos o poder de representação das CNNs no contexto do projeto de W-operadores de imagens. Integramos implementações públicas e bastante maduras de CNN a uma biblioteca de projeto de W-operadores desenvolvida pelo nosso grupo (TRIOS) e testamos diversas estratégias para segmentar imagens de níveis de cinza ou, ainda, classificar os padrões de intensidades em níveis de cinza observados através de uma janela W em poucos rótulos (em geral, dois rótulos, ou 0, ou 1).

Para validar a proposta, usamos 2 conjuntos de dados de imagens de fundo de olho, chamados de DRIVE e STARE, os quais já são um padrão na área de imagens para a segmentação das veias da retina e também em um conjunto de dados chamado de STAFF [KFV+13, VKFJ13], o qual é uma variação do banco de dados CVC-MUSCIMA [FDGL12] e tem o objetivo de segmentar notas musicais em partituras. Os resultados obtidos mostram que, para uma janela razoavelmente grande, os resultados são satisfatórias ao se comparar com soluções específicas do estado da arte, as quais utilizam heurísticas de pré e pós-processamento.

Palavras-chave: redes neurais, redes neurais convolucionais, deep-learning, operadores de imagem.

Abstract

LOPES, André Vinícius. **Convolutional neural networks model applied to construction of image operators.** 2017. 120 f. Dissertação - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2017.

The project of images W-operators requires the estimation of a local operator using training examples and the induction of a classifier based on machine learning to classify examples that are seldom or never seen during training.

In the last years, the area of machine learning advanced enormously due to the use of convolutional neural networks (CNN). This advance is caused mainly due to the power of representation of the neural networks and due to the fact that the convolutional neural networks are effective in the extraction of local characteristics. Consequently, they are present in many state of the art solutions for problems of computer vision [MPGC17, HGC+17, FTM+17, MZY+17, CGW+17].

In this work we study and explore the power of representation of the CNN's in the context of images w-operators project. We integrate public implementations and very mature libraries of CNN and w-operators developed by our group (TRIOS) and test several strategies to segment gray-level images or to classify the gray-level intensity patterns observed through a window w in a few labels (in general, 2 labels, either 0 or 1).

To validate this proposal we use 2 data sets of retinal images, called DRIVE and STARE which are commonly used for vessel segmentation of the retina and also in a data set called STAFF [KFV+13, VKFJ13], which is a variation of the database CVC-MUSCIMA [FDGL12] and has the objective of segmenting musical notes in partitures. The results have shown that, for a big window, the results are satisfactory when compared to specific state of the art solutions which use pre and post-processing.

Keywords: neural networks, convolutional neural networks, deep-learning, image operators.

Sumário

Lista de Abreviaturas

DL	Aprendizado profundo (*Deep Learning*)
RNA	Redes Neurais Artificiais (*Artificial Neural Networks*)
RELU	Unidades retificadoras lineares (*Rectified Linear Units*)
GPGPU	Unidade de Processamento Gráfico de Propósito Geral (*General Purpose Graphics Processing Unit*)
MLP	Perceptron de múltiplas camadas (*Multilayer Perceptron*)
CNN	Rede Neural Convolucional (*Convolutional Neural Networks*)
TANH	Tangente Hiperbólica
TP	Verdadeiro Positivo (*True Positive*)
FP	Falso Positivo (*False Positive*)
TN	Verdadeiro Negativo (*True Negative*)
FN	Falso Negativo (*False Negative*)
PPV	Valores preditivos positivos (*Positive predictive values*)
F1	Medida chamada de F-score que combina a sensitividade e a precisão (PPV)
GD	Gradiente Descendente
CLAHE	Histograma adaptativo com limitação de contraste (*Contrast limited adaptive histogram equalization*)

Lista de Símbolos

W	Janela de extração de padrões
w	Peso de uma rede neural (*weight*)
Ψ	Operador de imagem
I	Imagem ou matriz
I_w	Largura da imagem ou matriz
I_h	Altura da imagem ou matriz
O	Imagem de saída
p	Pixel de uma imagem
$\mathfrak{p}$	Preenchimento (*Padding*)
D	Conjunto de dados (*Dataset*)
i	Linha de uma matriz ou vetor
j	Coluna de uma matriz ou vetor
d	Dimensão de uma matriz ou vetor
n	Tamanho de uma dimensão de uma matriz ou vetor
X	Conjunto de exemplos de um conjunto de dado
x	Exemplo pertencente ao conjunto de exemplos X
Y	Conjunto de rótulos de um conjunto de dados
y	Rótulo pertencente ao conjunto de rótulos Y
E_{in}	Quantidade de erro no conjunto de treinamento
E_{out}	Quantidade de erro no conjunto de teste
$\mathbb{E}$	Esperança matemática ou expectância
$\mathfrak{h}$	Modelo de classificação ou regressão
$\mathfrak{f}$	Modelo ideal de classificação ou regressão
$\mathfrak{r}$	Probabilidade na camada de (*dropout*)
$\mathfrak{t}$	Extensão espacial na camada de (*max-pooling*)
ψ	Transformação de uma imagem por um operador de imagem
ϕ	Função de ativação

ϵ	Taxa de aprendizagem
b	Tamanho do lote
α	Momento (*Momentum*)
g	Imagem em níveis de cinza
h	Imagem binária
s	Passo (*Stride*)
τ	Deslocamento (W-Operadores)
$\mathcal{G}_i$	Conjunto de imagens de entrada
$\mathcal{H}_i$	Conjunto de imagens de saída

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Extrair informações de imagens não é uma tarefa trivial; na área de análise de imagens médicas, por exemplo, é necessário segmentar áreas de interesse, verificar agrupamento de objetos, analisar formas, tais como, retas, curvas e bordas, tratar estruturas subpíxel, etc. Em geral, para muitas dessas tarefas, um especialista com conhecimento e experiência em imagens é necessário.

Uma forma de mitigar o problema é usar o conhecimento do especialista para "ensinar" um sistema automático. Isso também tem seus problemas pois o tempo do especialista não é barato e, em geral, os sistemas automáticos necessitam de muitas imagens de entrada e saída para terem bons resultados. No entanto, o auxílio de sistema automáticos é desejável e existem formas de projetar operadores de imagens automaticamente para segmentar, ou rotular imagens.

Neste trabalho vamos estudar, implementar e testar uma classe específica de operadores de imagens chamadas W-operadores. Esses operadores são assim chamadas pois eles são localmente definidos dentro de um subconjunto do domínio da imagem (basta conhecer o valor dos pontos nesse subconjunto para saber o valor da saída do operador), e são invariantes por translação, ou seja, o operador local é o mesmo em qualquer ponto do domínio da imagem. Os W-operadores foram definidos originalmente por *Dougherty* [DD11, Dou92b], usando conceitos de estimação estatística [Dou90] e teoria da decisão [Ber13, Jr01].

O projeto de W-operadores foi bastante estudado desde os anos 90 e diversos métodos de classificação foram usados como modelo do operador. O nosso objetivo será estimar

operadores de imagens usando o modelo de redes neurais profundas, que constitui hoje em dia o estado da arte em termos de modelos de classificação. Essas redes foram usadas em diversos problemas da computação visual [MPGC17, HGC+17, FTM+17, MZY+17, CGW+17], e outros.

Em geral, o uso das redes neurais se dá para segmentação de imagens, ou classificação, mas o uso para representar W-operadores está apenas no início. Os modelos de redes neurais profundas são interessantes também pois eles podem aprender funções complexas e tem alto poder de expressão sobre os dados. Em especial, as redes neurais convolucionais, assim como descritas no Capítulo 2, fazem uma extração de características (*features*) dos dados, e isto é uma grande vantagem para achar um operador de imagem com boa acurácia. Além disso, diferentemente de tempos anteriores, atualmente há um poder computacional suficiente para treinar e utilizar estes modelos de forma eficiente, principalmente pelo trabalho de Hinton, G. [HOT06], o qual apresentou uma nova maneira de realizar o treinamento de redes neurais usando GPGPU's.

Portanto, para testar a abordagem proposta, aplicamos o método para 2 conjunto de imagens de fundo de olho chamado DRIVE [SAN+04b] e STARE [Hoo75], com o objetivo de segmentar as veias presentes nas imagens e também aplicamos o método para um conjunto de dados chamado de STAFF [KFV+13, VKFJ13], o qual é uma variação do banco de dados CVC-MUSCIMA [FDGL12] e tem o objetivo de segmentar notas musicais em partituras.

1.1 Organização do Trabalho

No Capítulo 2, apresentamos os conceitos fundamentais de aprendizagem de máquina e redes neurais artificiais. Também apresentamos as redes neurais profundas (*deep learning*) e as redes convolucionais profundas, que trouxeram uma grande melhora a este modelo, com a desvantagem de serem mais complexas.

No Capítulo 3, os conceitos de operadores de imagens são apresentados, inclusive os W-operadores, os quais são objetos de estudo neste trabalho.

No Capítulo 4, discutimos os resultados dos experimentos realizados com W-operadores e redes neurais convolucionais para a segmentação de veias da retina.

No Capítulo 5, discutimos os resultados dos experimentos realizados com W-operadores e redes neurais convolucionais para a segmentação de notas musicais em partituras.

No Apêndice A há a aplicação em código deste trabalho.

No Apêndice B há o conjunto de 40 imagens do banco de dados DRIVE.

No Apêndice C há o conjunto de 20 imagens do banco de dados STARE.

Capítulo 2

Conceitos de Aprendizado de Máquina

Aprendizado de máquina é o estudo de técnicas que utilizam um conjunto de dados para criar um programa de computador que é capaz de realizar estimações para uma tarefa, a partir de dados não vistos anteriormente pelo programa.

Neste capítulo, trataremos sobre os conceitos de aprendizado de máquina, modelo de redes neurais artificiais e modelo de redes neurais convolucionais.

2.1 O problema de aprendizado

Para um programa de computador realizar estimações E para uma tarefa T, é necessário um conjunto de dados com exemplos e rótulos respectivos. Por exemplo, suponha que, como uma tarefa T, quiséssemos classificar se um dígito manuscrito é 0 ou 1, neste caso, é necessário formar um conjunto de dados D formado com exemplos de cada dígito a ser reconhecido pelo programa e o rótulo respectivo de cada exemplo. Sendo n, o número de exemplos, definimos $D = \{x_{1..n}, y_{1..n}\}$, onde x_n é um exemplo e y_n é o seu rótulo respectivo. A figura 2.1 ilustra um conjunto de dados $D = \{X, Y\}$, onde cada $x_i \in X$ é a imagem de um dígito 0 ou 1, e cada $y_i \in Y$ é o rótulo correspondente a x_i. Note que, definimos aprendizado supervisionado como o treinamento de um modelo usando um conjunto de dados que contenha exemplos e rótulos respectivos.

(a) *Dígito 0*

(b) *Dígito 1*

(c) *Dígito 0*

(d) *Dígito 1*

Figura 2.1: *Ilustração de um conjunto de dados $D = \{X, Y\}$, onde cada $x_i \in X$ é uma imagem de um dígito (0 ou 1), e cada $y_i \in Y$ é o rótulo correspondente a x_i*

2.1.1 Funções de perda

Podemos afirmar que um modelo realiza uma boa ou ótima estimação se um conjunto de dados usado apenas para testar o modelo, fez o modelo inferir informações de dados que não estavam presentes no conjunto usado para treinamento. Portanto, um conjunto de de teste é utilizado para verificar e medir a qualidade de estimação do modelo.

Define-se que E_{in} é a quantidade de erro do modelo sobre o conjunto de dados usado para treinamento, e E_{out} é a quantidade de erro do modelo sobre os dados que estão no conjunto de teste e não estão no conjunto de treinamento.

Seja n a quantidade de exemplos de um conjunto de dados e $\mathbb{E}[\bullet]$ a esperança de $\bullet$ e seja $\mathfrak{h}(x_n)$ o valor de saída do modelo e $\mathfrak{f}(x_n)$ o valor correto esperado, define-se matematicamente :

$$E_{in}(\mathfrak{h}) = \frac{1}{n}\sum_{n=1}^{n}[\mathfrak{h}(X) \neq \mathfrak{f}(X)]$$

$$E_{out}(\mathfrak{h}) = \mathbb{E}[\mathfrak{h}(X) \neq \mathfrak{f}(X)]$$

E_{in} e E_{out} são calculados como uma função de perda, a qual é utilizada para medir a eficiência do classificador ou de um modelo de regressão. Porém, se esta função não é diferenciável, muitos algoritmos que necessitam desta propriedade para poder otimizar esta função, não poderão ser utilizados. Entre estas funções que necessitam dessa propriedade, a mais

utilizada para os modelos de redes neurais artificiais é chamada de gradiente descendente. Há diversas funções de perda que podem ser utilizadas, algumas são apenas para classificação e outras para problemas de regressão.

A entropia cruzada binária é utilizada para problemas de classificação em conjuntos de dados D em que Y é composto de valores entre $[0, 1]$. Esta função calcula o grau de confiança entre o que o modelo previu e o rótulo correto. Seja $\mathfrak{h}(x_n)$ o valor de saída do modelo, $\mathfrak{f}(x_n)$ o valor esperado e K o valor resultante da função da entropia cruzada binária, defini-se matematicamente :

$$K = -\mathfrak{f}(x_n) * \log(\mathfrak{h}(x_n)) - (1 - \mathfrak{f}(x_n)) * \log(1 - \mathfrak{h}(x_n))$$

Esta função é especialmente importante para os modelos de redes neurais, que veremos adiante, que tenham saída sigmoide.

2.2 Redes Neurais Artificiais

O modelo de redes neurais artificiais foi apresentado em 1943 por McCullock, W. e Pitts, [MP43]. Este modelo é inspirado no cérebro humano, mais precisamente no seu nível mais básico, o neurônio. È uma generalização do perceptron, e muito utilizado em problemas de regressão e visão computacional [MPGC17], [HGC+17], [FTM+17], [MZY+17], [CGW+17]. Uma rede neural artificial aprende padrões e dado um conjunto de entrada, teremos um conjunto de saída. Este modelo é formado por neurônios, bias e funções de ativação.

Definimos um neurônio como um modelo matemático de um neurônio biológico, o qual recebe como entrada um ou mais valores e então multiplica sua entrada por um peso w e soma as multiplicações e o valor resultante é passado para uma função de ativação ϕ. Uma função de ativação ϕ permite fazer uma transformação, em geral, não-linear dos dados de entrada e tem como domínio e contra domínio os números reais.

Cada conexão entre dois neurônios tem um peso, que chamamos de w. Seja cada $x_i \in X$ uma entrada para o neurônio, e w_i cada peso, o valor de saída do neurônio é definido

matematicamente como :

$$f(x_i, w_i) = \phi(\sum_{i}^{d}(w_i x_i))$$

Este modelo, ainda pode ser dividido em camadas : de entrada, escondidas e de saída. Sendo que cada camada é formada por 1 ou mais neurônios

A camada de entrada é responsável por aceitar os dados que vem de fora da rede neural. Cada neurônio nesta camada aceita um único dado por vez. Note que, dependendo da escolha de função de ativação na rede neural, os dados deverão ser normalizados antes de serem passados para a camada de entrada.

A camada de saída devolve um conjunto de valores que foi estimado pela rede neural.

As camadas escondidas permite que a rede neural aprenda representações mais complexas dos dados, estas recebem dados de uma camada anterior e enviam dados resultantes dos neurônios para a camada posterior.

A figura 2.2 ilustra a estrutura de uma rede neural artificial, com suas diferentes tipos de camadas.

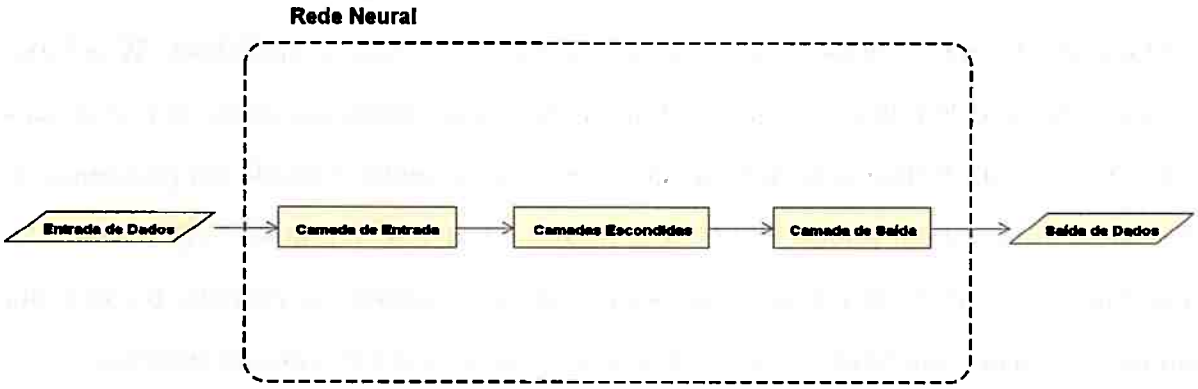


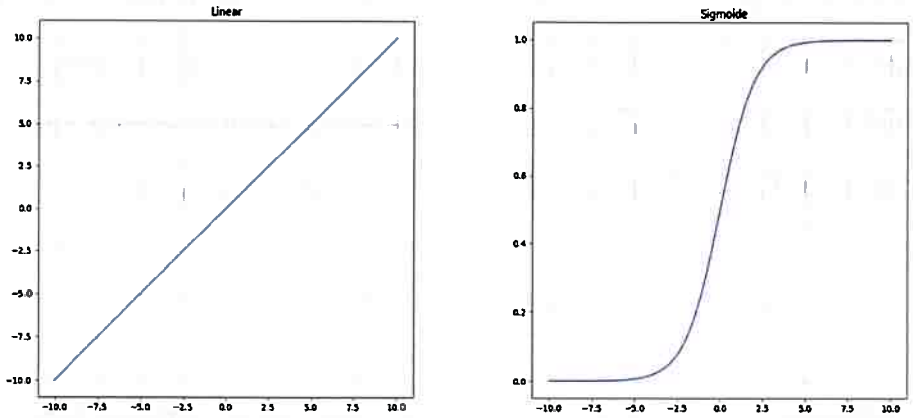
Figura 2.2: *Estrutura simplificada de uma rede neural*

2.2.1 Funções de Ativação

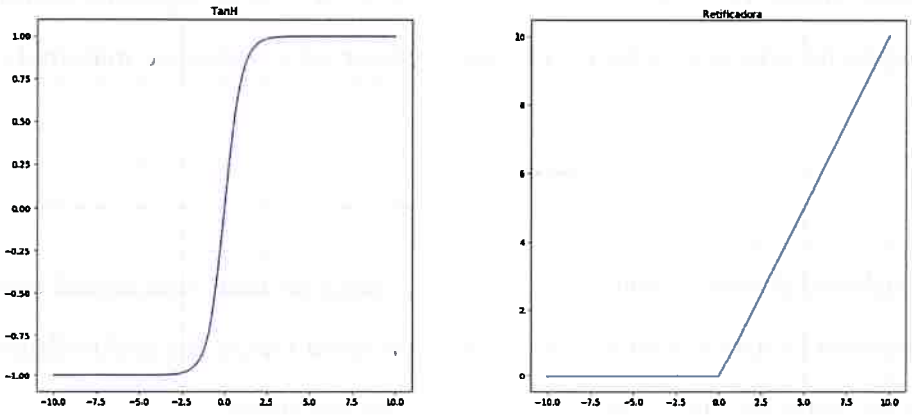
As funções de ativação definem a saída calculada pelo neurônio dado uma entrada x, sendo que x pode ser o dado vindo de um neurônio da camada de entrada ou de uma camada escondida. Escolher funções de ativação apropriadas é importante pois define se os dados de entrada deverão estar, por exemplo, normalizados entre 0 e 1. Ao usar funções do tipo sigmoide, por exemplo, um conjunto de dados D não deve ter valores negativos. Por

outro lado, caso isto seja necessário, pode-se usar a função tangente hiperbólica, a qual é descrita nesta seção. Além disso, há de se verificar qual função estará na camada de saída, pois caso seja um problema de classificação binária, por exemplo, pode-se usar a função sigmoide ou *softmax*.

A figura 2.3 ilustra as funções de ativação de uso mais comum.



(a) *Função linear e sigmoide, respectivamente*



(b) *Função tangente hiperbólica e retificadora linear, respectivamente*

Figura 2.3: *Gráfico da função de ativação linear, sigmoide, tangente hiperbólica e retificadora linear*

A função linear é a mais simples das funções de ativações, pois não muda a saída do neurônio, há apenas o retorno do valor de entrada. È comum ter esta função como saída de uma rede neural de regressão, define-se matematicamente :

$$\phi(x) = x$$

A função tangente hiperbólica [BDS07] permite valores negativos e positivos como saída, define-se matematicamente :

$$\phi(x) = tanh(x)$$

A função linear retificadora (RELU) [HSM+00] é a função mais utilizada nas redes neurais. Em diversas pesquisas, esta função causa os melhores resultados em comparação com a função sigmoide e tangente hiperbólica. Uma das causas de causar um melhor resultado é pelo fato de que esta função não satura. Uma função que satura vai em direção a um valor e se mantém neste valor eventualmente. A função sigmoide tende a saturar em 0, quando x diminui, e saturar em 1, quando x aumenta. Define-se matematicamente :

$$\phi(x) = max(0, x)$$

A função sigmoide [Ver45], também conhecida como função logística tem como saída apenas valores positivos. Esta, foi muito utilizada nas camadas escondidas das redes neurais quando a função linear retificadora ainda não existia. A função sigmoide continua sendo usada como saída de uma rede neural, nos casos apropriados, define-se matematicamente :

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

A função *softmax* [Bis06] é usada na camada de saída de uma rede neural classificadora e seu uso no modelo é ilustrada na figura 2.4. A saída desta função é a probabilidade de uma dada entrada faz parte de uma classe, define-se matematicamente :

$$\phi(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$$

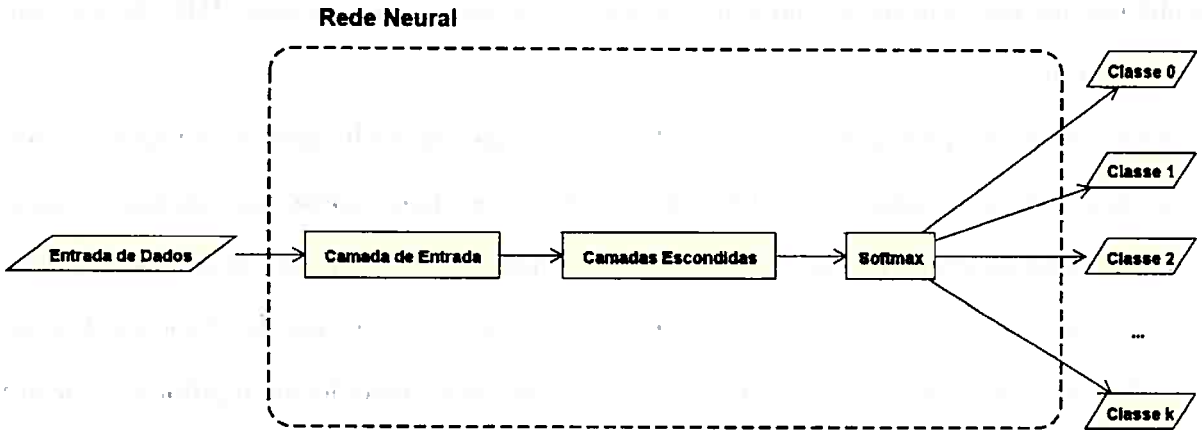


Figura 2.4: *Estrutura de rede Neural com* softmax

Note que, em geral, as redes neurais tem neurônios com função de bias, neste caso, cada camada escondida e de saída, tem um bias, o qual ajuda a rede a reconhecer padrões, pois estes permitem modificar a função de ativação. A figura 2.5 demonstra como o bias pode modificar uma função de ativação do tipo sigmoide.

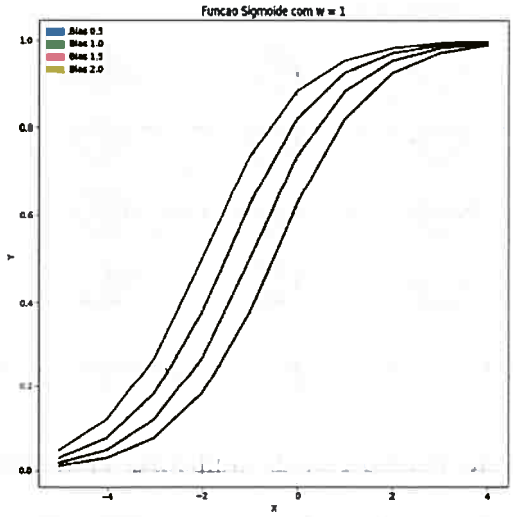


Figura 2.5: *Efeito do Bias na função sigmoide*

2.2.2 Perceptron de Múltiplas Camadas

O perceptron de múltiplas camadas (PMC) é uma classe de redes neurais que pode ser treinada utilizando aprendizado supervisionado e pode criar uma representação maior de dados do que o perceptron. Portanto, um programa P que tenha como tarefa T, um

problema que não seja trivial, pode apresentar um melhor resultado com PMC do que com o perceptron.

O teorema da aproximação universal[Csá01] diz que um PMC com uma única camada escondida com uma quantidade suficiente de neurônios, pode aproximar qualquer função continua. Este teorema, provado por George Cybenko em 1989, foi feito usando a função de ativação sigmoide e o teorema em si, fazia suposições sobre as funções de ativações. Porem, em 1991, Kurt Hornik [Hor91], mostrou que este teorema é valido devido a própria estrutura de múltiplas camadas e não devido a alguma escolha de função de ativação. A figura 2.6 ilustra um exemplo do modelo de perceptron de múltiplas camadas.

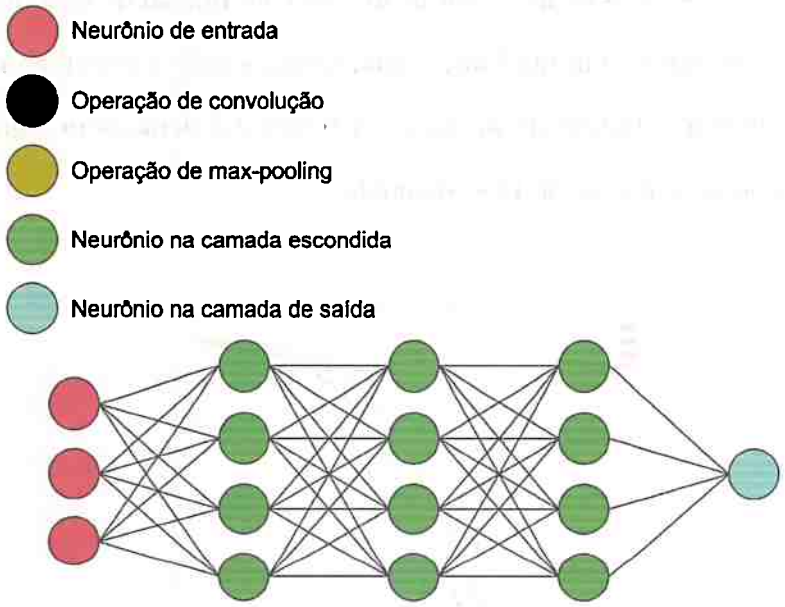


Figura 2.6: *Representação simplificada de um perceptron de múltiplas camadas*

Cada camada escondida do perceptron de múltiplas camadas é também chamado de camada densa. Uma camada densa conecta todos os neurônios da camada anterior com os da camada posterior. Quanto maior a quantidade de neurônios em cada camada densa e quanto mais camadas densas, maior o poder de representação dos dados do modelo. No entanto, se o modelo tiver uma quantidade excessiva de camadas ou neurônios este decora os dados ao invés de aprender e generalizar sobre eles, chamamos isto de má generalização do modelo ou super ajustamento (*overfitting*). Pode-se detectar *overfitting* quando o modelo apresenta um baixo E_{in} mas um alto E_{out}. Ou seja, este memorizou os dados de treinamento mas falha em generalizar para dados que estão fora da amostra usada no treinamento. Definimos

regularização como técnicas para mitigar o *overfitting*.

A experiencia E, ou seja, o treinamento deste modelo envolve encontrar valores de cada peso (w) que minimizem o erro de uma função de perda escolhida. Para encontrar estes valores é necessário ter um algoritmo que faça inicie o processo de treinamento com valores de pesos (w) que permitam que o modelo consiga alcançar um baixo nível de erro. Além disso, é necessário escolher um algoritmo de treinamento. Portanto, verificamos estes tópicos adiante.

2.2.3 Inicialização dos pesos

Antes do procedimento de treinamento, os valores da matriz de pesos do modelo, são inicializados de forma pseudoaleatória, dependendo dos valores, o treinamento pode não ser possível e não convergir a um valor aceitável e o modelo pode ficar parado em um minimo local. Em geral, quando uma rede neural falha, ou seja, fica preso em um minimo local ruim, pode-se começar o treinamento novamente, inicializando novos valores para a matriz de pesos. No entanto, este procedimento não permite a reprodutibilidade de um treinamento, pois ao mudar algum ponto da arquitetura da rede, não se poderá afirmar se o novo resultado foi afetado pela mudança na arquitetura ou pela inicialização dos pesos.
Devido a estes motivos, foi desenvolvido algoritmos de inicialização de pesos de forma a evitar que a rede neural falhe no treinamento. Além disso, muitos desses algoritmos permitem a inserção de um parâmetro *seed*, o qual, faz com que os valores inicializados pelo algoritmo seja o mesmo, portanto, permitindo a reprodutibilidade de execução.
Entre os algoritmos de inicialização, o algoritmo de *glorot* [GB10] inicializa os pesos com valores pseudoaleatórios em uma distribuição normal, centralizados em 0. Esta inicialização mantém os valores de uma maneira em que não são nem exageradamente grandes ou pequenos. O gradiente calculado pelo algoritmo de treinamento, quando os pesos tem valores muito altos ou muito baixos, fazem com que a rede neural falhe ou tenha um resultado ruim. Esta inicialização coloca valores em um alcance razoável.

Seu desvio padrão depende de quantas conexões de saída s e de quantas conexões de entrada e cada camada tem. De forma que, para cada camada da rede neural, o desvio padrão é calculado da seguinte forma :

$$desvio(w) = \frac{2}{e+s}$$

2.2.4 Treinamento

O treinamento de redes neurais envolve em se ter valores para a matriz de pesos que minimizem a função de custo. Sendo os valores de pesos números pertencentes ao conjunto dos números reais, mesmo para um modelo com poucos neurônios, há infinitas permutações possíveis. Portanto, é necessário um algoritmo que calcule valores para os pesos, de forma que, durante o treinamento, teremos cada vez mais, um menor erro de saída. Em geral, um modelo não terá como saída o valor correto de um conjunto de dados D antes do treinamento. Portanto, os pesos são ajustados para aproximar o modelo para as saídas corretas.
Os algoritmos mais eficientes para ajustar os pesos de um modelo de rede neural, utilizam o gradiente calculado pela derivada da função de erro, a saída esperada e a saída atual do modelo.

Gradiente Descendente

A derivada de uma função de custo permite calcular o gradiente para cada peso w, isto permite ajustá-los para um valor que minimiza o erro. O gradiente indica em como ajustar o valor atual de w para minimizar este erro. Se não há erro, o gradiente é 0. Se o sinal do gradiente é negativo, o valor de w deve ser aumentado e se o sinal é positivo, deve ser diminuído.
O algoritmo gradiente descendente permite ajustar todos os pesos w utilizando o gradiente calculado para cada um, utilizando os exemplos do conjunto de dados de treinamento D. A figura 2.7 ilustra de forma simples o algoritmo gradiente descendente.

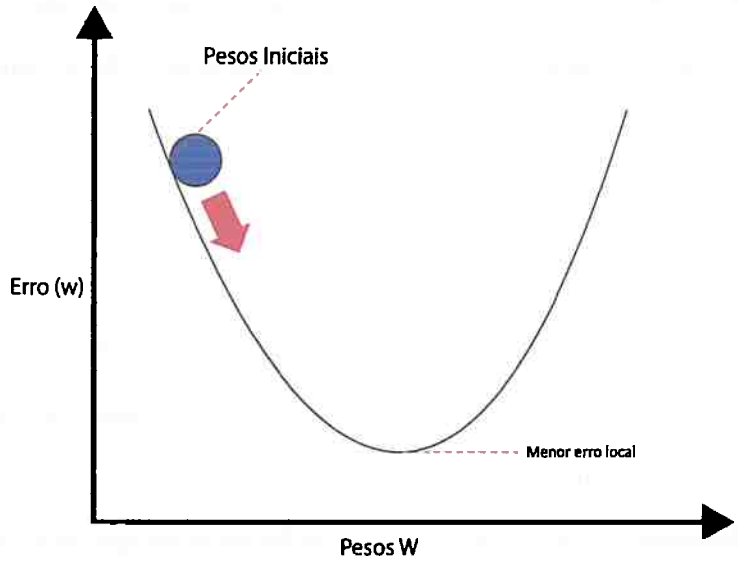


Figura 2.7: *Representação simplificada do gradiente descendente*

Algoritmos de retro propagação *backpropagation*

O algoritmo de retro propagação é utilizado com o calculo de gradiente de forma que a entrada, ou seja, o exemplo de treinamento de D é propagado por todas as camadas da rede neural, e a saída do modelo é utilizada para calcular o erro pela função de custo. Após isso, os valores dos erros são propagados de volta, começando pela camada de saída até a camada de entrada, com isto, os gradientes são calculados e os pesos ajustados. O objetivo é diminuir os gradientes para valores menores, desta forma, diminuindo o erro e aproximando os valores que saem do modelo, para os valores esperados de D.

Este algoritmo pode ser implementado em dois modos : online e em lote (*batch*). No modo online, os pesos são modificados para cada exemplo de treinamento visto pela rede neural. Ou seja, para cada exemplo, é aplicado o algoritmo de retro propagação e os pesos são ajustados.

No modo lote (*batch*), dado um conjunto de treinamento D com n exemplos, define-se um tamanho de lote b, tal que $b \leq n$. Os pesos são modificados para cada conjunto de exemplos de tamanho b : soma-se os gradientes de cada exemplo deste conjunto e só então se ajusta os pesos w.

Sendo ϵ a taxa de aprendizado, w cada peso da matriz de pesos da rede neural, α o

momento (*momentum*) e E uma função de custo e considerando os pesos e os gradientes calculados como um vetor unidimensional, a atualização dos pesos é feita conforme a equação abaixo :

$$\Delta_{w(t)} = -\epsilon \frac{\partial E}{\partial_{w(t)}} + \alpha \Delta_{w_{t-1}}$$

A taxa de treinamento aumenta ou diminui em escala, o gradiente. Uma taxa muito alta não permite que o modelo converja pra bons valores de pesos, e uma taxa baixa fara com que o modelo demore para adquirir bons valores de pesos.

O *momentum* determina a porcentagem da iteração anterior que deve ser aplicada na iteração. È um modo de permitir que o algoritmo de retro propagação, escape do minimo local, pois este tem como desvantagem, a tendencia de se manter em mínimos locais.

Esta equação calcula Δ, ou seja, a mudança para cada peso, como o produto do gradiente e a taxa de treinamento e adicionando o produto da mudança de peso da iteração anterior com o *momentum*.

Importante notar que a taxa de treinamento e o momento são hiper parâmetros, e em geral, são escolhidos de modo empírico, o que é considerado uma desvantagem deste tipo de algoritmo de treinamento.

Gradiente Descendente estocástico

O algoritmo de gradiente descendente estocástico é um tipo de procedimento de retro propagação, o qual também pode ser utilizado no modo online ou em lote, porém com algumas modificações. No modo online, escolhe-se os exemplos aleatoriamente para serem usados na retro propagação No modo de lote, o subconjunto, também chamado de mini lote (*mini batch*) é construído escolhendo-se elementos aleatórios do conjunto de treinamento, dado o tamanho deste conjunto. A diferença entre o gradiente descendente e o gradiente descendente estocástico é a escolha aleatória de exemplos. O modo lote não é implementado no algoritmo de gradiente descendente, apenas no gradiente descendente estocástico.
A principal vantagem é que no modo lote, ele é muito mais rápido do que o gradiente descendente. Apesar de, em geral, o GD apresentar um menor erro, pode-se aproximar

deste, aumentando o número de iterações do algoritmo, e ainda assim, tende a finalizar mais rapidamente.

Gradiente Descendente estocástico de Nesterov

Um algoritmo de treinamento que é muito utilizado em redes neurais é o gradiente descendente estocástico de Nesterov [Nes03], apresentado por Yurii Nesterov. Sua motivação, é devido a que o algoritmo de Gradiente Descendente estocástico pode produzir resultados erráticos, pois em uma iteração, uma escolha aleatória de exemplos pode fazer com que os valores de pesos melhorem muito a saída do modelo, porém, em uma próxima iteração, uma má escolha de exemplos pode arruinar esta melhora, ou seja, piorando muito os valores de pesos, fazendo com que a rede neural fique estagnada, varie muito sua taxa de erro, demore para convergir, e em alguns casos, fazer com que o treinamento falhe, devido aos exemplos e ao tamanho de lote escolhido.

A técnica de Nesterov, utiliza o momento (*momentum*) para mitigar este problema, pois permite o algoritmo a não seguir o gradiente cegamente e de forma brusca. O algoritmo segue na direção do gradiente acumulado, e após fazer uma medição neste gradiente, é possível corrigir a direção. Em outras palavras, o algoritmo de Nesterov é composto de duas partes: calcula-se o gradiente para a mudança de peso e então utiliza-se este valor para seguir em uma direção baseado na atualização dos pesos da última iteração.

A atualização dos pesos com gradiente descendente estocástico de Nesterov é feito da seguinte maneira:

Seja α_t o momento (*momentum*) em uma iteração t, inicializamos α_0 em 0 e em seguida calculamos α_t :

$$\alpha_0 = 0$$

$$\alpha_t = \frac{1 + \sqrt{1 + 4\alpha_{t-1}^2}}{2}$$

Aonde, ϵ é a taxa de aprendizado, w cada peso da matriz de pesos da rede neural e E uma função de custo e considerando os pesos e os gradientes calculados como um vetor

unidimensional. Após isto, calcula-se a atualização dos pesos pelo gradiente. Seja y_{t+1} os valores dos pesos antes da aplicação do momento (*momentum*), os valores finais dos pesos (w_{t+1}) são calculados desta maneira:

$$y_{t+1} = w_t - \epsilon \frac{\partial E}{\partial w_t}$$

$$w_{t+1} = (1 - \alpha_t) y_{t+1} + \alpha_t y_t$$

2.2.5 Ajuste de taxa de treinamento (*Learning Rate*)

Conforme o modelo vai aproximando os valores de pesos para valores ótimos, é interessante diminuir a taxa de aprendizado, ou seja, diminuir esta taxa conforme o treinamento progride. Esta ideia originou de Jacobs (1988) [Jac88], e atualmente é uma boa prática de treinamento de redes neurais. Se considerarmos que o modelo esta em uma colina, na qual, o ponto mais alto é onde o erro é maior e o ponto mais baixo ocorre o menor erro, o gradiente faz com que o modelo desça esta colina. Caso a taxa de aprendizado seja muito alta, o modelo ira "saltar" do ponto mais baixo, "passando direto". Portanto, seguindo a analogia, a ideia é que conforme o modelo for se aproximando da parte mais baixa da colina, diminui-se a taxa de aprendizado. Na figura 2.8, em vermelho há a representação de uma alta taxa de treinamento, e em azul, taxa de treinamento muito baixa.

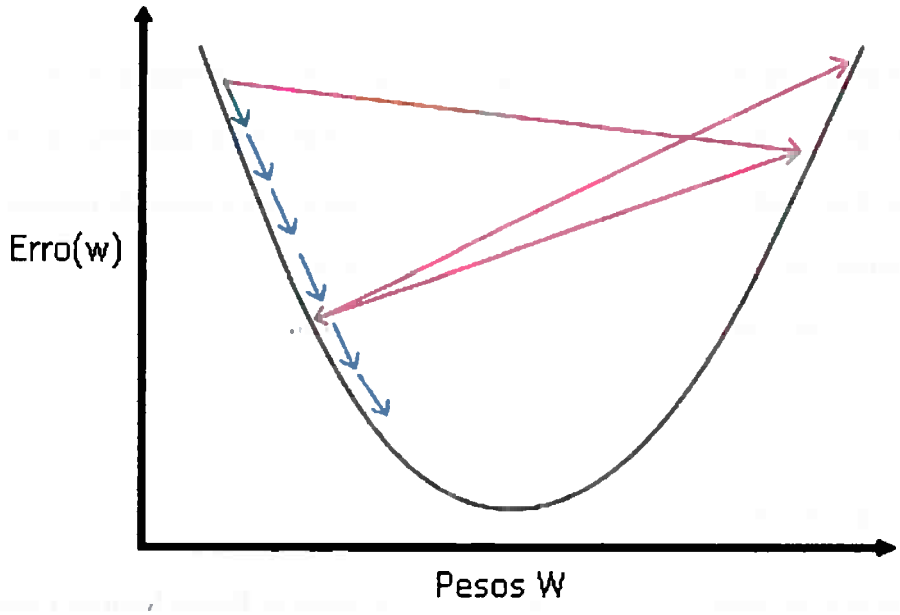


Figura 2.8: *Representação simplificada do ajuste da taxa de treinamento*

2.2.6 Regularização com early-stopping

Uma outra boa prática ao utilizar redes neurais é fazer uso do *early-stopping* [Pre98a] [Pre98b] [DBDJH14] [CLG01] como método de regularização. Para aplicar este método, é necessário dividir o conjunto de treinamento D em dois subconjuntos : treinamento e de validação. O conjunto de validação é usado para medir o erro durante o treinamento do modelo com o conjunto de treinamento. Quando o algoritmo de aprendizado é iniciado, é comum que o erro dos dois conjuntos diminuam progressivamente. Em algum momento, pode ocorrer de o erro do conjunto de validação estagnar e começar a aumentar enquanto o erro do conjunto de treinamento continua a diminuir. Nesta situação, é detectado o *overfitting*. Ou seja, o conjunto de validação é usado para verificar se o modelo esta generalizando ou memorizando dados.

Neste contexto, define-se *paciência* como a quantidade máxima de épocas em que não há melhora no erro do conjunto de validação. Quando a paciência chega ao limite, o algoritmo é finalizado e é utilizado o modelo na época de menor erro do conjunto de validação.

2.3 Aprendizado profundo (*Deep learning*)

Aprendizado profundo, *Deep Learning*, é uma maneira de treinar qualquer rede neural profunda. Em geral, uma rede neural para ser considerada profunda, tem de ter mais que duas camadas. Em 1943, Pitts [MP43], introduziu o modelo perceptron de múltiplas camadas *MLP*, desde então, podia-se criar redes neurais profundas.
Deep Learning é composto de técnicas inovativas, tais como :

- Uso de Unidades retificadoras lineares ($ReLU$)
- Uso de regularização com Dropout (*Dropout*)
- Convoluções em redes neurais, as quais são chamadas de Redes Neurais Convolucionais [LB95]

2.3.1 *Dropout*

Para diminuir o *overfitting*, os modelos de *deep learning* tendem a utilizar *dropout*. Esta técnica consiste em que, alguns neurônios e suas conexões nas camadas escondidas são ignorados e não tem seus pesos alterados ou usados, periodicamente, em algumas iterações no processo de treinamento. Foi apresentada em 2012 por Hinton, Srivastava, Krizhevsky, Sutskever e Salakhutdinov [HSK+12], e tem como principal característica não permitir a co-adaptação de neurônios. Dado 2 neurônios, quando um deles é removido temporariamente, a rede neural terá que ajustar o outro neurônio para compensar esta remoção.
Isto ajuda a mitigar a situação do modelo memorizar os dados de treinamento.
È comum que esta técnica seja implementada em forma de camada escondida. Dada uma camada escondida, pode-se ter uma camada de dropout em que cada neurônio se comporta como uma camada densa, no entanto, esta terá um hiper parâmetro τ, que é a probabilidade de um neurônio desta camada ser escolhido para ser removido temporariamente, junto com suas conexões.

2.3.2 Redes Neurais Convolucionais e Camadas Convolucionais

A rede neural convolucional foi desenvolvida tendo o olho biológico como inspiração e devido a seus resultados, é um modelo variante do perceptron de múltiplas camadas, muito importante na área de computação visual, pois este modelo tem um ótimo desempenho para reconhecimento e classificações de imagens. Foi apresentada pela primeira vez por Fukushima em 1980 [Fuk80].

Em 1998, este modelo de rede neural foi melhorado significantemente por LeCun, Bottou, Bengio e Haffner [LBBH01].

As redes neurais convolucionais utilizam camadas convolucionais em sua estrutura. Estas camadas tem como objetivo extrair características (*features*) do conjunto de dados de treinamento. Esta camada é composta de diversos hiper parâmetros, sendo estes :

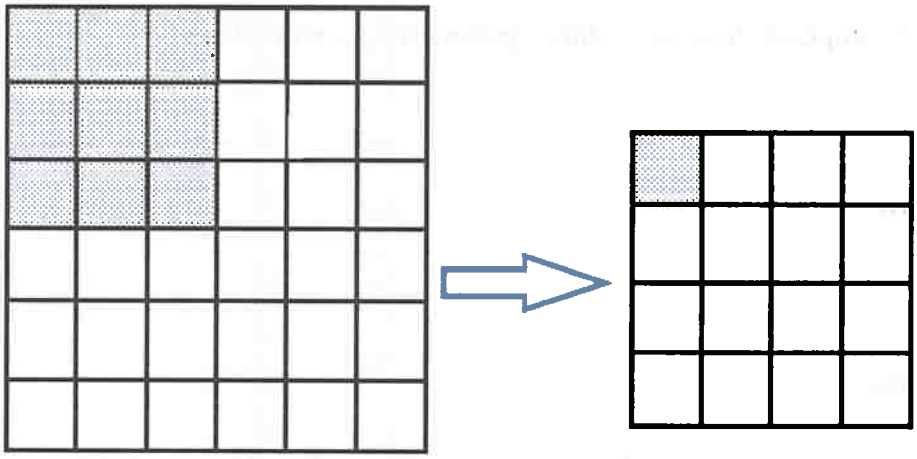
- Numero de filtros
- Dimensão dos filtros
- *Stride*
- Dimensão de *Padding*

As camadas convolucionais apresentam filtros que tem a forma de polígonos de 4 lados com ângulos retos, sendo mais comum, o formato de um quadrado. Cada filtro, também chamado de *kernel* pode ter seu tamanho configurado por um parâmetro de largura e altura. Sendo a imagem de entrada, uma matriz formada por pixeis, então cada filtro é uma matriz de tamanho menor que a imagem de entrada que transita por cada linha da imagem e calcula o produto escalar, e a matriz formada por este processo é chamada de mapa de características ou mapa de ativação. Portanto, definimos mapa de ativação como o resultado de aplicar o filtro em todas as sub-regiões da matriz de entrada, adicionando o termo bias e aplicando uma função de ativação.

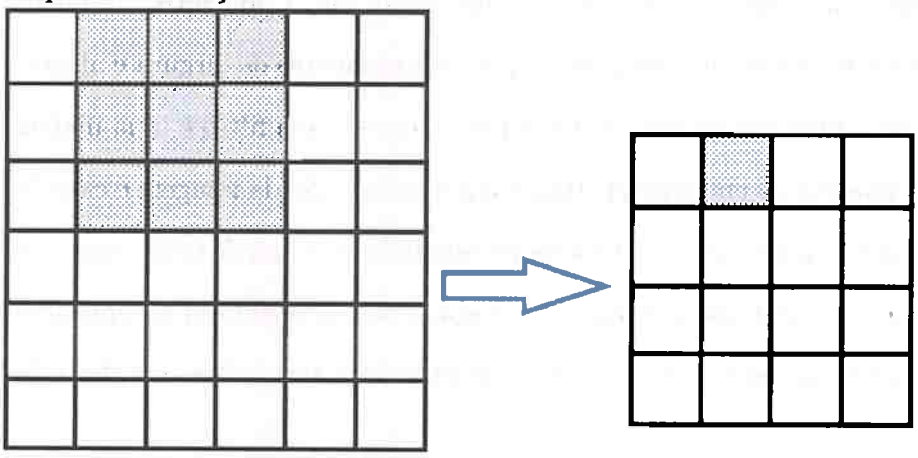
Passo (*stride*) é a quantidade de pixels que o filtro saltará a cada movimentação. Para um valor de *stride* igual a 1, o filtro ira se mover um pixel por vez para a direita e ao chegar na extremidade, um pixel para baixo. O *stride* não pode ter valor igual a 0, senão o filtro não ira transitar.

Preenchimento (*Padding*) é uma técnica de se adicionar valores 0's nas bordas da matriz de entrada da camada convolucional. Isto permite que o filtro não perca informação das bordas, pois a cada vez que a matriz passa por uma camada convolucional, sua dimensão diminui. O *Padding* permite mitigar este problema e aumentar a dimensão. Desta maneira, se mantém informação das bordas e permite ter uma rede com mais camadas convolucionais, ou seja, mais profunda.

Importante notar que o *stride* e tamanho do filtro não podem ser maior que a dimensão da matriz de entrada. A figura 2.9 mostra um exemplo de filtro de dimensão 3x3 e stride 1 em uma imagem 6x6. Note que na figura 2.9a há a primeira iteração do filtro e na figura 2.9b temos a segunda iteração.



(a) *Exemplo de filtro 3x3, passo* (stride) *= 1 e preenchimento* (padding) *= 1 em uma imagem 6x6 na primeira iteração*



(b) *Exemplo de filtro 3x3, passo* (stride) *= 1 e preenchimento* (padding) *= 1 em uma imagem 6x6 na segunda iterações*

Figura 2.9: *Exemplo de 2 iterações com filtro 3x3, stride = 1 e preenchimento* (padding) *= 1 em uma imagem 6x6*

Podemos calcular a dimensão resultante da matriz de saída I após passagem do filtro convolucional. Para cada d, tal que d é a dimensão de altura ou largura, e seja $\mathfrak{s}$ o stride e $\mathfrak{p}$ o preenchimento (*padding*), então :

$$I_d = \frac{I_d - f_d + 2\mathfrak{p}}{\mathfrak{s}} + 1$$

Portanto, note que na figura 2.9a, podemos calcular qual será a dimensão resultante por esta fórmula. Abaixo, temos um exemplo do uso desta fórmula na figura 2.9a.

Para a dimensão de altura :

$$I_d = \frac{I_d - f_d + 2\mathfrak{p}}{\mathfrak{s}} + 1 \tag{2.1}$$

$$I_d = \frac{6 - 3 + 2 * 0}{1} + 1 \tag{2.2}$$

$$I_d = 4 \tag{2.3}$$

Note que, para a dimensão de largura obtemos o mesmo valor devido aos valores utilizados e conforme ilustrado na figura 2.9a.

2.3.3 Camada de Max-Pooling

A camada de *max-pooling* permite diminui a dimensão da matriz que passa por esta camada, pois sua função é dividi-la em submatrizes de tamanhos menores e manter apenas o maior valor presente nesta submatriz Ou seja, a matriz resultante terá os valores maiores das submatrizes e portanto, a matriz resultante sera composta de submatrizes com um único valor.

Apresenta 2 hiper parâmetros, a extensão espacial $\mathfrak{t}$ e *stride* $\mathfrak{s}$. A extensão espacial define a dimensão das submatrizes, portanto, para $\mathfrak{t} = 2$ e $\mathfrak{s} = 2$, e uma matriz de entrada de tamanho $4x4$, a matriz resultante desta camada terá dimensão $2x2$.

A dimensão da matriz que resulta desta camada é calculada da seguinte maneira: Dado a matriz de entrada que irá passar por esta camada e para cada d, tal que d é a dimensão de altura ou largura, então sua dimensão de saída I_d é tal :

$$I_d = \frac{I_d - \mathtt{t}}{\mathtt{s} + 1}$$

Importante notar, que caso a matriz de entrada seja tridimensional, sendo a terceira dimensão a quantidade de canais de cor, esta dimensão permanecerá e não sera removida. Em geral, esta camada é utilizada após uma camada convolucional, porém, há um consenso na comunidade em se usar uma camada *max-pooling* a cada 2 ou 3 camadas de convolução [NH10, LBBH98, LBBH98, SZ14].

Esta camada não tem pesos e, portanto, não sofre mudanças pelo algoritmo de treinamento. Além disso, promove um tipo de invariância a translação e reduz o custo computacional para as outras camadas. A figura 2.10 ilustra o uso do max-pooling com $\mathtt{t} = 2$ e $\mathtt{s} = 2$.

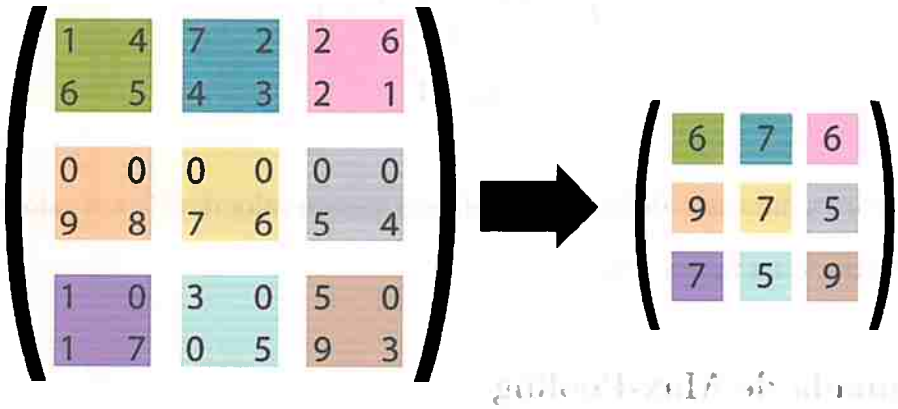


Figura 2.10: *Exemplo de max-pooling com* $\mathtt{t} = 2$ *e* $\mathtt{s} = 2$

A figura 2.11 ilustra um exemplo de modelo de rede convolucional.

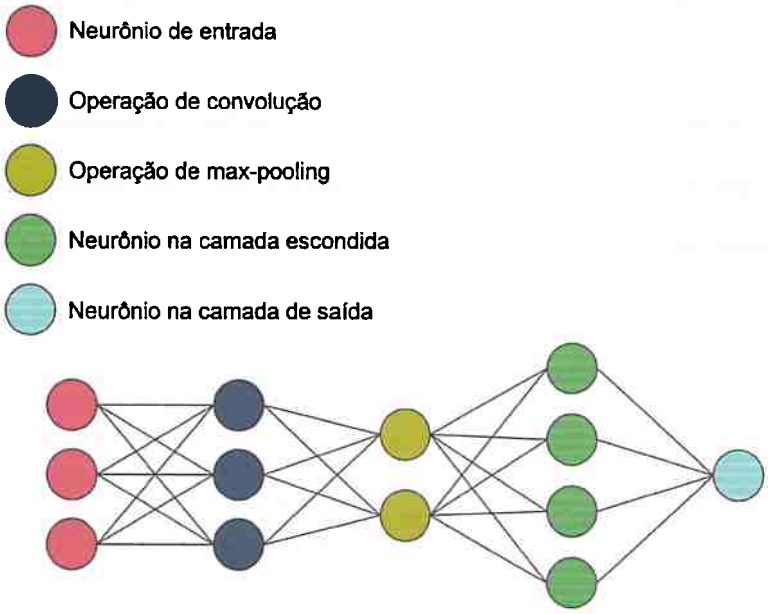


Figura 2.11: *Representação simplificada de um perceptron de múltiplas camadas*

2.4 Outras Medidas para o conjunto de teste

Para auxiliar na avaliação dos resultados é possível calcular a acurácia, a sensitividade, a especificidade, Valor Preditivo Positivo *(PPV)* e F-score *(F1)* do conjunto de testes. A sensitividade e especificidade são medidas usadas para avaliar o desempenho de uma classificação binária.

Seja TP *(True Positive)* a quantidade de positivos verdadeiros, FP *(False Positive)* a quantidade de falsos positivos, FN *(False Negative)* a quantidade de falsos negativos e TN *(True Negative)* a quantidade de negativos verdadeiros, podemos calcular a sensitividade, especificidade e precisão (PPV). Ou seja, se definimos 'positivo' como 'aceito' e 'negativo' como 'rejeitado', podemos interpretar da seguinte maneira :

- TP como corretamente aceito.
- FP como incorretamente aceito.
- TN como corretamente rejeitado.
- FN como incorretamente rejeitado.

A sensitividade, também chamada de probabilidade de detecção, permite medir a proporção de positivos corretamente detectados. A especificidade permite medir a proporção de negativos corretamente detectados. O F-score *(F1)* combina a sensitividade e a precisão (PPV), permitindo que se verifique a robustez do modelo que esta sendo analisado.

Definimos matematicamente :

$$Sensitividade = \frac{TP}{TP + FN}$$

$$Especificidade = \frac{TN}{TN + FP}$$

$$PPV = \frac{TP}{TP + FP}$$

$$F_{score} = \frac{PPV * Sensitividade}{PPV + Sensitividade}$$

Para ilustrar, suponha que queremos usar estas medidas em exames para verificar doença em indivíduos. Neste exemplo, dizemos que :

- TP ou positivo verdadeiro, significa que um indivíduo doente foi identificada corretamente como doente.
- TN ou negativo verdadeiro, significa que um indivíduo saudável foi identificada corretamente como saudável.
- FP ou falso positivo, significa que um indivíduo saudável foi identificada incorretamente como doente.
- FN ou falso negativo, significa que um indivíduo doente foi identificada incorretamente como saudável.
- Sensibilidade significa a proporção de indivíduos positivos verdadeiros(TP).
- Especificidade significa a proporção de indivíduos negativos verdadeiros(TN).

Capítulo 3

Projeto de operadores locais de imagem usando CNN

No capítulo 2 discutimos os conceitos necessários para compreender o uso de redes neurais artificiais para estimar uma função a a partir de exemplos de treinamento. Neste capítulo vamos discutir sobre os operadores de imagens locais e o uso de redes convolucionais para estimá-los.

Um operador de imagem Ψ é uma transformação entre dois espaços de imagens e será denotado normalmente por uma letra grega Ψ maiúscula. Em particular, neste trabalho, estamos interessados em operadores no domínio das imagens em níveis de cinza, denotada por $[0, K]^E$ (imagens em qual k = 255) e tendo como contra-domínio as imagens binárias $[0, 1]^E$.

Seja g uma imagem em níveis de cinza, g $\in [0, K]^E$, e h uma imagem binária, h $\in [0, 1]^E$ então Ψ leva g em h através da relação $\Psi(g) = h$. A figura 3.1 apresenta um exemplo desta transformação.

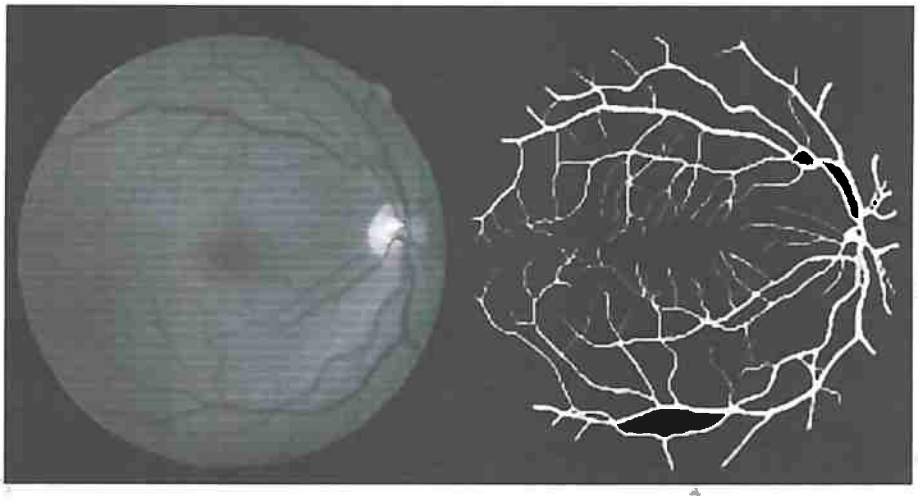


Figura 3.1: *Exemplo de imagem de entrada (I) e de saída (ideal) (O).*

Os operadores Ψ que são invariantes por translação e localmente definidos são conhecidos como W-operadores, ou "operadores de janela", onde W é a "janela", a qual define a saída do operador. Um operador $\tau \in E$, é invariante à uma translação se possue a seguinte propriedade:

$$\Psi(g + \tau) = \Psi(g) + \tau$$

Em outras palavras, a translação de uma imagem por um deslocamento τ no domínio das imagens não afeta o resultado do operador, a menos da translação. Uma outra classe importante de operadores de imagens é a dos operadores locais, isto é, operadores que dada uma vizinhança $W \subset E$, $o \in W$ (sendo o a origem de E), possuem a seguinte propriedade:

$$\Psi(g)(\tau) = \Psi(g|W_\tau)(\tau)$$

Os operadores de janela definidos no domínio $[0, K]^E$ com contra-domínio $[0, 1]^E$, podem ser representados por função $\psi : [0, K]^W \to [0, 1]$.

A aplicação de um operador de janela numa imagem $g \in [0, K]^E$ faz se transladando-se W sobre todos os possíveis pontos de E para todos os pontos p de E, observando-se $g|W_p$ e aplica-se a função ψ sobre essa subimagem. O resultado dessa aplicação é o valor de $\psi(g)(p)$

A figura 3.2 ilustra uma janela sobre uma posição em uma imagem.

Figura 3.2: *Exemplo de uma janela (W) em I e o padrão observado.*

O operador gradiente morfológico [Dou92a], [Hei11], com elemento estruturante cruz, assim como na figura 3.3, é um exemplo de operador de janela pois é localmente definido pelo elemento estruturante cruz. A figura 3.4 mostra a operação de segmentação de borda com este operador.

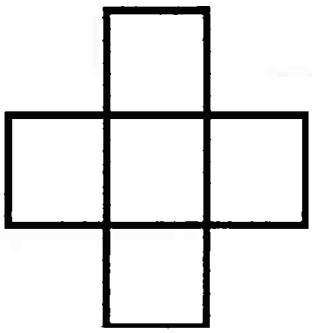


Figura 3.3: *Elemento estruturante cruz*

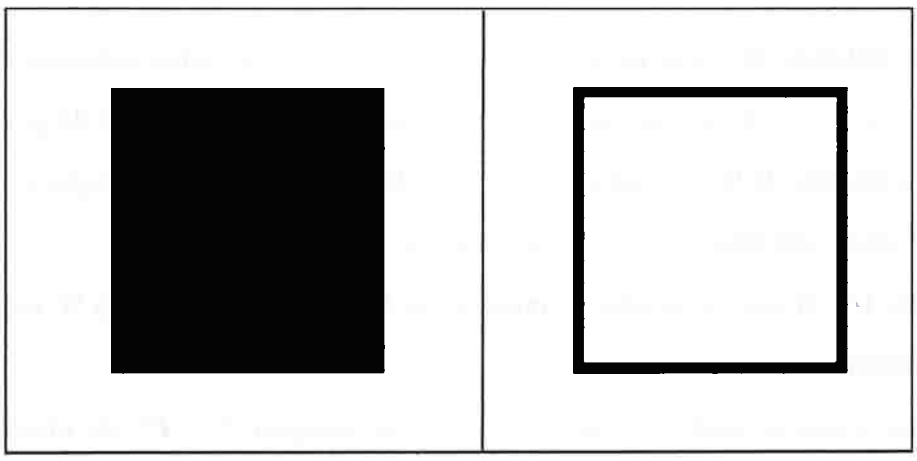


Figura 3.4: *Exemplo do operador gradiente sobre uma imagem fazendo uma segmentação de borda*

3.1 Processo de aprendizagem de operadores

Na seção anterior, vimos que os operadores de janela podem ser representados por funções de imagens locais $\psi : [0, 255]^W$ em $\{0, 1\}$. Nesta seção e na próxima vamos recordar o processo

de estimação dessas funções como proposto em trabalhos anteriores do grupo e apresentar nossa proposta de estimação das funções usando redes neurais convolucionais.

No trabalho de Barrera et al [BDT97] é apresentado o projeto de W-operadores de imagens através de estimação usando aprendizado de maquina. A figura 3.5 ilustra a ideia.

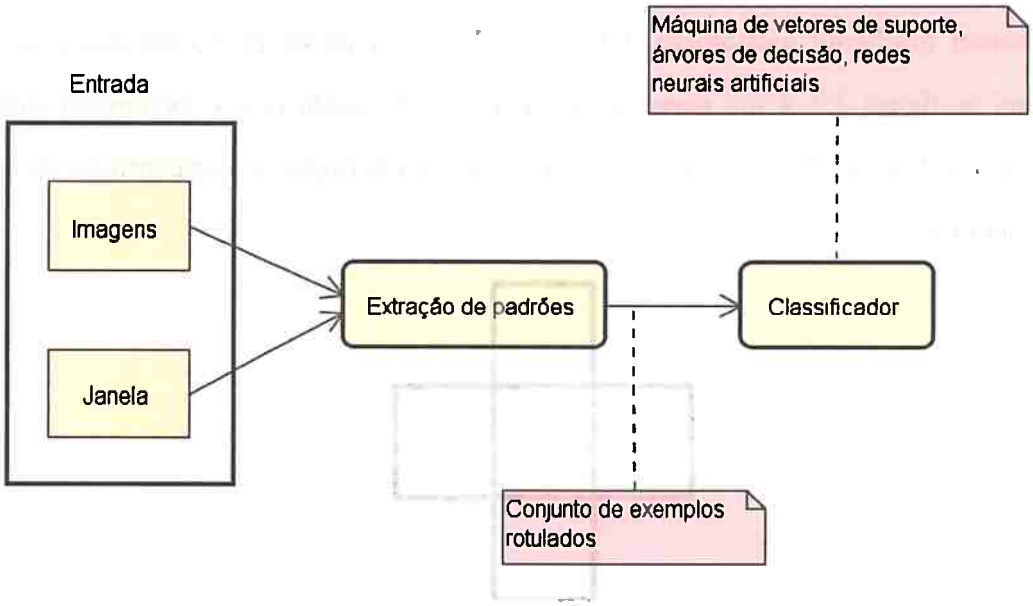


Figura 3.5: *Processo de aprendizagem de Ψ Operadores*

Dado um conjunto de imagens de entrada $\{\mathcal{G}_i\}, i \in 1, ..., n$, outro conjunto de imagens de saída $\{\mathcal{H}_i\}, i \in 1, ..., n$ tal que cada $\{h_i\}$ corresponde a transformação de g_i por um W-operador Ψ, o projeto de W-operadores é um método para estimar uma função $\hat{\psi}$ que melhor aproxima a função que representa o W-operador ψ.

As famílias $\mathcal{G}$ e $\mathcal{H}$ são chamadas de imagens de treinamento e a janela W simplesmente de janela na figura 3.5.

Para obter todas os padrões possíveis para uma imagem $f \in \mathcal{L}^E$ de níveis de cinza, fazemos a translação com a janela $W|W \subset E$, sobre f e observamos cada padrão visto por esta janela. Ou seja, restringimos f a W no ponto $\tau|\tau \in E$, e variando τ para todos os pontos de E, obtendo todas os padrões.A quantidade de operadores do subconjunto formado pelos W-operadores é $|\mathcal{L}|^{(|\mathcal{L}|^{|W|})}$, pois para cada padrão de janela pode haver $|\mathcal{L}|$ valores de saída para o operador, sendo $|\mathcal{L}|$ os níveis de cinza possíveis.

Com as imagens e a janela é feito a extração de padrões e de seus rótulos. Cada padrão é

a vizinhança definida pela janela W, ou seja, os pixeis na vizinhança de um pixel p, aonde W é centralizado. A extração de padrões é feita da seguinte maneira : a janela W se movimenta pixel a pixel sobre uma imagem I, de forma que, W na posição (i, j) de I extrai pixeis da vizinhança, tal que, a vizinhança é definida pela dimensão de W e o rótulo é extraído de O na mesma posição (i, j).

A definição do tamanho da janela W é feita de forma manual através de tentativa e erro. Desta maneira, há de se verificar situações em que a janela é excessivamente grande e quando é extremamente pequena. Uma janela excessivamente grande causa um aumento no tempo de processamento e diminui a acurácia do operador devido a que a janela começa a ter informações pouco relevantes [Jr01]. Usar janelas muito pequenas impedirá o operador de imagem Ψ de observar diversos padrões que são importantes para efetuar sua transformação $f(I_i) = O_i$, assim, resultando em um conjunto de treinamento insuficiente, com pouca variação de padrões [Jr01].

Para ilustrar este procedimento, na tabela 3.1 há exemplos de padrões vistos por uma cruz, assim como na figura 3.3, de dimensão 3x3 sobre a imagem e rótulo presente na figura 3.4.

Tabela 3.1: *Padrões extraídos para segmentação de borda.*

Padrões Extraídos			
Índice	Padrão	Rótulo	
		0	255
0	255, 0, 0, 0, 0	237	0
1	0, 255, 0, 0, 0	249	0
2	0, 0, 0, 0, 0	2880	56133
3	255, 255, 255, 255, 0	0	239
4	0, 0, 0, 255, 255	1	0
5	255, 255, 255, 0, 255	0	251
6	0, 255, 255, 255, 255	0	239
7	0, 0, 0, 255, 0	249	0
8	255, 255, 255, 255, 255	0	996
9	255, 0, 0, 255, 0	1	0
10	0, 0, 0, 0, 255	237	0
11	0, 255, 0, 0, 255	1	0
12	255, 0, 255, 255, 255	0	251
13	255, 255, 0, 0, 0	1	0

Na tabela 3.1 vimos que pode haver padrões repetidos com rótulos iguais e/ou diferentes, assim como no índice 0 desta tabela. Quando isto ocorre, há um problema de decisão, o qual é visto como parte do processo na figura 3.5. Portanto, uma solução possível, é remover os exemplos repetidos e manter apenas um exemplo com o rótulo mais frequente. Para exemplificar, podemos realizar este processo na tabela 3.1, o resultado esta na tabela 3.2. Outra solução possível, é manter os exemplos repetidos, inclusive com rótulos divergentes e deixar que o classificador faça inferência sobre eles.

Tabela 3.2: *Padrões extraídos para segmentação de borda*

Padrões Extraídos			
Índice	Padrão	Rótulo	
		0	255
0	255, 0, 0, 0, 0	1	0
1	0, 255, 0, 0, 0	1	0
2	0, 0, 0, 0, 0	0	1
3	255, 255, 255, 255, 0	0	1
4	0, 0, 0, 255, 255	1	0
5	255, 255, 255, 0, 255	0	1
6	0, 255, 255, 255, 255	0	1
7	0, 0, 0, 255, 0	1	0
8	255, 255, 255, 255, 255	0	1
9	255, 0, 0, 255, 0	1	0
10	0, 0, 0, 0, 255	1	0
11	0, 255, 0, 0, 255	1	0
12	255, 0, 255, 255, 255	0	1
13	255, 255, 0, 0, 0	1	0

Com o problema de decisão tratado, os padrões, com seus respectivos rótulos, são enviados para um classificador, e este será o responsável pelo aprendizado do operador de imagem. Cada padrão e rótulo respectivos formam um conjunto de treinamento definitivo.

3.2 W-operadores e Redes Convolucionais

O classificador utilizado no processo de aprendizagem do W-operador neste trabalho é o modelo de rede neural convolucional. A figura 3.6 ilustra este processo.

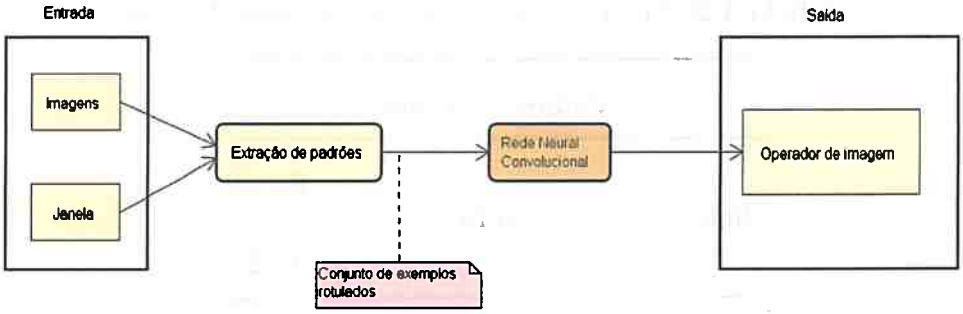


Figura 3.6: *Processo de aprendizagem de Operadores usando Rede neural convolucional*

Cada janela rotulada é passada para a camada de entrada da rede neural convolucional e esta é responsável por enviar a imagem para a primeira camada convolucional, a qual terá seus filtros e outros parâmetros que irão fazer a convolução. A figura 3.7 ilustra este procedimento.

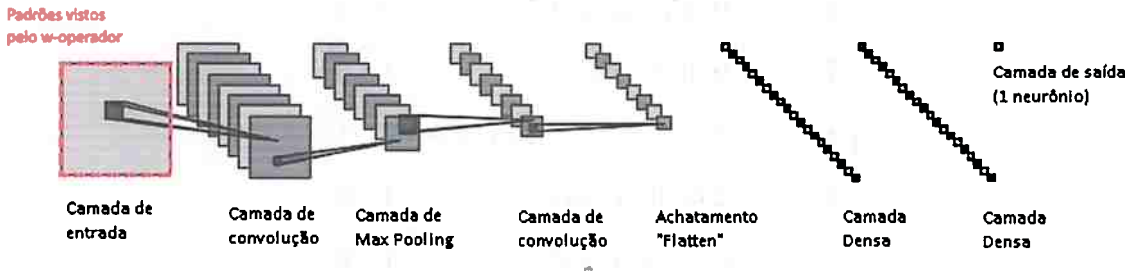


Figura 3.7: *Ilustração de uma Rede neural convolucional. Imagem feita com referencia da imagem presente em github.com/gwding/draw_convnet*

Note que, em geral, nos trabalhos científicos que usam este modelo convolucional, utiliza-se uma imagem completa como entrada para o modelo, assim como ilustrado na figura 3.8.

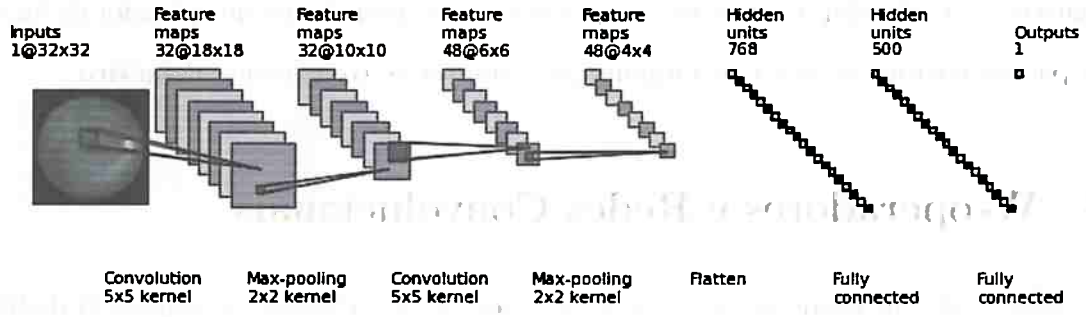


Figura 3.8: *Ilustração de uma Rede neural convolucional. Imagem feita com referencia da imagem presente em github.com/gwding/draw_convnet*

Neste trabalho, no entanto, utilizamos os padrões vistos pelo W-Operador como entrada

e estas são partes de imagem. Isto é uma diferença do nosso trabalho para outros trabalhos publicados. Além disso, nesta abordagem é possível ter como saída do modelo apenas um neurônio sigmoide.

3.3 Escolha da arquitetura

A escolha da arquitetura de uma rede neural convolucional para ser aplicada com W-operadores, necessita de atenção na quantidade de camadas convolucionais e de max-pooling.

Em geral, a dimensão das janelas do W-operador são relativamente pequenas, portanto, não faz sentido ter uma rede convolucional com uma quantidade excessiva de camadas, pois estas não irão conseguir extrair características suficiente dos dados devido a dimensão insuficiente da entrada. Além disso, há o problema da diminuição de dimensionalidade dos dados a cada passagem por uma camada convolucional e de max-pooling, a qual pode ser mitigada por uso de padding, no entanto, o seu uso excessivo por muitas camadas, irá prejudicar a quantidade de informação que as camadas convolucionais irão conseguir extrair.

Para exemplificar, suponha a escolha de uma janela 21x21, a qual é considerada razoavelmente grande para um W-operador. Neste primeiro exemplo, verificamos o que acontece com a dimensão da imagem de entrada ao passar por 2 camadas convolucionais e em seguida por uma camada de max-pooling, 2 vezes. Note as dimensões na tabela 3.3.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x21x21				
Convolucional	20x23x23		3x3	2x2	Retificadora linear
Convolucional	20x25x25		3x3	2x2	Retificadora linear
Max-pooling	20x12x12	2x2		0x0	
Convolucional	40x14x14		3x3	2x2	Retificadora linear
Convolucional	40x16x16		3x3	2x2	Retificadora linear
Max-pooling	40x8x8	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

Tabela 3.3: *Exemplo de arquitetura de rede neural convolucional com janela 21x21*

Note que na última camada de max-pooling, a dimensão já chega a 8x8, o que é considerado pequeno para que uma camada convolucional consiga extrair características de forma eficiente. Portanto, adicionar mais camadas, com esta dimensão de janela, não é interessante. Vejamos o que acontece quando utiliza-se uma janela mais comum ao se trabalhar com W-operadores, por exemplo, com dimensão 11x11, assim como descrito na tabela 3.4

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x11x11				
Convolucional	20x13x13		3x3	2x2	Retificadora linear
Convolucional	20x15x15		3x3	2x2	Retificadora linear
Max-pooling	20x7x7	2x2		0x0	
Convolucional	40x9x9		3x3	2x2	Retificadora linear
Convolucional	40x11x11		3x3	2x2	Retificadora linear
Max-pooling	40x5x5	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

Tabela 3.4: *Exemplo de arquitetura de rede neural convolucional com janela 11x11*

Verifique na tabela 3.4 que usando uma janela 11x11, a dimensão resultante na ultima camada de max-pooling é ainda menor. Utilizar muitos filtros nas camadas convolucionais, neste caso, é ineficiente, pois ocorrerá de alguns filtros terem pouca influência no processo de inferência. Em outras palavras, o uso de muitos filtros quando há uma dimensão pequena de imagem pode fazer com que muitos filtros não aprendam algo sobre os dados.

Ao realizar experiências cientificas no âmbito acadêmico, sabemos que há a necessidade de controlar a quantidade de parâmetros que irá ser mudado entre cada experimento. Além disso, é necessário ter uma arquitetura que irá ser íntegra e eficiente para diferentes tipos de dimensão de janelas de W-operadores escolhidos e que não irá ter sua dimensão reduzida a 0 ou 1 em alguma camada convolucional ou max-pooling, assim como ilustrado na tabela 3.5.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x11x11				
Convolucional	20x11x11		5x5	2x2	Retificadora linear
Convolucional	20x11x11		5x5	2x2	Retificadora linear
Max-pooling	20x5x5	2x2		0x0	
Convolucional	40x5x5		5x5	2x2	Retificadora linear
Convolucional	40x5x5		5x5	2x2	Retificadora linear
Max-pooling	40x2x2	2x2		0x0	
Convolucional	40x2x2		5x5	2x2	Retificadora linear
Convolucional	40x2x2		5x5	2x2	Retificadora linear
Max-pooling	40x1x1	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

Tabela 3.5: *Exemplo de problema na arquitetura de rede neural convolucional com janela 11x11, note a camada em tom vermelho representando a situação descrita.*

Note também que estes exemplos, apesar de terem sido focados na dimensão das janelas dos W-operadores, servem como informação ao criar uma arquitetura de rede convolucional para outros tipos de problemas que não envolvam W-operadores. Atualmente não há uma forma única de criar estes modelos e portanto, testa-los e adapta-los, mudando a quantidade de camadas, filtros convolucionais e neurônios, ainda é uma tarefa importante ao se trabalhar com eles.

Outro ponto importante é a questão entre criar uma arquitetura própria (customizada) ou utilizar um modelo pronto, por exemplo, a VGGnet [SZ14], AlexNet [NH10, LBBH98], ResNet[HZRS16] e GoogleNet [SLJ+15], as quais têm milhões de parâmetros que exigem alta capacidade de computação e memória. Estes modelos não podem ser utilizados em casos em que não há a possibilidade de se ter um processador e GPU de alto desempenho [HRVS16]. Além disso, em muitos casos, não há uma grande quantidade de dados disponível, como por exemplo, a que os autores destes modelos utilizaram para treinar estes modelos e portanto,

uma rede de menor profundidade é desejável. È possível criar redes de menos profundidades com acurácia próxima a modelos extremamente profundos [HRVS16], usando algoritmos de regularização e normalização ou até mesmo fazendo o aumento do conjunto de dados de forma artificial, por exemplo.

De qualquer forma, há uma troca entre acurácia e processamento, sendo que, quando é necessário uma alta capacidade de computação, o tempo de inferência e treinamento é altamente afetado [CPC16], tornando inviável sua aplicação direta. Desta maneira, quando há a possibilidade de se obter resultados razoáveis com redes não tão profundas, estas devem ser prioridade devido ao menor custo computacional envolvido, e um exemplo pode ser verificado no trabalho de Yoon Kim [Kim14] aonde uma rede convolucional simples é utilizada para classificar frases e consegue resultados excelentes.

Capítulo 4

Segmentação de veias da retina

Para testar a abordagem proposta, aplicamos o método para 2 conjuntos de imagens de fundo de olho. Um dos conjuntos é chamado DRIVE [SAN+04b] e o outro de STARE [Hoo75].

A segmentação de veias é um processo difícil de ser feito, inclusive para humanos. Mudanças nas características das veias de uma retina, podem indicar problemas de saúde, como por exemplo, hipertensão, diabetes, doenças cardiovasculares e arteriosclerose. A segmentação de veias é o primeiro passo para uma análise médica, e portanto, sua automatização pode diminuir o custo e o tempo gasto com a segmentação manual e suas inconsistências.

O conjunto de dados DRIVE é formado por 40 imagens de retina, sendo que 33 não demonstram nenhuma característica de retinopatia diabética [SAN+04b] porém, 7 mostram sinais desta doença, no estado inicial. Cada imagem tem dimensão 768 por 584 pixeis e foram adquiridas em RBG. O conjunto foi dividido em 20 imagens de treinamento e 20 imagens de teste.

O conjunto de dados STARE é formado por 20 imagens de retina, sendo que 10 imagens não demonstram nenhuma característica de retinopatias porém, 10 mostram sinais de doença. Cada imagem tem dimensão 700 por 605 pixeis e foram adquiridas em RBG. Todas as imagens foram segmentadas manualmente para formar o ground-truth por 2 observadores. O observador utilizado nos nossos experimentos, Adam Hoover [Hoo75], priorizou a segmentação de veias com maior dimensão.

A figura 4.1 apresenta algumas imagens de treinamento do conjunto de dados DRIVE,

para cada linha a primeira imagem é a original adquirida em RGB, a segunda é a imagem convertida para tons de cinza, a terceira imagem é uma imagem "máscara" que delimita a área de interesse da imagem para o problema. Finalmente, a quarta imagem contém a vasculatura segmentada manualmente.

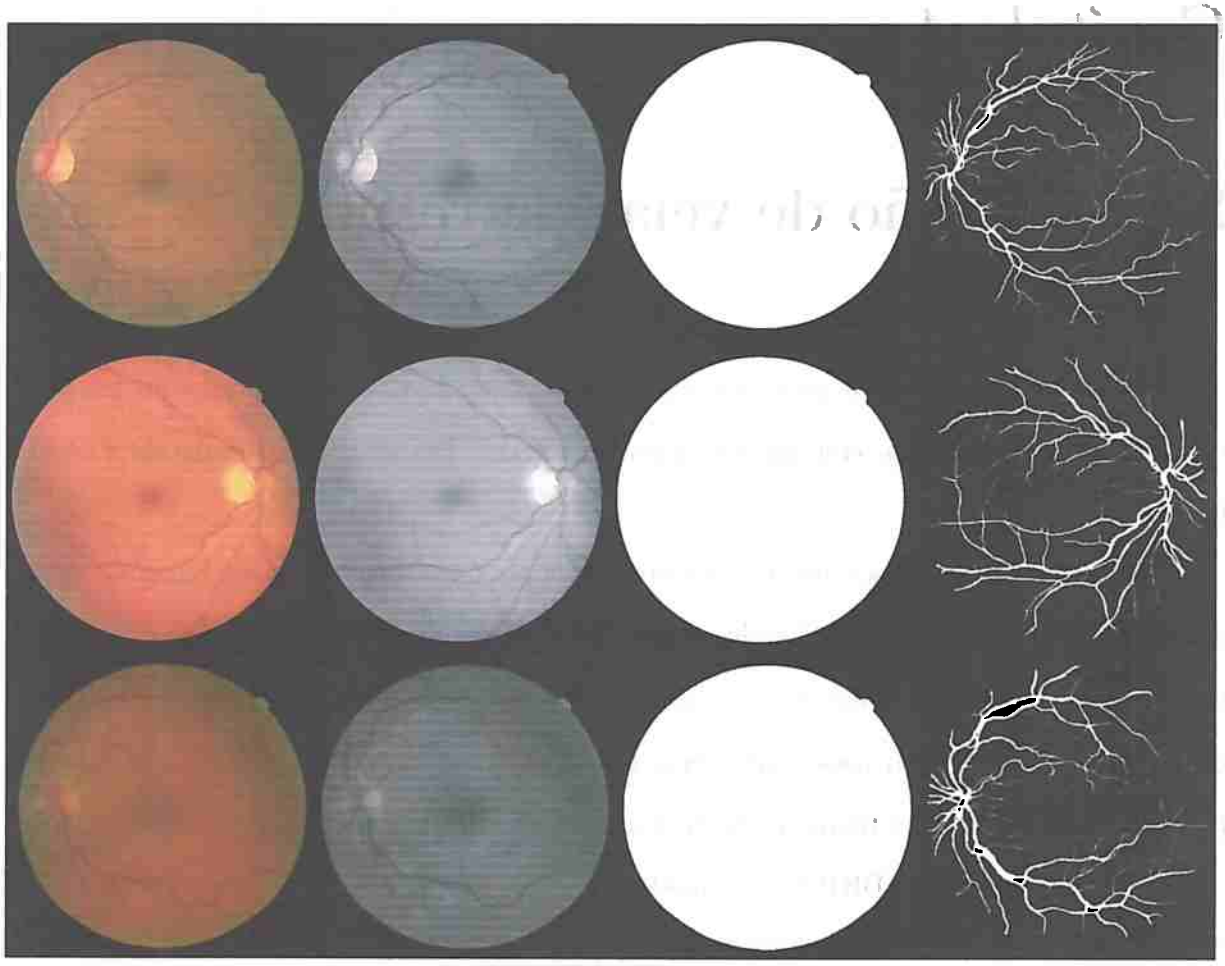


Figura 4.1: *Primeiras 3 imagens do conjunto de treinamento do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscara respectiva e a quarta coluna contém a vasculatura segmentada*

A figura 4.2 apresenta algumas imagens de treinamento do conjunto de dados STARE, para cada linha a primeira imagem é a original adquirida em RGB, a segunda é a imagem convertida para tons de cinza, a terceira imagem é uma imagem "máscara"que delimita a área de interesse da imagem para o problema. Finalmente, a quarta imagem contém a vasculatura segmentada manualmente.

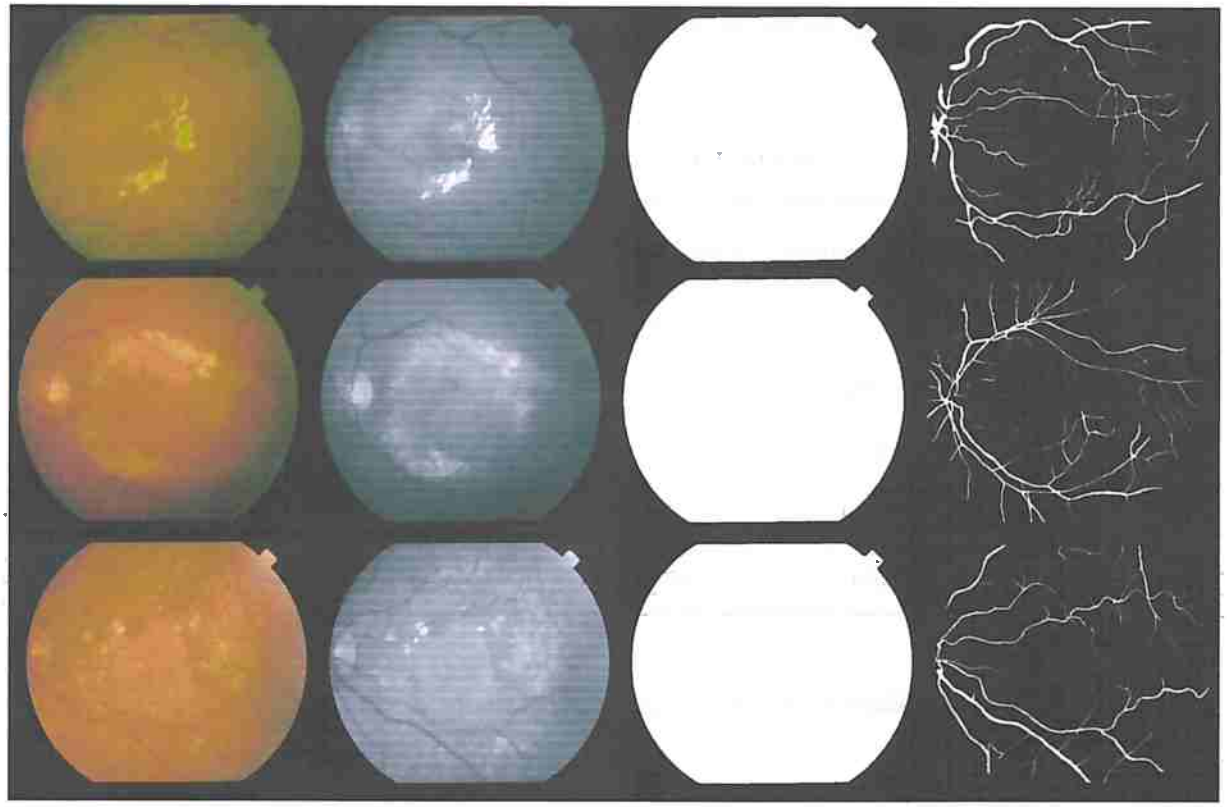


Figura 4.2: *Primeiras 3 imagens do conjunto de treinamento do banco de dados STARE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscara respectiva e a quarta coluna contém a vasculatura segmentada*

4.1 Experimentos

O modelo utilizado nos W-operadores testados é uma rede neural convolucional com saída sigmoide, pois estamos interessados apenas na segmentação da imagem em duas partições: veias e fundo. Este modelo de rede convolucional foi apresentado no capítulo 2

Em todos os experimentos, os parâmetros fixados, foram :

- Tamanho do lote *(mini batch)* : 6000
- Número de épocas : 1000
- Paciência : 100
- Conjunto de validação é formado por 30% dos dados do conjunto de treinamento.
- Inicialização dos pesos do modelo usando algoritmo Glorot [GB10].

As taxas de aprendizado foram alteradas de acordo com o passo das épocas, de acordo com a tabela 4.1.

Tabela 4.1: *Ajuste de aprendizado*

Época inicial	Época final	Taxa de aprendizado
1	49	0.15
50	99	0.10
100	299	0.01
300	799	0.001
800	1000	0.0001

Para facilitar a visualização dos resultados do operador de imagem, estas foram colorizadas da seguinte maneira:

- Tom azul = Positivo verdadeiro (TP)
- Tom laranja = Falso positivo (FP)
- Tom vermelho = Falso negativo (FN)
- Tom verde = Negativo verdadeiro (TN)

A figura 4.3 apresenta um exemplo de um resultado de uma experiencia com esta colorização descrita.

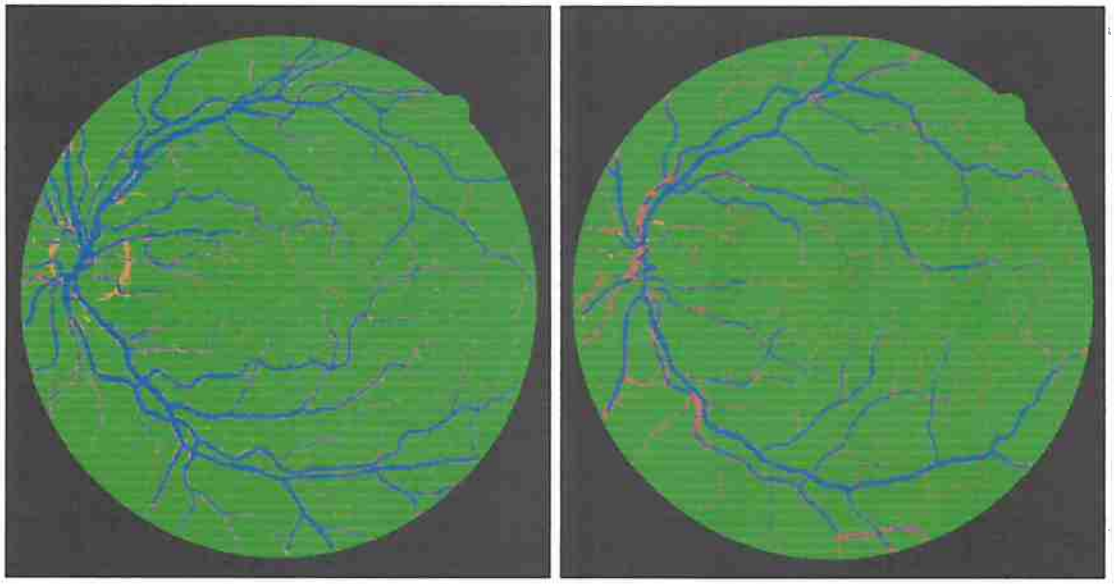


Figura 4.3: *Duas imagens resultantes de um operador de imagem, colorizadas para ajudar na visualização*

Nos experimentos 1, 2 e 3, utilizando apenas o banco de dados DRIVE, fixamos a janela do W-operador em $11x11$ e testamos uma estratégia de aumentar os exemplos vistos pelo modelo, rotacionando em 90 graus no experimento 2, e em 45 graus no experimento 3.

No experimento 4, utilizando apenas o banco de dados DRIVE, fixamos o número de exemplos e variamos a dimensão da janela em $5x5$, $7x7$, $21x21$, $31x31$.

No experimento 5, utilizando apenas o banco de dados DRIVE, fixamos a dimensão da janela em $11x11$ e variamos a quantidade de imagens utilizadas para o treinamento do modelo.

No experimento 6, utilizando apenas o banco de dados DRIVE, utilizamos uma estratégia de pré-processamento do conjunto de dados, no caso, equalização adaptativa de histograma com limitação de contraste (*CLAHE*) [PAA+87].

No experimento 7, testamos a robustez do modelo ao treinar usando o banco de dados DRIVE e testando no banco de dados STARE e vice-versa.

Nas subseções a seguir são descritos os experimentos realizados.

4.1.1 Experimento 1

Neste experimento, verificamos os resultados obtidos para uma janela de dimensão $11x11$ e com as 3 primeiras imagens de treinamento do banco de dados DRIVE.

Estas imagens, com a janela 11x11 resultam em 681.473 exemplos de treinamento.

A arquitetura da rede tem 9 camadas, sendo 4 camadas convolucionais, 2 camadas de max-pooling, 3 camadas densas. Os detalhes desta arquitetura estão descritas na tabela 4.2.

O modelo foi testado nas 20 imagens de teste e a figura 4.5 apresenta alguns resultados obtidos, os quais foram colorizados de acordo com a explicação do início do capítulo. E a tabela 4.3 contém as medidas realizadas para este conjunto de teste.

A figura 4.4 contém os gráficos da função de perda no conjunto de treinamento e do conjunto de validação e da acurácia no conjunto de validação para as épocas no treinamento do modelo.

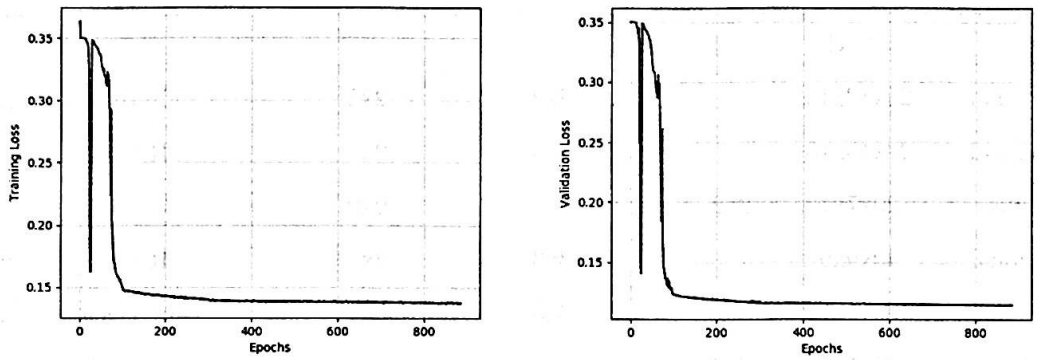
O modelo estabiliza o seu treinamento próximo a época 200, no entanto, o modelo continua melhorando até aproximadamente a época 700 e então o treinamento é interrompido próximo a 800 épocas, isto ocorre devido o uso da técnica *early-stopping*.

Note que as veias com espessura muito pequena não são segmentadas corretamente, porém as com grande espessura são, em geral, bem segmentadas. Além disso, há a presença de ruído, a qual pode ser verificada pela alta quantidade de falsos positivos.

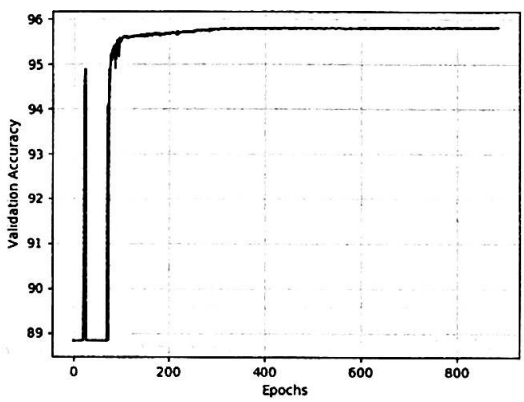
Foi obtido 91.20 % de acurácia, 78.63 % de sensitividade e 93.04 % de especificidade. Note que houve uma grande quantia de verdadeiros negativos : 3 684 910.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x11x11					
Convolucional	20x13x13		3x3	2x2		Retificadora linear
Convolucional	20x15x15		3x3	2x2		Retificadora linear
Max-pooling	20x7x7	2x2		0x0		
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Convolucional	40x11x11		3x3	2x2		Retificadora linear
Max-pooling	40x5x5	2x2		0x0		
Dropout	40x5x5				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.2: *Modelo de rede convolucional utilizada no experimento 1*



(a) *Perda no conjunto de treinamento e perda no conjunto de validação*

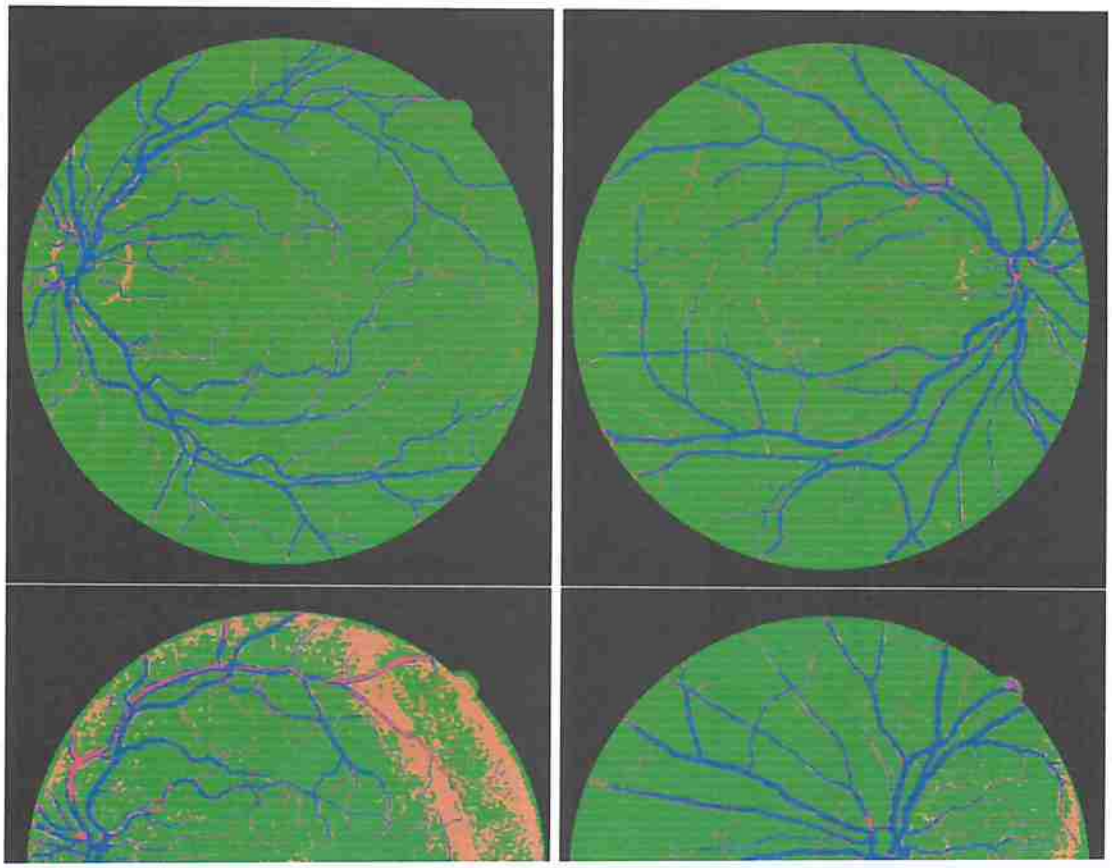


(b) *Acurácia no conjunto de validação*

Figura 4.4: *Gráfico da função de perda no conjunto de treinamento e no conjunto de validação e da acurácia no conjunto de validação*

Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN
DRIVE	DRIVE	0.9120	0.7863	0.9304	0.6223	454.221	3.684.910	275.584	123.428

Tabela 4.3: *Resultados dos experimentos*



Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x11x11					
Convolucional	20x13x13		3x3	2x2		Retificadora linear
Convolucional	20x15x15		3x3	2x2		Retificadora linear
Max-pooling	20x7x7	2x2		0x0		
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Convolucional	40x11x11		3x3	2x2		Retificadora linear
Max-pooling	40x5x5	2x2		0x0		
Dropout	40x5x5				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.2: *Modelo de rede convolucional utilizada no experimento 1*

4.1.2 Experimento 2

Neste experimento, a janela utilizada foi também de dimensão $11x11$ e as imagens de treinamento foram as respectivas 3 primeiras imagens do conjunto de treinamento, as quais estão presentes no apêndice B. O conjunto de dados foi limitado a 681.473 exemplos de treinamento e estes foram formados por imagens rotacionadas em 90 graus e não rotacionadas.

A arquitetura da rede é idêntica a do experimento 1. Os detalhes desta arquitetura estão descritas na tabela 4.4.

Novamente, o modelo foi testado nas 20 imagens de teste e a figura 4.7 apresenta alguns resultados obtidos, os quais foram colorizados de acordo com a explicação do início do capítulo. E a tabela 4.5 contém as medidas realizadas para este conjunto de teste.

A figura 4.6 contém os gráficos da função de perda no conjunto de treinamento e do conjunto de validação e da acurácia no conjunto de validação para as épocas no treinamento do modelo.

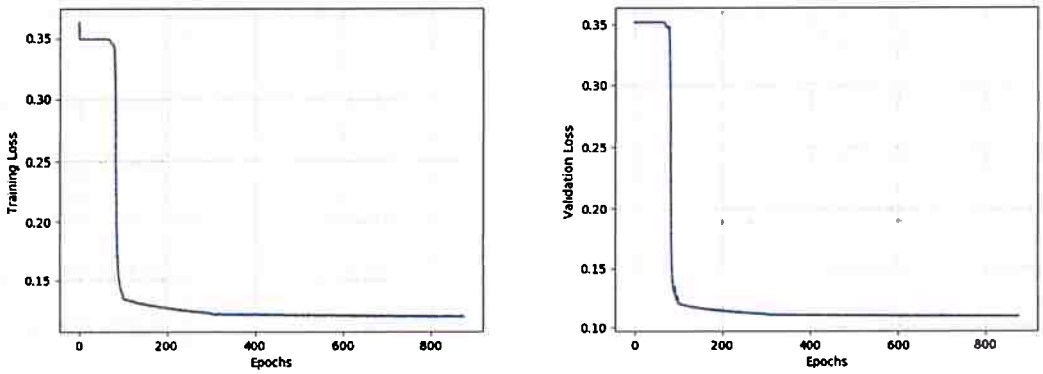
O modelo estabiliza o seu treinamento próximo a época 200, no entanto, o modelo continua melhorando até aproximadamente a época 700 e então o treinamento é interrompido próximo a 800 épocas.

Foi obtido 92.03 % de acurácia, 76.62 % de sensitividade e 94.28 % de especificidade. Note que houve uma grande quantia de verdadeiros negativos : 3 734 065.

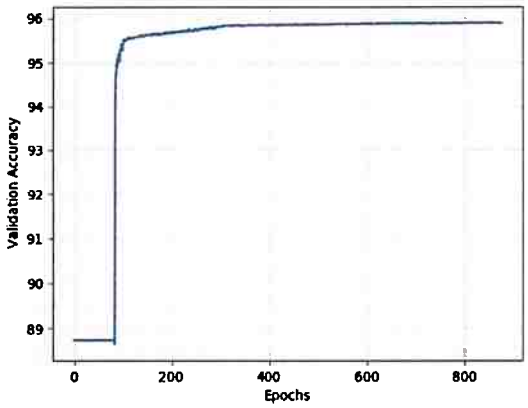
A acurácia e especificidade obtida neste experimento, é maior do que a do experimento 1 4.1.1, no entanto, obtivemos uma menor sensitividade.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x11x11					
Convolucional	20x13x13		3x3	2x2		Retificadora linear
Convolucional	20x15x15		3x3	2x2		Retificadora linear
Max-pooling	20x7x7	2x2		0x0		
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Convolucional	40x11x11		3x3	2x2		Retificadora linear
Max-pooling	40x5x5	2x2		0x0		
Dropout	40x5x5				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.4: *Rede Convolucional Utilizada No Experimento 2*



(a) *Perda no conjunto de treinamento e perda no conjunto de validação*



(b) *Acurácia no conjunto de validação*

Figura 4.6: *Gráfico da função de perda no conjunto de treinamento e no conjunto de validação e da acurácia no conjunto de validação*

Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN
DRIVE	DRIVE	0.9203	0.7662	0.9428	0.6615	442.643	3.734.065	226.429	135.006

Tabela 4.5: *Resultados dos experimentos*

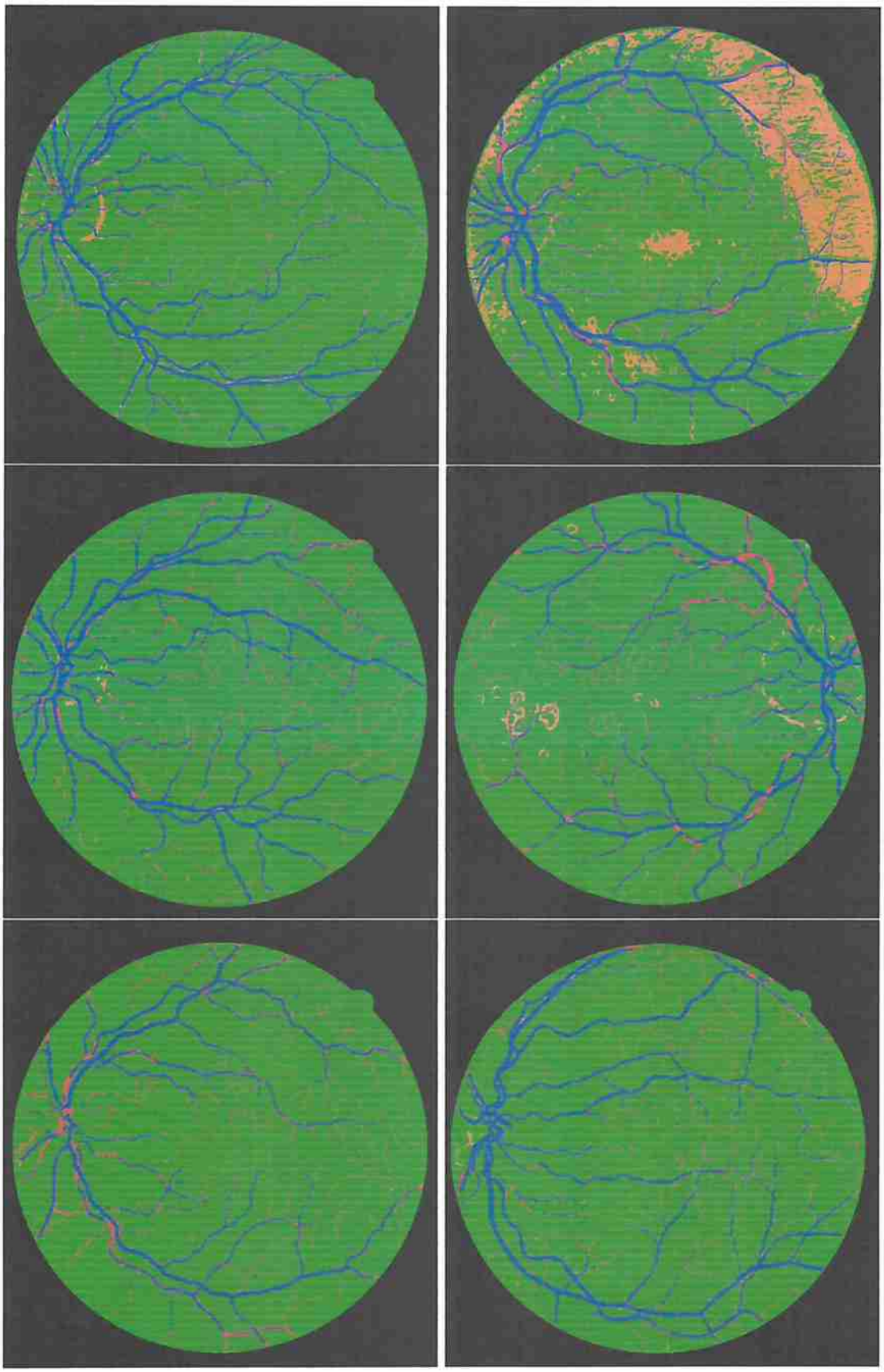


Figura 4.7: *Algumas Imagens de teste*

4.1.3 Experimento 3

Neste experimento, a janela utilizada foi também de dimensão $11x11$ e as imagens de treinamento foram as respectivas 3 primeiras imagens do conjunto de treinamento, as quais estão presentes no apêndice B. O conjunto de dados foi limitado a 681.473 exemplos de treinamento e estes foram formados por imagens rotacionadas em 45 graus e não rotacionadas.

A arquitetura da rede é idêntica a do experimento 1 e 2. Os detalhes desta arquitetura estão descritas na tabela 4.6.

A figura 4.8 contém os gráficos da função de perda no conjunto de treinamento e do conjunto de validação e da acurácia no conjunto de validação para as épocas no treinamento do modelo.

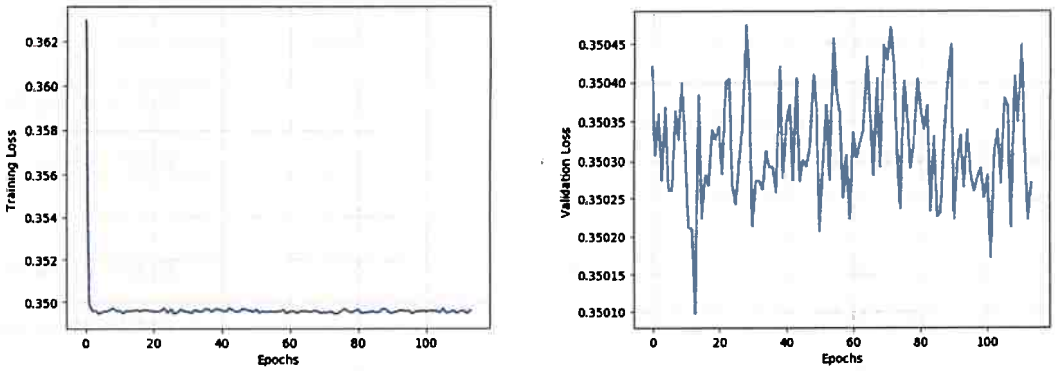
O modelo foi testado nas 20 imagens de teste e a figura 4.9 apresenta alguns resultados obtidos, os quais foram colorizados de acordo com a explicação do início do capítulo. E a tabela 4.7 contém as medidas realizadas para este conjunto de teste.

O modelo não estabiliza o seu treinamento em um nível de erro desejável, note que na função de perda no conjunto de treinamento, o modelo interrompe o treinamento em uma taxa de erro final com pouca diferença do inicial. Além disso, a função de perda no conjunto de validação não tende a diminuir o erro e apresenta muito ruído e apresenta muita oscilação.

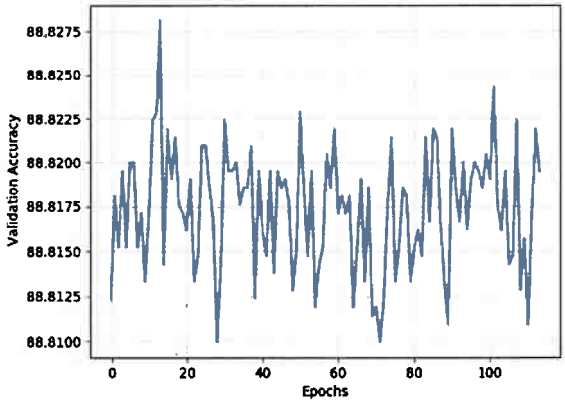
Os resultados obtidos neste experimento mostram que a acurácia de 0.8727 ocorre quando infere-se o valor 0 para qualquer exemplo dado.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x11x11					
Convolucional	20x13x13		3x3	2x2		Retificadora linear
Convolucional	20x15x15		3x3	2x2		Retificadora linear
Max-pooling	20x7x7	2x2		0x0		
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Convolucional	40x11x11		3x3	2x2		Retificadora linear
Max-pooling	40x5x5	2x2		0x0		
Dropout	40x5x5				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.6: *Rede Convolucional Utilizada No Experimento 3*



(a) *Perda no conjunto de treinamento e perda no conjunto de validação*



(b) *Acurácia no conjunto de validação*

Figura 4.8: *Gráfico da função de perda no conjunto de treinamento e no conjunto de validação e da acurácia no conjunto de validação*

Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN
DRIVE	DRIVE	0.8727	0.0	1.0	0	0	3.960.494	0	577.649

Tabela 4.7: *Resultados dos experimentos*

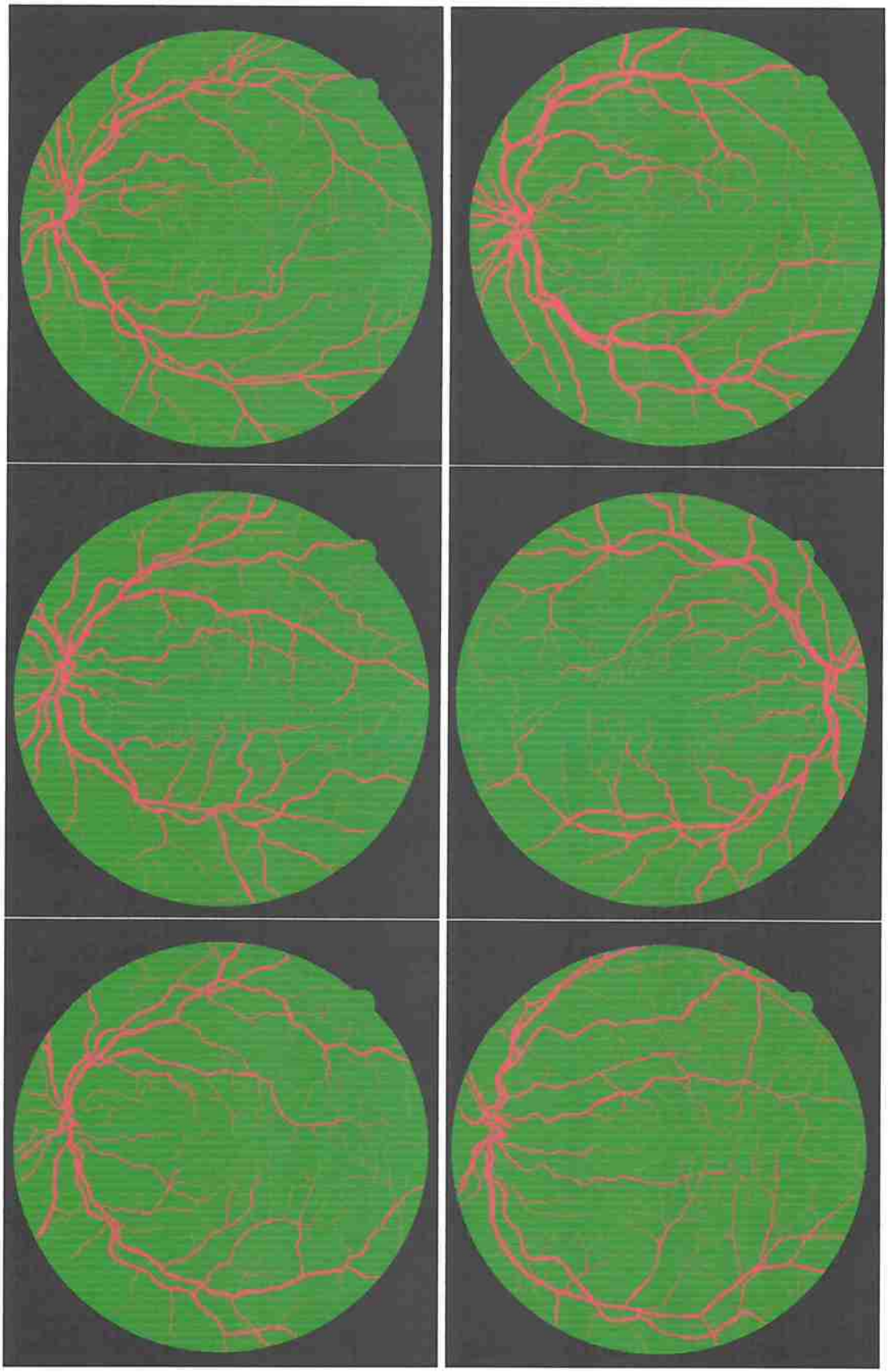


Figura 4.9: *Algumas Imagens de teste*

4.1.4 Experimento 4

Neste experimento, comparamos o que ocorre ao variar a dimensão da janela do W-operador. È utilizado as 3 primeiras imagens de treinamento, presentes no apêndice B. As medidas realizadas neste experimento são verificadas na tabela 4.8.

A arquitetura da rede, apesar de ser a mesma dos experimentos 1, 2 e 3, tem sua quantidade de parâmetros alteradas conforme a dimensão da janela utilizada, pois há a necessidade de adequar a camada de entrada do modelo para a janela utilizada. Portanto, é importante verificar a mudança na quantidade de pesos do modelo ao alterar o tamanho da janela, verifica-se estes pesos na tabela 4.9 para a janela 5x5, 4.10 para a janela 7x7, 4.11 para a janela 11x11, 4.12 para a janela 21x21 e na tabela 4.13 para a janela 31x31.

Na figura 4.10, há diferentes tamanhos de janelas utilizadas nos experimentos sobre uma mesma imagem. È possível verificar que com a janela $31x31$, por exemplo, há um grande aumento do contexto que pode ser visto pelo modelo, porém, com uma janela $5x5$ ocorre o oposto.

A acurácia aumenta conforme a dimensão da janela aumenta. Há uma melhora significativa nos resultados entre as janelas 11x11 e 21x21, porém pouca em relação a janela 21x21 e 31x31.

Note que com a janela de dimensão 31x31 obtemos 94.59% de acurácia, 74.03% de sensitividade e 97.59% de especificidade, no entanto, com uma janela de dimensão 5x5, obtemos 88.68% de acurácia, 72.92% de sensitividade e 90.97% de especificidade . Além disso, ao usar uma janela maior o número de falsos positivos teve tendencia de queda.

Dimensão Da Janela	Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN
5x5	DRIVE	DRIVE	0.8868	0.7292	0.9097	0.5410	421.257	3.603.216	357.278	156.392
7x7	DRIVE	DRIVE	0.8961	0.7389	0.9190	0.5711	426.863	3.639.917	320.577	150.786
11x11	DRIVE	DRIVE	0.9120	0.7863	0.9304	0.6223	454.221	3.684.910	275.584	123.428
21x21	DRIVE	DRIVE	0.9419	0.7677	0.9673	0.7740	443.473	3.831.029	129.465	134.176
31x31	DRIVE	DRIVE	0.9459	0.7403	0.9759	0.8179	427.690	3.865.287	95.207	149.959

Tabela 4.8: *Resultados dos experimentos com diferente tamanhos de Janelas*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x5x5					
Convolucional	20x7x7		3x3	2x2		Retificadora linear
Convolucional	20x9x9		3x3	2x2		Retificadora linear
Max-pooling	20x4x4	2x2		0x0		
Convolucional	40x6x6		3x3	2x2		Retificadora linear
Convolucional	40x8x8		3x3	2x2		Retificadora linear
Max-pooling	40x4x4	2x2		0x0		
Dropout	40x4x4				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.9: *Rede Convolucional Utilizada No Experimento 4 com janela 5x5*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x7x7					
Convolucional	20x9x9		3x3	2x2		Retificadora linear
Convolucional	20x11x11		3x3	2x2		Retificadora linear
Max-pooling	20x5x5	2x2		0x0		
Convolucional	40x7x7		3x3	2x2		Retificadora linear
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Max-pooling	40x4x4	2x2		0x0		
Dropout	40x4x4				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.10: *Rede Convolucional Utilizada No Experimento 4 com janela 7x7*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x11x11					
Convolucional	20x13x13		3x3	2x2		Retificadora linear
Convolucional	20x15x15		3x3	2x2		Retificadora linear
Max-pooling	20x7x7	2x2		0x0		
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Convolucional	40x11x11		3x3	2x2		Retificadora linear
Max-pooling	40x5x5	2x2		0x0		
Dropout	40x5x5				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

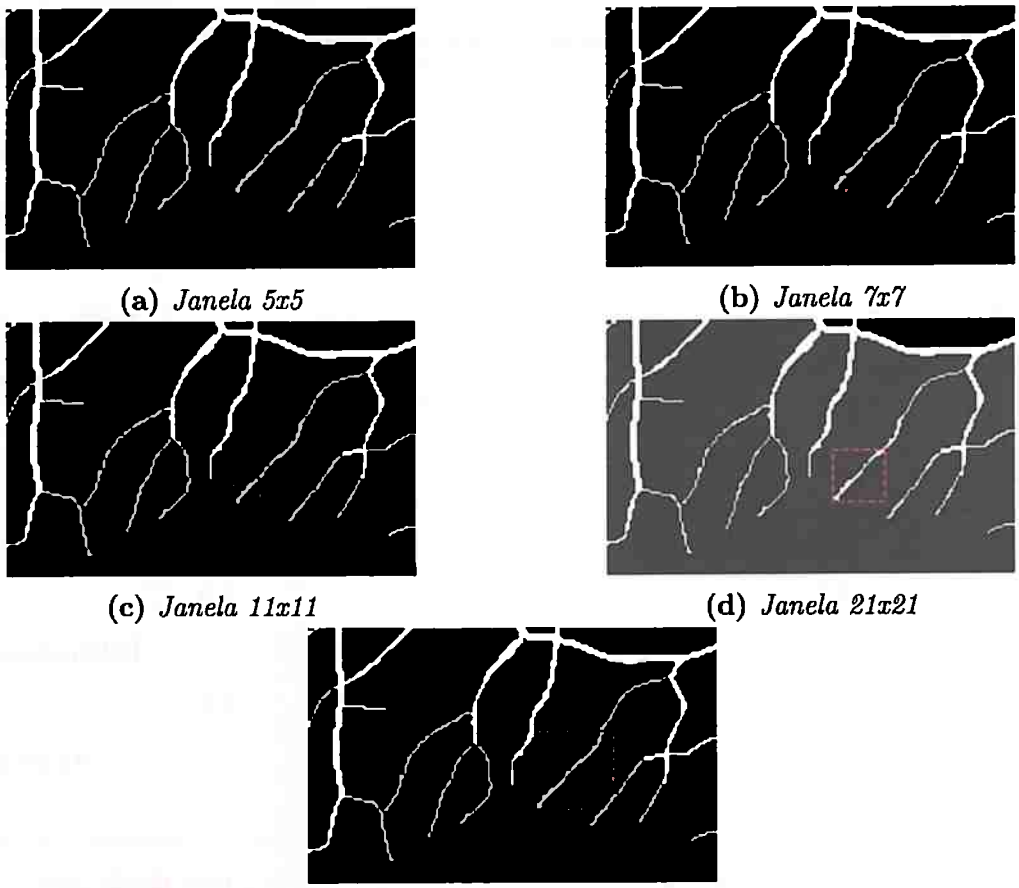
Tabela 4.11: *Rede Convolucional Utilizada No Experimento 4 com janela 11x11*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x21x21					
Convolucional	20x23x23		3x3	2x2		Retificadora linear
Convolucional	20x25x25		3x3	2x2		Retificadora linear
Max-pooling	20x12x12	2x2		0x0		
Convolucional	40x14x14		3x3	2x2		Retificadora linear
Convolucional	40x16x16		3x3	2x2		Retificadora linear
Max-pooling	40x8x8	2x2		0x0		
Dropout	40x8x8				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.12: *Rede Convolucional Utilizada No Experimento 4 com janela 21x21*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x31x31					
Convolucional	20x33x33		3x3	2x2		Retificadora linear
Convolucional	20x35x35		3x3	2x2		Retificadora linear
Max-pooling	20x17x17	2x2		0x0		
Convolucional	40x19x19		3x3	2x2		Retificadora linear
Convolucional	40x21x21		3x3	2x2		Retificadora linear
Max-pooling	40x10x10	2x2		0x0		
Dropout	40x10x10				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.13: *Rede Convolucional Utilizada No Experimento 4 com janela 31x31*



(a) *Janela 5x5* (b) *Janela 7x7*

(c) *Janela 11x11* (d) *Janela 21x21*

(e) *Janela 31x31*

Figura 4.10: *Ilustração de janelas de diferentes tamanhos sobre uma certa posição de uma imagem*

4.1.5 Experimento 5

Neste experimento, comparamos o que ocorre ao variar a quantidade de imagens de treinamento. Fixamos a dimensão da janela em 11x11. As medidas realizadas neste experimento são verificadas na tabela 4.14.

A arquitetura da rede é descrita na tabela 4.15 e é a mesma utilizada anteriormente nos experimentos 1,2 e 3. Importante notar que 1 imagem de treinamento gerou 225.600 exemplos, 3 imagens geraram 681.473 exemplos, 10 imagens geraram 2.271.374 exemplos e 20 imagens geraram 4.541.006 exemplos.

Na figura 4.11 ha a comparação de 3 imagens de teste para cada quantidade de imagens de treinamento utilizadas.

Ao aumentar a quantidade de imagens de treinamento a acurácia melhora, principalmente ao comparar 3 e 20 imagens.

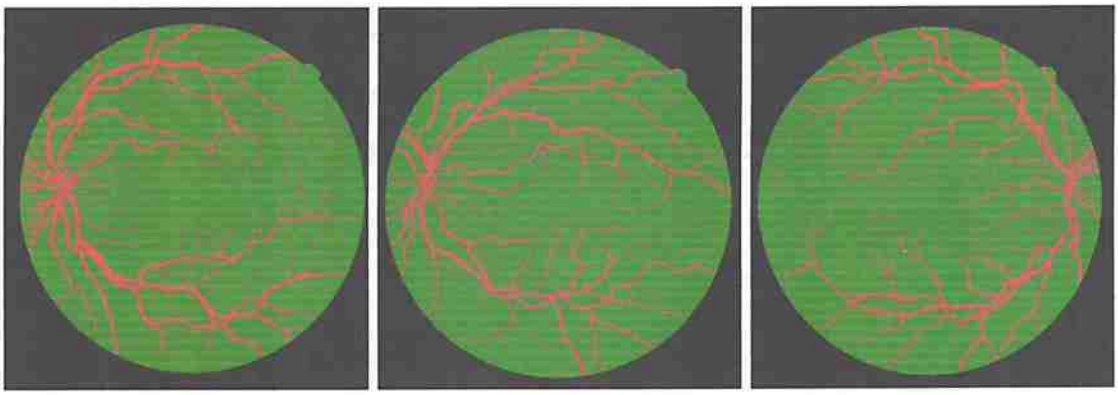
Note que com 3 imagens de treinamento obtemos 91.20% de acurácia, 78.63% de sensitividade e 93.04% de especificidade, no entanto, ao utilizar todas as 20 imagens de treinamento, obtemos 94.93% de acurácia, 68.72% de sensitividade e 98.75% de especificidade . Além disso, ao usar um número maior de imagens de treinamento, a quantidade de verdadeiros positivos e falsos positivos tenderam a diminuir.

Dimensão Da Janela	Nº Imagens	Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN
11x11	1	DRIVE	DRIVE	0.8727	0.0	1.0	0.0	0	3.960.494	0	577.649
11x11	3	DRIVE	DRIVE	0.9120	0.7863	0.9304	0.6223	454.221	3.684.910	275.584	123.428
11x11	10	DRIVE	DRIVE	0.9487	0.7377	0.9794	0.8399	426.174	3.879.276	81.218	151.475
11x11	20	DRIVE	DRIVE	0.9493	0.6872	0.9875	0.8895	397.005	3.911.184	49.310	180.644

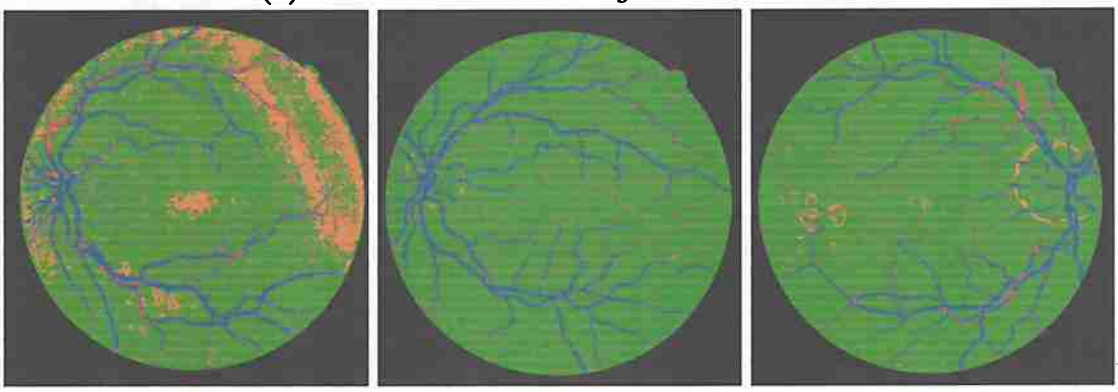
Tabela 4.14: *Resultados dos experimentos modificando o conjunto de treinamento*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x11x11					
Convolucional	20x13x13		3x3	2x2		Retificadora linear
Convolucional	20x15x15		3x3	2x2		Retificadora linear
Max-pooling	20x7x7	2x2		0x0		
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Convolucional	40x11x11		3x3	2x2		Retificadora linear
Max-pooling	40x5x5	2x2		0x0		
Dropout	40x5x5				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

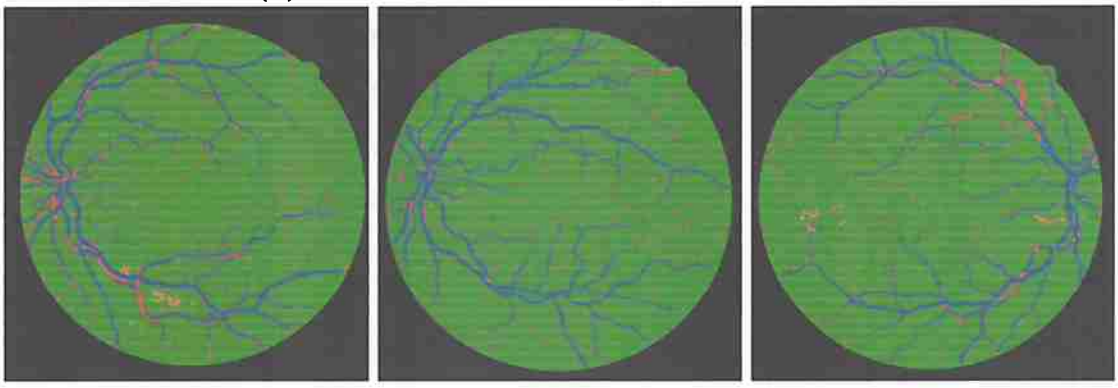
Tabela 4.15: *Rede Convolucional Utilizada No Experimento 4 com janela 11x11*



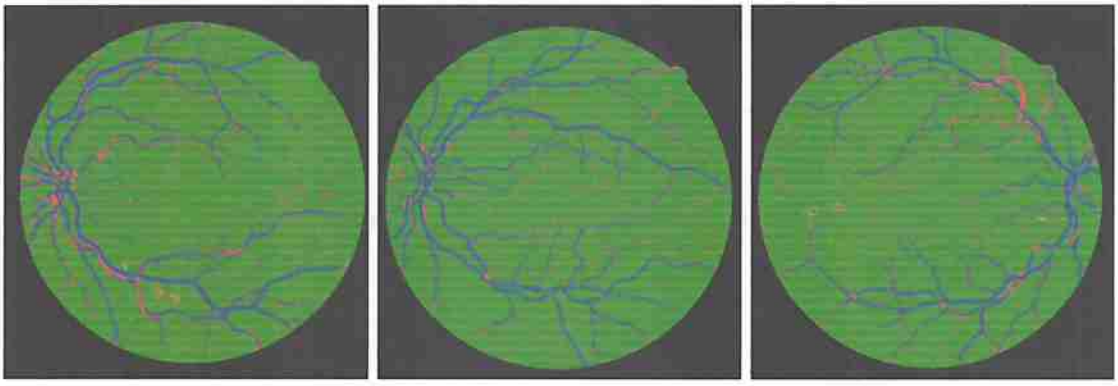
(a) *Resultados usando 1 imagem de treinamento*



(b) *Resultados usando 3 imagens de treinamento*



(c) *Resultados usando 10 imagens de treinamento*



(d) *Resultados usando 20 imagens de treinamento*

Figura 4.11: *Comparação de imagens de teste para diferentes quantidade de imagens de treinamento*

4.1.6 Experimento 6

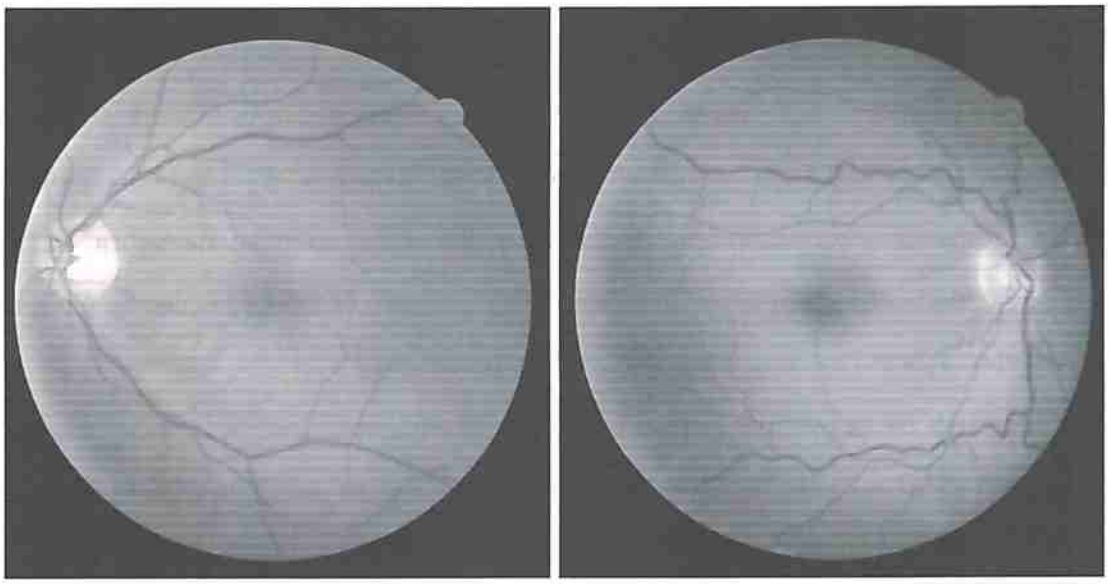
Neste experimento, o conjunto de dados DRIVE é preprocessado utilizando o algoritmo *CLAHE* (contrast limited adaptive histogram equalization) [PAA+87], também chamado de histograma adaptativo com limitação de contraste. Para exemplificar, na figura 4.12 há 2 imagens de treinamento pre-processadas com este algoritmo. O modelo utilizado é o mesmo do experimento 1, 2 e 3 e é descrito com detalhes na tabela 4.17.

Na tabela 4.16 há as medidas realizadas neste experimento.

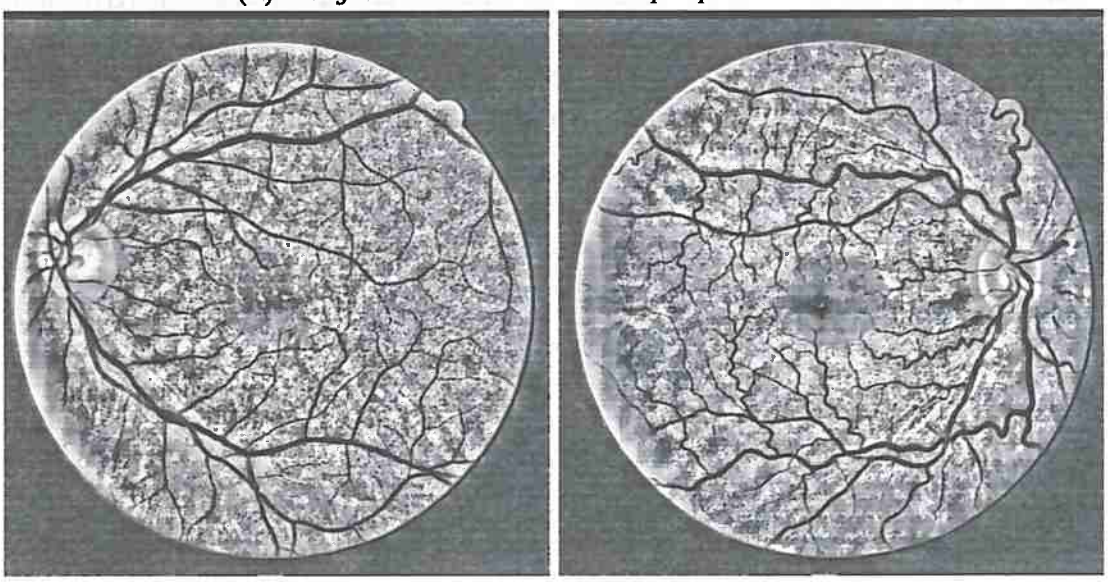
Na figura 4.12 há a comparação de 3 imagens de teste para cada quantidade de imagens de treinamento utilizadas.

A maior acurácia ocorre com 20 imagens de treinamento, utilizando CLAHE, obtendo acurácia de 95.04%.

Note que com 20 imagens de treinamento sem CLAHE obtemos 95.04% de acurácia, 73.19% de sensitividade e 98.22% de especificidade, no entanto, ao utilizar o CLAHE e todas as 20 imagens de treinamento, obtemos 95.30% de acurácia, 76.50% de sensitividade e 98.04% de especificidade .



(a) *Imagens de treinamento sem pré-processamento*

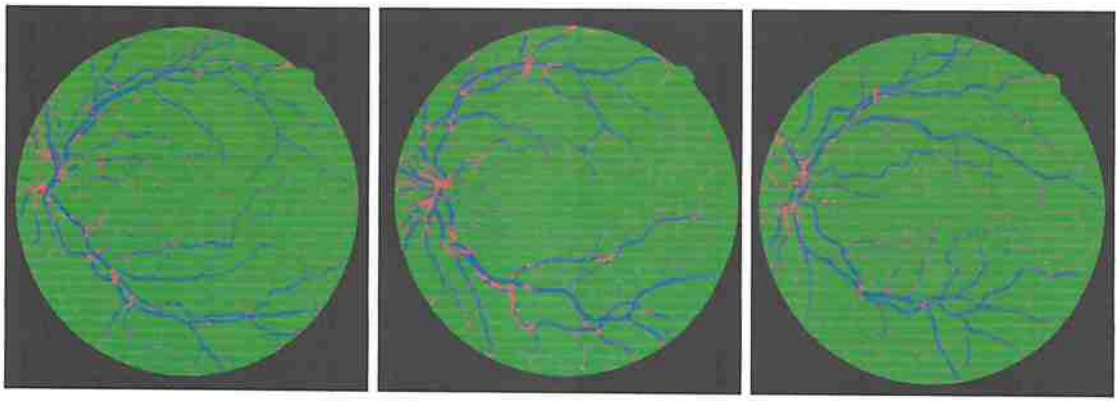


(b) *Imagens de treinamento com pré-processamento usando CLAHE*

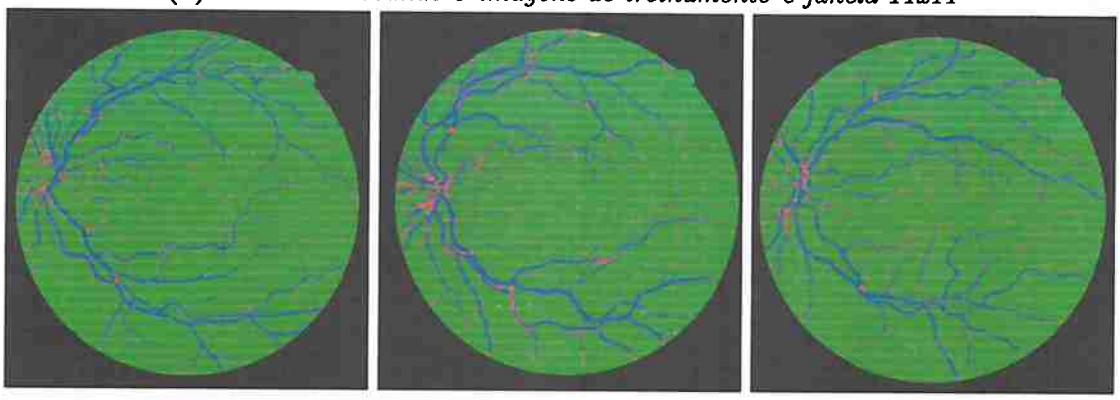
Figura 4.12: *Diferença de algumas imagens de treinamento ao usar CLAHE*

Dimensão Da Janela	Nº Imagens	Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN
11x11	3	DRIVE	DRIVE	0.9422	0.6906	0.9789	0.8273	398.929	3.877.229	83.265	178.720
11x11	10	DRIVE	DRIVE	0.9475	0.7515	0.9761	0.8214	434.153	3.866.140	94.354	143.496
11x11	20	DRIVE	DRIVE	0.9504	0.7319	0.9822	0.8576	422.785	3.890.298	70.196	154.864
31x31	20	DRIVE	DRIVE	0.9530	0.7650	0.9804	0.8506	441.951	3.882.901	77.593	135.698

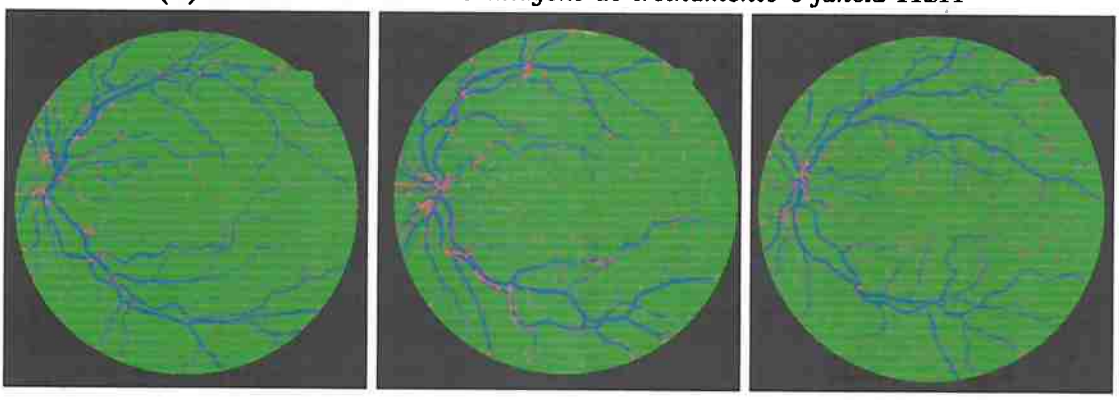
Tabela 4.16: *Resultados do experimento 6 usando o algoritmo de pre-processamento CLAHE*



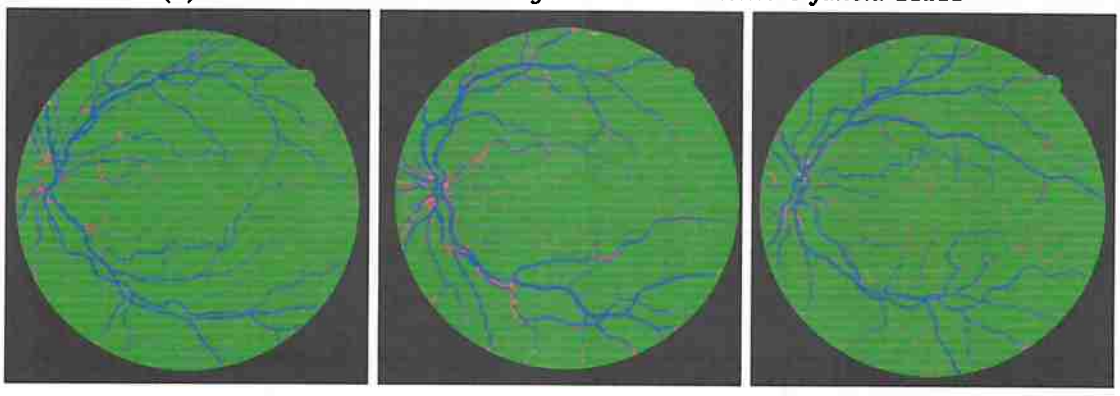
(a) *Resultados usando 3 imagens de treinamento e janela 11x11*



(b) *Resultados usando 10 imagens de treinamento e janela 11x11*



(c) *Resultados usando 20 imagens de treinamento e janela 11x11*



(d) *Resultados usando 20 imagens de treinamento e janela 31x31*

Figura 4.13: *Alguns resultados de imagens de teste*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x11x11					
Convolucional	20x13x13		3x3	2x2		Retificadora linear
Convolucional	20x15x15		3x3	2x2		Retificadora linear
Max-pooling	20x7x7	2x2		0x0		
Convolucional	40x9x9		3x3	2x2		Retificadora linear
Convolucional	40x11x11		3x3	2x2		Retificadora linear
Max-pooling	40x5x5	2x2		0x0		
Dropout	40x5x5				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

Tabela 4.17: *Rede Convolucional Utilizada No Experimento 6*

4.1.7 Experimento 7

Neste experimento testamos a robustez do modelo ao treinar usando o banco de dados DRIVE e testando no banco de dados STARE e vice-versa. Usamos o algoritmo CLAHE nos dois banco de dados e janela de tamanho 31x31.

Dimensão Da Janela	Nº Imagens	Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN
11x11	3	DRIVE	STARE	0.9281	0.6153	0.9642	0.6655	396.279	5.376.631	199.098	247.683
31x31	3	DRIVE	STARE	0.9397	0.6586	0.9722	0.7326	424.164	5.420.976	154.753	219.798
31x31	20	DRIVE	STARE	0.9465	0.7688	0.9765	0.7295	495.133	5.392.225	183.504	148.829
31x31	3	STARE	DRIVE	0.9338	0.5534	0.9893	0.8832	319.672	3.918.239	42.255	257.977
11x11	3	STARE	DRIVE	0.9266	0.4959	0.9894	0.8730	286.499	3.918.843	41.651	291.150

Tabela 4.18: *Resultados do experimento 7 usando o banco de dados STARE e DRIVE*

4.2 Discussão Experimentos

Na tabela 4.19 verificamos as medidas de todos os experimentos deste trabalho, realizados com o banco de dados DRIVE e STARE.

Descrição	Experimento	Dimensão Da Janela	Nº Imagens	Conjunto de treinamento	Conjunto de teste	Acurácia	Sensitividade	Especificidade	PPV
Sem pré-processamento	1	11x11	3	DRIVE	DRIVE	0.9120	0.7863	0.9304	0.6223
Rotações de 90 graus	2	11x11	3	DRIVE	DRIVE	0.9203	0.7662	0.9428	0.6615
Rotações de 45 graus	3	11x11	3	DRIVE	DRIVE	0.8727	0.0	1.0	0.0
Sem pré-processamento	4	5x5	3	DRIVE	DRIVE	0.8868	0.7292	0.9097	0.5410
Sem pré-processamento	4	7x7	3	DRIVE	DRIVE	0.8961	0.7389	0.9190	0.5711
Sem pré-processamento	4	21x21	3	DRIVE	DRIVE	0.9419	0.7677	0.9673	0.6223
Sem pré-processamento	4	31x31	3	DRIVE	DRIVE	0.9459	0.7403	0.9759	0.8179
Sem pré-processamento	5	11x11	1	DRIVE	DRIVE	0.8727	0.0	1.0	0.0
Sem pré-processamento	5	11x11	10	DRIVE	DRIVE	0.9487	0.7377	0.9794	0.8399
Sem pré-processamento	5	11x11	20	DRIVE	DRIVE	0.9493	0.6872	0.9875	0.8895
Pré-processamento com CLAHE	6	11x11	3	DRIVE	DRIVE	0.9422	0.6906	0.9789	0.8273
Pré-processamento com CLAHE	6	11x11	10	DRIVE	DRIVE	0.9475	0.7515	0.9761	0.8214
Pré-processamento com CLAHE	6	11x11	20	DRIVE	DRIVE	0.9504	0.7319	0.9822	0.8576
Pré-processamento com CLAHE	6	31x31	20	DRIVE	DRIVE	0.9530	0.7650	0.9804	0.8506
Pré-processamento com CLAHE	7	11x11	3	DRIVE	STARE	0.9231	0.6153	0.9642	0.6655
Pré-processamento com CLAHE	7	31x31	3	DRIVE	STARE	0.9397	0.6586	0.9722	0.7326
Pré-processamento com CLAHE	7	31x31	20	DRIVE	STARE	0.9465	0.7688	0.9765	0.7295
Pré-processamento com CLAHE	7	11x11	3	STARE	DRIVE	0.9266	0.4959	0.9894	0.8730
Pré-processamento com CLAHE	7	31x31	3	STARE	DRIVE	0.9338	0.5534	0.9893	0.8832

Tabela 4.19: *Compilação dos resultados de todos os experimentos*

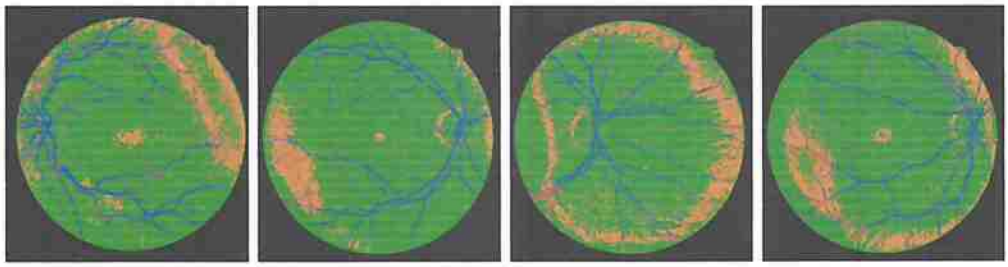
Nos experimentos 1, 2 e 3, foi verificado que modificar os exemplos para treinar o modelo,

ou seja, rotacionando os exemplos em 45 ou 90 graus, altera os resultados de forma a melhorar ou até mesmo não permitir a aprendizagem do modelo. Ao usar o experimento 1 como linha de base, verifica-se que ao usar exemplos rotacionados em 90 graus, melhora-se a acurácia em $0,83\%$ e a especificidade em $1,24\%$ e o PPV em 3.92%. No entanto, a sensitividade piorou em $2,01\%$.

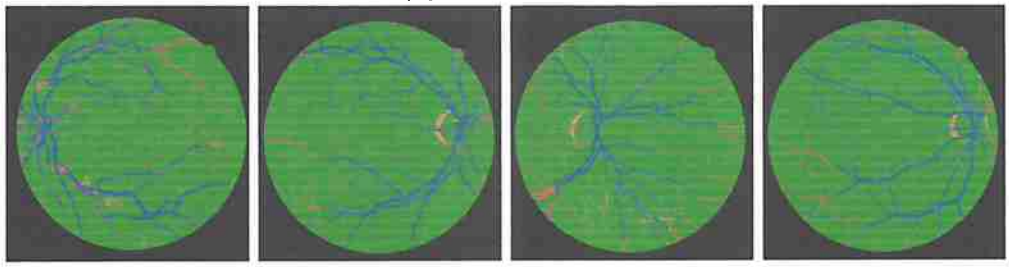
No experimento 3, o uso de exemplos rotacionados em 45 graus, fez com que o treinamento terminasse com uma função de perda de treinamento ruim, pois a diferença entre a perda na época 0 e época final é muito pequena, além disso, pode-se verificar que a acurácia no conjunto de validação ficou pior do que na época inicial, em algumas épocas.

Portanto, de acordo com os gráficos da função de perda do treinamento e da validação deste experimento, pode-se afirmar que o treinamento falhou. Entre as motivos disto ocorrer, envolve a escolha dos pesos iniciais, os quais foram fixados para todos os experimentos e ao conjunto de dados escolhido.

No experimento 4, tendo fixados os mesmos parâmetros dos experimentos anteriores porém, alterando o tamanho da janela do W-operador, pode ser verificado que os resultados melhoram conforme a dimensão da janela aumenta, é esperado que em algum momento este aumento não ira melhorar os resultados devido ao excesso do tamanho da janela adicionar informações irrelevantes a segmentação. No entanto, para os experimentos realizados, a janela de dimensão $31x31$ tem o aumento de 3.39% de acurácia e 4.55% de especificidade e 19.56% de PPV, tendo uma piora de 4.6% de sensitividade, ao comparar com a janela de dimensão $11x11$. Observando algumas imagens de teste na figura 4.14 é possível ver a diminuição de ruído, a qual é ilustrada pelo tom laranja.



(a) *Janela 11x11*



(b) *Janela 31x31*

Figura 4.14: *comparação de algumas imagens de teste com janelas diferentes*

No experimento 5, comprova-se que a utilização de mais imagens de treinamento permite estimar com melhor acurácia o operador de imagem. Observa-se que ao usar 1 única imagem, o modelo não consegue aprender, a acurácia resulta em 87.27% e a sensitividade em 0.0% e a especificidade em 1.0%, assim como ocorreu no experimento 3. Note que, não foi apenas o fato de ter sido usado 1 imagem de treinamento que causou este resultado, mas a arquitetura, o algoritmo de inicialização dos pesos, etc, assim como visto no capitulo 2.

Porém, ao usar 3 imagens há uma acurácia de 91.20%, no entanto, nota-se um pouco de ruído nas imagens resultantes. Utilizando 10 imagens o ruído observável diminui consideravelmente. Utilizando todo o conjunto de treinamento disponível, 20 imagens, há um pequeno aumento da acurácia porém uma grande melhora no PPV e uma diminuição na sensitividade. Sendo que para 3, 10 e 20 imagens de treinamento, a especificidade aumentou.

No experimento 6 se obteve o melhor resultado de todos os experimentos ao utilizar o pre-processamento com o algoritmo CLAHE [PAA+87], 20 imagens para treinamento e janela com dimensão 31x31 , obtendo 95.30% de acurácia. O algoritmo CLAHE (*Contrast-limited adaptive histogram equalization*) foi utilizado em outros estudos para a segmentação de veias [ZWWS14] [SEM11] [ABPS08] [RR14].

O uso do algoritmo CLAHE neste experimento foi utilizado para melhorar a acurácia do classificador, sendo que, este tipo de pre-processamento, permite compensar os efeitos da

diferença de iluminação em cada imagem da retina [ZWWS14] melhorando o seu contraste.

No experimento 7 testamos a robustez do modelo ao treinar usando o banco de dados DRIVE e testando no banco de dados STARE e vice-versa. Utilizamos o pre-processamento CLAHE para os 2 conjunto de dados e janela de tamanho 31x31.

Verifica-se que pode-se melhorar a estimação de um W-operador ao aumentar a dimensão da janela para um tamanho que permita o modelo ter suficiente contexto para a classificação do pixel e aumentar quantidade de imagens utilizadas para o treinamento.

4.2.1 Comparação com outros trabalhos com o banco de dados DRIVE

Método	Tipo de Aprendizado	Descrição	Ano	Sensitividade	Especificidade	Acurácia
Meng Li et al. [MLLZH17]	Supervisionado	Descrições locais de reforço	2016	0.7680	0.9827	0.9630
Lupascu et al. [LTT10]	Supervisionado	Classificador AdaBoost	2010	0.7200		0.9597
Ricci and Perfetti [RP07]	Supervisionado	Operadores de linha e Classificação de vetores de suporte	2007			0.9563
Wang et al. [WYC+15]	Supervisionado	Rede neural convolucional e Random Forest	2013	0.8173	0.9733	0.9533
Andre Lopes. 4.1.6	**Supervisionado**	**Aprendizado de operador de imagem com deep learning**	**2017**	**0.7650**	**0.9804**	**0.9530**
Li et al. [LFX+16]	Supervisionado	Rede neurais	2016	0.7569	0.9816	0.9527
Fraz et al. [FRH+12]	Supervisionado	Ensemble de classificadores de árvores de decisão reforçadas e ensacadas	2012	0.7406	0.9807	0.9480
Qian Zhao et al. [ZWWS14]	Não supervisionado	Conjunto de níveis e crescimento de região	2014	0.7354	0.9789	0.9477
2nd Observador humano [SAN+04b]		Observador humano independente		0.7746	0.9724	0.9472
Lam et al. [LGL10]	Não supervisionado	Modelagem de Multiconcavidade Baseada em Regularização	2010			0.9472
Soares et al.[SLC+06]	Supervisionado	2-D Gabor Wavelet e classificação supervisionada	2006			0.9466
Mendonça and Campilho [MC06]	Não supervisionado	Combinação de Detecção de Linhas Centrais e Reconstrução Morfológica	2006	0.7344	0.9764	0.9452
Marin et al. [MAGAB11]	Supervisionado	Recursos baseados em níveis de cinza e momentos invariantes	2011	0.7067	0.9801	0.9452
Staal et al. [SAN+04a]	Supervisionado	Segmentação de vasos com base em Ridge em imagens de cor da retina	2004			0.9442
Montegner. [Mon17a]	Supervisionado	Aprendizado de operador de imagem usando modelos lineares	2017			0.9440
You et al. [YPY+11]	Semi-supervisionado	Projeção radial	2011	0.7410	0.9751	0.9434
Welikala et al. [WDH+XX]	Supervisionado	Operador de linha modificado e classificação dupla	2014	0.7370	0.9521	0.9420
Nguyen et al. [NBPR13]	Supervisionado	Detecção de linha de escala múltipla	2013	0.7321	0.9487	0.9407
Delibasis et al. [DKTAXX]	Não supervisionado	Algoritmo de rastreamento automático baseado em modelo	2010			0.9407
Montegner et al. [ISMJ16]	Supervisionado	Aprendizado de operador de imagem	2016			0.9391
Zana [ZK01]		Avaliação matemática de morfologia e curvatura	2001			0.9377
Alonso-Montes et al. [AMVDP08]	Não supervisionado	Uma abordagem paralela de pixels	2008			0.9185
Hong Tan et al. [TAB+17]	Supervisionado	Uso de rede neural convolucional	2017	0.7357	0.9694	0.9268

Tabela 4.20: *Resultado de diferentes métodos de segmentação no banco de dados DRIVE*

A tabela 4.20 contém resultados da estado da arte e próximos dele com informações resumidas dos artigos publicados. Ao verificar os resultados desta tabela, verificamos que :

- A média das acurácias é : 94.49 %
- A média das sensitividades disponíveis é : 74.61 %
- A média das especificidades disponíveis é : 97.26 %

Este trabalho ficou acima da média em relação a acurácia, sensitividade e especificidade. A figura 4.15 ilustra as acurácias da tabela 4.20 em forma de gráfico em barras, mostrando a diferença das acurácias de outros autores em relação a este trabalho.

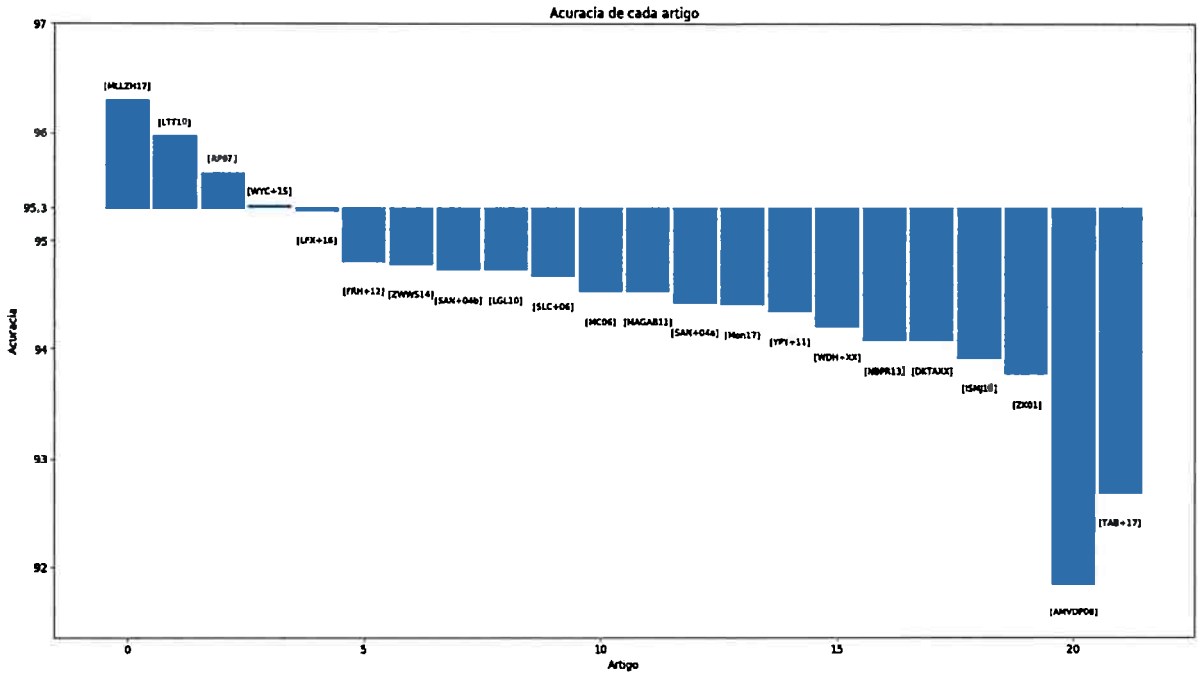


Figura 4.15: *Gráfico em barras comparando a acurácia dos outros trabalhos da tabela 4.20 com o resultado deste trabalho 4.1.6. Note que o valor de* 95.3 *no eixo das acurácias, se refere a acurácia deste trabalho.*

A tabela 4.20 apresenta resultados, incluindo o estado da arte, com informações resumidas de cada artigo publicado pelos autores. Ao comparar o melhor resultado obtido em nossos experimentos com esta tabela, verificamos que obtivemos resultados próximos ao estado da arte. Nesta tabela, ficamos em 5° lugar em ordem de acurácia, 4° lugar em ordem de sensitividade e em especificidade. Nosso trabalho usando W-operadores apresenta resultados próximos a outros trabalhos que fazem uso de técnicas específicas para esta tarefa de segmentação de veias. O uso de W-operadores, no entanto, pode ser usado para diversos casos

de segmentação. Portanto, uma vantagem é a generalização do nosso trabalho a outros problemas, inclusive permitindo a fácil mudança de modelo, por exemplo, de rede convolucional para árvore de decisão.

4.2.2 Comparação com outros trabalhos com o banco de dados STARE

Método	Tipo de Aprendizado	Descrição	Ano	Sensitividade	Especificidade	Acurácia
Meng Li et al. [MLLZH17]	Supervisionado	Descrições locais de reforço	2016	0.7890	0.9883	0.9710
Li et al. [LFX+16]	Supervisionado	Rede neurais	2016	0.7726	0.9844	0.9628
Wang et al. [WYC+15]	Supervisionado	Rede neural convolucional e Random Forest	2013	0.8104	0.9791	0.9621
Ricci and Perfetti [RP07]	Supervisionado	Operadores de linha e Classificação de vetores de suporte	2007			0.9584
Lam et al. [LGL10]	Não supervisionado	Modelagem de Multiconcavidade Baseada em Regularização	2010			0.9567
Fraz et al. [FRH+12]	Supervisionado	Ensemble de classificadores de árvores de decisão reforçadas e ensacadas	2012	0.7548	0.9763	0.9534
Marin et al. [MAGAB11]	Supervisionado	Recursos baseados em níveis de cinza e momentos invariantes	2011	0.6944	0.9819	0.9526
Staal et al. [SAN+04a]	Supervisionado	Segmentação de vasos com base em Ridge em imagens de cor da retina	2004			0.9516
Qian Zhao et al. [ZWWS14]	Não supervisionado	Conjunto de níveis e crescimento de região	2014	0.7187	0.9767	0.9509
You et al. [YPY+11]	Semi-supervisionado	Projeção radial	2011	0.7260	0.9756	0.9497
Soares et al.[SLC+06]	Supervisionado	2-D Gabor Wavelet e classificação supervisionada	2006			0.9480
Andre Lopes. 4.1.7	**Supervisionado**	**Aprendizado de operador de imagem com deep learning**	**2017**	**0.7688**	**0.9765**	**0.9465**
Mendonça and Campilho [MC06]	Não supervisionado	Combinação de Detecção de Linhas Centrais e Reconstrução Morfológica	2006	0.6996	0.9730	0.9440
Welikala et al. [WDH+XX]	Supervisionado	Operador de linha modificado e classificação dupla	2014	0.7448	0.9533	0.9431
2nd Observador humano [SAN+04b]		Observador humano independente		0.8951	0.9384	0.9348
Nguyen et al. [NBPR13]	Supervisionado	Detecção de linha de escala múltipla	2013	0.7314	0.9429	0.9324
Delibasis et al. [DKTAXX]	Não supervisionado	Algoritmo de rastreamento automático baseado em modelo	2010			0.9324

Tabela 4.21: *Resultado de diferentes métodos de segmentação no banco de dados STARE*

A tabela 4.21 contém resultados da estado da arte e próximos dele com informações resumidas dos artigos publicados. Ao verificar os resultados desta tabela, verificamos que :

- A média das acurácias é : 95.02 %
- A média das sensitividades disponíveis é : 75.78 %
- A média das especificidades disponíveis é : 96.99 %

Este trabalho ficou acima da média em relação sensitividade e especificidade e abaixo em relação a acurácia. A figura 4.16 ilustra as acurácias da tabela 4.21 em forma de gráfico em barras, mostrando a diferença das acurácias de outros autores em relação a este trabalho.

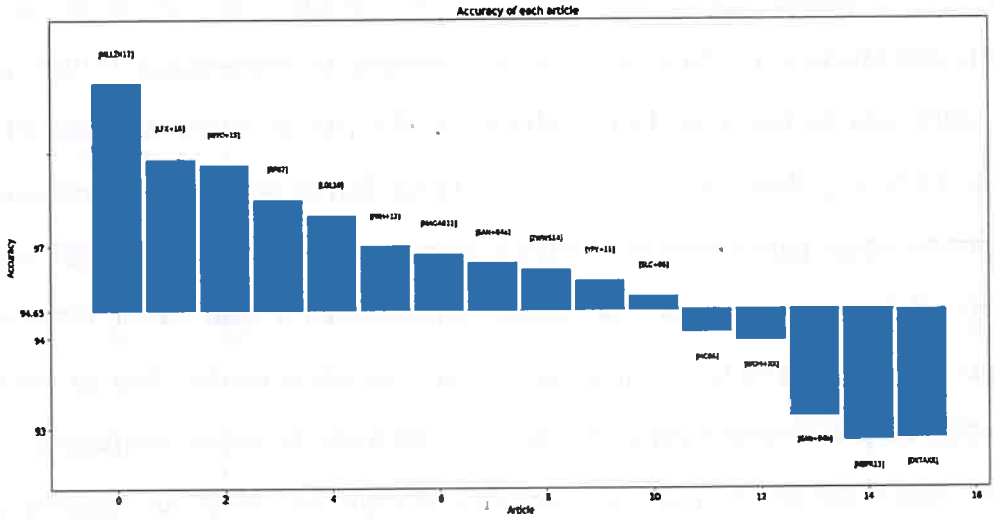


Figura 4.16: *Gráfico em barras comparando a acurácia dos outros trabalhos da tabela 4.21 com o resultado deste trabalho 4.1.6. Note que o valor de* 95.3 *no eixo das acurácias, se refere a acurácia deste trabalho.*

4.2.3 Comparação deste trabalho com os outros trabalhos com o banco de dados STARE e DRIVE

Observando os resultados de diversos autores nas duas tabelas 4.20 e 4.21 notamos que o mesmo autor Meng Li et al. [MLLZH17] consegue os melhores resultados para os dois banco de dados. Além disso, há outros trabalhos que utilizam redes neurais convolucionais, portanto vamos verificar estes trabalhos e a abordagem de Meng Li et al. a seguir e compará-los com a nossa abordagem.

Meng Li et al. [MLLZH17] na sua abordagem de segmentação de veias faz a construção de um vetor de características para cada pixel e então faz a tarefa de classificação com máquinas de vetores de suporte e em seguida realiza um pós-processamento baseado em reconstrução morfológica para melhorar situações em que haja veias desconexas. O interessante neste trabalho de Meng Li et al. é a construção dos vetores de características, os quais são feitos utilizando a informação local para cada pixel, e então é extraído uma característica baseada em gradiente morfológico. Ao final, estas características são combinadas para formar os descritores locais de reforço. Meng Li et al. [MLLZH17] apresenta resultados no estado da arte tanto para o banco de dados DRIVE, quanto para o STARE. Note que há apenas a preocupação de segmentar veias em imagens de retina e as técnicas utilizadas para tanto são focadas no objetivo proposto. Em contrapartida, nosso trabalho com o uso de W-operadores permite a transferência e uso fácil para outros problemas de segmentação. Obtivemos um resultado 1.00% pior no banco de dados DRIVE e 2.45% pior no banco de dados STARE.

Li et al. [LFX+16], Wang et al. [WYC+15] e Hong Tan et al. [TAB+17] utilizam redes neurais convolucionais para tratar o problema proposto de segmentar veias da retina. Li et al. [LFX+16] faz o uso de uma rede neural convolucional a qual recebe como entrada sub-imagens e como saída cada sub-imagem com as veias segmentadas. Não há uso de pre-processamento ou pós-processamento. No entanto, diferente da nossa abordagem com W-operadores e com o uso de redes neurais convolucionais que faz uso de sub-imagens que são formadas por vizinhanças de cada pixel, a arquitetura de Li et al. não utiliza camadas de convolução e a saída da rede neural tem a mesma dimensão da imagem de entrada com as veias já segmentadas. No nosso trabalho, obtemos uma acurácia semelhante do trabalho de Li et al, sendo que obtivemos um resultado 0.03% melhor no banco de dados DRIVE e 1.63% pior no banco de dados STARE.

Com a abordagem de Wang et al. [WYC+15] há o uso de filtro gaussiano e de adaptação de histograma como pre-processamento, em seguida se utiliza uma rede neural convolucional, sem as camadas densas, para extrair características dos dados e então há o uso de florestas randômicas (*random forests*) para o processo de classificação a qual não ocorre pixel-a-pixel pois é utilizado um processo de clusterização de pixeis, chamado de superpixel, aonde a vizinhança de um pixel classificado obtém o mesmo rótulo. Estas são as principais diferenças

do trabalho de Wang et al. da nossa proposta, pois nós utilizamos apenas a rede neural convolucional com as redes densas para classificação pixel-a-pixel, no entanto, obtivemos uma acurácia 0.03% pior no banco de dados DRIVE e 1.56% pior no banco de dados STARE. A arquitetura da rede neural convolucional utilizada por Wang et al. [WYC+15] pode ser verificada na tabela 4.22, note que além desta arquitetura há as florestas randômicas conectadas em cada camada.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding
Entrada	1x25x25			
Convolucional	12x22x22		4x4	0x0
Max-pooling	12x11x11	2x2		0x0
Convolucional	12x8x8		4x4	0x0
Max-pooling	12x4x4	2x2		0x0
Densa	100			
Densa	1			

Tabela 4.22: *Rede Convolucional Utilizada No trabalho de Wang et al. [WYC+15]*

Hong Tan et al. [TAB+17] utiliza uma única rede neural convolucional para segmentar veias, disco ótico e fóvea. Há o uso de pre-processamento para corrigir problemas de iluminação. A classificação é feita pixel-a-pixel, no entanto, a saída da rede neural é constituída de 4 neurônios com função de ativação softmax, enquanto em nossa abordagem utilizamos uma saída sigmoide. No entanto, também faz uma classificação pixel-a-pixel. Hong Tan et al. [TAB+17] propondo utilizar apenas um classificador para 3 objetivos, não consegue uma acurácia próxima ao estado da arte. Ao comparar com nosso melhor resultado, obtivemos ma acurácia 2.62% melhor no banco de dados DRIVE. A arquitetura da rede neural convolucional utilizada por Hong Tan et al. [TAB+17] pode ser verificada na tabela 4.23.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding
Entrada	1x33x33			
Convolucional	30x29x29		5x5	0x0
Max-pooling	30x14x14	2x2		0x0
Convolucional	45x10x10		5x5	0x0
Max-pooling	45x5x5	2x2		0x0
Densa	100			
Densa	4			

Tabela 4.23: *Rede Convolucional Utilizada No trabalho de Hong Tan et al. [TAB+17]*

Capítulo 5

Segmentação de Notas Musicais

Aplicamos a abordagem proposta em outro conjunto de dados chamado de STAFF, o qual tem como objetivo remover linhas de partituras musicais. Sistemas de reconhecimento musical ótico tem como objetivo reconhecer partituras musicais para converte-las em uma linguagem que pode ser trabalhada no computador, no entanto, para que isto seja feito de forma eficaz, deve ser removido imperfeições e outros artefatos, tal como as linhas verticais da partitura. Com estes sistemas é possível editar e renovar partituras, produzir arquivos de áudio a partir de partituras manuscritas e em grande quantidade. Além disso, há o interesse na área de ciência Forense [FDGL12] de verificar se uma partitura foi realmente produzida por alguém que se diga autor do mesmo.

O conjunto de dados STAFF é uma variação do banco de dados CVC-MUSCIMA [FDGL12], o qual contém 1000 partituras musicais de 20 músicas, sendo que as partituras foram escritas por 50 músicos escolhidos de maneira que o grupo fosse o mais heterogêneo possível. Em contrapartida, o conjunto STAFF teve imagens deformadas para simular situações que ocorrem em documentos, tal como, degradação natural do documento e particularidades de escritas do músico, resultando em 4000 imagens para o conjunto de treinamento e 1000 imagens para o conjunto de teste. O STAFF foi disponibilizado em uma competição do ICDAR 2013 [KFV+13, VKFJ13], e diversas abordagens foram propostas e portanto é um conjunto de dados que podemos usar para experimentar com a abordagem do nosso trabalho. A imagem 5.1 ilustra um exemplo do STAFF.

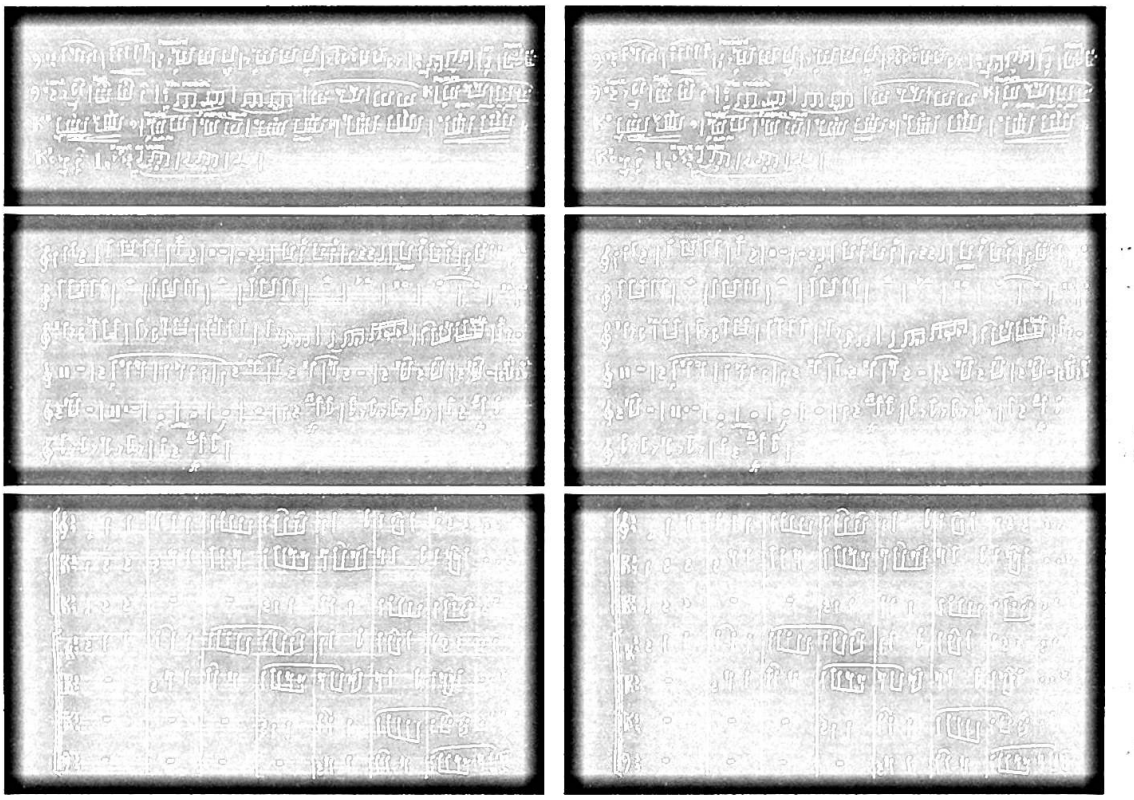


Figura 5.1: *3 Exemplos do conjunto de dados STAFF. Na primeira coluna há a imagem de entrada e na segunda coluna a saída esperada.*

5.1 Experimentos

O modelo utilizado nos W-operadores testados é uma rede neural convolucional com saída sigmoide, pois estamos interessados apenas na segmentação da imagem em duas partições: notas musicais e fundo. Este modelo de rede convolucional foi apresentado no capítulo 2.

Em todos os experimentos, os parâmetros fixados, foram :

- Tamanho do lote *(mini batch)* : 6000
- Número de épocas : 1000
- Paciência : 100
- Conjunto de validação é formado por 30% dos dados do conjunto de treinamento.
- Inicialização dos pesos do modelo usando algoritmo Glorot [GB10].

As taxas de aprendizado foram alteradas de acordo com o passo das épocas, de acordo com a tabela 5.1.

Tabela 5.1: *Ajuste de aprendizado*

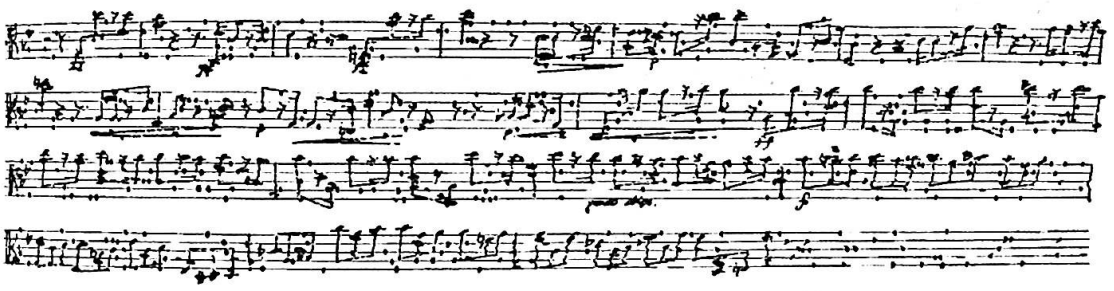
Época inicial	Época final	Taxa de aprendizado
1	49	0.15
50	99	0.10
100	299	0.01
300	799	0.001
800	1000	0.0001

Para facilitar a visualização dos resultados do operador de imagem, estas foram colorizadas da seguinte maneira:

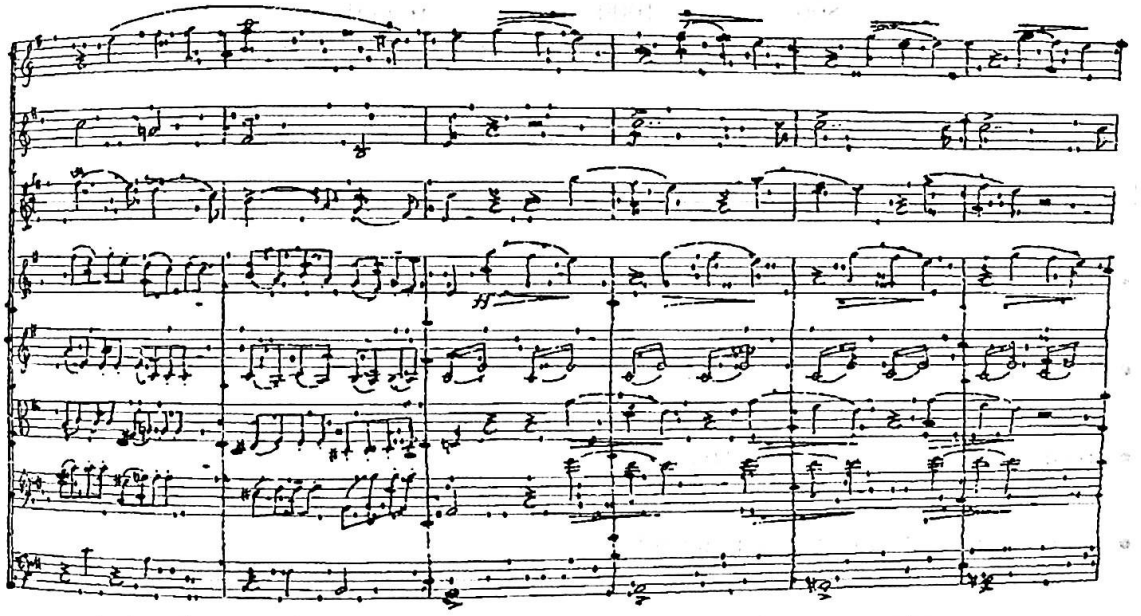
- Tom azul = Positivo verdadeiro (TP)
- Tom laranja = Falso positivo (FP)
- Tom vermelho = Falso negativo (FN)
- Tom verde = Negativo verdadeiro (TN)

Consideramos TP como as linhas de partitura corretamente identificadas, FP como os pixels de notas musicais classificados como linha de partitura, FN como pixels de linhas de partitura classificados como notas musicais e TN como pixels de notas musicais corretamente identificadas. Nesta configuração, modelos com altos valores de sensitividade conseguem remover mais linhas de partitura e modelos com altos valores de especificidade degradam menos as notas musicais. Portanto, a medida F1 é interpretada como um balanço entre a sensitividade e especificidade, sendo que, um modelo com valor 1 de F1 tem perfeita sensitividade e especificidade.

A figura 5.2 apresenta um exemplo de um resultado de uma experiencia com esta colorização descrita.



(a) *Imagens resultante de um operador de imagem, colorizada para ajudar na visualização*



(b) *Imagens resultante de um operador de imagem, colorizada para ajudar na visualização*

Figura 5.2: *Duas imagens resultantes de um operador de imagem, colorizadas para ajudar na visualização*

No experimento 1, fixamos o número de exemplos e variamos a dimensão da janela em $7x7$, $11x11$ e $21x21$.

No experimento 2, fixamos a dimensão da janela em $11x11$ e variamos a quantidade de imagens utilizadas para o treinamento do modelo.

No experimento 3, utilizamos a janela $21x21$ e 6 imagens e verificamos se obtemos o melhor resultado entre os experimentos 1 e 2.

No experimento 4, utilizamos o modelo convolucional utilizado no capítulo anterior, no

qual fizemos segmentação de veias da retina.

Nas subseções a seguir são descritos os experimentos realizados.

5.1.1 Experimento 1

Neste experimento, usamos um modelo de rede neural convolucional menos profunda do que a utilizada nos experimentos de segmentação de veias da retina 5. Com este modelo menos profundo, experimentamos com janela 7x7, 11x11 e 21x21 com 3 imagens de treinamento, sendo utilizado 1.570.460 exemplos de treinamento.

A arquitetura da rede tem 6 camadas, sendo 2 camadas convolucionais, 1 camada de max-pooling e 3 camadas densas. Os detalhes desta arquitetura estão descritas nas tabelas 5.3, 5.4 e 5.2 para cada dimensão de janela.

O modelo foi testado nas 1000 imagens de teste e a figura 5.5 apresenta alguns resultados para cada experimento nas diferentes dimensões de janela, sendo que, estes resultados foram colorizados de acordo com a explicação do início deste capítulo, a tabela 5.5 contém as medidas realizadas para este conjunto de teste.

As figuras 5.4 contém os gráficos da função de perda no conjunto de treinamento e do conjunto de validação para as épocas no treinamento do modelo.

O modelo se estabiliza próximo a época 200 e prossegue melhorando até próximo a época 400, para as diferentes janelas usadas.

O melhor resultado foi obtido usando a maior janela (21x21), aonde a acurácia foi de 96.71% e F1 de 95.60%. Note que entre as janelas 7x7 e 21x21 a diferença na especificidade e sensitividade mostram que o modelo com janela 21x21 degrada menos as notas musicais com pouca piora na sensitividade. Isto resulta na maior medida F1 entre estes experimentos.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x7x7				
Convolucional	20x9x9		3x3	2x2	Retificadora linear
Convolucional	20x11x11		3x3	2x2	Retificadora linear
Max-pooling	20x5x5	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

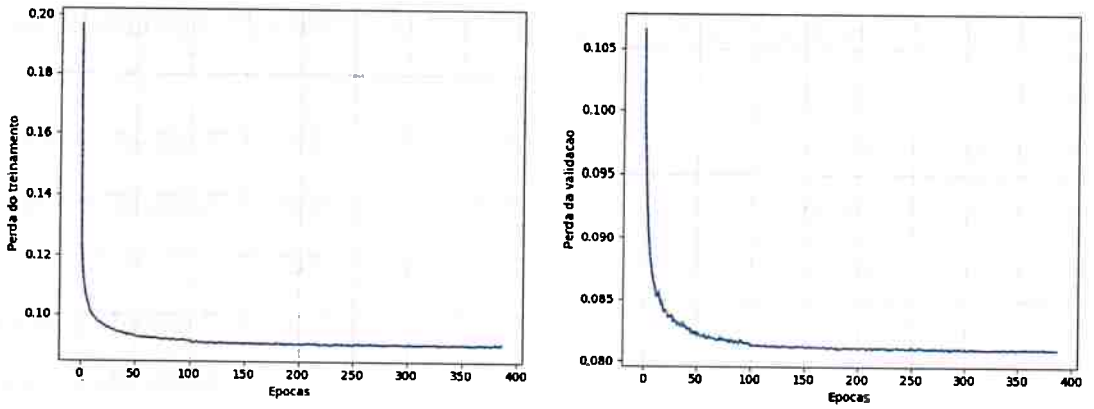
Tabela 5.2: *Exemplo de arquitetura de rede neural convolucional com janela 7x7*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x11x11				
Convolucional	20x13x13		3x3	2x2	Retificadora linear
Convolucional	20x15x15		3x3	2x2	Retificadora linear
Max-pooling	20x7x7	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

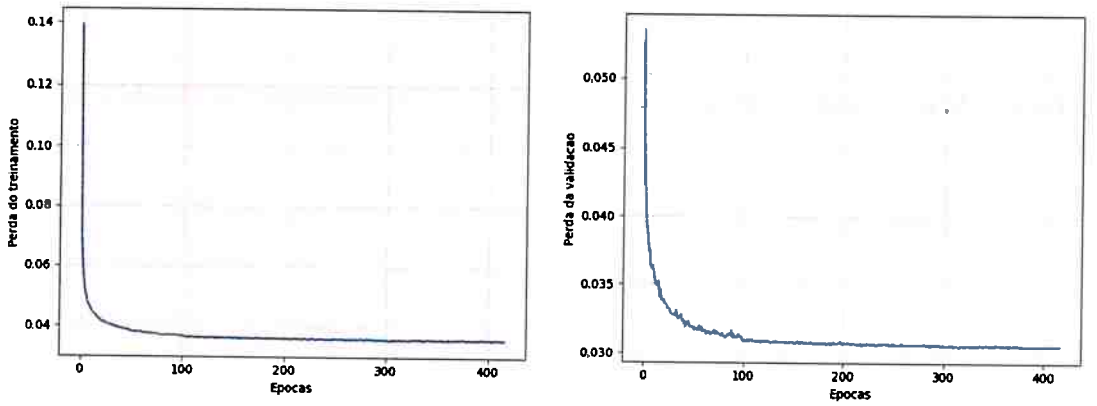
Tabela 5.3: *Exemplo de arquitetura de rede neural convolucional com janela 11x11*

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x21x21				
Convolucional	20x23x23		3x3	2x2	Retificadora lincar
Convolucional	20x25x25		3x3	2x2	Retificadora linear
Max-pooling	20x12x12	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

Tabela 5.4: *Exemplo de arquitetura de rede neural convolucional com janela 21x21*

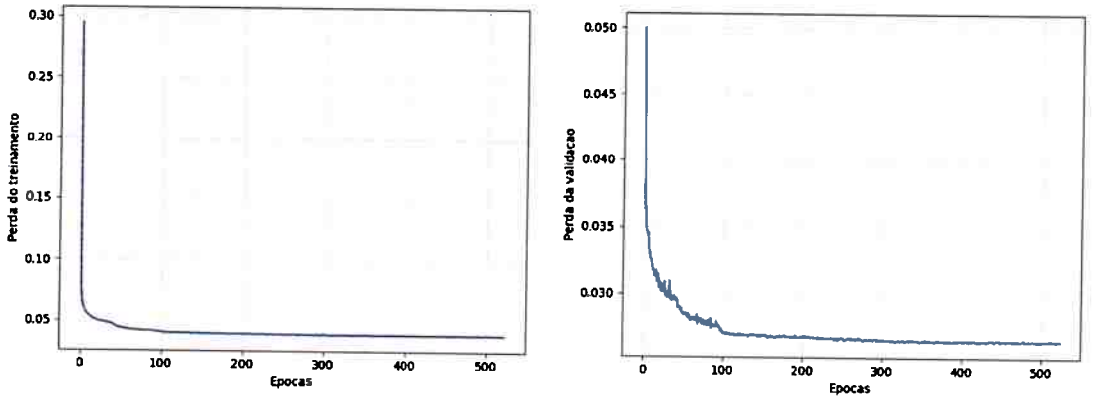


(a) *Perda no conjunto de treinamento e perda no conjunto de validação usando janela 7x7*



(b) *Perda no conjunto de treinamento e perda no conjunto de validação usando janela 11x11*

Figura 5.3: *Gráfico da função de perda no conjunto de treinamento e no conjunto de validação*

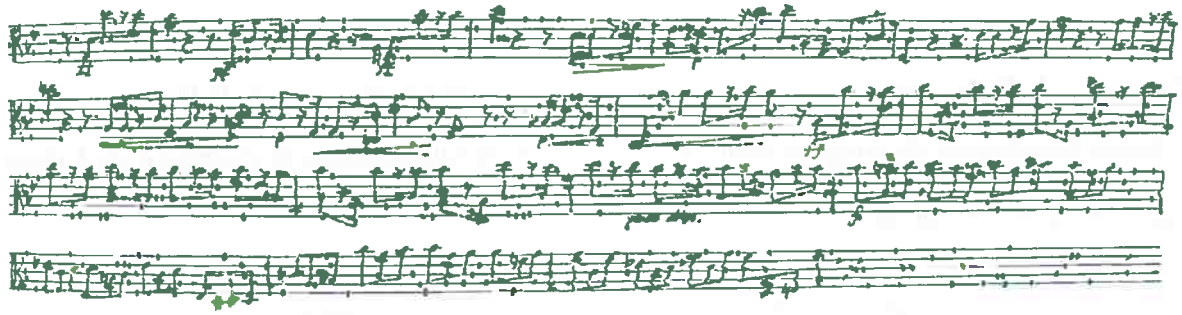


(a) *Perda no conjunto de treinamento e perda no conjunto de validação usando janela 21x21*

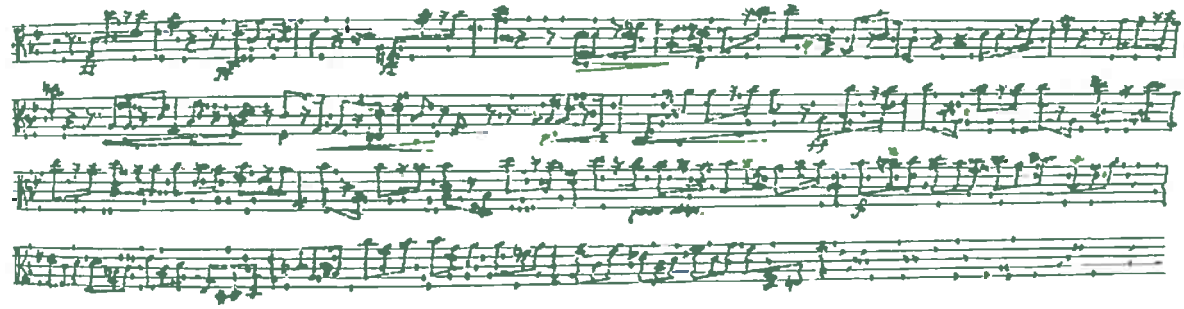
Figura 5.4: *Gráfico da função de perda no conjunto de treinamento e no conjunto de validação*

Dimensão Da Janela	Nº Imagens	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN	F1
7x7	3	0.9407	0.9538	0.9328	0.8953	234830614	381419778	27437601	11357741	0.9236
11x11	3	0.9602	0.9404	0.9722	0.9532	231516860	397493551	11363828	14671495	0.9467
21x21	3	0.9671	0.9497	0.9776	0.9624	233818472	399722575	9134804	12369883	0.9560

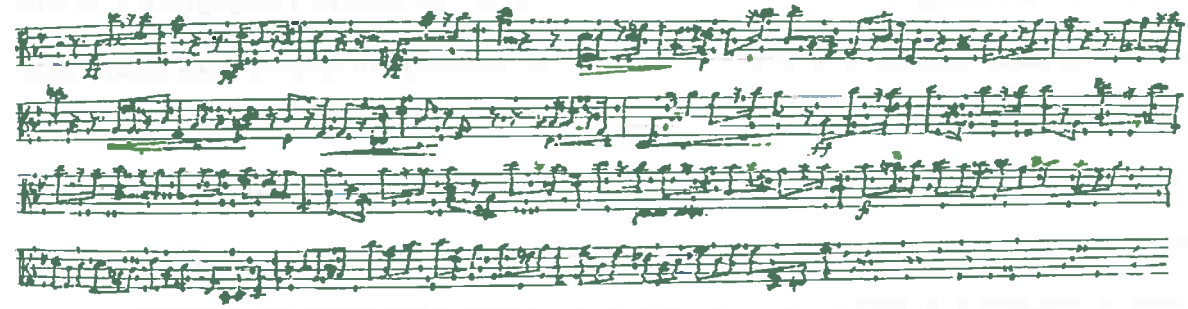
Tabela 5.5: *Resultados dos experimentos modificando o tamanho da janela do W-Operador*



(a) *Imagem de teste resultante ao usar janela 7x7*



(b) *Imagem de teste resultante ao usar janela 11x11*



(c) *Imagem de teste resultante ao usar janela 21x21*

Figura 5.5: *Uma imagem de teste resultante do experimento 1 para cada configuração de janela testada*

5.1.2 Experimento 2

Neste experimento, usamos um modelo de rede neural convolucional do experimento anterior 5.1.1. Experimentamos com janela 11x11 e variamos as imagens de treinamento

Estas quantidade de imagens resultam em diferentes quantidades de exemplos de treinamento, assim como na tabela 5.6.

Quantidade de imagens	Número de exemplos
1	431.673
3	1.570.460
6	3.152.022

Tabela 5.6: *Quantidade de exemplos para diferentes quantidades de imagens de treinamento usando janela 11x11.*

Os detalhes da arquitetura desta rede esta descrita nas tabela 5.7.

Neste experimento, o modelo foi novamente testado nas 1000 imagens de teste e a figura 5.7 apresenta alguns resultados para cada experimento nas diferentes dimensões de janela, sendo que, estes resultados foram colorizados de acordo com a explicação do início deste capítulo, A tabela 5.8 contém as medidas realizadas para este conjunto de teste.

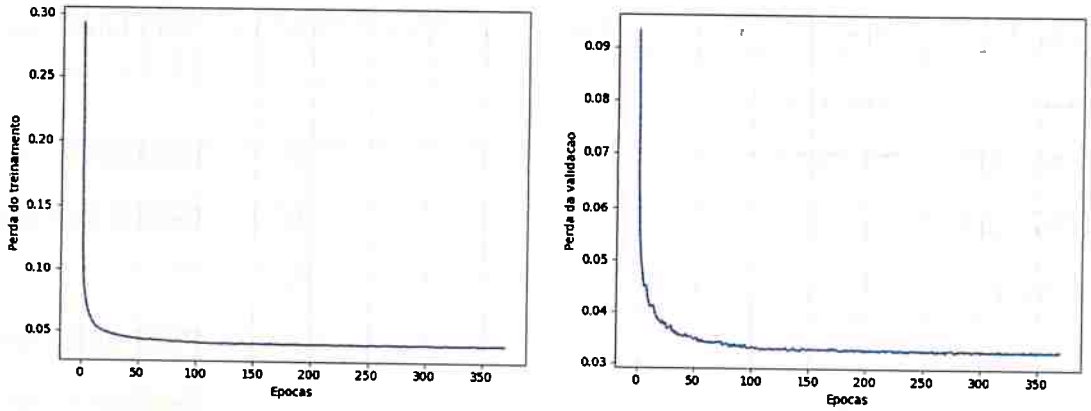
A figura 5.6 contém os gráficos da função de perda no conjunto de treinamento e do conjunto de validação para as épocas no treinamento do modelo para as diferentes quantidades de imagens.

O modelo se estabiliza próximo a época 200 para as diferentes imagens utilizadas. No entanto, devido a técnica do early-stopping, continua até aproximadamente a época 400. O modelo se comporta de forma parecida para as diferentes quantidades de imagens.

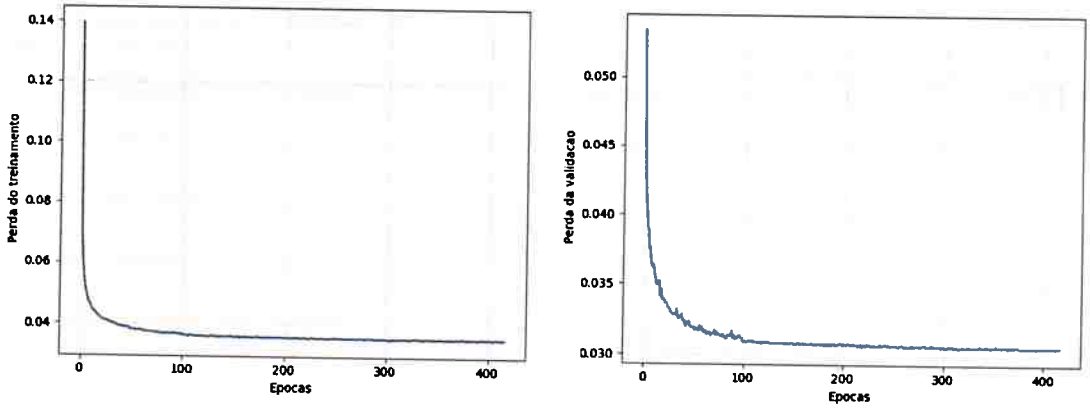
O modelo apresenta a melhor especificidade quanto utilizada apenas 1 imagem. No entanto, a melhor acurácia, sensitividade e F1 ocorre quando se utiliza 6 imagens de treinamento. Ao comparar os experimentos de 3 e 6 imagens de treinamento, verificamos que a acurácia, a sensitividade, a especificidade, o PPV e a medida F1 melhoram consideravelmente, portanto, treinar um modelo com janela grande e com 6 imagens de treinamento pode gerar resultados interessantes, assim como veremos no próximo experimento.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x11x11				
Convolucional	20x13x13		3x3	2x2	Retificadora linear
Convolucional	20x15x15		3x3	2x2	Retificadora linear
Max-pooling	20x7x7	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

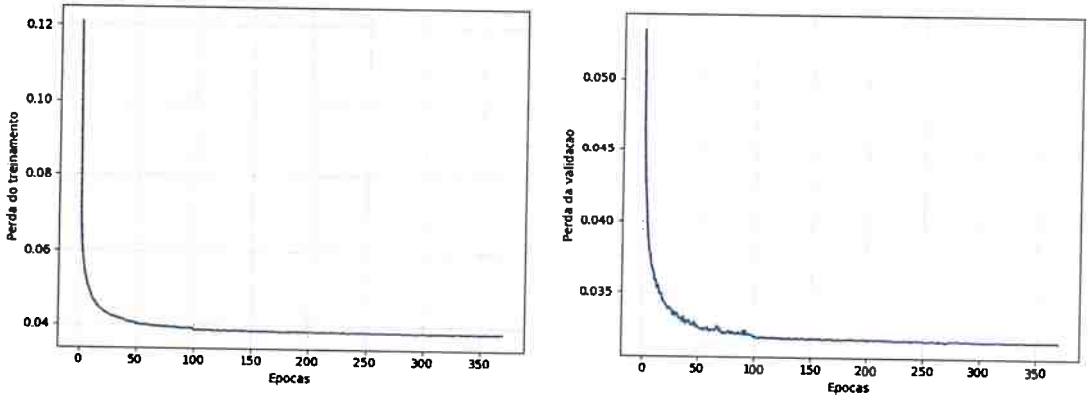
Tabela 5.7: *Exemplo de arquitetura de rede neural convolucional com janela 11x11*



(a) *Perda no conjunto de treinamento e perda no conjunto de validação usando 1 imagem de treinamento*



(b) *Perda no conjunto de treinamento e perda no conjunto de validação usando 3 imagens de treinamento*

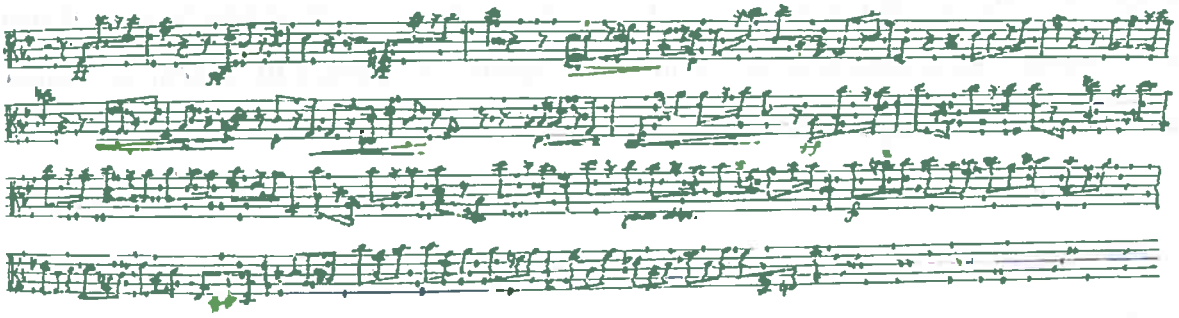


(c) *Perda no conjunto de treinamento e perda no conjunto de validação usando 6 imagens de treinamento*

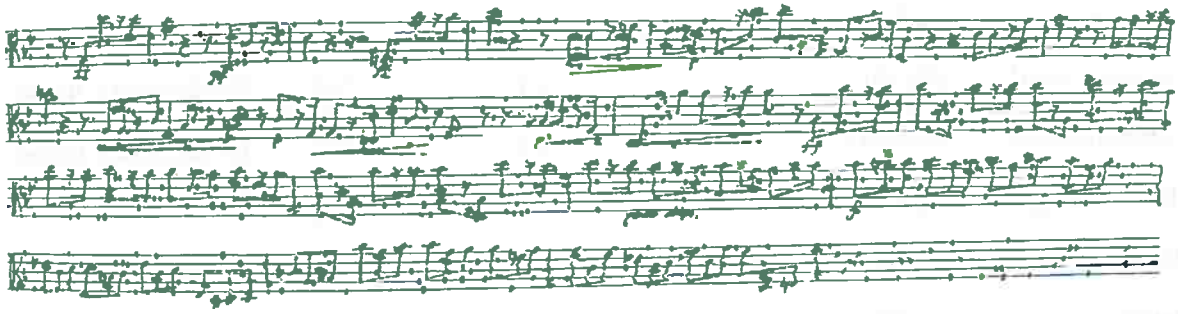
Figura 5.6: *Gráfico da função de perda no conjunto de treinamento e no conjunto de validação*

Dimensão Da Janela	Nº Imagens	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN	F1
11x11	1	0.9367	0.8500	0.9889	0.9788	209266525	404341852	4515527	36921830	0.9099
11x11	3	0.9602	0.9404	0.9722	0.9532	231516860	397493551	11363828	14671495	0.9467
11x11	6	0.9622	0.9411	0.9749	0.9577	231688838	398627793	10229586	14499517	0.9493

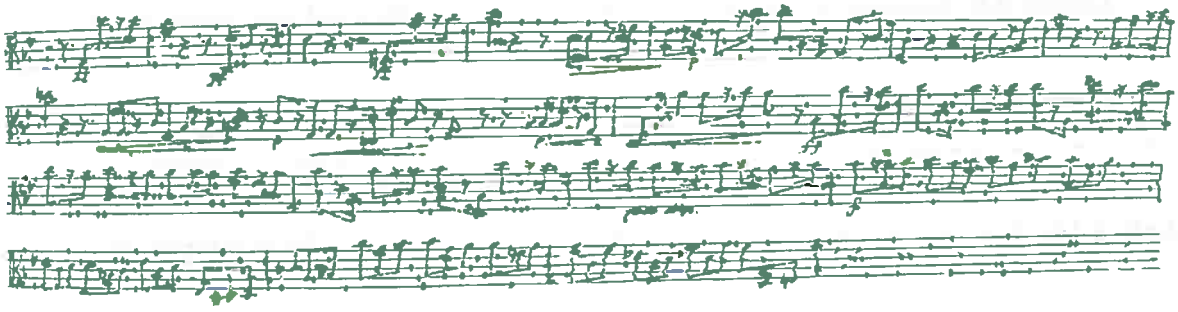
Tabela 5.8: *Resultados dos experimentos modificando o tamanho da janela do W-Operador*



(a) *Imagem de teste resultante ao usar 1 imagem de treinamento*



(b) *Imagem de teste resultante ao usar 3 imagens de treinamento*



(c) *Imagem de teste resultante ao usar 6 imagens de treinamento*

Figura 5.7: *Uma imagem de teste resultante do experimento 2 para cada quantidade de imagens de treinamento utilizada*

Neste experimento, o modelo foi novamente testado nas 1000 imagens de teste e a figura 5.8 apresenta alguns resultados do experimento, sendo que, estes resultados foram colorizados de acordo com a explicação do início deste capítulo, A tabela 5.10 contém as medidas realizadas para este conjunto de teste.

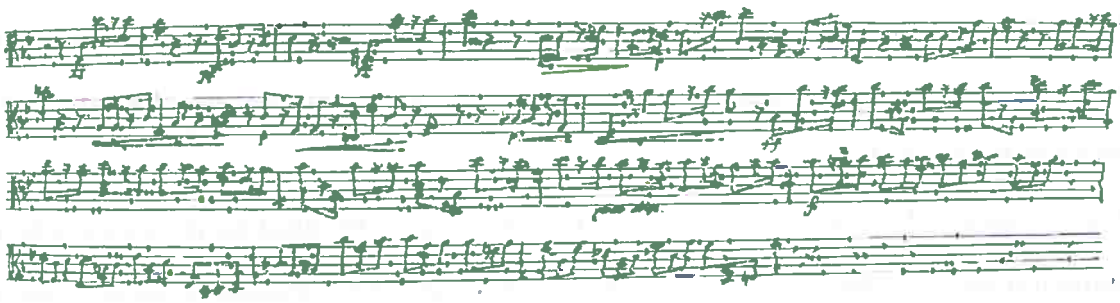
Ao compararmos os resultados obtidos com o experimento 1, observando os resultados 5.5, há uma pequena piora da sensitividade e em contrapartida uma melhora na especificidade. Ainda assim, o uso da janela 21x21 e 6 imagens de treinamento resultou na melhor medida de acurácia e F1 de todos os experimentos realizados.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Não-Linearidade
Entrada	1x11x11				
Convolucional	20x13x13		3x3	2x2	Retificadora linear
Convolucional	20x15x15		3x3	2x2	Retificadora linear
Max-pooling	20x7x7	2x2		0x0	
Densa	64				Retificadora linear
Densa	32				Retificadora linear
Densa	1				Sigmoide

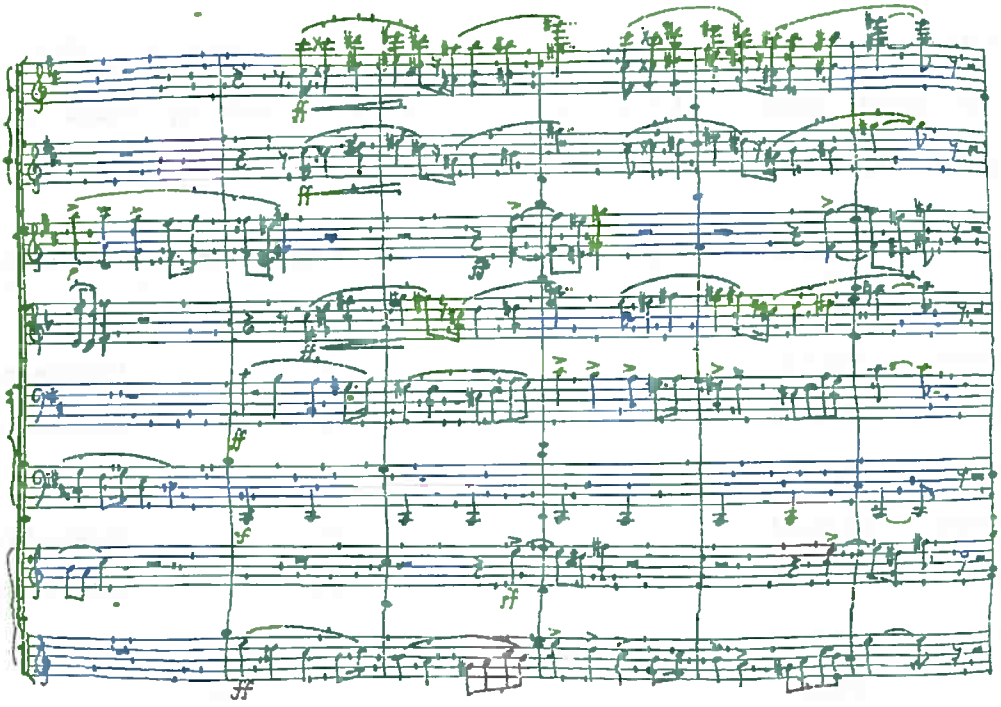
Tabela 5.9: *Exemplo de arquitetura de rede neural convolucional com janela 11x11*

Dimensão Da Janela	Nº Imagens	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN	F1
21x21	6	0.9678	0.9480	0.9797	0.9657	233391478	400576503	8280876	12796877	0.9567

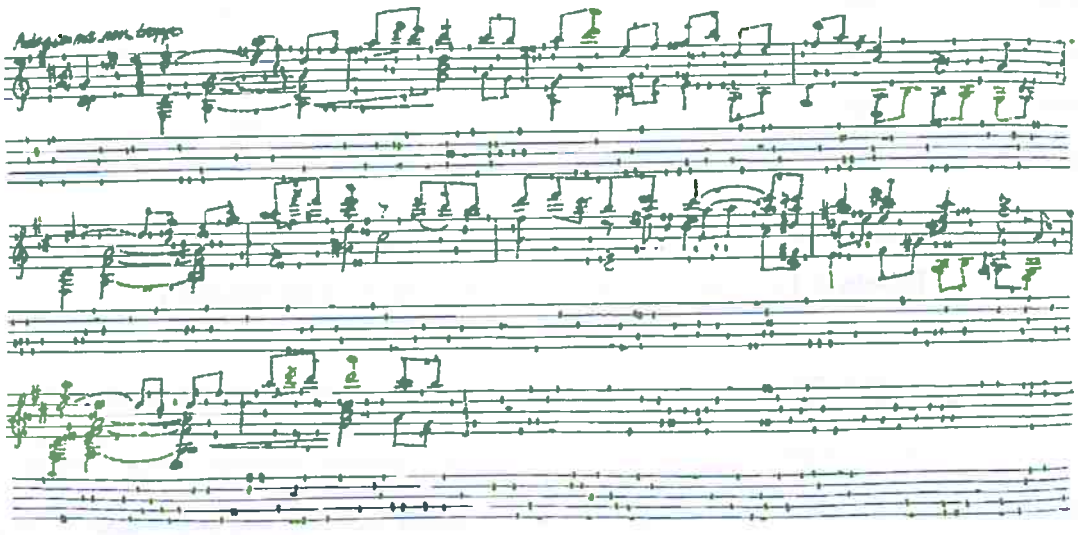
Tabela 5.10: *Resultado do experimento 3 usando uma janela grande e alta quantidade de imagens*



(a) *Imagem de teste resultante do experimento 3*



(b) *Imagem de teste resultante do experimento 3*



(c) *Imagem de teste resultante do experimento 3*

Figura 5.8: *Três imagens de teste resultantes do experimento 3*

5.1.4 Experimento 4

Neste experimento, testamos com o modelo de rede convolucional utilizado nos experimentos com segmentação de veias da retina 4. Os detalhes da arquitetura desta rede esta descrita nas tabela 5.11.

Neste experimento, o modelo foi novamente testado nas 1000 imagens de teste e a figura 5.9 apresenta alguns resultados do experimento, sendo que, estes resultados foram colorizados de acordo com a explicação do início deste capítulo, a tabela 5.12 contém as medidas realizadas para este conjunto de teste.

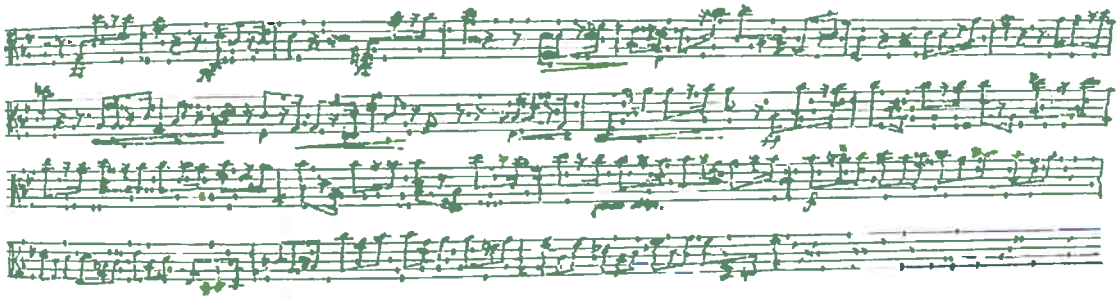
Verificamos que o uso de um modelo mais profundo não pressupõe melhores resultados. O experimento anterior 5.1.3 apresenta um melhor resultado em relação a acurácia, especificidade, sensitividade, PPV e F1.

Camada (Layer)	Dimensão	Pooling	Tamanho do Filtro	Padding	Dropout	Não-Linearidade
Entrada	1x21x21					
Convolucional	20x23x23		3x3	2x2		Retificadora linear
Convolucional	20x25x25		3x3	2x2		Retificadora linear
Max-pooling	20x12x12	2x2		0x0		
Convolucional	40x14x14		3x3	2x2		Retificadora linear
Convolucional	40x16x16		3x3	2x2		Retificadora linear
Max-pooling	40x8x8	2x2		0x0		
Dropout	40x8x8				0.5	
Densa	64					Retificadora linear
Dropout	64				0.5	
Densa	32					Retificadora linear
Dropout	32				0.5	
Densa	1					Sigmoide

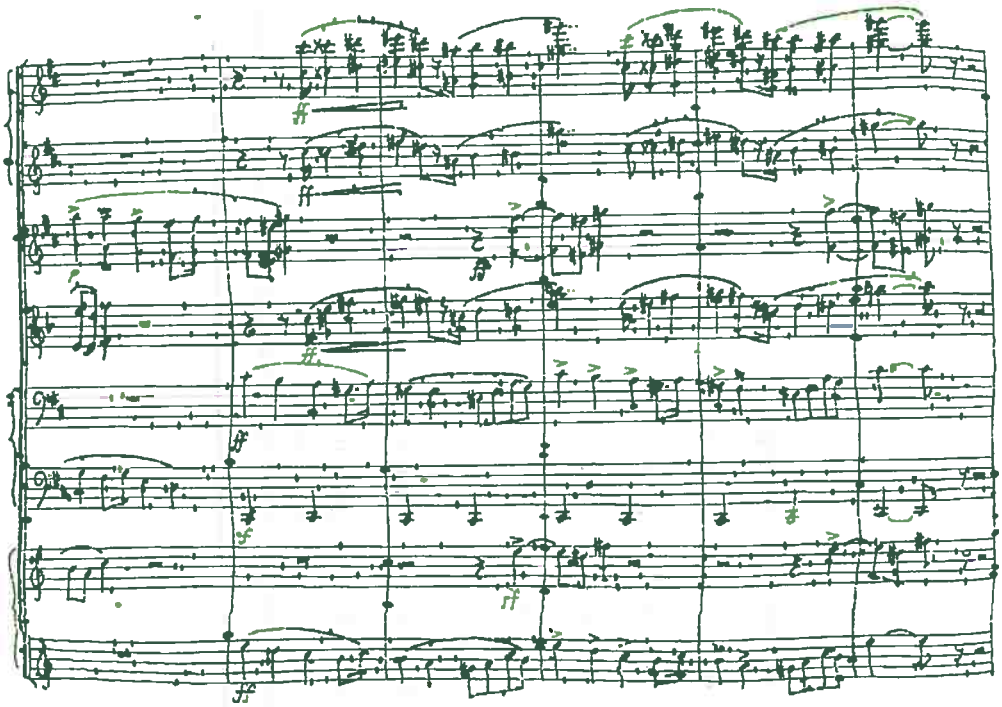
Tabela 5.11: *Rede Convolucional Utilizada No Experimento 4*

Dimensão Da Janela	Nº Imagens	Acurácia	Sensitividade	Especificidade	PPV	TP	TN	FP	FN	F1
21x21	6	0.9645	0.9436	0.9772	0.9614	232308892	399540549	9316830	13879463	0.9524

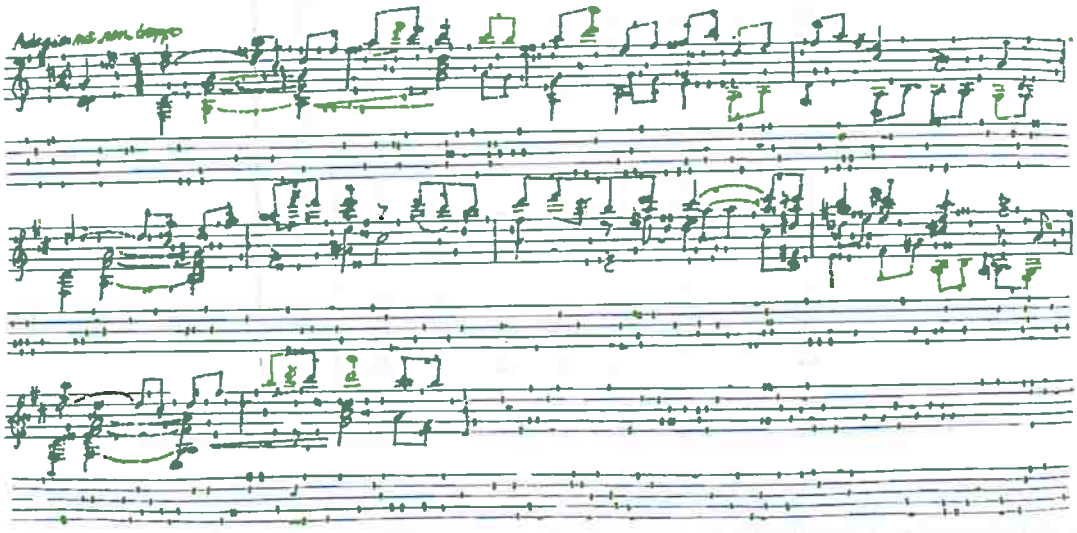
Tabela 5.12: *Resultado do experimento 4 usando uma janela grande e alta quantidade de imagens*



(a) *Imagem de teste resultante do experimento 4*



(b) *Imagem de teste resultante do experimento 4*



(c) *Imagem de teste resultante do experimento 4*

Figura 5.9: *Três imagens de teste resultantes do experimento 4*

5.2 Discussão Experimentos

Na tabela 5.13 verificamos as medidas de todos os experimentos deste trabalho, realizados com o banco de dados STAFF.

Experimento	Dimensão Da Janela	Nº Imagens	Acurácia	Sensitividade	Especificidade	PPV	F1
1	7x7	3	0.9407	0.9538	0.9328	0.8953	0.9236
1	11x11	3	0.9602	0.9404	0.9722	0.9532	0.9467
1	21x21	3	0.9671	0.9497	0.9776	0.9624	0.9560
2	11x11	1	0.9367	0.8500	0.9889	0.9788	0.9099
2	11x11	3	0.9602	0.9404	0.9722	0.9532	0.9467
2	11x11	6	0.9622	0.9411	0.9749	0.9577	0.9493
3	21x21	6	0.9678	0.9480	0.9797	0.9657	0.9567
4	21x21	6	0.9645	0.9436	0.9772	0.9614	0.9524

Tabela 5.13: *Compilação dos resultados de todos os experimentos*

No experimento 1 analisamos os resultados ao modificar o tamanho da janela do W-

operador, e assim como no experimento 4 (4.1.4) do capítulo 4, há também a melhora das medidas ao aumentar o tamanho da janela. Note que a diferença não é tão brusca entre a janela 11x11 e 21x21, no entanto, pode ser verificada mais facilmente ao se comparar a janela 7x7 com 21x21. Ainda assim, vemos que o melhor resultado ocorreu com a maior janela, ou seja, 21x21.

No experimento 2, confirmamos nossas conclusões vistas também no capítulo 4 no experimento 5 (4.1.5), pois novamente houve a melhora dos resultados ao aumentar a quantidade de imagens de treinamento.

No experimento 3, conseguimos o melhor resultado de todos os experimentos ao utilizar a janela de dimensão 21x21 e 6 imagens de treinamento.

No experimento 4, repetimos o experimento 3 porém utilizando o modelo de rede neural convolucional do capítulo 4. No entanto, obtemos um resultado pior aonde nota-se a diminuição da acurácia, sensitividade especificidade, PPV e F1.

Outro ponto importante é que as imagens do banco de dados STAFF geraram muito mais exemplos do que as imagens do banco de dados DRIVE, por exemplo, 3 imagens do STAFF geraram 1.570.460 exemplos contra 681.473 do DRIVE.

5.2.1 Comparação com outros trabalhos com o banco de dados STAFF

Método	Tipo de Aprendizado	Descrição	Ano	Sensitividade	Especificidade	Acurácia	F1
Rede Neural Convolucional [JAH17]	Supervisionado	Aprendizado de operador de imagem com deep learning	2017	0.9572	0.9898	0.9796	N/D
LRDE [MHH17, Gér14, VKF313]	N/A	Uso de operadores morfológicos	2014	0.9402	0.9884	0.9703	0.9597
FS-MI [MHH17]	Supervisionado	Estimação de operadores de imagens	2017	0.9448	0.9846	0.9696	N/D
Andre Lopes. 5.1.4	**Supervisionado**	**Aprendizado de operador de imagem com deep learning**	**2017**	**0.9480**	**0.9797**	**0.9678**	**0.9567**
Skeleton [MHH17, DDPF08]	N/A	*Line Tracking*	2014	0.8697	0.9903	0.9450	0.9224
LTC [MHH17, BB97, MB91, DDPF08]	N/A	*Line Tracking*	2008	0.6776	0.9952	0.8758	0.8040

Tabela 5.14: *Resultado de diferentes métodos de segmentação no banco de dados STAFF*

A tabela 4.20 contém resultados da estado da arte e próximos dele com informações resumidas dos artigos publicados. Ao verificar os resultados desta tabela, verificamos que :

- A média das acurácias é : 94.80 %
- A média das sensitividades disponíveis é : 87.79 %
- A média das especificidades disponíveis é : 98.96 %
- A média das medidas F1's disponíveis é : 91.07 %

Este trabalho ficou acima da média em relação a acurácia, sensitividade e F1, porém abaixo na questão de especificidade. A figura 5.10 ilustra as acurácias da tabela 5.14 em forma de gráfico em barras, mostrando a diferença de outros autores em relação a este trabalho.

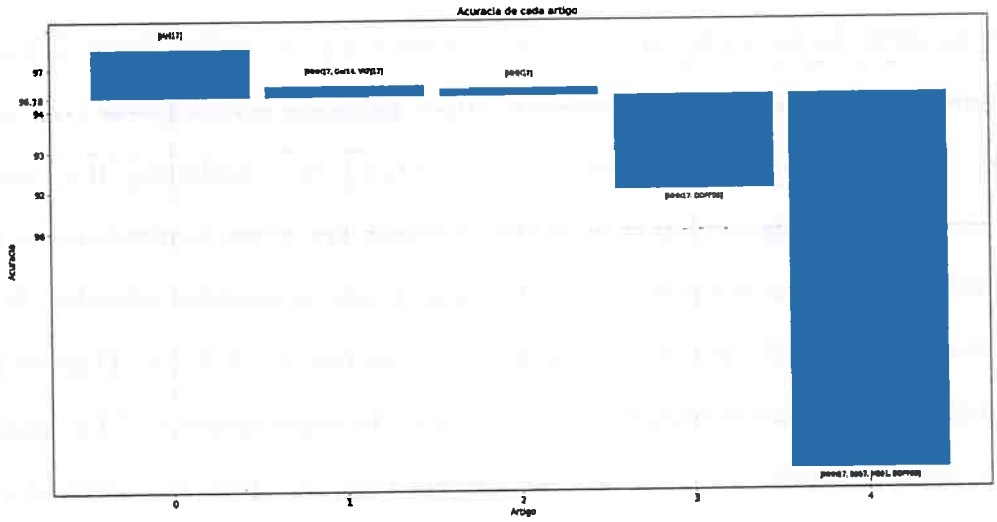


Figura 5.10: *Gráfico em barras comparando a acurácias dos outros trabalhos da tabela 5.14 com o resultado deste trabalho 5.1.3. Note que o valor de* 97.43 *no eixo dos valores de acurácia, se refere a melhor acurácia deste trabalho.*

Observando os resultados de diversos autores na tabela 5.14 notamos que trabalho de redes neurais convolucionais com W-Operadores [JAH17] consegue os melhores resultados. Vamos verificar a abordagem de deste e dos outros trabalhos e compará-los com a nossa abordagem.

Julca-Aguilar et al. [JAH17] na sua abordagem de segmentação de notas musicais utiliza uma rede neural convolucional com arquitetura construída a partir de uma busca em grade em um conjunto de validação. A arquitetura é limitada em 2 blocos formados por 1 camada

de convolução seguida por 1 camada de max-pooling, aonde cada camada de convolução tem 32 filtros com tamanho 5x5. È utilizado a saída softmax e um mini-batch de tamanho 50. Na busca em grade, cada modelo é treinado durante 50 épocas e é variada a taxa de aprendizado, os valores de drop-out, e o tamanho da janela do W-Operador. A melhor arquitetura escolhida neste trabalho de Julca-Aguilar et al. faz a utilização de uma janela de dimensão 19x19. No mesmo trabalho, há a discussão de que filtros de tamanho 3x3 obtiveram melhor acurácia, porém não foram utilizados no trabalho final pois requerem maior estudo. Em suma, [JAH17] utiliza a mesma técnica do nosso trabalho.

Entre as diferenças perceptíveis do modelo escolhido por Julca-Aguilar et al. e o nosso utilizado nos experimentos 1, 2 e 3, nota-se que ele utiliza uma camada de max-pooling após uma única camada de convolução e há o uso da função de ativação softmax na saída do modelo ao invés da sigmoide. Apesar disso, notamos que ele utiliza filtros 5x5 ao invés de 3x3. Importante notar, que este modelo é pouco profundo ao comparar com modelos conhecidos, como por exemplo, VGGnet [SZ14], AlexNet [NH10, LBBH98], R e GoogleNet [SLJ+15], no entanto, Julca-Aguilar et al. [JAH17] atingiu um ótimo resultado e obviamente o modelo utilizado exige menos poder computacional do que os modelos referidos. Note que obtemos uma acurácia 1.18% pior ao comparar com o experimento 3 (5.1.3). O nosso modelo convolucional é semelhante em relação a profundidade. No experimento 4 (5.1.4) utilizamos um modelo mais profundo, no entanto, obtemos um resultado pior ao comparar com os resultados de acurácia de Julca-Aguilar et al [JAH17] (1.51%) e com o F1 do experimento 3 (0.33%).

No método FS-MI, o qual é apresentado no trabalho de Montagner [MHH17, Mon17b] há o uso de uma técnica que utiliza o conceito de W-operador. O FS-MI faz uso de métodos de escolhas de características para gerar janelas (W) de primeiro-nível usando o algoritmo *relief* [KR92, KRSP96] e determinação automática de janela. Ao comparar com nosso trabalho, verificamos que o método FS-MI obtém uma melhor acurácia (0.18%), especificidade (0.49%) e pior sensitividade (0.32%).

Os métodos LTC [MHH17, BB97, MB91, DDPF08] e Skeleton [MHH17, DDPF08] utilizam a técnica de *Line Tracking*, a qual envolve localizar as linhas da partitura por algum método específico e em seguida, cada linha é rastreada para verificar se algum pixel da

vizinhança precisa ser removido baseado em um critério específico.

No método de LTC [MHH17, BB97, MB91, DDPF08] é detectado o esqueleto de cada nota musical e em seguida é removido pixeis ao redor do mesmo, baseado em um critério que indica se o pixel em questão pertence ou não ao símbolo musical. Com o método Skeleton [MHH17, DDPF08], há o uso da informação do esqueleto para a análise e então a segmentação.

O método proposto em nosso trabalho com rede convolucional supera tanto o método LTC, quanto o método Skeleton nas medidas F1, acurácia e sensitividade, porém, apresenta valores piores em relação a especificidade. Portanto, podemos afirmar que na nossa abordagem, removemos melhor as linhas de partitura porém degradamos mais as notas musicais.

O método LRDE [MHH17, Gér14, VKFJ13] faz apenas uso de operadores morfológicos simples em um processo em cadeia, para remover as linhas das partituras. Ainda assim, consegue uma acurácia 0.25% e especificidade 0.87% melhor que nossa abordagem, no entanto, obtemos um valor superior em relação a sensitividade (0.78%).

Em suma, verificamos que a principal dificuldade desta tarefa envolve em configurar a arquitetura e os parâmetros da rede convolucional para que esta obtenha ótimos resultados. Apesar de este objetivo ser possível, envolve alto esforço computacional, no entanto, assim como notado na tabela 5.14 e no trabalho de Julca-Aguilar et al. [JAH17], os resultados podem ser melhores que o estado da arte. È possível concluir que o uso de redes neurais convolucionais proporciona uma nova maneira de se usar W-operadores, com a vantagem de se obter melhores resultados e com a desvantagem da necessidade de um entendimento avançado da estrutura e a necessidade de se fazer algum tipo de busca ou construção automatizada da arquitetura. O entendimento dos parâmetros da estrutura de uma rede neural convolucional não é trivial, pois a mínima mudança em um parâmetro de uma certa camada ira mudar o aprendizado do modelo.

Capítulo 6

Conclusão

Neste capítulo fazemos as considerações finais dos resultados deste trabalho e propomos ideias para trabalhos futuros.

Neste trabalho, mostramos que o uso de projeto automático de W-operadores usando redes neurais convolucionais traz diversos benefícios e resultados próximos ao estado da arte.

Entre os benefícios da abordagem vista neste trabalho, podemos destacar o fato de que as camadas convolucionais fazem uma extração automática de características *(features)*, o que aumenta muito a eficácia deste modelo utilizado ao comparar com as redes neurais artificiais.

Em relação aos resultados obtidos, notamos que para problemas complexos, tal como a segmentação de retina, o uso da rede convolucional com W-Operador obtém resultados excelentes, e usando um algoritmo de pré-processamento, há resultados ainda melhores. Na arquitetura do modelo convolucional para este problema, houve a necessidade de um alto número de filtros e camadas convolucionais, no entanto, ainda é um modelo que pode ser treinado e usado com a tecnologia atual. Nos experimentos para segmentação de notas musicais, notamos que pode-se obter resultados razoáveis sem pré-processamento e com uma rede convolucional simples, ou seja, com poucas camadas convolucionais e max-pooling. Portanto, verifica-se que outros pesquisadores podem se interessar por este tipo de abordagem ao estudar problemas de segmentação.

Portanto, a principal contribuição deste trabalho é o estudo e uso dessas redes convolucionais para aprendizagem de operadores de imagem.

6.1 Sugestões para Pesquisas Futuras

Com a evidencia do modelo das redes neurais desde 2006, com o trabalho de Hinton, G. [HOT06], o qual apresentou uma nova maneira de realizar o treinamento de redes neurais usando GPGPU's, este modelo recebeu diversas contribuições de diversos pesquisadores e inclusive de empresas, tal como a NVIDIA, com seu pacote CUDA [Nvi10] para programação paralela em GPGPU'S.

Como resultado das contribuições de pesquisas e da evidência em que este modelo esta na comunidade, atualmente pode-se construir e arquitetar redes neurais de inúmeras maneiras. Podemos ilustrar esta situação ao verificar novas funções de ativações, tal como a função de Unidade Linear Exponencial Escalonada *(SELU)* [KUMH17] a qual promete criar modelos com uma regularização de maior eficácia. Outro exemplo, é o uso de diferentes camadas de convolução, tal como a camada de convolução transposta [DV16] ou camada de convolução dilatada [YK15].

Portanto, como uma sugestão para futuras pesquisas, sugerimos usar diferentes modelos de redes convolucionais, observando estudos de estado da arte nos quais há inovações, assim como o exemplo da função *(SELU)*, e assim, estudar o uso estas inovações com os W-operadores.

Outra sugestão é o algoritmo XGBOOST [CG16], o qual foi utilizado como solução de diversos problemas [CH15, CS9, MTD, MK16], e portanto, é um algoritmo candidato a pesquisas com W-operador. Além disso, pode ser utilizado em conjunto com redes neurais convolucionais, usando as camadas de convolução como extrator de características e XGBOOST como classificador.

Apêndice A

Apêndice

```
1   from lasagne.nonlinearities import rectify, softmax, sigmoid
2   import lasagne
3   import theano
4   import theano.tensor as T
5   from trios.feature_extractors import RAWFeatureExtractor
6   from sklearn.model_selection import train_test_split
7   import numpy as np
8
9   #Definimos o tamanho do mini batch
10  batch_size = 6000
11
12
13  #Garantimos a reprodutibilidade
14  random_state = 8000
15  lasagne.random.set_rng(np.random.RandomState(random_state))
16
17  #Criamos a janela do w-operador
18  window = np.ones(shape=(11, 11), dtype=np.uint8)
19
20  # Carregamos os dados de treinamento
21  imageset = trios.Imageset.read('images/train.set')
22
23  # Extraimos o padrao do conjunto de treinamento
24  X, Y = RAWFeatureExtractor(window).extract_dataset(imageset, True)
25
26  # Carregamos os dados de teste
27  imageset = trios.Imageset.read('images/test.set')
28
29  # Extraimos o padrao do conjunto de teste
30  X_teste, Y_teste = RAWFeatureExtractor(window).extract_dataset(
        imageset, True)
31
32
33  #Separamos os dados de treinamento e validacao
34  X, Y, X_val, Y_val = train_test_split(X, Y, test_size=0.30)
35
36
37  # Criamos variaveis do Theano para entrada e saida com minibatches.
38  input_var = T.tensor4('inputs')
```

```
39      target_var = T.fmatrix('targets')
40
41      # Criamos uma pequena rede convolucional
42      network = lasagne.layers.InputLayer((None, 1, window.shape[0], window.
            shape[1]), input_var)
43      network = lasagne.layers.Conv2DLayer(network, num_filters=64,
            filter_size=(3, 3),nonlinearity=rectify, W=lasagne.init.
            GlorotUniform())
44      network = lasagne.layers.Conv2DLayer(network, num_filters=32,
            filter_size=(3, 3),nonlinearity=rectify, W=lasagne.init.
            GlorotUniform())
45      network = lasagne.layers.Pool2DLayer(network, pool_size=(2, 2))
46      network = lasagne.layers.DenseLayer(lasagne.layers.dropout(network,
            0.5), num_units=128, nonlinearity=rectify, W=lasagne.init.
            GlorotUniform())
47      network = lasagne.layers.DenseLayer(lasagne.layers.dropout(network,
            0.5), num_units=1,  nonlinearity=sigmoid)
48
49      # Criamos a funcao de perda
50      prediction = lasagne.layers.get_output(network)
51      loss = lasagne.objectives.binary_crossentropy(predictions=lasagne.
            layers.get_output(network, deterministic=False), targets=target_var
            )
52      loss = loss.mean()
53
54  test_acc = lasagne.objectives.binary_accuracy(predictions=lasagne.layers
        .get_output(network, deterministic=True), targets=target_var).mean()
55  test_loss = lasagne.objectives.binary_crossentropy(predictions=lasagne.
        layers.get_output(network, deterministic=True), targets=target_var).
        mean()
56
57      # Criamos a funcao de treinamento
58      params = lasagne.layers.get_all_params(network, trainable=True)
59      updates = lasagne.updates.nesterov_momentum(loss_or_grads=
            loss_or_grads, params=params, learning_rate=0.01, momentum=0.9)
60
61      # Compilamos a funcao de treinamento e validacao
62      train_fn = theano.function([input_var, target_var], loss, updates=
            updates)
63  val_fn = theano.function([input_var, target_var], [test_loss, test_acc])
64
65      # Ajustamos as dimensoes dos conjuntos de dados.
66      input_layer = lasagne.layers.get_all_layers(self.network)[0]
67
68      # Reshape do conjunto de treinamento
69      X = X.reshape(-1, input_layer.shape[1], input_layer.shape[2],
            input_layer.shape[3])
70      Y = Y.reshape(-1, 1)
71
72      # Reshape do conjunto de validacao
73      X_val = X_val.reshape(-1, input_layer.shape[1], input_layer.shape[2],
            input_layer.shape[3])
74      Y_val = Y_val.reshape(-1, 1)
75
76      #Reshape do conjunto de teste
77      X_teste = X_teste.reshape(-1, input_layer.shape[1], input_layer.shape
            [2], input_layer.shape[3])
78      Y_teste = Y_teste.reshape(-1, 1)
79
```

```
80      # Loop de treinamento da rede neural
81      for epoch in range(100):
82            # A cada epoca passamos por todos os exemplos do conjunto de
                  treinamento:
83              train_err = 0
84              train_batches = 0
85              start_time = time.time()
86              for batch in minibatch_iterator(X, Y, batch_size):
87                  inputs, targets = batch
88                  train_err += train_fn(inputs, targets)
89                  train_batches += 1
90                  total_train_batches = total_train_batches + 1
91
92              # E um passo completo no conjunto de validacao:
93              val_err = 0
94              val_acc = 0
95              val_batches = 0
96              for batch in minibatch_iterator(X_val, Y_val, batch_size):
97                  inputs, targets = batch
98                  err, acc = val_fn(inputs, targets)
99                  val_err += err
100                 val_acc += acc
101                 val_batches += 1
102
103             # Calculamos os resultados
104             trainingLoss = (train_err / train_batches)
105             validationLoss = (val_err / val_batches)
106             validationAccuracy = (val_acc / val_batches * 100)
107
108     # Usamos a rede treinada para avaliar o modelo no conjunto de testes
109
110     # Computamos e imprimimos o resultado do conjunto de teste:
111     test_err = 0
112     test_acc = 0
113     test_batches = 0
114     for batch in minibatch_iterator(X_teste, Y_teste, batch_size):
115             inputs, targets = batch
116             err, acc = val_fn(inputs, targets)
117             test_err += err
118             test_acc += acc
119             test_batches += 1
120
121     print("Evaluation results:")
122     print("  Test Loss:\t\t\t{:.6f}".format(test_err / test_batches))
123     print("  Test Accuracy:\t\t{:.4f} %".format(test_acc / test_batches *
            100))
```

Listing A.1: *blabla*

Apêndice B

Apêndice

As figuras B.1, B.2, B.3 e B.4 apresentam as imagens de treinamento, sendo que em cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada.

As figuras B.5, B.6, B.7, B.8 apresentam as imagens de teste, sendo que em cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada.

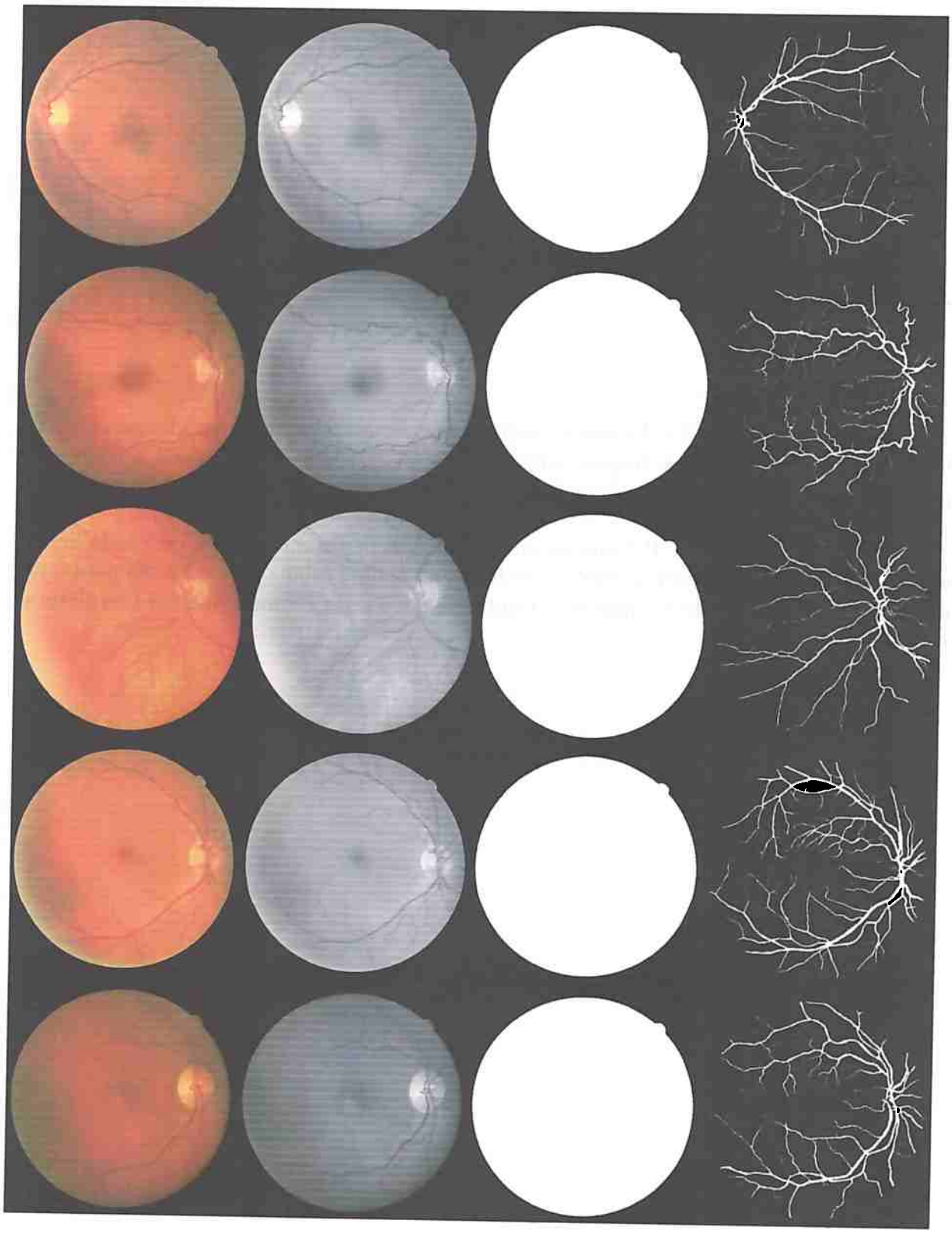


Figura B.1: *Primeiras cinco imagens do conjunto de treinamento do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

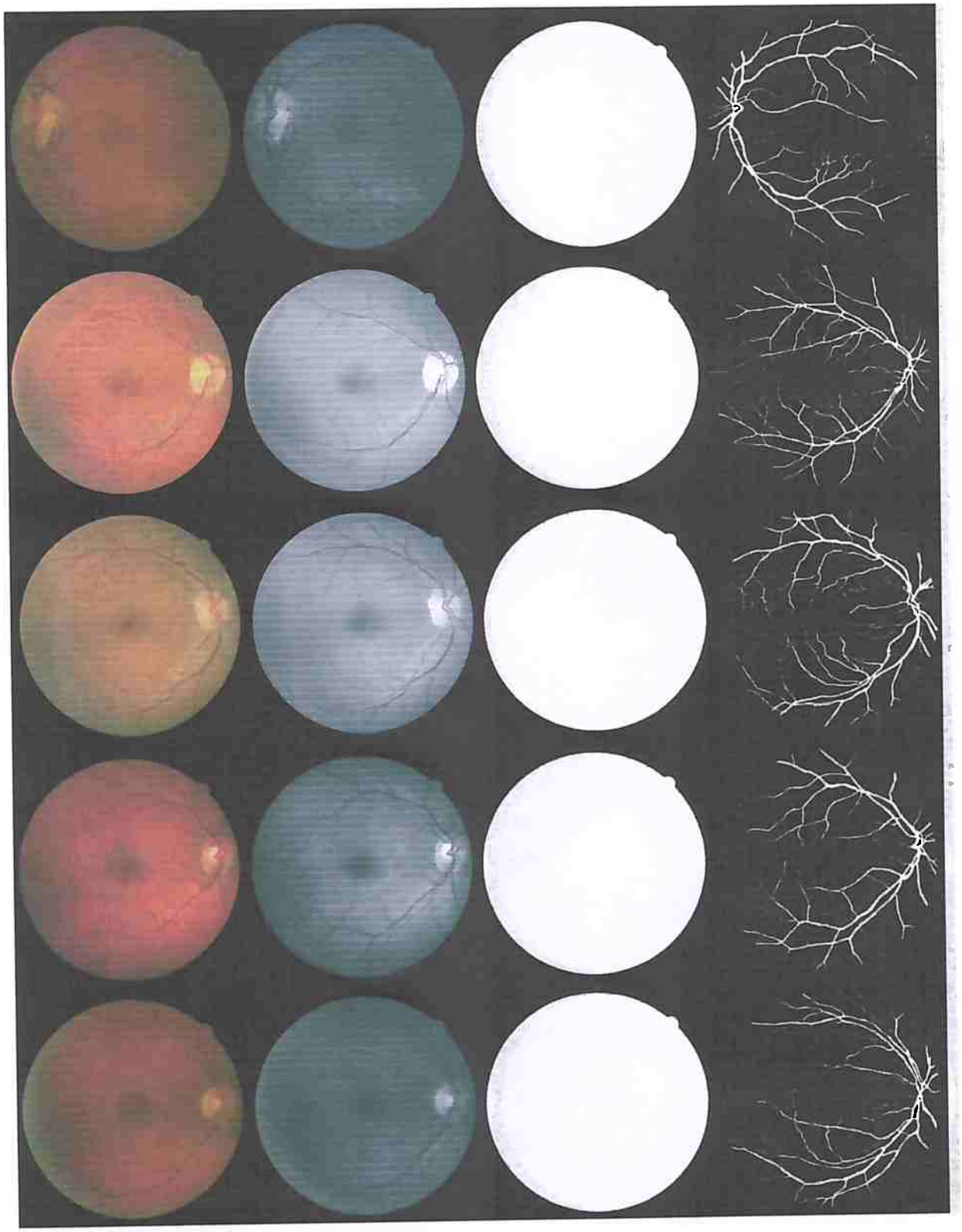


Figura B.2: *Imagens 6 a 10 do conjunto de treinamento do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

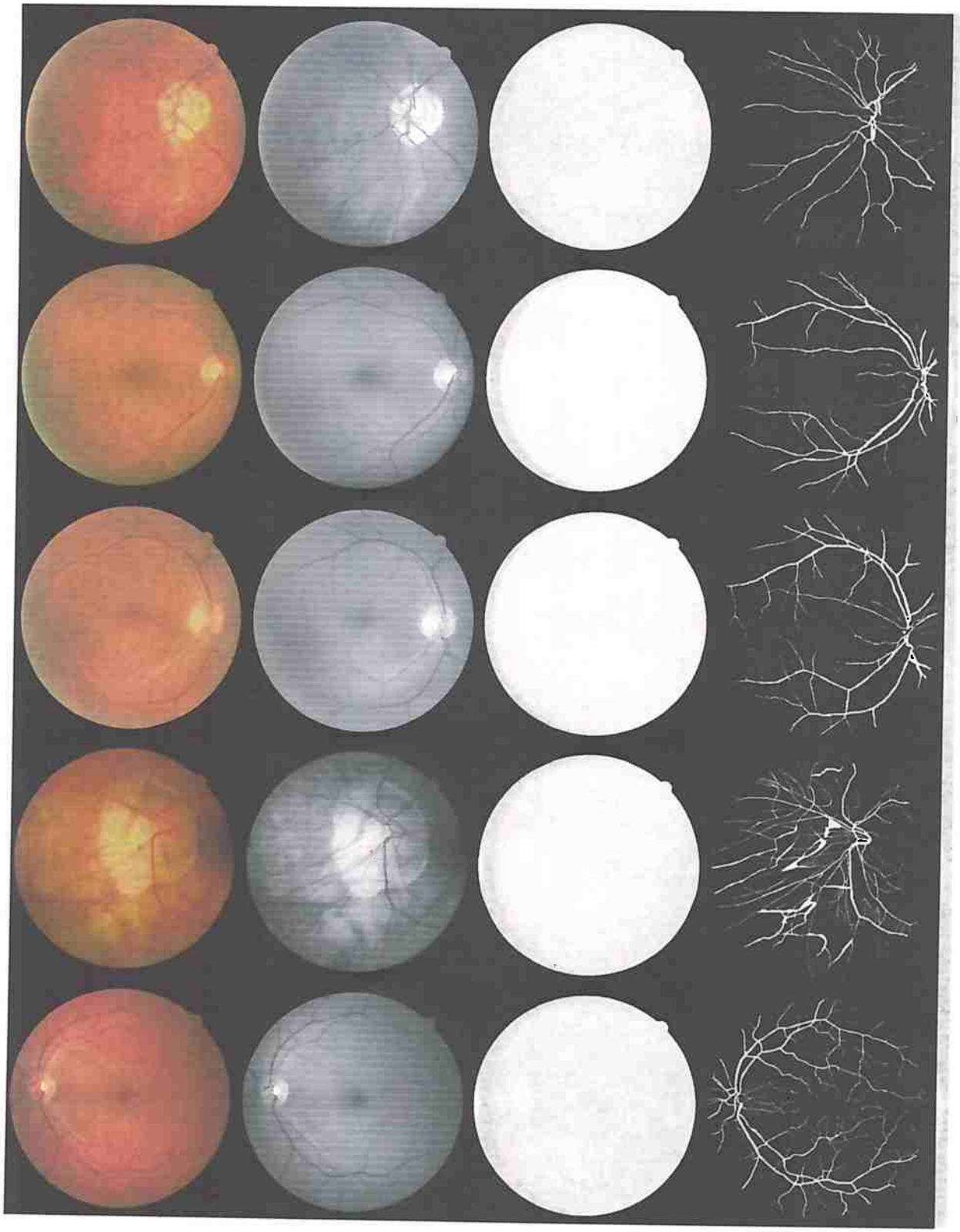


Figura B.3: *Imagens 11 a 15 do conjunto de treinamento do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

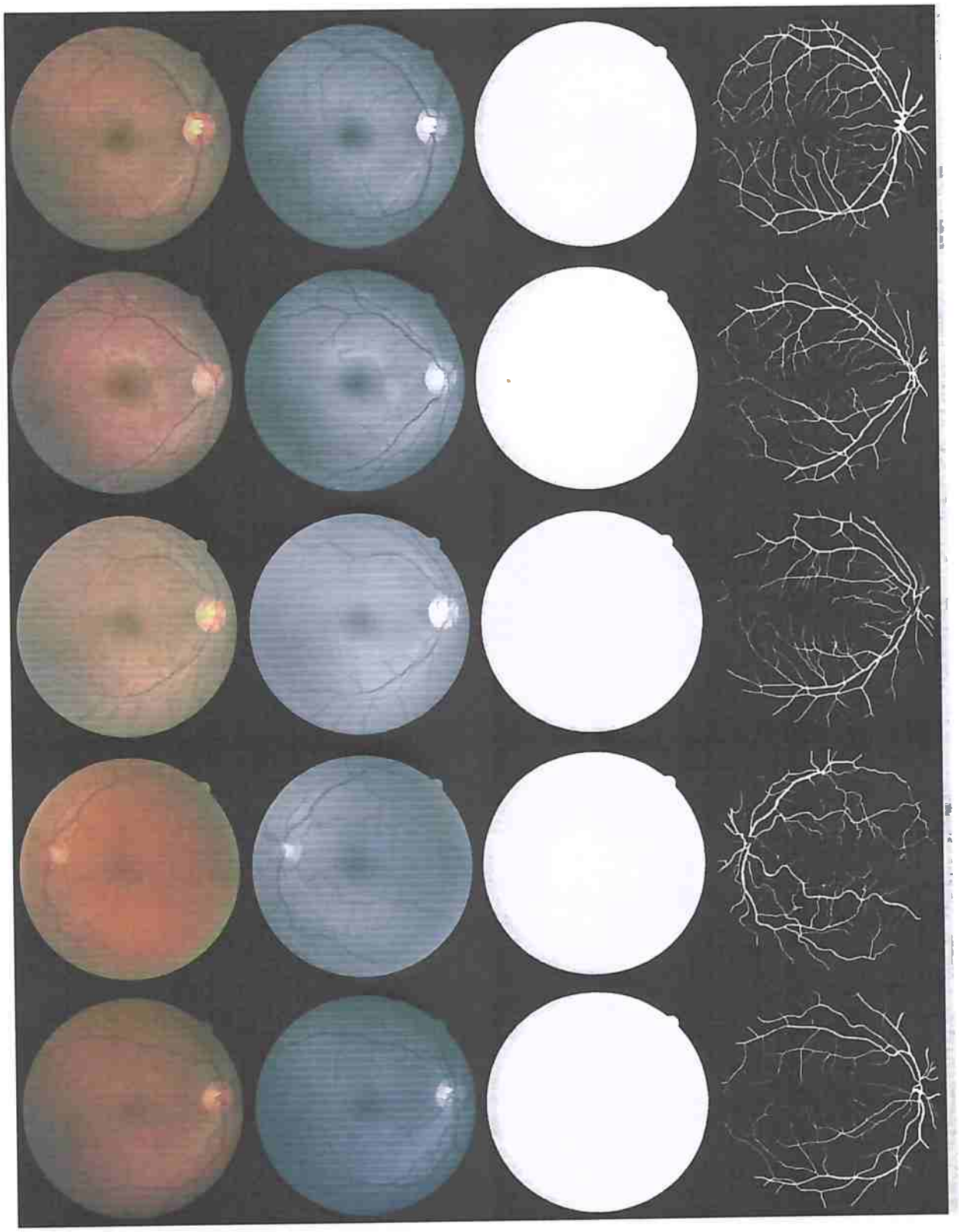


Figura B.4: *Imagens 16 a 20 do conjunto de treinamento do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

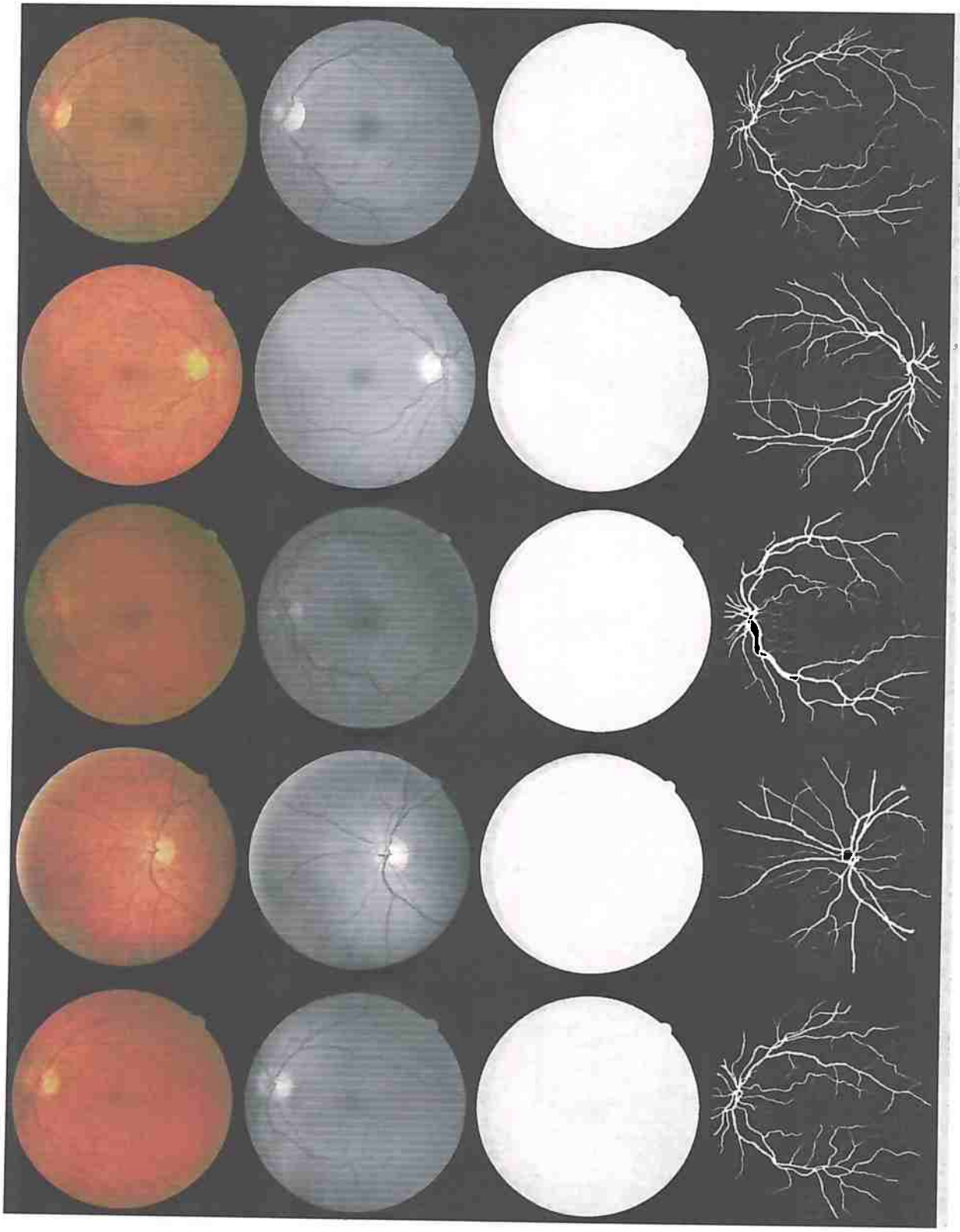


Figura B.5: *Primeiras cinco imagens do conjunto de teste do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

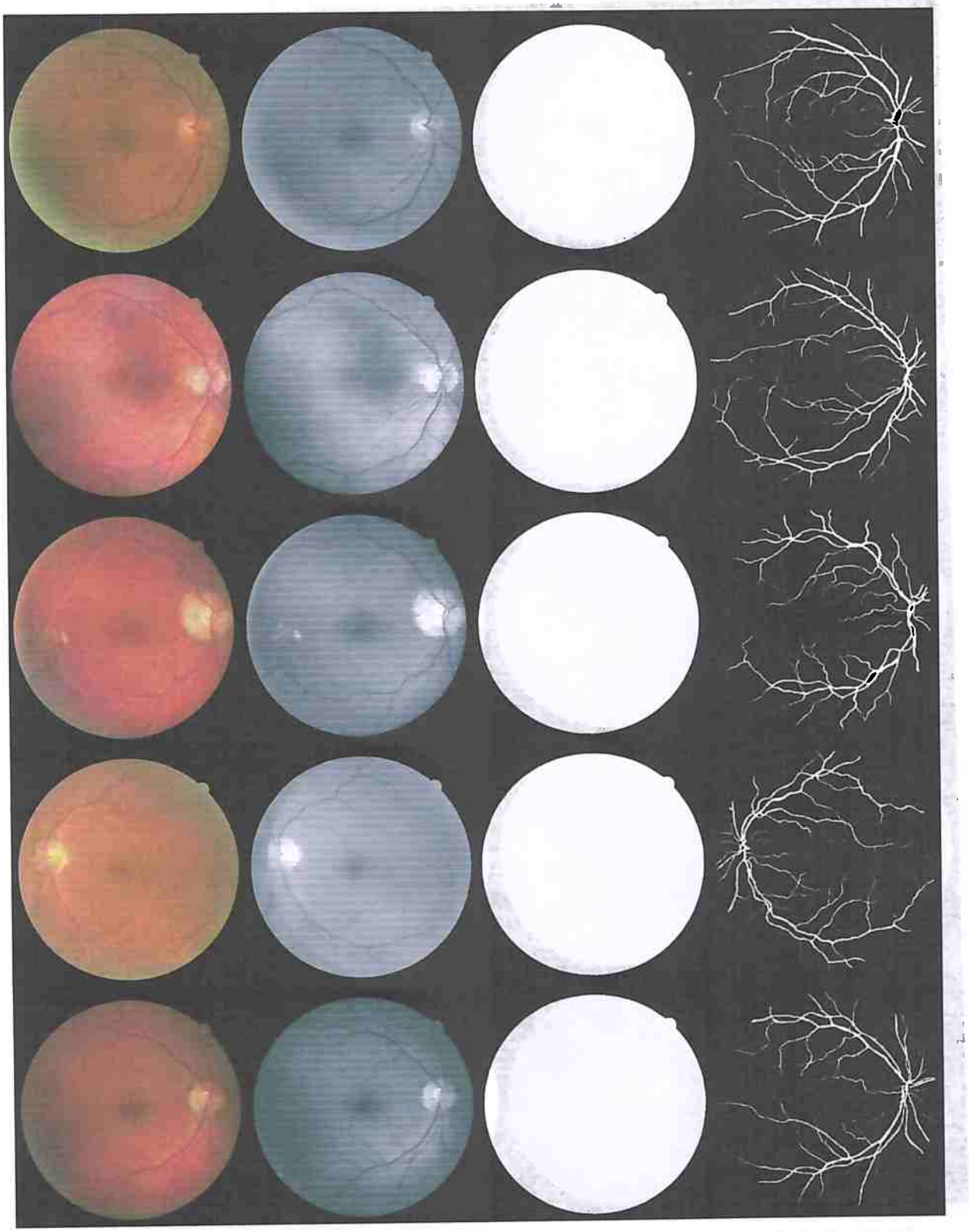


Figura B.6: *Imagens 6 a 10 do conjunto de teste do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

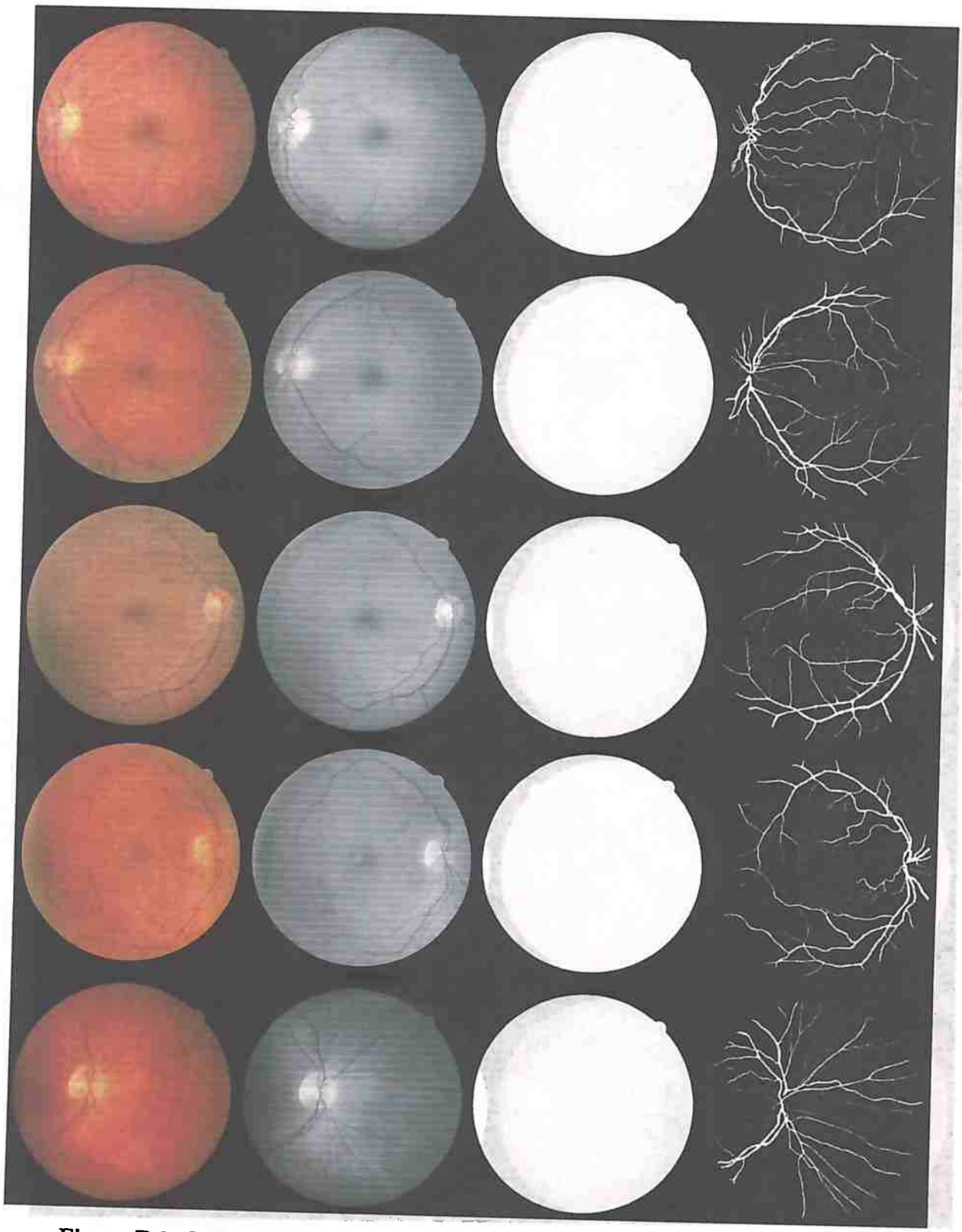


Figura B.7: *Imagens 11 a 15 do conjunto de teste do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

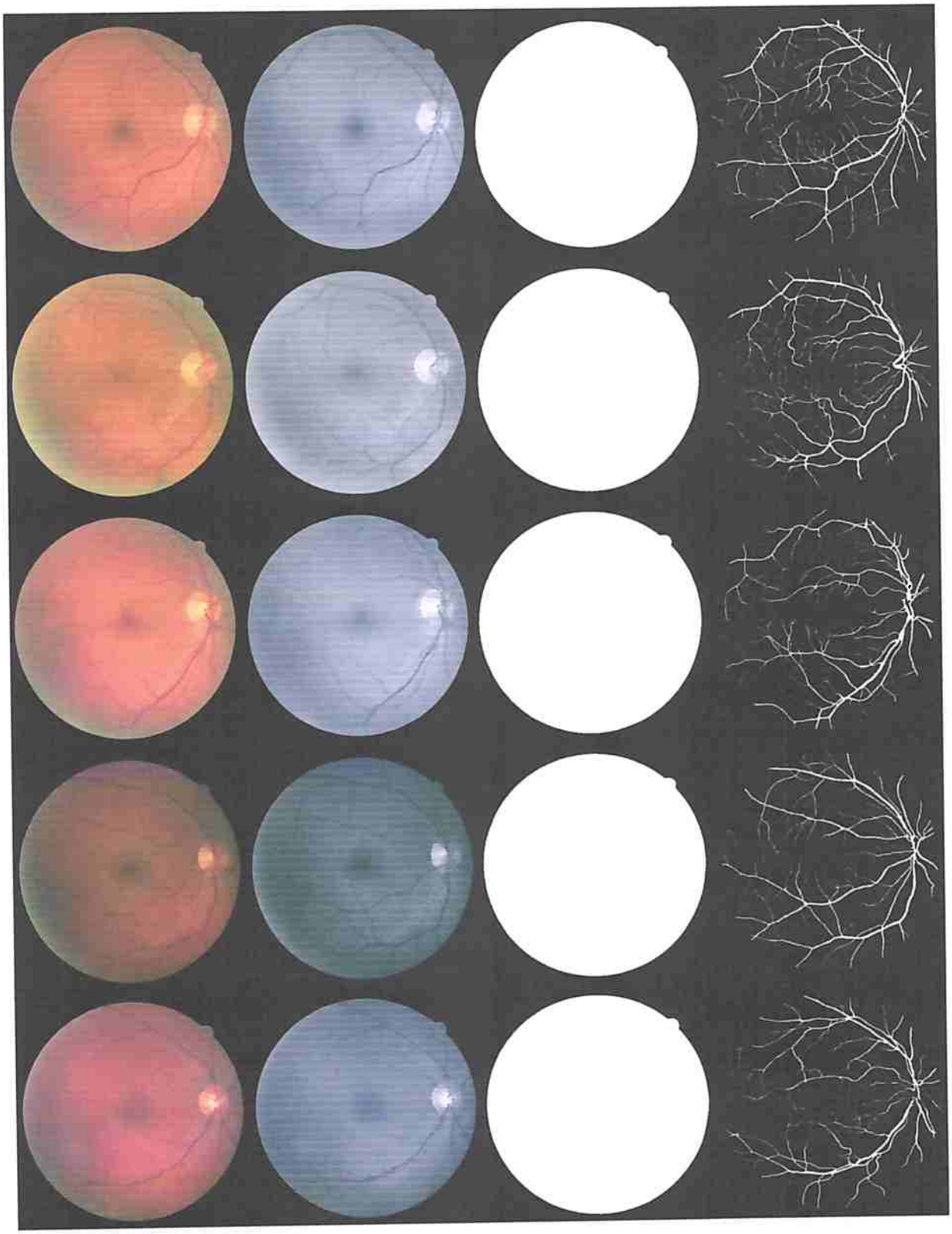
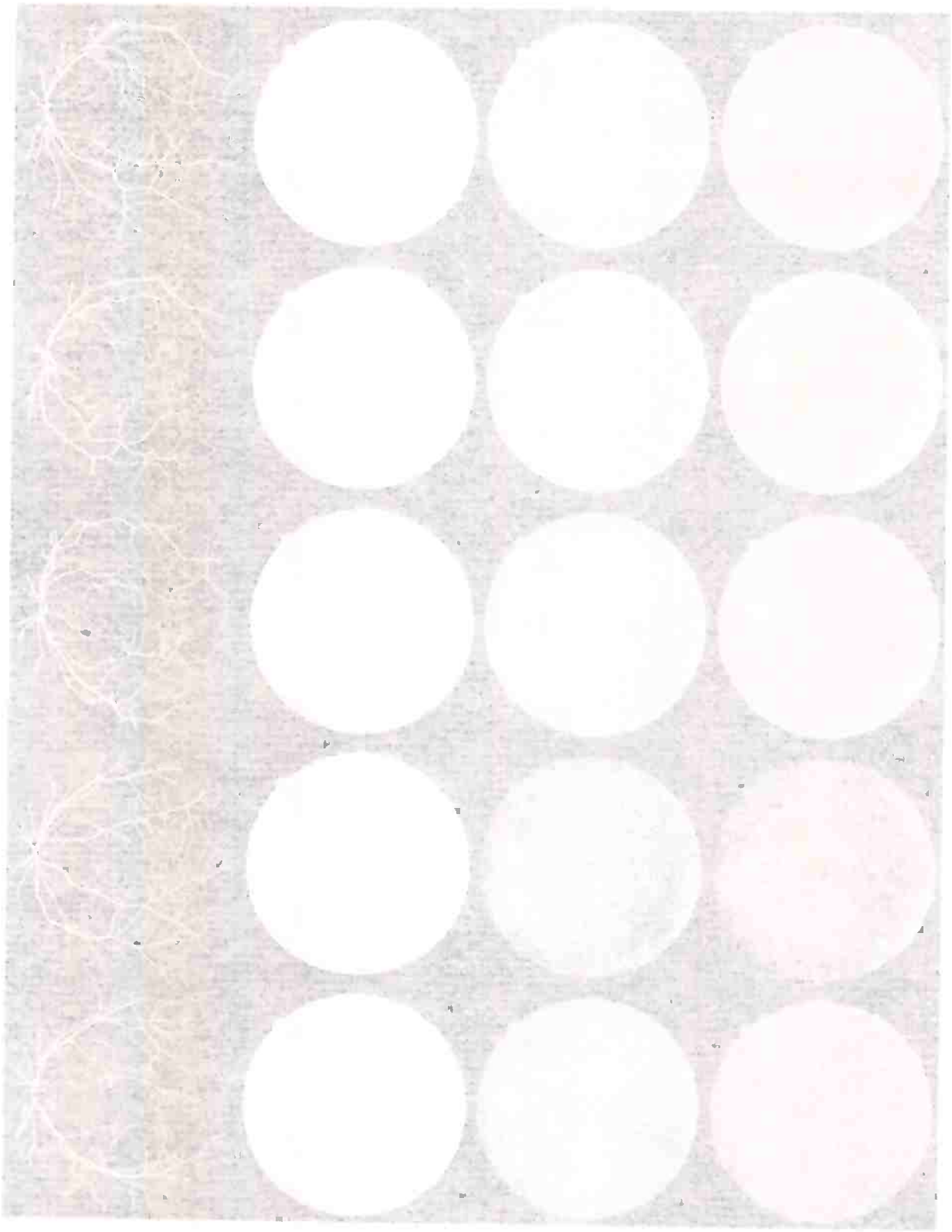


Figura B.8: *Imagens 16 a 20 do conjunto de teste do banco de dados DRIVE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*



Apêndice C

Apêndice

As figuras C.1, C.2, C.3 e C.4 apresentam as imagens de treinamento, sendo que em cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada.

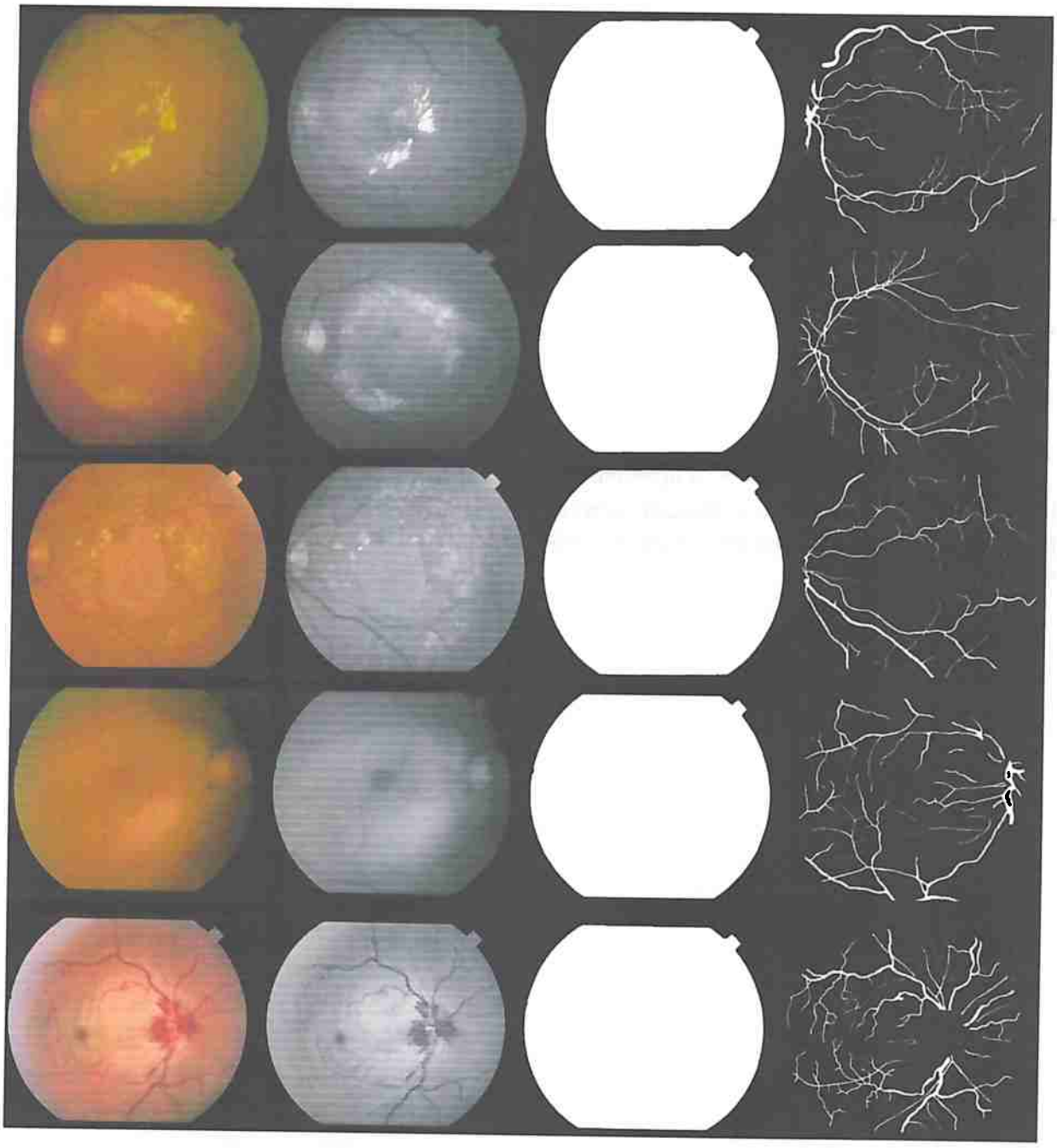


Figura C.1: *Primeiras cinco imagens do conjunto de treinamento do banco de dados STARE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

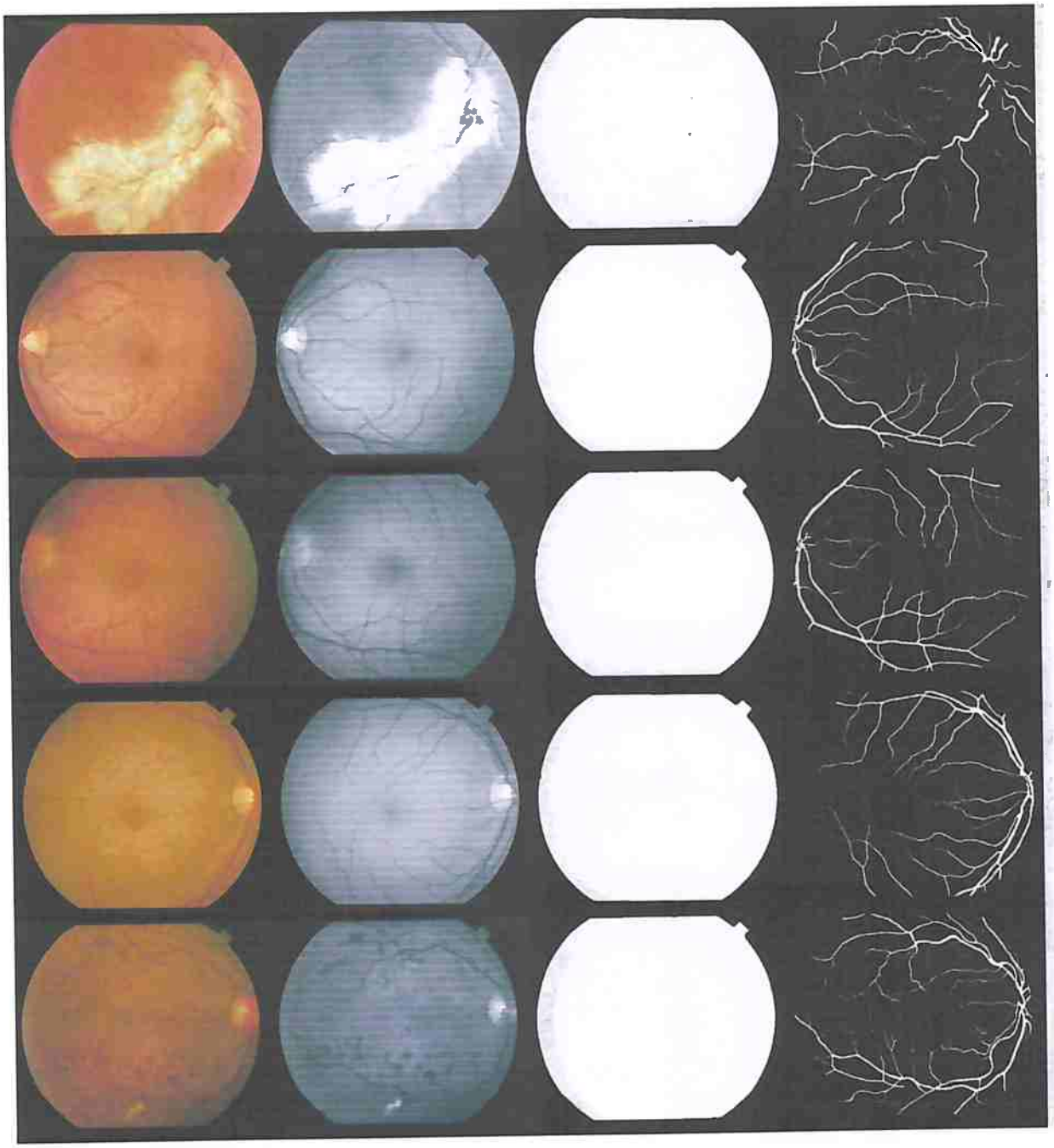


Figura C.2: *Imagens 6 a 10 do conjunto de treinamento do banco de dados STARE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

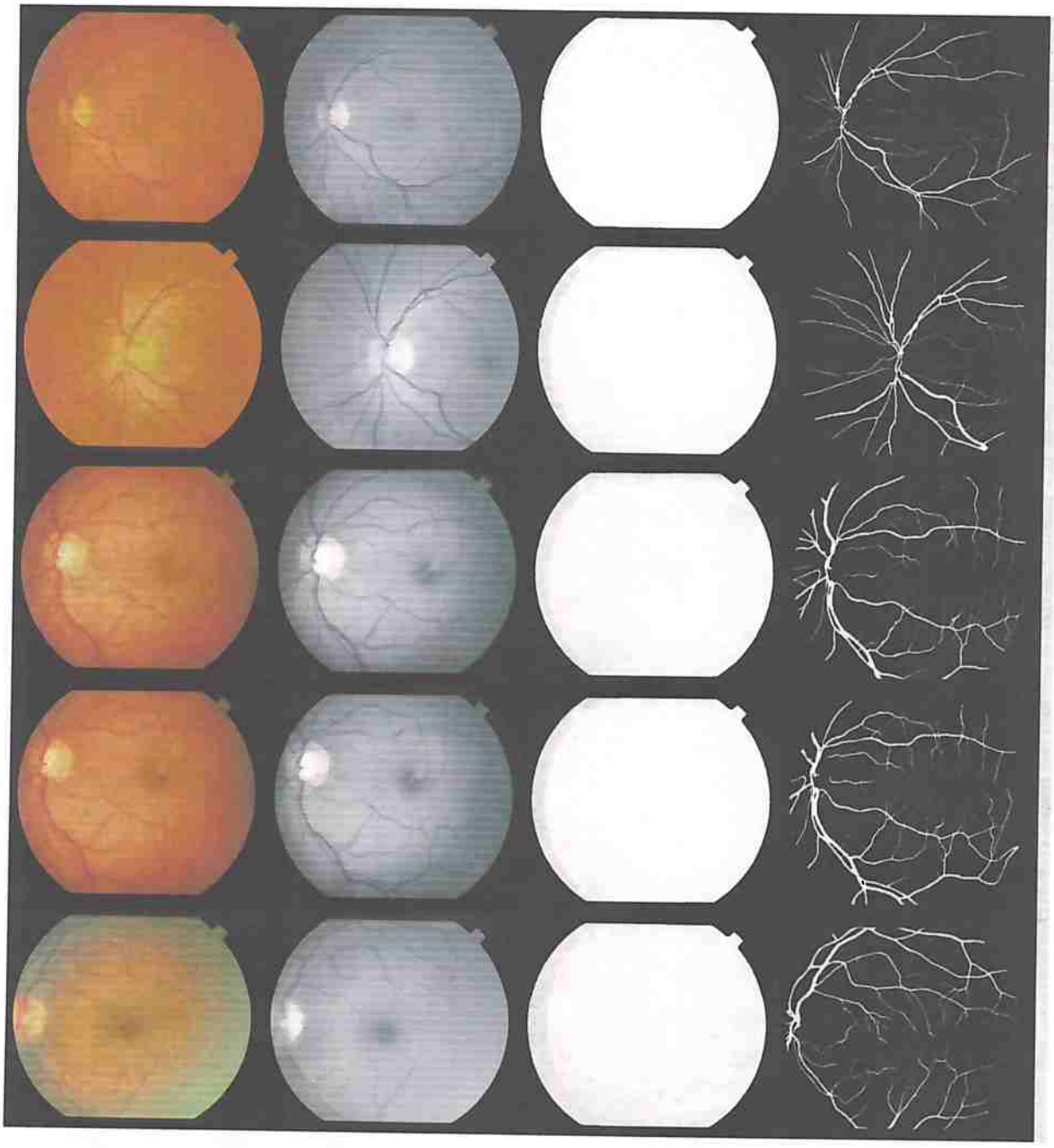


Figura C.3: *Imagens 11 a 15 do conjunto de treinamento do banco de dados STARE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*

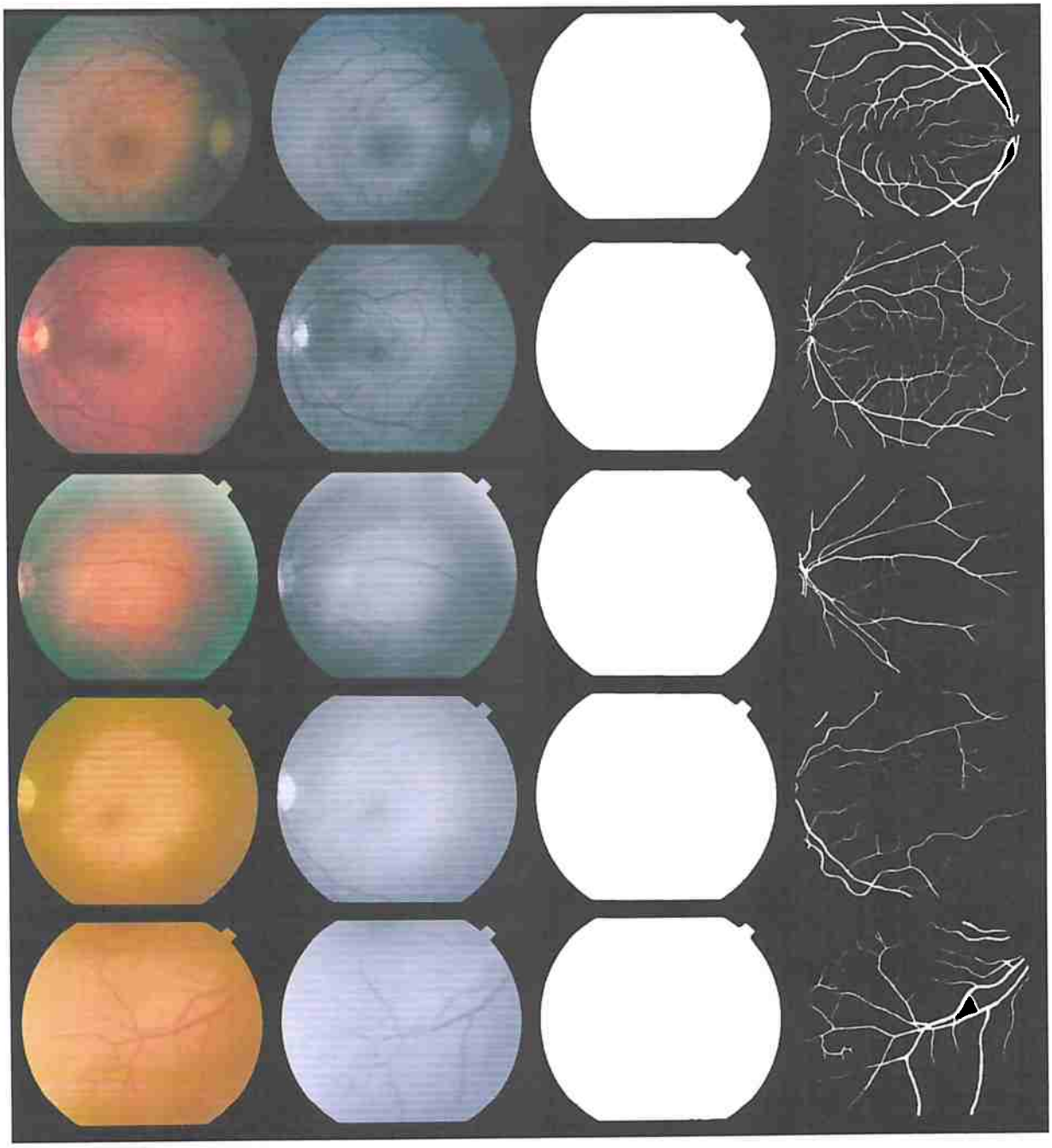
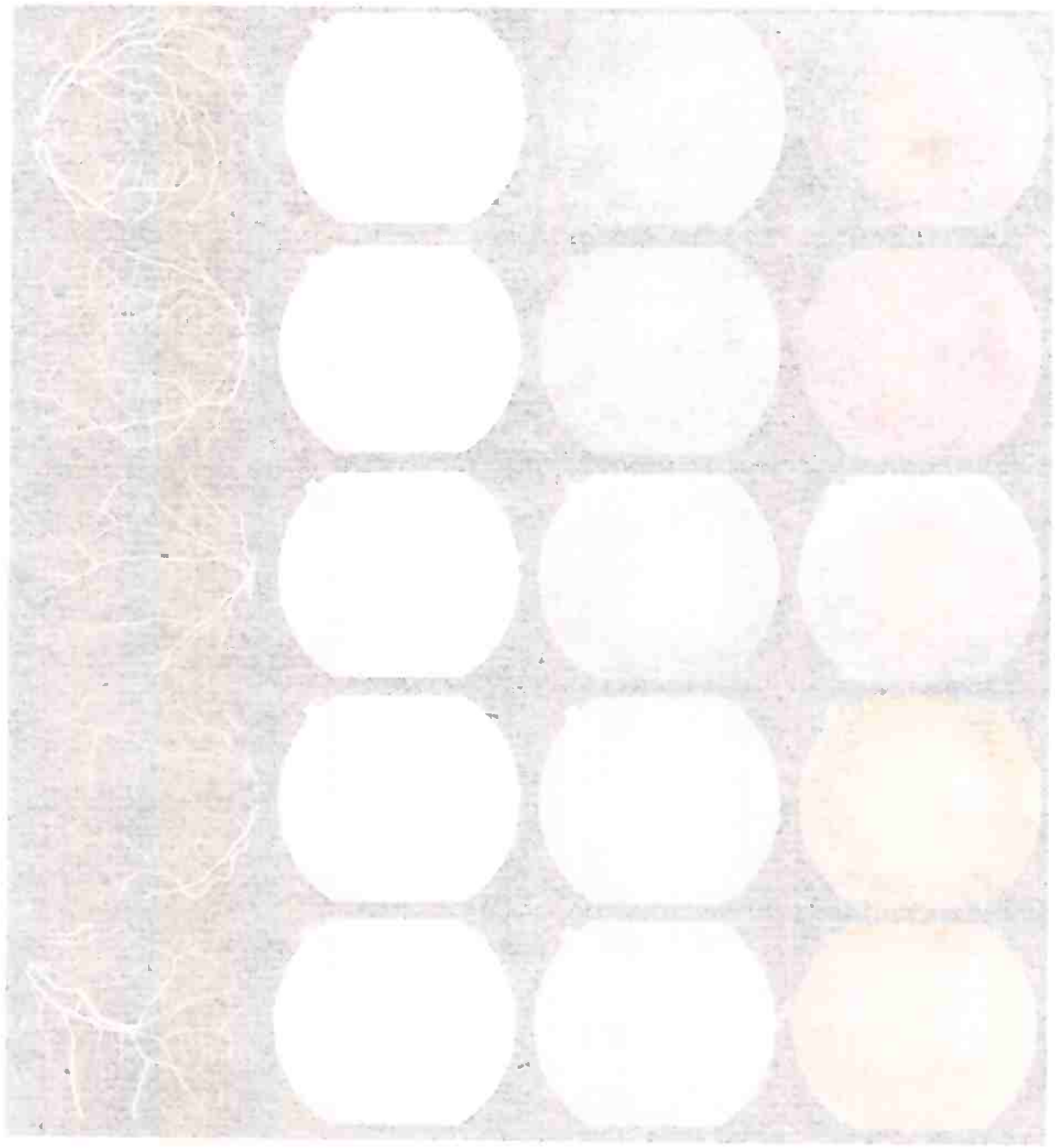


Figura C.4: *Imagens 16 a 20 do conjunto de treinamento do banco de dados STARE. Para cada linha, a primeira coluna é a imagem adquirida em RGB, a segunda coluna é a conversão para tons de cinza, a terceira coluna é a máscaras respectiva e a quarta coluna contém a vasculatura segmentada*



Referências Bibliográficas

[ABPS08] Andrea Anzalone, Federico Bizzarri, Mauro Parodi e Marco Storace. A modular supervised algorithm for vessel segmentation in red-free retinal images. *Computers in biology and medicine*, 38(8):913–922, 2008. 71

[AMVDP08] C. Alonso-Montes, D. L. Vilariño, P. Dudek e M. G. Penedo. Fast retinal vessel tree extraction: A pixel parallel approach. *International Journal of Circuit Theory and Applications*, 36(5-6):641–651, 2008. 74

[BB97] David Bainbridge e TC Bell. Dealing with superimposed objects in optical music recognition. 1997. 102, 104, 105

[BDS07] Robert E Bradley, Lawrence A D'Antonio e C Edward Sandifer. *Euler at 300: an appreciation*, volume 5. StatSoft, Inc., 2007. 10

[BDT97] Junior Barrera, Edward R Dougherty e Nina S Tomita. Automatic programming of binary morphological machines by design of statistically optimal operators in the context of computational learning theory. *Journal of Electronic Imaging*, 6(1):54–67, 1997. 30

[Ber13] James O Berger. *Statistical decision theory and Bayesian analysis*. Springer Science & Business Media, 2013. 1

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006. 10

[CG16] Tianqi Chen e Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. 108

[CGW+17] Vincent Christlein, Florin C. Ghesu, Tobias Würfl, Andreas Maier, Fabian Isensee, Peter Neher e Klaus Maier-Hein. *Tutorial: Deep Learning Advancing the State-of-the-Art in Medical Image Analysis*, páginas 6–7. Springer Berlin Heidelberg, Berlin, Heidelberg, 2017. iii, v, 2, 7

[CH15] Tianqi Chen e Tong He. Higgs boson discovery with boosted trees. Em *NIPS 2014 Workshop on High-energy Physics and Machine Learning*, páginas 69–80, 2015. 108

[CLG01] Rich Caruana, Steve Lawrence e C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. Em *Advances in neural information processing systems*, páginas 402–408, 2001. 19

[CPC16] Alfredo Canziani, Adam Paszke e Eugenio Culurciello. An analysis of deep neural network models for practical applications. *arXiv preprint arXiv:1605.07678*, 2016. 39

[CS9] Machine Learning CS933. Kaggle competition: Product classification. 108

[Csá01] Balázs Csanád Csáji. Approximation with artificial neural networks. *MSc Thesis, Eötvös Loránd University (ELTE), Budapest, Hungary*, 2001. 12

[DBDJH14] Howard B Demuth, Mark H Beale, Orlando De Jess e Martin T Hagan. *Neural network design*. Martin Hagan, 2014. 19

[DD11] L. A. Dalton e E. R. Dougherty. Bayesian minimum mean-square error estimation for classification error x2014;part i: Definition and the bayesian mmse error estimator for discrete classification. *IEEE Transactions on Signal Processing*, 59(1):115–129, Jan 2011. 1

[DDPF08] Christoph Dalitz, Michael Droettboom, Bastian Pranzas e Ichiro Fujinaga. A comparative study of staff removal algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):753–766, 2008. 102, 104, 105

[DKTAXX] Konstantinos K. Delibasis, Aristides I. Kechriniotis, C. Tsonos e Nicholas Assimakis. Automatic model-based tracing algorithm for vessel segmentation and diameter estimation. *Computer Methods and Programs in Biomedicine*, 100(2):108–122, 2017/05/16 XXXX. 74, 78

[Dou90] Edward R Dougherty. *Probability and statistics for the engineering, computing, and physical sciences*. Prentice-Hall, Inc., 1990. 1

[Dou92a] Edward Dougherty. *An introduction to morphological image processing*. SPIE Optical Engineering Press, Bellingham, Wash., USA, 1992. 29

[Dou92b] Edward R Dougherty. Optimal mean-square n-observation digital morphological filters: ii. optimal gray-scale filters. *CVGIP: Image Understanding*, 55(1):55–72, 1992. 1

[DV16] Vincent Dumoulin e Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016. 108

[FDGL12] Alicia Fornés, Anjan Dutta, Albert Gordo e Josep Lladós. Cvc-muscima: a ground truth of handwritten music score images for writer identification and staff removal. *International Journal on Document Analysis and Recognition*, páginas 1–9, 2012. iii, v, 2, 83

[FRH+12] M. M. Fraz, P. Remagnino, A. Hoppe, B. Uyyanonvara, A. R. Rudnicka, C. G. Owen e S. A. Barman. An ensemble classification-based approach applied to retinal blood vessel segmentation. *IEEE Transactions on Biomedical Engineering*, 59(9):2538–2548, Sept 2012. 74, 78

[FTM+17] Z. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue e K. Mizutani. State-of-the-art deep learning: Evolving machine intelligence toward tomorrow x2019;s intelligent network traffic control systems. *IEEE Communications Surveys Tutorials*, PP(99):1–1, 2017. iii, v, 2, 7

[Fuk80] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. 21

[GB10] Xavier Glorot e Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. Em *Aistats*, volume 9, páginas 249–256, 2010. 13, 43, 84

[Gér14] Thierry Géraud. A morphological method for music score staff removal. Em *Image Processing (ICIP), 2014 IEEE International Conference on*, páginas 2599–2603. IEEE, 2014. 102, 105

[Hea08] Jeff Heaton. *Introduction to Neural Networks for Java, 2nd Edition.* Heaton Research, Inc., 2° edição, 2008.

[Hea15] Jeff Heaton. *Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks*, volume 3. Heaton Research, Inc, 1° edição, December 2015.

[Hei11] Henk Heijmans. *Mathematical morphology and its applications to image and signal processing.* Springer, Dordrecht London, 2011. 29

[HGC+17] B. Harwood, V. K. B G, G. Carneiro, I. Reid e T. Drummond. Smart Mining for Deep Metric Learning. *ArXiv e-prints*, Abril 2017. iii, v, 2, 7

[Hoo75] A Hoover. Stare database. *Available: Available: http://www. ces. clemson. edu/˜ ahoover/stare*, 1975. 2, 41

[Hor91] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257, 1991. 12

[HOT06] Geoffrey E Hinton, Simon Osindero e Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 2, 108

[HRVS16] Seyyed Hossein HasanPour, Mohammad Rouhani, Javad Vahidi e Reza Saadati. Lets keep it simple: using simple architectures to outperform deeper architectures. *arXiv preprint arXiv:1608.06037*, 2016. 38, 39

[HSK+12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever e Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012. 20

[HSM+00] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas e H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947, 2000. 10

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren e Jian Sun. Deep residual learning for image recognition. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 770–778, 2016. 38

[ISMJ16] Nina S. T. Hirata Igor S. Montagner e Roberto Hirata Jr. Image operator learning and applications. *SIBGRAPI 2016*, 2016. 74

[Jac88] Robert A Jacobs. Increased rates of convergence through learning rate adaptation. *Neural networks*, 1(4):295–307, 1988. 18

[JAH17] Frank D Julca-Aguilar e Nina ST Hirata. Image operator learning coupled with cnn classification and its application to staff line removal. *arXiv preprint arXiv:1709.06476*, 2017. 102, 103, 104, 105

[Jr01] Roberto Hirata Jr. Projeto de operadores morfológicos para imagens e sinais, Novembro 2001. 1, 31

[KFV+13] Van Cuong Kieu, Alicia Fornés, Muriel Visani, Nicholas Journet e Dutta Anjan. The icdar/grec 2013 music scores competition on staff removal. Em *10th IAPR International Workshop on Graphics Recognition*, 2013. iii, v, 2, 83

[Kim14] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. 39

[KR92] Kenji Kira e Larry A Rendell. The feature selection problem: Traditional methods and a new algorithm. Em *Aaai*, volume 2, páginas 129–134, 1992. 104

[KRSP96] Igor Kononenko, Marko Robnik-Sikonja e Uros Pompe. Relieff for estimation and discretization of attributes in classification, regression, and ilp problems. *Artificial intelligence: methodology, systems, applications*, páginas 31–40, 1996. 104

[KUMH17] Gunter Klambauer, Thomas Unterthiner, Andreas Mayr e Sepp Hochreiter. Self-normalizing neural networks. *CoRR*, abs/1706.02515, 2017. 108

[LB95] Yann Lecun e Yoshua Bengio. *Convolutional Networks for Images, Speech and Time Series*, páginas 255–258. The MIT Press, 1995. 20

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio e Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 24, 38, 104

[LBBH01] Y. LeCun, L. Bottou, Y. Bengio e P. Haffner. Gradient-based learning applied to document recognition. Em *Intelligent Signal Processing*, páginas 306–351. IEEE Press, 2001. 21

[LFX+16] Q. Li, B. Feng, L. Xie, P. Liang, H. Zhang e T. Wang. A cross-modality learning approach for vessel segmentation in retinal images. *IEEE Transactions on Medical Imaging*, 35(1):109–118, Jan 2016. 74, 78, 80

[LGL10] B. S. Y. Lam, Y. Gao e A. W. C. Liew. General retinal vessel segmentation using regularization-based multiconcavity modeling. *IEEE Transactions on Medical Imaging*, 29(7):1369–1381, July 2010. 74, 78

[LTT10] C. A. Lupascu, D. Tegolo e E. Trucco. Fabc: Retinal vessel segmentation using adaboost. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1267–1274, Sept 2010. 74

[MAGAB11] D. Marin, A. Aquino, M. E. Gegundez-Arias e J. M. Bravo. A new supervised method for blood vessel segmentation in retinal images by using gray-level and moment invariants-based features. *IEEE Transactions on Medical Imaging*, 30(1):146–158, Jan 2011. 74, 78

[MB91] Philippe Martin e Carniue Bellissant. Low-level analysis of music drawing images. Em *First International Conference Document Analysis and Recognition*, páginas 417–425, 1991. 102, 104, 105

[MC06] A. M. Mendonca e A. Campilho. Segmentation of retinal blood vessels by combining the detection of centerlines and morphological reconstruction. *IEEE Transactions on Medical Imaging*, 25(9):1200–1213, Sept 2006. 74, 78

[MHH17] Igor S. Montagner, Nina S.T. Hirata e Roberto Hirata. Staff removal using image operator learning. *Pattern Recognition*, 63(Supplement C):310 – 320, 2017. 102, 104, 105

[MK16] Ankita Mangal e Nishant Kumar. Using big data to enhance the bosch production line performance: A kaggle challenge. Em *Big Data (Big Data), 2016 IEEE International Conference on*, páginas 2029–2035. IEEE, 2016. 108

[MLLZH17] Zhenshen Ma Meng Li, Chao Liu, Guang Zhang e Zhe Han. Robust retinal blood vessel segmentation based on reinforcement local descriptions. *BioMed Research International*, 2017(2028946):9, 2017. 74, 78, 79, 80

[Mon17a] Igor Montagner. W-operator learning using linear models for both gray-level and binary inputs, Fevereiro 2017. 74

[Mon17b] Igor dos Santos Montagner. *W-operator learning using linear models for both gray-level and binary inputs.* Tese de Doutorado, Universidade de São Paulo, 2017. 104

[Mos12] Yaser Mostafa. *Learning from data : a short course.* AMLBook.com, United States, 2012.

[MP43] Warren S. McCulloch e Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943. 7, 20

[MPGC17] N. Majumder, S. Poria, A. Gelbukh e E. Cambria. Deep learning-based document modeling for personality detection from text. *IEEE Intelligent Systems*, 32(2):74–79, Mar 2017. iii, v, 2, 7

[MTD] Zachary Maurer, Tanuj Thapliyal e Shloka Desai. Land cover classification in the amazon. 108

[MZY+17] Chunwei Ma, Zhiyong Zhu, Jun Ye, Jiarui Yang, Jianguo Pei, Shaohang Xu, Ruo Zhou, Chang Yu, Fan Mo, Bo Wen et al. Deeprt: deep learning for peptide retention time prediction in proteomics. *arXiv preprint arXiv:1705.05368*, 2017. iii, v, 2, 7

[NBPR13] Uyen T.V. Nguyen, Alauddin Bhuiyan, Laurence A.F. Park e Kotagiri Ramamohanarao. An effective retinal blood vessel segmentation method using multi-scale line detection. *Pattern Recognition*, 46(3):703 – 715, 2013. 74, 78

[Nes03] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course (Applied Optimization).* Springer, 2003. 17

[NH10] Vinod Nair e Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. Em *Proceedings of the 27th international conference on machine learning (ICML-10)*, páginas 807–814, 2010. 24, 38, 104

[Nvi10] CUDA Nvidia. Programming guide, 2010. 108

[PAA+87] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B. Zimmerman e Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 9 1987. 45, 65, 71

[Pre98a] Lutz Prechelt. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks*, 11(4):761–767, 1998. 19

[Pre98b] Lutz Prechelt. Early stopping-but when? *Neural Networks: Tricks of the trade*, páginas 553–553, 1998. 19

[RP07] E. Ricci e R. Perfetti. Retinal blood vessel segmentation using line operators and support vector classification. *IEEE Transactions on Medical Imaging*, 26(10):1357–1365, Oct 2007. 74, 78

[RR14] CG Ravichandran e J Benadict Raja. A fast enhancement/thresholding based blood vessel segmentation for retinal image using contrast limited adaptive histogram equalization. *Journal of Medical Imaging and Health Informatics*, 4(4):567–575, 2014. 71

[SAN+04a] J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever e B. van Ginneken. Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, April 2004. 74, 78

[SAN+04b] J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever e B. van Ginneken. Ridge based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, 2004. 2, 41, 74, 78

[SEM11] Marwan D Saleh, C Eswaran e Ahmed Mueen. An automated blood vessel segmentation algorithm using histogram equalization and automatic threshold selection. *Journal of digital imaging*, 24(4):564–572, 2011. 71

[SLC+06] J. V. B. Soares, J. J. G. Leandro, R. M. Cesar, H. F. Jelinek e M. J. Cree. Retinal vessel segmentation using the 2-d gabor wavelet and supervised classification. *IEEE Transactions on Medical Imaging*, 25(9):1214–1222, Sept 2006. 74, 78

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke e Andrew Rabinovich. Going deeper with convolutions. Em *Proceedings of the IEEE conference on computer vision and pattern recognition*, páginas 1–9, 2015. 38, 104

[SZ14] Karen Simonyan e Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 24, 38, 104

[TAB+17] Jen Hong Tan, U. Rajendra Acharya, Sulatha V. Bhandary, Kuang Chua Chua e Sobha Sivaprasad. Segmentation of optic disc, fovea and retinal vasculature using a single convolutional neural network. *Journal of Computational Science*, páginas –, 2017. xvii, 74, 80, 81, 82

[Ver45] PF Verhulst. Recherches mathnmatiques sur la loi d'accroissement de la population, nouveaux mnmoires de l acadnmie royale des sciences et belles-lettres de bruxelles, 18, art. 1, 1-45. 1845. 10

[VKFJ13] Muriel Visaniy, Van Cuong Kieu, Alicia Fornés e Nicholas Journet. Icdar 2013 music scores competition: Staff removal. Em *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, páginas 1407–1411. IEEE, 2013. iii, v, 2, 83, 102, 105

[WDH+XX] R. A. Welikala, J. Dehmeshki, A. Hoppe, V. Tah, S. Mann, T. H. Williamson e S. A. Barman. Automated detection of proliferative diabetic retinopathy using a modified line operator and dual classification. *Computer Methods and Programs in Biomedicine*, 114(3):247–261, 2017/05/16 XXXX. 74, 78

[WYC+15] Shuangling Wang, Yilong Yin, Guibao Cao, Benzheng Wei, Yuanjie Zheng e Gongping Yang. Hierarchical retinal blood vessel segmentation based on feature and ensemble learning. *Neurocomputing*, 149, Part B:708 – 717, 2015. xvii, 74, 78, 80, 81

[YK15] Fisher Yu e Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 108

[YPY+11] Xinge You, Qinmu Peng, Yuan Yuan, Yiu ming Cheung e Jiajia Lei. Segmentation of retinal blood vessels using the radial projection and semi-supervised approach. *Pattern Recognition*, 44(10?11):2314 – 2324, 2011. Semi-Supervised Learning for Visual Content Analysis and Understanding. 74, 78

[ZK01] F. Zana e J. C. Klein. Segmentation of vessel-like patterns using mathematical morphology and curvature evaluation. *IEEE Transactions on Image Processing*, 10(7):1010–1019, Jul 2001. 74

[ZWWS14] Yu Qian Zhao, Xiao Hong Wang, Xiao Fang Wang e Frank Y. Shih. Retinal vessels segmentation based on level set and region growing. *Pattern Recognition*, 47(7):2437 – 2446, 2014. 71, 72, 74, 78