



UNIVERSIDADE FEDERAL DO MARANHÃO

Curso de Ciência da Computação

Roberto Matheus Pinheiro Pereira

**Estudo do Desempenho de Redes Neurais
Convolucionais Aplicada ao Reconhecimento de
Símbolos Musicais, Glaucoma e Texto**

São Luís - MA

2017

Roberto Matheus Pinheiro Pereira

**Estudo do Desempenho de Redes Neurais Convolucionais
Aplicada ao Reconhecimento de Símbolos Musicais,
Glaucoma e Texto**

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Geraldo Braz Junior

São Luís - MA

2017

Pinheiro Pereira, Roberto Matheus.

Estudo do Desempenho de Redes Neurais Convolucionais Aplicada ao Reconhecimento de Símbolos Musicais, Glaucoma e Texto / Roberto Matheus Pinheiro Pereira. - 2017.
90 f.

Orientador(a): Geraldo Braz Junio.

Monografia (Graduação) - Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luis, 2017.

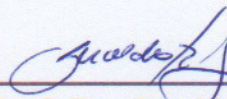
1. Aprendizado de Máquina. 2. Deep Learning. 3. Redes Neurais Convolucionais. I. Junio, Geraldo Braz. II. Título.

Roberto Matheus Pinheiro Pereira

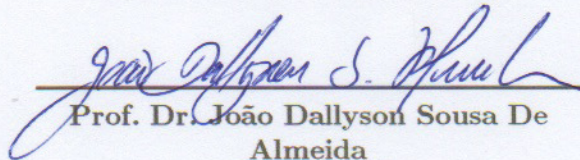
**Estudo do Desempenho de Redes Neurais Convolucionais
Aplicada ao Reconhecimento de Símbolos Musicais,
Glaucoma e Texto**

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

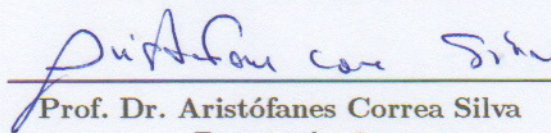
Trabalho Aprovado em: São Luís - MA, 26 de Janeiro de 2016:



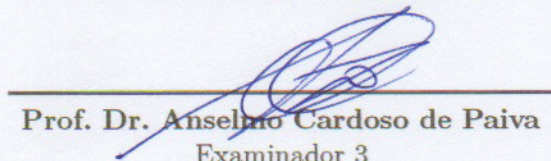
Prof. Dr. Geraldo Braz Junior
Orientador
Universidade Federal do Maranhão



Prof. Dr. João Dallyson Sousa De Almeida
Examinador 1
Universidade Federal do Maranhão



Prof. Dr. Aristófanes Correa Silva
Examinador 2
Universidade Federal do Maranhão



Prof. Dr. Anselmo Cardoso de Paiva
Examinador 3
Universidade Federal do Maranhão

São Luís - MA

2017

Agradecimentos

Agradeço em primeiro lugar a Deus que iluminou o meu caminho durante esta jornada. Aos meus pais e meu irmão que, em todos os momentos da minha vida, sempre estiveram presentes me apoiando, incentivando e orientando.

Às minhas tias, tios e familiares pela atenção e ensinamentos que recebi desde criança.

À minha namorada que me apoia a cada dia e todos amigos com quem compartilhei minha graduação. Em especial, a Lucas Maia e Nigel Lima pelo apoio e amizade sincera e a Jordan Boaz e Polyana Costa pelos momentos de força e inspiração na escrita deste e de outros trabalhos.

Ao meu orientador, Prof. Dr. Geraldo Braz Junior, quem me inspirou e orientou em todos momentos da minha graduação, sempre motivador, disponível, paciente e ensinando mesmo fora da sala de aula.

Aos meus professores que me inspiraram, aconselharam e que colaboraram com muito mais do que meu conhecimento acadêmico, em especial aos Prof. Dr. Carlos Salles, Prof. Dr. Anselmo Cardoso de Paiva e Prof. Dr. João Dallyson os quais terão meu eterno agradecimento e admiração.

Resumo

Diversas aplicações vem utilizando diferentes tipos de Redes Neurais Convolucionais (CNN) em tarefas de classificação e detecção. Este tipo especial de rede neural permite a extração automática de características, assim, reduzindo esforços na escolha das melhores características que representam um conjunto. Contudo, estes esforços normalmente são transmitidos para a escolha dos hiper-parâmetros e da arquitetura das redes neurais convolucionais.

Este trabalho tem como objetivo realizar um estudo sobre variações de diferentes arquiteturas de Redes Neurais Convolucionais (LeNet, VGGNet e ResNet) e de seus hiper-parâmetros aplicados em diferentes cenários. Foram escolhidos três problemas a serem estudados: classificação de notas musicais manuscritas, identificação de glaucoma em imagens oculares e a análise de sentimentos em avaliações textuais. Os melhores resultados obtidos foram de 95.16% de acurácia para classificação de símbolos musicais, 88.57% para diagnóstico de glaucoma e 88.30% para análise de sentimento em texto.

Palavras-chaves: Deep Learning; Redes Neurais Convolucionais; Aprendizado de Máquina

Abstract

Many applications have recently been developed using Convolutional Neural Networks (CNN) in the fields of classification and detection. This special type of neural network are capable of extracting features automatically. Thus, reducing the need to do previous engineering in order to extract them. However, choosing the correct architecture and its hyper-parameters to solve a task may be hard and the key to solve many open problems.

This work aims to study different variations of CNN's architectures (LeNet, VGGNet and ResNet) as well as their hyper-parameters' variations used to solve different tasks. A total of three were chosen: musical symbol classification, glaucoma diagnosis and text sentiments analysis. The best models for each task reached 95.16% of accuracy for musical symbol classification, 88.57% for glaucoma diagnosis and 88.30 for text sentiments analysis.

Keywords: Deep Learning; Convolutional Neural Networks; Machine Learning

Lista de ilustrações

Figura 1 – Neurônio biológico	20
Figura 2 – Neurônio artificial	21
Figura 3 – Rede Neural Multicamada totalmente conectada	22
Figura 4 – Gráficos da função de custo $J(w_1, w_2)$	25
Figura 5 – Exemplo - mapa de características por convoluções	29
Figura 6 – Compartilhamento de parâmetros na operação de convolução	30
Figura 7 – Exemplo - operação de convolução	30
Figura 8 – Convolução em uma dimensão	31
Figura 9 – Exemplo - sub-amostragem de máxima	32
Figura 10 – Exemplo - de modelo <i>dropout</i>	34
Figura 11 – Relação semântica entre um vocabulário	37
Figura 12 – Arquitetura da rede neural convolucional LeNet	39
Figura 13 – Arquitetura da rede neural convolucional LeNet	40
Figura 14 – Aprendizado residual.	42
Figura 15 – Metodologia proposta	44
Figura 16 – Exemplo - nota Mínima escrita por diferentes músicos	45
Figura 17 – Exemplo - imagem gerada da base de dados HOMUS	46
Figura 18 – Retinografias de olho glaucomatoso e normal	47
Figura 19 – Exemplo - imagem resultado da base RIM-ONE	47
Figura 20 – Rede Neural Residual utilizada	50
Figura 21 – Gráfico - Resultados com a base HOMUS e a arquitetura LeNet utilizando a função de custo entropia cruzada	61
Figura 22 – Gráfico - Resultados com a base HOMUS e a arquitetura LeNet utilizando a função de custo adaptada	62
Figura 23 – Resumo dos resultados obtidos com a arquitetura LeNet	66
Figura 24 – Gráfico - Resultados com a base RIM-ONE e a arquitetura LeNet utilizando a função de custo entropia cruzada	68
Figura 25 – Gráfico - Resultados com a base RIM-ONE e a arquitetura LeNet utilizando a função de custo adaptada	69
Figura 26 – Gráfico - Resultados com a base IMDB e a arquitetura LeNet utilizando a função de custo entropia cruzada	72
Figura 27 – Gráfico - Resultados com a base IMDB e a arquitetura LeNet utilizando a função de custo adaptada	73
Figura 28 – Gráfico - Resultados com a base HOMUS e a arquitetura ResNet-30	77
Figura 29 – Gráfico - Resultados com a base RIM-ONE e a arquitetura ResNet-30	79
Figura 30 – Gráfico - Resultados com a base IMDB e a arquitetura ResNet-30	80

Figura 31 – Resumo dos resultados obtidos com a arquitetura ResNet-30 81

Lista de tabelas

Tabela 1 – Resumo dos trabalhos relacionados	18
Tabela 2 – Exemplo - Codificação de frase	37
Tabela 3 – Estrutura das redes VGG-16 e VGG-19	41
Tabela 4 – Exemplo - Análise de sentimento em texto	49
Tabela 5 – LeNet: Combinação de campos receptivos.	52
Tabela 6 – LeNet: Conjuntos de hiper-parâmetros <i>stride</i>	52
Tabela 7 – Instâncias da rede LeNet para a base RIM-ONE	53
Tabela 8 – Instâncias da rede LeNet para a base HOMUS	53
Tabela 9 – Instâncias da rede LeNet para a base IMDB	54
Tabela 10 – LeNet: Quantidade de filtros nas camadas de convolução	54
Tabela 11 – VGG-16: Campo receptivo para as camadas de convolução	55
Tabela 12 – Combinações válidas da redes VGG-16 para a base HOMUS e RIM-ONE	55
Tabela 13 – VGG-16: Quantidade de filtros nas camadas de convolução	56
Tabela 14 – ResNet: Quantidade de canais nas camadas de convolução	56
Tabela 15 – Resultados obtidos com os modelos baseados na VGG-16	58
Tabela 16 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo entropia cruzada para o reconhecimento de símbolos musicais.	63
Tabela 17 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo adaptada para o reconhecimento de símbolos musicais.	64
Tabela 18 – Resumo das melhores arquiteturas baseadas na LeNet	66
Tabela 19 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo entropia cruzada para o diagnóstico de glaucoma.	70
Tabela 20 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo adaptada para o diagnóstico de glaucoma.	71
Tabela 21 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo entropia cruzada para a análise de sentimentos em texto.	74
Tabela 22 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo adaptada para a análise de sentimentos em texto.	75
Tabela 23 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo entropia cruzada para o reconhecimento de símbolos musicais.	76

Tabela 24 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo adaptada para o reconhecimento de símbolos musicais.	78
Tabela 25 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo adaptada para o diagnóstico de glaucoma.	78
Tabela 26 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo entropia cruzada para o diagnóstico de glaucoma.	78
Tabela 27 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo entropia cruzada para a análise de sentimentos em texto.	80
Tabela 28 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo adaptada para a análise de sentimentos em texto.	81
Tabela 29 – Resumo das melhores arquiteturas baseadas na ResNet-30	81

Lista de abreviaturas e siglas

CNN	<i>Convolutional Neural Network</i>
CDR	<i>Cup-to-disc</i>
DNN	<i>Deep Neural Network</i>
DTW	<i>Dynamic time warping</i>
GRU	<i>Gated Recurrent Unit</i>
GPU	<i>Graphics Processing Unit</i>
HOS	<i>Higher Order Spectra</i>
K-NN	<i>K-Nearest Neighbours</i>
LSTM	<i>Long Short-Term Memory</i>
LSVRC	<i>Large Scale Visual Recognition Competition</i>
MSE	<i>Mean Square Error</i>
MLP	<i>Multilayer Perceptron</i>
MVS	Máquina de Vetores de Suporte
OMR	<i>Optical Music Recognition</i>
PLN	Processamento de Linguagem Natural
RNA	Redes Neurais Artificiais
ReLu	<i>Rectified Linear Unit</i>
RNN	<i>Recurrent Neural Network</i>
ResNet	<i>Residual Network</i>
SGD	<i>Stochastic Gradient Descendent</i>
SVD	<i>Singular Value Decomposition</i>

Sumário

1	INTRODUÇÃO	14
1.1	Objetivo	15
1.1.1	Objetivos Específicos	15
1.2	Contribuições	16
1.3	Trabalhos Relacionados	16
1.4	Organização do Trabalho	19
2	FUNDAMENTAÇÃO TEÓRICA	20
2.1	Redes Neurais Artificiais	20
2.1.1	Redes Neurais Multicamada	22
2.1.2	Função de Custo	24
2.1.3	Gradiente Descendente Estocástico	24
2.1.4	<i>Backpropagation</i>	26
2.2	Redes Neurais Convolucionais	28
2.2.1	Convolução	29
2.2.2	Sub-amostragem	32
2.2.3	Camadas de Ativação	33
2.2.4	<i>Dropout</i>	34
2.2.5	Treinamento	34
2.3	Texto e CNNs	36
2.3.1	Encoders	36
2.3.2	<i>Embedding</i>	37
2.4	Arquiteturas CNN	38
2.4.1	LeNet	38
2.4.2	AlexNet	39
2.4.3	Redes VGG	40
2.4.4	Redes Neurais Residuais	41
3	METODOLOGIA	44
3.1	Aquisição	44
3.1.1	HOMUS: Classificação de Símbolos Musicais	44
3.1.2	RIM-ONE: Diagnóstico de Glaucoma	46
3.1.3	IMDb: Análise de Sentimentos em Avaliações de Filmes	48
3.2	Escolha das Redes Neurais Convolucionais	49
3.3	Estimação de Hiper-Parâmetros	51
3.3.1	LeNet: Hiper-Parâmetros	52

3.3.2	VGG: Hiper-Parâmetros	54
3.3.3	ResNet: Hiper-Parâmetros	56
3.4	Treino e Validação	57
4	RESULTADOS	58
4.1	Resultados: VGG-16	58
4.2	Resultados: LeNet	59
4.2.1	Reconhecimento de Símbolos Musicais	59
4.2.2	Diagnóstico de Glaucoma	64
4.2.3	Análise de Sentimentos em Avaliações de Filmes	65
4.3	Resultados: ResNet-30	75
4.3.1	Reconhecimento de Símbolos Musicais	75
4.3.2	Diagnóstico de Glaucoma	76
4.3.3	Análise de Sentimentos em Avaliações de Filmes	79
5	PUBLICAÇÕES	83
6	CONCLUSÃO	84
6.1	Trabalhos Futuros	85
	REFERÊNCIAS	86

1 Introdução

Deep Learning é uma técnica de aprendizado de máquina que utiliza-se de processamentos em camadas para tentar representar modelos matemáticos complexos. Na década de 1970 pesquisadores já utilizavam redes neurais de múltiplas camadas para aproximar hiperplanos. Anos depois, em 1998, LeCun utilizou uma nova abordagem de *Deep Learning*, atribuindo a cada uma dessas camadas uma função específica, com o objetivo de classificar números manuscritos (LECUN; CORTES; BURGESS, 1998). Essa nova abordagem hoje é chamada de Redes Neurais Convolucionais (*Convolutional Neural Networks - CNN*) e funciona de maneira similar às Redes Neurais Artificiais de Múltiplas Camadas.

Uma das principais características dessa nova abordagem é o uso de camadas de convolução¹ para a extração automática de características relevantes na geração de um modelo robusto e eficiente. Por outro lado, esse esforço é traduzido na escolha da arquitetura da rede neural a ser utilizada. Algumas arquiteturas CNN ganharam destaque nos últimos anos devido aos seus resultados obtidos durante o concurso ImageNet (DENG et al., 2009), dentre elas, estão a AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), VGGNet (SIMONYAN; ZISSERMAN, 2014), GoogleNet Inception (SZEGEDY et al., 2015) e as Redes Residuais (HE et al., 2015) que, em seus respectivos anos, ganharam o primeiro lugar nas categorias de classificação, segmentação ou localização.

O sucesso das redes neurais convolucionais impulsionou vários pesquisadores à proporem diversas arquiteturas e aplicações que utilizam redes neurais convolucionais. Muitas destas atingindo o estado da arte em diferentes áreas como visão computacional, reconhecimento de voz e processamento de linguagem natural (PLN), contudo, um consenso geral de qual a melhor arquitetura para um determinado grupo de problemas ainda não foi proposto. Além de variações arquitetônicas, redes convolucionais ainda podem sofrer variações internas, nos hiper-parâmetros de cada camada, alguns destes são a quantidade de filtros nas camadas de convolução, o tamanho deste filtro e seu deslocamento.

Tendo em vista o conjunto de possibilidades existentes quando se trabalha com redes neurais convolucionais, acreditamos que um estudo comparativo utilizando diversas redes neurais aplicadas em diferente problemas, se faz necessário para esclarecer relações entre arquiteturas, hiper-parâmetros e problemas. Com esta finalidade, este trabalho tem como objetivo encontrar a melhor adaptação de três arquiteturas de redes neurais convolucionais para três problemas distintos.

As CNNs utilizadas são: LeNet, VGG-16 e ResNet e os problemas de reconhecimento

¹ Principal camada das redes neurais convolucionais, possui filtros treináveis que realizam a extração automática de características.

estudados são a classificação de símbolos musicais, a identificação de glaucoma em imagens de fundo de olho e a análise de sentimento em texto. Assim como as redes utilizadas durante este trabalho, cada um destes problemas são discutidos no Capítulo 2.

1.1 Objetivo

O principal objetivo deste trabalho é fazer um estudo comparativo entre diferentes arquiteturas de Redes Neurais Convolucionais. Assim como analisar a influência dos hiper-parâmetros das camadas de *pooling* e convolução no desempenho das redes neurais convolucionais. Para isso, utilizaremos três redes neurais bem conhecidas: LeNet (LECUN et al., 1998), VGG-16 (SIMONYAN; ZISSERMAN, 2014) e ResNet (HE et al., 2015).

1.1.1 Objetivos Específicos

Especificamente este trabalho tem como objetivos:

- Implementar diversas abordagens de Redes Neurais Convolucionais, com variações de seus hiper-parâmetros, sobre três problemas diferentes: classificação de símbolos musicais, análise de sentimento de texto e a identificação retinografias com glaucoma;
- Observar e quantificar como as redes neurais convolucionais reagem em diferentes cenários.
 - O objetivo do reconhecimento de símbolos musicais é submeter as redes à análise de formas e contornos;
 - O objetivo de usá-la em imagens médicas permitem intercalar o problema de formas e tecidos através de imagens de origem não natural e de baixo contraste. Ainda, analisar o comportamento de diferentes modelos treinados com poucas amostras;
 - O objetivo de reconhecimento de sentimento tem como finalidade estudar como as redes neurais podem tratar linguagens naturais;
- Utilizar uma função de custo adaptada para reduzir o efeito de bases de dados desbalanceadas na geração dos modelos.
- Aprofundar conhecimentos de *deep learning* e contribuir para geração de ciência;
- Estudar e desenvolver mecanismos de aprendizado automático;
- Estudar e analisar o desempenho de modelos treinados com poucas amostras.
- Contribuir para o resgate de acervos históricos de música e com o tratamento de doenças regionais.

1.2 Contribuições

As principais contribuições deste trabalho são:

- Estudo comparativo de diferentes redes neurais convolucionais e de seus hiperparâmetros.
- Função de custo adaptada com o uso da sensibilidade média entre as classes de um problema. O objetivo é reduzir o efeito de *overfitting* em modelos treinados com bases desbalanceadas.
- Uso de redes neurais convolucionais no reconhecimento de símbolos musicais, diagnóstico de glaucoma e análise de texto.
- Análise de modelos treinados com poucas amostras.

1.3 Trabalhos Relacionados

Redes Neurais Convolucionais vem sendo responsáveis por importantes avanços na área de visão computacional. Sua capacidade de extrair características sem nenhum conhecimento prévio das mesmas tem inspirado trabalhos nas áreas de classificação de imagem (HE et al., 2015), detecção (JADERBERG et al., 2016), (FARFADE; SABERIAN; LI, 2015), segmentação (GIRSHICK et al., 2014), (DAI; HE; SUN, 2015) e geração de imagens sintéticas (DOSOVITSKIY; SPRINGENBERG; BROX, 2015), (GATYS; ECKER; BETHGE, 2015).

Utilizando uma CNN com uma camada de parâmetros dinâmicos por *Gated Recurrent Unit* (GRU), Noh, Seo e Han (2016) afirmam obter o estado da arte nas principais bases *question-answering*: DAQUAR, COCO-QA e VQA. O objetivo da camada dinâmica é adaptar o modelo utilizado para responder perguntas sobre a imagem de entrada. Esse problema é popularmente conhecido como *ImageQA* e se aproxima bastante do que se considera o completo entendimento de uma imagem.

Reed et al. (2016) aborda o problema de descrição de cenas utilizando CNN gerando diferentes codificadores de texto. O modelo proposto supera o atual estado da arte, na base de dados Caltech-UCSD Birds-200-2011, para classificação *zero-shot*, ou seja, sem total conhecimento das possíveis respostas.

Contudo, nem sempre a construção de uma arquitetura específica se faz necessária para solucionar um problema, em (HE et al., 2015) e (CHEN et al., 2016) podemos perceber o uso da mesma CNN sofrendo pequenas alterações para realizar tarefas distintas. Outro exemplo é a utilização das arquiteturas AlexLeNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) e GoogLeNet (SZEGEDY et al., 2015) que foram, originalmente, projetadas

para a competição Imagenet e, em (PEREIRA et al., 2016a), são utilizadas na análise de lesões mamárias. Ainda, outras aplicações com arquiteturas desse tipo podem ser encontradas em (HAFEMANN, 2014), onde utiliza-se redes neurais convolucionais, dentre outras, na classificação de áudios e em (GINNEKEN et al., 2015) para a detecção de nódulos pulmonares em tomografias.

Diversos trabalhos já foram propostos na área de classificação de símbolos musicais, mas poucos utilizando técnicas de *deep learning*. Miyao e Maruyama (2004) dividiram os símbolos musicais em notas e pauta. A detecção e classificação são realizadas utilizando Código Derivativo de 8 Direções (*Freeman Code*) e Máquinas de Vetores de Suporte obtendo um resultado de 98,80% utilizando uma base de dados com 250 símbolos.

Já em (REBELO; CAPELA; CARDOSO, 2010), quatro métodos de classificação foram estudados: máquinas de vetores de suporte (MVS), redes neurais artificiais (RNA), *K-Nearest Neighbor* e modelo oculto de Markov. O estudo foi feito sobre 3.222 notas musicais manuscritas e 2.521 símbolos musicais impressos por um computador, cada conjunto possuía 14 classes.

Em (CALVO-ZARAGOZA; ONCINA, 2014) a base de dados HOMUS é introduzida consistindo de 32 símbolos musicais, produzidos por 100 músicos diferentes, resultando em 15.200 amostras. Durante os experimentos, foram testados diferentes algoritmos para detectar símbolos musicais, utilizando SVM, K-NN com *Dynamic time warping* (DTW) como medida de dissimilaridade e outros cenários, os experimentos obtiveram uma taxa de erro superior a 15%. Valero-Mas et al. (2016) propõe métodos de ranqueamento como *Farthest Neighbor* e *Nearest Neighbor* como protótipo para seleção de características na mesma base de dados. A acurácia máxima obtida para a tarefa de classificação foi de 89,90%.

Ainda, em (PEREIRA et al., 2016c), avalia-se as redes LeNet, AlexNet e GoogLeNet na classificação de símbolos musicais sobre a mesma base, HOMUS. O melhor resultado foi obtido com a arquitetura GoogLeNet com acurácia de 96,01% e sensibilidade de 96,56%.

No diagnóstico de glaucoma, Maninis et al. (2016) propõe um método de segmentação de vasos sanguíneos e disco óptico utilizando uma rede convolucional baseada nas redes VGG e no modelo Inception. Testado em quatro bases públicas distintas, o método se mostrou eficiente nas tarefas e, segundo os autores, demonstra resultados mais consistentes do que anotações feitas por humanos para controle. Com objetivo similar, Zilly, Buhmann e Mahapatra (2015) faz uso de CNNs como extratores de características utilizadas na segmentação de discos ópticos.

Já Nayak et al. (2009) geram características relacionando o tamanho e distância do disco óptico, nervo óptico e os vasos sanguíneos, e então, classifica essas características utilizando uma rede neural artificial para diagnosticar pacientes com glaucoma. De maneira

similar em (MOOKIAH et al., 2012) utiliza-se características de *Higher Order Spectra* (HOS) e *Discrete Wavelet Transform* (DWT) que são, então, inseridas em um classificador MVS o que resultou em um modelo com acurácia de 95%. Utilizando a base RIM-ONE, Haleem et al. (2016) usa características regionais para obter um modelo automático denominado, pelos autores, de RIFM e acurácia de 94,40%.

Redes neurais convolucionais também vem sendo utilizadas em análise de sentimentos em textos. Trabalhos como (SANTOS; GATTI, 2014) propõem redes neurais convolucionais específicas para a análise de sentimentos em textos. E (KIM, 2014), utiliza uma rede simples de uma camada de convolução e *word vectors* obtendo modelos satisfatórios na tarefa de análise de sentimentos. Contudo, para a base IMDb Haleem et al. (2016) possuem os melhores resultado, com 94.4% de acurácia, utilizando redes neurais convolucionais e *bag-of-words* para realizar a codificação do texto.

Para fins de comparação, a Tabela 1 apresenta um resumo das técnicas e aplicações citadas nessa seção.

Tabela 1 – Resumo dos trabalhos relacionados.

Referência	Método	Aplicação
(HE et al., 2015)	ResNet	ImageNet
(NOH; SEO; HAN, 2016)	CNN + GRU	ImageQA
(REED et al., 2016)	CNN como codificadores de texto	Descrição de cenas
(CHEN et al., 2016)	ResNet	Segmentação de cérebro
(PEREIRA et al., 2016a)	AlexNet + GoogLeNet	Classificação de lesões mamarias
(HAFEMANN, 2014)	CNN	Classificador de áudio
(GINNEKEN et al., 2015)	CNN OverFeat + SVM	Deteção de nódulos em mamografias
(MIYAO; MARUYAMA, 2004)	Free man code + SVM	Classificação de símbolos Musicais
(PEREIRA et al., 2016c)	LeNet + AlexNet + GoogLeNet	
(MANINIS et al., 2016)	VGG + Inception	Segmentação de vasos em retinografias
(HALEEM et al., 2016)	RIFM	Diagnóstico de glaucoma
(MOOKIAH et al., 2012)	HOS + DWT + SVM	
(SANTOS; GATTI, 2014)	CNN própria	Análise de sentimentos em texto
(KIM, 2014)	CNN + word vectors	
(HALEEM et al., 2016)	CNN + <i>bag-of-words</i>	

1.4 Organização do Trabalho

Esta monografia está dividida em cinco capítulos. O primeiro contém a introdução aos problemas, objetivos gerais e específicos.

No segundo são abordados os conceitos de redes neurais e *deep learning* necessários para o entendimento deste trabalho, dentre eles: neurônio, convolução, sub-amostragem e hiper-parâmetros de uma rede neural convolucional.

No terceiro capítulo estão contidas as metodologias utilizadas, ordenadas de acordo com os problemas abordados. Em seguida, o quarto capítulo apresenta os resultados obtidos. Por último, apresenta-se as conclusões obtidas com a realização deste trabalho.

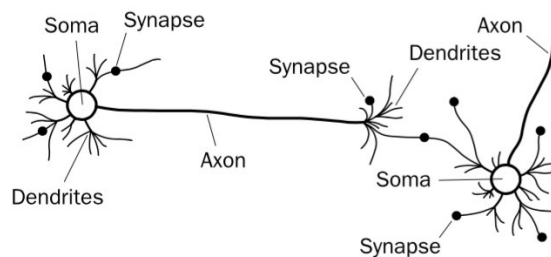
2 Fundamentação Teórica

Neste capítulo são discutidos os principais conceitos utilizados na realização deste trabalho e relacionados com o entendimento das Redes Neurais Artificiais. Junto com esses conceitos, apresenta-se como esses modelos podem ser representados matematicamente. O conceito de Redes Neurais Convolucionais é, então, formalizado juntamente com os tipos de camadas que mais aparecem nestas redes.

2.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA) são modelos computacionais que tentam simular o processo de aprendizagem presentes em organismos biológicos inteligentes. Geralmente, esses organismos aprendem por meio de experiência e repetição. O principal componente destes modelos são os chamados neurônios. Em seres vivos os neurônios são a unidade básica de um sistema nervoso, eles são constituídos de: um corpo principal, detritos, axônios e se comunicam por meio de sinapses nervosas, como apresentado na Figura 1.

Figura 1 – Estrutura de um neurônio biológico.

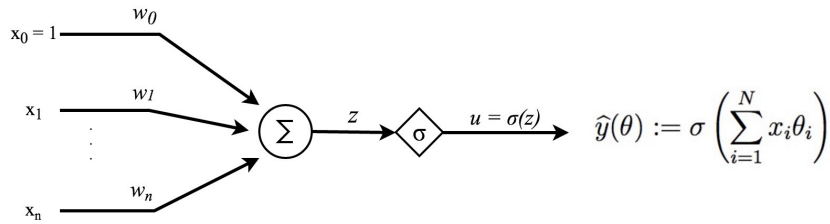


Fonte: (MICHAEL, 2005)

De maneira geral, cada neurônio processa no seu corpo principal o que recebe por meio das sinapses e, então, transmite uma cópia do sinal resultante para outras células que realizam o mesmo processo. Assim, o sinal é alterado até chegar no seu destino. Um processo similar acontece em neurônios artificiais onde o corpo do neurônio é representado por funções matemáticas e as sinapses por valores reais, geralmente denominados de pesos ou parâmetros.

A primeira concepção de neurônio artificial foi proposta por McCulloch e Pitts em (MCCULLOCH; PITTS, 1943) durante a segunda guerra mundial, mas só dez anos depois este modelo foi implementado no modelo chamado de *perceptron*. Através da Figura 2, pode-se verificar que em um neurônio artificial, a soma ponderada das entradas é

Figura 2 – Estrutura de um neurônio artificial, onde w_0 representa o *bias* e σ a função de ativação.



Fonte: Autor

transmitida à função de ativação que mapeia a entrada em um intervalo conhecido. Esse intervalo representa o nível de excitação do neurônio em relação a entrada s e parâmetros w . O funcionamento de um neurônio artificial pode ser modelado segundo a Equação 2.1:

$$z(w) = w_0 + x_1 w_1 + x_2 w_2 + \dots + x_n w_n = \sum_{i=1}^N x_i w_i + w_0 \quad (2.1)$$

onde, x_i é a entrada i , w_i é o peso associado a entrada i , w_0 é o termo bias e, σ é uma função não linear. Inserindo uma constante $x_0 = 1$, podemos escrever a Equação 2.2:

$$z(w) = \sum_{i=0}^N x_i w_i \quad (2.2)$$

e, adicionando a função de ativação do neurônio, podemos calcular o valor predito pelo neurônio (\hat{y}) como sendo:

$$\hat{y} = \sigma(z(w)) \rightarrow \hat{y}(w) = \sigma \left(\sum_{i=0}^N x_i w_i \right). \quad (2.3)$$

A função σ representa uma distribuição da entrada, no caso $z(w)$. Geralmente, em um neurônio artificial, essa função é determinada pelas funções sigmoide, ReLu ou pela função *hard-limit*. Essa última mapeia uma entrada em dois valores, ativo (1) ou inativo (0) e quando utilizadas, representam um *perceptron*. A função sigmoide pode ser definida como:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \quad (2.4)$$

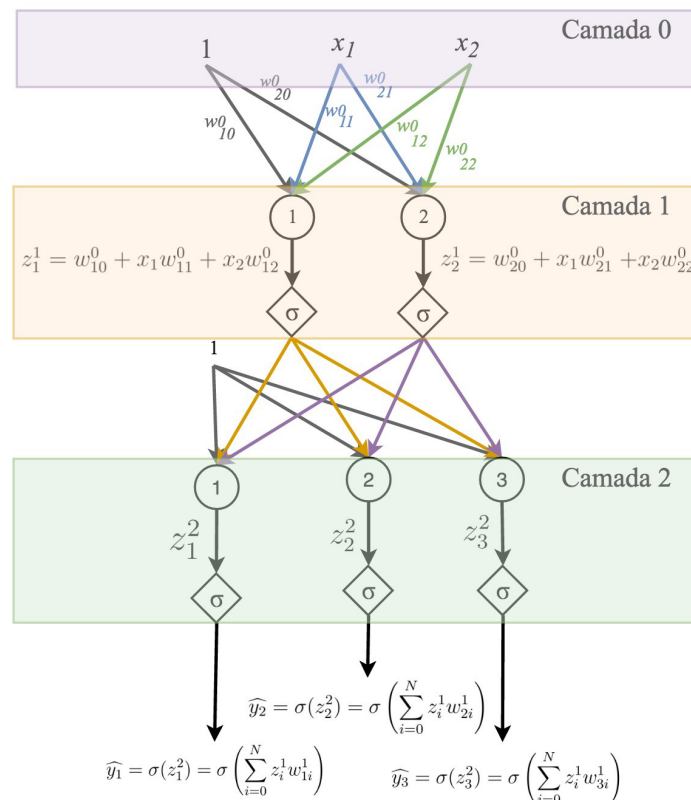
Mais detalhes sobre funções de ativação serão discutidos na Seção 2.2.3. Por enquanto, é necessário entender que camadas de ativação podem ser utilizadas para realizar a normalização da saída de um neurônio.

2.1.1 Redes Neurais Multicamada

Neurônios artificiais têm a capacidade de representar e solucionar problemas separáveis linearmente (MINSKY, 1969), como os operadores lógicos OR, NOR e AND, contudo, não são capazes de representar operadores mais complexos, como o XOR. Portanto, se faz necessário o uso de múltiplos neurônios, gerando o que chamamos de Rede Neural Artificial Multicamada (RNA), também chamada de *Multilayer Perceptron (MLP)* para neurônios do tipo *perceptron*.

Em modelos multicamada, a entrada da rede compõe a primeira camada da rede e é seguida de uma ou mais camadas intermediárias, chamadas de camadas ocultas. Por fim, a última camada contém a saída da rede. Essa arquitetura está exemplificada na Figura 3 em uma rede totalmente conectada, ou seja, todos os neurônios da camada l se ligam aos da camada seguinte $l + 1$, geralmente denotados por índices superiores z^l . No caso dos pesos, o primeiro índice inferior indica o neurônio de destino e o segundo o de origem. As camadas roxo, laranja e verde representam, respectivamente, a camada de entrada, uma camada oculta e a saída da rede.

Figura 3 – Rede Neural Multicamada totalmente conectada. Os índices superiores representam a camada a qual o elemento está relacionado.



Fonte: Autor

Quando uma rede neural artificial possui mais de uma camada escondida, ela é denominada de Rede Neural Profunda (*Deep Neural Network* - DNN). Uma rede neural artificial também pode ser representada por um conjunto de camadas empilhadas, onde, as camadas superiores representam as entrada, as ocultas são camadas nas quais ocorre a aprendizagem de características e, por último, as inferiores projetam uma saída ou predição.

Arquiteturas desse tipo são conhecidas por serem *feedforward*, ou seja, cada camada se conecta à próxima camada, porém não há caminho de volta. Redes nas quais há um caminho de voltas são ditas recorrentes, mas vão além do escopo deste trabalho.

O processo conhecido como *feedforward* consiste na transformação do conjunto de entrada em um conjunto de saída. Para isso, as saídas de cada neurônio de cada camada (l) tornam-se entrada dos neurônios da camada seguinte ($l + 1$), seguindo a Equação 2.3 que modela a ativação de um neurônio artificial. Assim, pode-se reescrever essa equação como:

$$u_j^{l+1}(w) = \sigma(z_j^{l+1}(w)) \rightarrow u_j^{l+1}(w) = \sigma\left(\sum_{i=0}^n u_i^l w_{ji}^l\right), \quad (2.5)$$

onde w_{ji}^l representa os pesos da camada l que servirão como entrada do neurônio j da camada $l + 1$. Como podemos perceber no exemplo apresentado na Figura 3 quando l é zero, os valores de u^0 correspondem a entrada da rede neural, por outro lado, quando l é última camada da rede, u^l equivale à \hat{y} .

Utilizando a notação de vetores, podemos representar a ativação de um neurônio qualquer como sendo:

$$u^{l+1} = \sigma(z^l \cdot (w) + b^{l+1}) \quad (2.6)$$

ou, como

$$u^{l+1} = \sigma(z^l \cdot (w^l)). \quad (2.7)$$

Por fim, para que a Equação 2.7 seja válida, devemos considerar que a função σ realiza suas operações elemento a elemento:

$$\sigma([v_1, v_2, v_3]) = [\sigma(v_1), \sigma(v_2), \sigma(v_3)] \quad (2.8)$$

Assim, podemos calcular os valores de predição (\hat{y}) como uma multiplicação de vetores (Equação 2.7).

2.1.2 Função de Custo

Em aprendizado de máquina, quando queremos avaliar um modelo, utilizamos uma função de custo. O objetivo de tais funções é avaliar o quão bom o modelo atual é em prever o conjunto \hat{y} . Podemos pensar que uma função de custo $J(y, \hat{y})$ realiza comparações entre o vetor y e o vetor \hat{y} e nos diz o quão diferente eles são.

Duas funções de custo bem conhecidas são o erro quadrático e a entropia cruzada. O primeiro método, descrito pela Equação 2.9 mede o quadrado da diferença entre o valor encontrado e o esperado para cada elemento i .

$$J(y, \hat{y}) = \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2.9)$$

Uma variação da função quadrática é o Erro Quadrático Médio (MSE), definida pela média da equação anterior, resultando em:

$$J(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \quad (2.10)$$

Já a entropia cruzada é definida como uma relação entre a distribuição verdadeira y e a distribuição predita \hat{y} como:

$$J(y, \hat{y}) = H(y, \hat{y}) = - \sum_{i=1}^N y(x_i) - \log(\hat{y})(x_i) \quad (2.11)$$

Na teoria, qualquer uma das duas funções pode ser utilizada com RNA. Na prática, contudo, utilizar o método de entropia cruzada tem ajudado modelos a convergirem mais rapidamente se tornando, assim mais popular ao longo dos anos (HAFEMANN, 2014).

Em estudos preliminares (PEREIRA et al., 2016b), observou-se que uma função de custo associando a sensibilidade dos termos com o custo total gera bons resultados quando aplicado a problemas em geral. Essa função de custo é definida como:

$$J_{se}(y, \hat{y}) = \frac{H(y, \hat{y})}{\frac{1}{C} \sum_{c=1}^C Se(y, \hat{y}, c)} \quad (2.12)$$

onde Se calcula a sensibilidade da classe c . Perceba que quanto maior a média da sensibilidade, menor o erro. O objetivo é analisar como uma função de custo que leva em consideração explicitamente a sensibilidade se comporta nos diferentes cenários estudados. Ao longo desse capítulo utilizaremos a Equação 2.9 para facilitar o entendimento do leitor.

2.1.3 Gradiente Descendente Estocástico

Com os conceitos vistos até agora, podemos prever, através do *feedforward*, e avaliar, por meio das funções de custo, o resultado de uma rede neural artificial. Contudo,

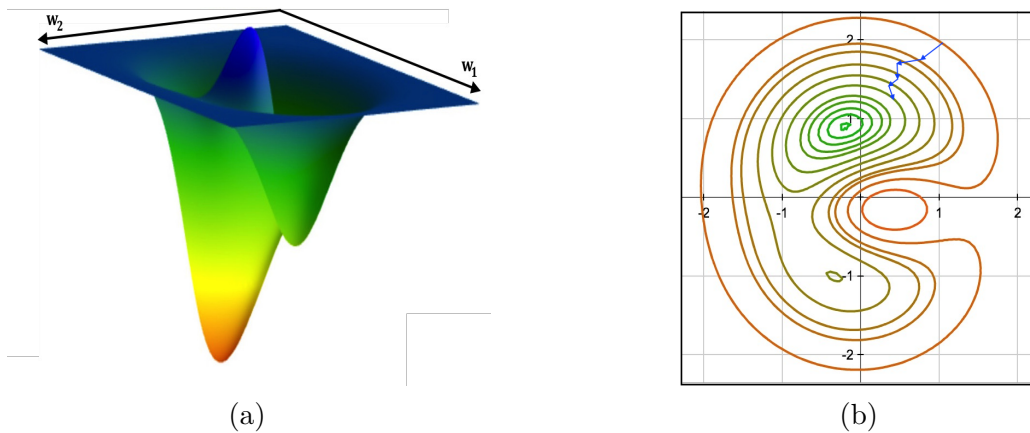
para o pleno funcionamento das redes neurais artificiais ainda é necessário definir um algoritmo que proporcione a melhor maneira de ajuste dos pesos w . O melhores pesos são aqueles que diminuem a diferença entre \hat{y} e y . Em outras palavras, precisamos de um método que nos permita encontrar o mínimo local ou global da função de custo $J(y, \hat{y})$ em função dos parâmetros w de uma rede neural artificial. Um método bem conhecido é o Gradiente Descendente Estocástico (*Stochastic Gradient Descendent* - SGD).

A Figura 4a exemplifica graficamente uma função de custo (J) baseada nos pesos w^l de uma camada l . Isso é possível já que \hat{y} está diretamente associado aos pesos de todas as camadas da rede (Equação 2.3). No exemplo dado, w é um vetor no espaço \mathbb{R}^2 e a função de custo definida por:

$$J(w_1, w_2) = -(w_1^2 - w_1 + 3 * w_2^2 + w_2) * e^{-w_1^2 - w_2^2}. \quad (2.13)$$

Na Figura 4a é possível visualizar o gráfico tridimensional desta função de custo, quanto mais próximo da cor laranja ou verde, mais próximo o valor do mínimo local. Já na Figura 4b, é possível perceber as curvas de nível da função de custo e como o SGD procura o mínimo local, cada interação do algoritmo é representado por uma linha em azul.

Figura 4 – Gráficos da função de custo representa da Equação 2.13.



Fonte: Autor

Desejamos encontrar um valor mínimo de J , ou seja um ponto p na curva, que faça $J(w)$ mínimo e $\frac{\partial J(p)}{\partial(w)} = 0$. Para tanto, podemos inicialmente considerar um ponto qualquer na curva, no nosso exemplo p_0 , e caminhar em direção ao valor mínimo local tomando a primeira derivada como direção. Uma analogia frequentemente utilizada é considerar uma bola lançada em uma malha qualquer, a tendência é que a bola chegue em um ponto mínimo, a força da resultante na bola coordena o sentido e a direção da mesma.

Quando variamos w_1 ou w_2 , fazemos com que o ponto p_i se mova para um novo ponto p_{i+1} representada por uma variação ΔJ , a relação entre ΔJ e Δw pode ser definida

como:

$$\Delta J \simeq \nabla J \cdot \Delta w, \quad (2.14)$$

onde ∇J é definida como o gradiente de J . Como estamos à procura de um mínimo, devemos gerar um valor $\Delta J \leq 0$, para tanto, podemos escolher Δw como sendo:

$$\Delta w := -\alpha \nabla J, \quad (2.15)$$

ou seja, os pesos w são movidos α unidades na direção contrária a ∇J (Figura 4b). Assim, a Equação 2.14 fica:

$$\Delta J \simeq \nabla J \cdot \Delta w = -\alpha \|\nabla J\|^2 \quad (2.16)$$

Já que $\|\nabla J\|^2$ é um valor positivo, temos sempre $\Delta C \leq 0$, ou seja, se mudarmos J de acordo com a Equação 2.16 o valor de J sempre irá se aproximar de um valor mínimo. Para calcular o novo valor w' de w , basta utilizar a correção:

$$w' = w - \alpha \nabla J \quad (2.17)$$

Se repetirmos esse processo iterativamente, temos o algoritmo de busca local SGD. Normalmente, este método é repetido por uma quantidade pré-definida de vezes ou até que o erro J seja suficientemente pequeno. Uma variação desse algoritmo consiste em realizar a busca local baseando-se no gradiente $\nabla \hat{J}$, estimado a partir de um subconjunto aleatório, de tamanho m , formado por elementos da entrada. Essa estimação é realizada por meio da média dos elementos. Esse subconjunto é chamado de *mini-batch*.

2.1.4 Backpropagation

O algoritmo anterior nos permite encontrar um mínimo local em um espaço utilizando somente o gradiente $\nabla J(w)$. Contudo, não chegamos a definir formalmente o que é o gradiente $\nabla J(w)$ e, mais importante ainda, como encontrá-lo. Nesta seção o faremos utilizando o algoritmo de retro-propagação (RUMELHART; HINTON; WILLIAMS, 1988), também chamado de *backpropagation*.

Matematicamente podemos definir $\nabla J(w)$ como sendo:

$$\nabla J(w) = \nabla J(w_1, w_2, \dots, w_n) = \left(\frac{\partial J(w)}{\partial w_1}, \frac{\partial J(w)}{\partial w_2}, \dots, \frac{\partial J(w)}{\partial w_n} \right) \quad (2.18)$$

onde

$$\frac{\partial J(w_i)}{\partial w_i} = \lim_{\delta \rightarrow 0} \frac{J(w_i + \delta) - J(w_i)}{\delta} \quad (2.19)$$

tal que δ_i é erro calculado no neurônio i . Relembrando a notação de vetores, utilizada na Equação 2.5, pode-se dizer que δ_i^l é o erro de um neurônio i na camada l e que δ^l é o erro associado à esta camada.

A partir da Equação 2.18, que define o valor de $u^{l+1}(w)$, e da definição de J segundo o erro quadrático (Equação 2.9), podemos definir a derivada de J em relação a saída i como sendo:

$$\frac{\partial J}{\partial \hat{y}_i} = \frac{\partial (y_i - \hat{y}_i(x, w))^2}{\partial \hat{y}_i} = -2(y_i - \hat{y}_i) \quad (2.20)$$

ou, ainda, por:

$$\frac{\partial J}{\partial u_i^l} = 2(u_i^l - y_i) \quad (2.21)$$

Utilizaremos essa definição mais a frente. Por ora, focaremos em $\frac{\partial J}{\partial w_i}$. Como dito anteriormente, δ_i representa o erro do neurônio i , podemos generalizar isso por meio da Equação 2.22:

$$\delta_i^l \equiv \frac{\partial J}{\partial z_i^l} \quad (2.22)$$

É importante ressaltar que, δ_i^l também pode ser escrito em função de u_i^l , mas dificulta consideravelmente o entendimento e desenvolvimento matemático, sendo assim continuaremos com a definição proposta. Utilizando a regra da cadeia, obtém-se:

$$\delta_i^l = \frac{\partial J}{\partial a_i^l} \frac{\partial u_i^l}{\partial z_i^l} \rightarrow \delta_i^l = \frac{\partial J}{\partial u_i^l} \sigma'(z_i^l) \quad (2.23)$$

Lembrando que σ é uma função de ativação. O primeiro termo da direita $\frac{\partial J}{\partial a_i^l}$ mede o quão rápido a função de custo varia em relação à ativação do neurônio i na camada l , quanto menos J é influenciado por esse neurônio, menor o valor de δ_i^l . O valor de $\frac{\partial J}{\partial u_i^l}$ varia de acordo com a definição da função de custo, no nosso caso, pode ser calculado pela Equação 2.21. Substituindo, chegamos a seguinte equação final:

$$\delta_i^l = 2(a_i^l - y) \sigma'(z_i^l) \quad (2.24)$$

que calcula o erro em uma camada l segundo a função de custo do erro quadrático. Analisando a Figura 3 é possível perceber que o erro δ^l está diretamente relacionado com o erro na camada posterior δ^{l+1} , já que só se pode conhecer o erro produzido nas camadas mais rasas quando se conhece o erro das camadas mais profundas. A partir da Equação 2.22, demonstra-se que:

$$\delta_j^l = \frac{\partial J}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_{jk}^{l+1}} \frac{\partial z_{jk}^{l+1}}{\partial z_j^l} \quad (2.25)$$

Em outras palavras, o valor de z na camada l pode ser calculado em função de z na camada subsequente $l + 1$. Pela definição proposta na Equação 2.22 é possível reescrever a

Equação 2.25 como sendo:

$$\delta_j^l = \sum_k \delta_{jk}^{l+1} \frac{\partial z_{jk}^{l+1}}{\partial z_{jk}^l} \quad (2.26)$$

e, baseando-se nas equações 2.3 e 2.2, que definem o valor de z , pode-se concluir que:

$$\frac{\partial z^l}{\partial z^{l+1}} = w^{l+1} \sigma'(z^l + 1) \quad (2.27)$$

Substituindo esse resultado na Equação 2.26, obtemos a relação entre o erro da camada l e a camada $l + 1$ descrita por:

$$\delta^l = (w^{l+1} \delta^{l+1}) \odot \sigma'(z^l) \quad (2.28)$$

onde \odot representa a multiplicação Hadamard, ou seja, multiplicação elemento a elemento. Um caso especial da Equação 2.22, que só agora é possível compreender, é quando $i = 0$, então esta variável referi-se ao valor do *bias* o que implica em:

$$\frac{\partial J}{\partial z_{i=0}^l} = \delta^l \quad (2.29)$$

Por fim, pode-se agora encontrar o valor de $\frac{\partial J}{\partial w_{jk}^l}$, que é o nosso problema original, como sendo a relação:

$$\frac{\partial J}{\partial w_{jk}^l} = u_k^{l-1} \delta_j^l \quad (2.30)$$

Da Equação 2.30 podemos concluir que, primeiramente, deve-se encontrar todos os valores de $u = \delta(z)$ das camadas de ativação, para só então calcular o erro δ^l que, por sua vez, será retro-propagado para as camadas anteriores a fim de ajustar os pesos das mesmas. Esse processo se repete até que seja alcançado um erro aceitável, o que pode demorar muito, ou por uma quantidade finita de vezes. Esse processo é conhecido como o processo de aprendizagem ou treinamento de uma RNA.

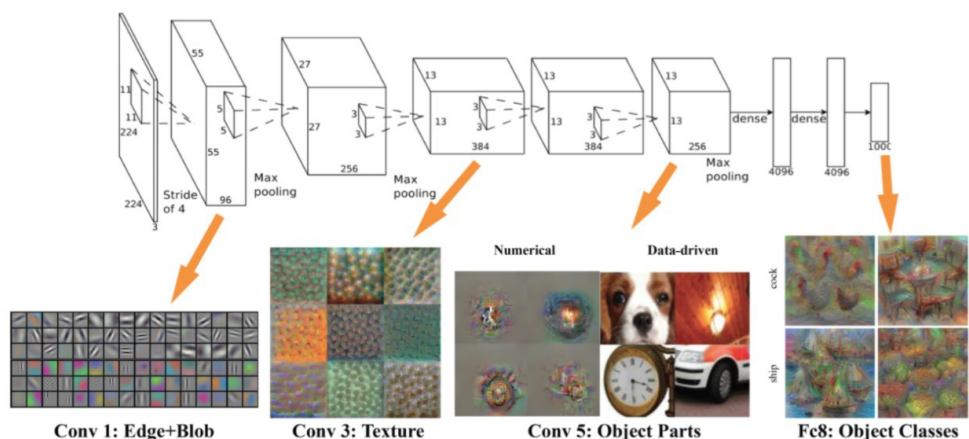
2.2 Redes Neurais Convolucionais

Inspirado no córtex visual de animais (HUBEL; WIESEL, 1968), redes neurais convolucionais (CNNs) continuamente subdividem imagens com o objetivo de extrair características da mesma. Uma rede neural convolucional realiza o aprendizado por meio dos métodos *feedforward* e *backpropagation* (RUMELHART; HINTON; WILLIAMS, 1988). Geralmente, existem quatro camadas distintas em uma CNN: convolução, sub-amostragem, ativação e uma RNA completamente conectada, as quais são objeto de explicação das seções a seguir.

2.2.1 Convolução

Camadas de convolução podem ser entendidas como conjuntos de filtros a serem aprendidos representados em forma de matrizes de ativação. Cada filtro é aprendido de uma parte diferente da imagem e, então, utilizado como extrator de características resultando em um mapa de ativação daquela característica.

Figura 5 – Exemplo de características aprendidas por cada convolução de uma rede neural baseada na AlexNet.



Fonte: Adaptado de: (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

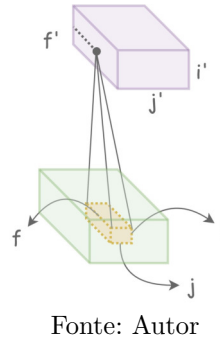
Cada camada de convolução tem como objetivo extrair um conjunto de características da camada imediatamente anterior a ela. É possível perceber, da Figura 5, que as camadas mais rasas da rede tendem a aprender características de mais baixo nível, como filtros de passa baixa e, a medida que convoluções em camadas mais profundas são realizadas, esses filtros envolvem para características mais complexas, como texturas e partes de objetos.

Além do aumento do poder computacional nos últimos anos, redes neurais convolucionais só são viáveis na prática devido ao compartilhamento de neurônios feito em cada camada de convolução. Em uma imagem colorida de tamanho 64x64, uma RNA totalmente conectada de 2 camadas, teria de lidar com mais de 12 mil pesos por neurônio, uma vez que cada entrada possui um peso que o conecta à um neurônio de saída. Essa quantidade de pesos cresce rapidamente a medida que se aumenta o tamanho da imagem ou a quantidade de camadas da RNA tornando o processo impraticável.

Uma CNN compartilha pesos. Ao invés de conectar um neurônio com todos os neurônios da camada consecutiva, o neurônio é conectado a um volume específico. A área desse volume é chamada de campo receptivo e é definida pelo tamanho do filtro (ou *kernel*) da convolução. A profundidade deste volume é sempre igual a profundidade do volume de entrada em cada camada. Sendo assim, para um filtro de tamanho 5x5 em uma imagem 64x64x3 (RGB), cada neurônio teria 75 ($5 * 5 * 3$) pesos à serem treinados (Figura 6).

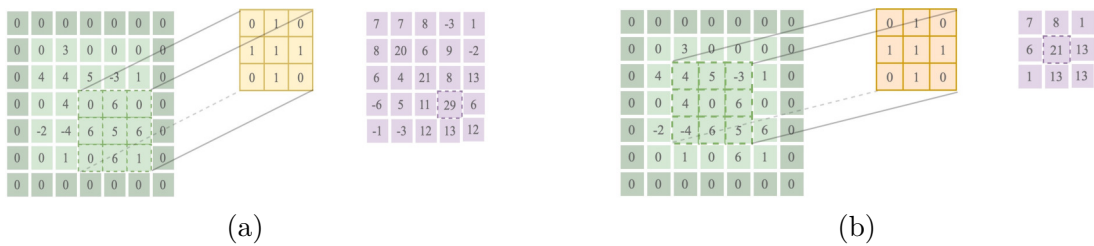
Além do tamanho do *kernel*, existem outros hiper-parâmetros nas camadas de

Figura 6 – Compartilhamento de parâmetros na operação de convolução. O volume em verde é o volume de entrada e o em roxo o de saída. Todo o volume em laranja possui somente um representante na saída.



convolução. Eles são: passo, *zero-padding*, e quantidade de filtros. A Figura 7 nos ajuda a visualizar e entender cada um deles e como eles influenciam na saída. No exemplo, ambas as convoluções possuem filtros de tamanho 3x3 e *zero-padding* 1x1. O filtro (amarelo) possui forma de "+", portanto, procura esse símbolo na entrada (verde). Na Figura 7a a operação é realizada com *stride* 1x1 e a saída possui tamanho 5x5. Já na Figura 7b a operação é realizada com *stride* 2x2 e a saída possui tamanho 3x3.

Figura 7 – Exemplo de convolução.



Fonte: Autor

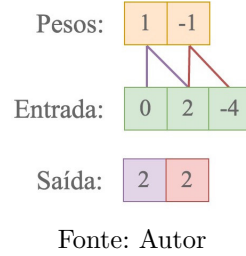
O passo, também chamado, em inglês, de *stride* representa o quão rápido os filtros irão se mover pela imagem. Por padrão, em convoluções, o passo é definido em 1x1, ou seja, os filtros se move 1 pixel por vez em uma das direções. Quando não queremos dispensar as bordas de uma imagem ou característica é conveniente adicionar uma nova borda zeros na imagem. O tamanho desta borda é definida pelo hiper-parâmetro *zero-padding*. Por fim, o número de filtros em uma camada de convolução determina quantos filtros devem ser treinados. Cada um deles observa e tenta aprender uma característica diferente no volume de entrada.

Para calcular a dimensão do volume de saída de uma camada de convolução, todos estes hiper-parâmetros são levados em conta. A Equação 2.31 formaliza essa relação.

$$o(w) = \frac{e - k + 2p}{s} + 1 \tag{2.31}$$

Os parâmetros e , k , p e s representam, respectivamente, o volume de entrada e os hiper-parâmetros tamanho do filtro, *zero-padding* e passo.

Figura 8 – Convolução em uma dimensão.



Observando a Figura 8, é possível perceber um padrão na saída. Cada operação que gera uma saída está representada por sua cor na ligação entre pesos e entrada. Neste caso, temos o tamanho do filtro como sendo 2 e seu passo em 1 unidade.

Entendendo cada quadrado como sendo um neurônio obtemos a ideia de uma convolução. Para uma dimensão, podemos modelar matematicamente essa ideia como:

$$z_{i'} = \sum_{i=1}^k w_i x_{i'+i-1} \quad (2.32)$$

tal que $i' + i - 1$ remete ao campo receptivo, ou seja, a área sobre a qual a operação será realizada. Inserimos a notação i' para representar o índice da saída, k como vimos anteriormente é o tamanho do filtro. A Equação 2.32 é denominada de correlação e mede o quanto um mesmo padrão se repete em uma entrada. A operação de convolução é uma variação dessa definição:

$$z_{i'} = \sum_{i=1}^k w_{k-i+1}^{(2)} x_{i'+i-1} \quad (2.33)$$

onde $w^{(2)}$ agora é o mesmo vetor w rotacionado, ou seja, se antes tínhamos $w = (w_1, w_2, \dots, w_n)$ agora temos $w^{(2)} = (w_n, w_{n-1}, \dots, w_1)$. Como as duas equações são iguais, manteremos a primeira definição para calcular a saída em K dimensões.

Primeiramente, vamos estender a definição para duas dimensões, para fazer isso, definiremos o valor de k como sendo (k_x, k_y) no espaço \mathbb{N}^2 . Assim, a Equação 2.32 fica:

$$z_{i',j'} = \sum_{j=1}^{k_y} \sum_{i=1}^{k_x} w_{i,j} x_{i'+i-1,j'+j-1} \quad (2.34)$$

A nova equação, simplesmente possui uma quantidade maior de índices. Se quisermos adicionar K camadas na operação de convolução, temos que manter o controle das camadas de entrada f e saída (f') dos neurônios em questão, como exemplificado na Figura 6. Assim, obtemos a equação abaixo:

$$z_{i',j',f'} = \sum_{j=1}^{k_y} \sum_{i=1}^{k_x} \sum_{f=1}^K w_{i,j,f,f'} x_{i'+i-1,j'+j-1,f} \quad (2.35)$$

que relaciona os neurônios da entrada i, j, f (largura, altura, profundidade) com um neurônio de saída $i' j' f'$.

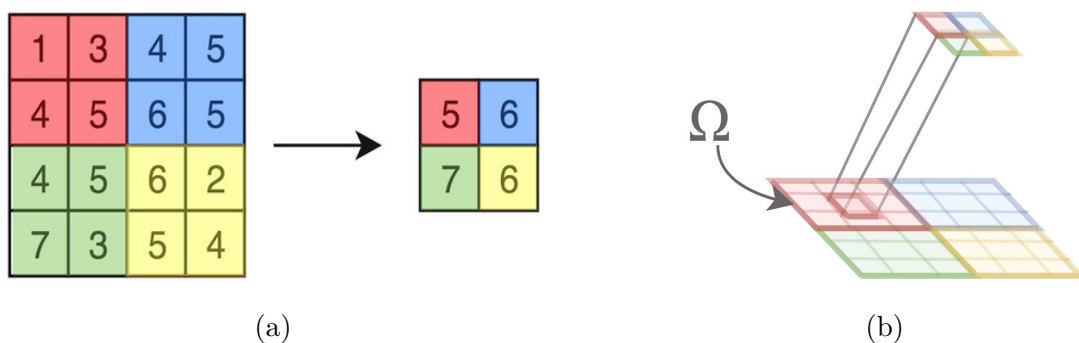
2.2.2 Sub-amostragem

Geralmente, camadas de convolução são seguidas de uma camada de sub-amostragem, também conhecidas como *pooling* ou seleção. Camadas desse tipo têm como objetivo representar estatisticamente a camada anterior. Utilizando funções tais como máximo ou média, a quantidade de dados é significativamente reduzida sem grandes perdas. Camadas de *pooling* geram filtros mais confiáveis e ajudam a prevenir *overfitting*.

Com exceção da quantidade de filtros que é sempre a mesma da entrada, uma camada de *pooling* qualquer possui todos os hiper-parâmetros presentes nas camadas de convolução. As funções são as mesmas, controlar o passo e o tamanho dos filtros.

A Figura 9a exemplifica a operação de *pooling*. Uma característica importante, que muitas vezes passa despercebida, é o fato de que nessas camadas geralmente não existem pesos à serem treinados. A entrada é analisada de acordo com seus valores atuais e resumida para um volume de saída de acordo com o tipo de *pooling* escolhido. Reduzindo a resolução dos mapas torna as características invariantes a translação, rotação e escala.

Figura 9 – Exemplo - sub-amostragem de máxima. **Em (a):** Mapeamento da matriz de entrada (esquerda) para a matriz de saída (direita). **Em (b):** Visualização de uma convolução durante a retro-propagação, Ω representa o campo receptivo de cor vermelha.



Fonte: Autor

As dimensões do volume de saída após sofrer uma operação de *pooling* pode ser calculado da mesma maneira das camadas de convolução (Equação 2.31). Nas camadas de sub-amostragem, normalmente, o tamanho do passo é o mesmo tamanho do filtro (k), de modo que não exista sobreposição em passos consecutivos. A profundidade ou número de canais é sempre a mesma do volume inicial.

Quando o passo é menor do que o tamanho do filtro, algumas vezes chamado de *kernel*, diz-se que há sobreposição na operação. Algumas redes utilizam essa abordagem

com o objetivo de analisar características vizinhas entre si, contudo, é mais comum encontrar camadas que não sofrem sobreposição.

Para definir matematicamente a camada de sub-amostragem, definiremos o símbolo Ω como sendo o campo receptivo, ou seja, a área que corresponde ao tamanho do *kernel* do *pooling*, essa definição pode ser vista na Figura 9b. Sendo assim, temos:

$$z_{i'j'} = \max_{ij \in \Omega(i'j')} x_{ij}, \quad (2.36)$$

onde $i'j'$ representa o neurônio de saída e $i'j'$ o neurônio selecionado com a função max. A expressão $ij \in \Omega(i'j')$ nos diz que o campo de busca para ij é o campo receptivo de $i'j'$, denotado por $\Omega(i'j')$.

Por fim, é importante perceber que, dependendo da escolha de hiper-parâmetros, uma camada pode resultar em uma saída inválida. Para isso, basta que os hiper-parâmetros escolhidos em uma camada resultem em um volume nulo ou negativo, neste caso, considera-se que a operação impossível de ser realizada.

2.2.3 Camadas de Ativação

Como no início do capítulo, camadas de ativação são funções matemáticas com o objetivo de mapear a entrada em uma distribuição conhecida. Para tanto, algumas funções utilizadas são: linear, *hard-limit* (ou limiar), a *Rectified Linear Unit* (ReLU) (NAIR; HINTON, 2010) e a sigmoide. As três primeiras estão representadas nas equações 2.37, 2.38 e 2.39, respectivamente. A função sigmoide foi definida anteriormente pela Equação 2.4.

$$f(x) = x \quad (2.37)$$

$$f(x) = \begin{cases} x, & \text{se } x \geq \theta \\ 0, & \text{caso contrário} \end{cases} \quad (2.38)$$

A função mais utilizada atualmente, a ReLu, geralmente diminui o tempo de treino. Na sua forma tradicional, sua equação é a mesma do limiar, mas com o valor de θ fixo em 0. Normalmente, as camadas de convolução quando implementadas por bibliotecas e *frameworks* possuem uma função linear na sua saída. Formalmente, podemos definir a Equação 2.39 como o modelo matemático das camadas ReLu.

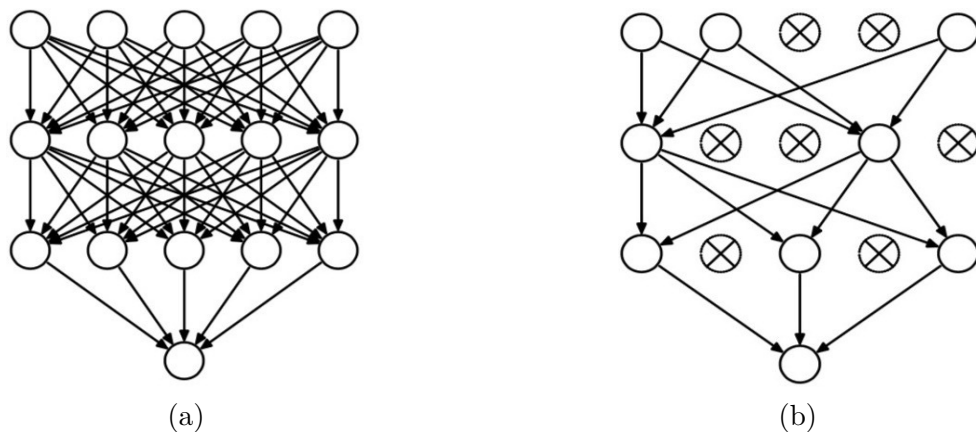
$$f(x) = \max(0, x) \quad (2.39)$$

onde $\max(0, x)$ retorna o maior valor entre 0 e x .

2.2.4 Dropout

Outra camada importante presente em algumas redes neurais é a camada de *dropout* (SRIVASTAVA et al., 2014). Este tipo de camada afeta somente o treino tentando prevenir o super-ajuste de neurônios nas redes neurais artificiais. A ideia é simples, porém, eficaz. A cada época de treino ou *batch* treinado, um sorteio sobre os neurônios envolvidos na MLP é feito e alguns são selecionados e excluídos da próxima interação com probabilidade p que é o único hiper-parâmetro da camada. A Figura 10 ilustra o funcionamento deste método, na esquerda (Figura 10a) é ilustrado uma MLP completamente conectada e na direita (Figura 10b) a mesma rede após a aplicação do *dropout*.

Figura 10 – Exemplo de modelo *dropout*.



Fonte: (SRIVASTAVA et al., 2014)

2.2.5 Treinamento

Agora que já conhecemos as principais camadas para extração de características, só nos falta uma etapa: classificar essas características extraídas. Geralmente, para realizar tal tarefa utiliza-se uma RNA completamente conectada como última camadas. Estudo com o uso de outros classificadores também vem sendo realizados (GENG et al., 2016), mas vão além do escopo deste trabalho.

Já vimos que para treinar uma rede neural temos de calcular duas coisas, o erro $J^l(w^l)$ para cada camada l e suas derivadas parciais em relação ao vetor w^l . Em outras palavras, para realizar o ajuste de pesos em uma rede é necessário realizar os processos de *feedforward* e de retro-propagação (Seções 2.1.3 e 2.1.4). Então tudo que precisamos descobrir é como calcular esses valores para as novas camadas: convolução, *pooling* e, eventualmente, para ativações do tipo ReLU.

Também já sabemos como encontrar o valor de z_{ijk}^l (Equação 2.35 e Equação 2.36) para as camadas de convolução e *pooling*. Portanto, podemos calcular as derivadas necessárias para realizar o algoritmo de retro-propagação nessas camadas. Voltemos nossa

atenção primeiramente para as camadas de convolução, podemos calcular o valor de $\frac{\partial J}{\partial w}$ por meio da Equação 2.40.

$$\frac{\partial J}{\partial w_{ijkff'}} = \sum_{i'j'f'} \delta_{i'j'f'}^{l+1} \frac{\partial z_{i'j'f'}(x, w_{f'})}{\partial w_{ijkff'}} \quad (2.40)$$

De maneira similar ao que fizemos na Seção 2.1.3, podemos chegar a conclusão que a Equação 2.40 pode ser traduzida em:

$$\frac{\partial J}{\partial w_{ijkff'}} = \sum_{i'j'f'} \delta_{i'j'f'}^{l+1} x_{i'+i-1, j'+j-1, f'} \quad (2.41)$$

Agora, só nos resta calcular o valor de $\delta_{i'j'f'}^{l-1}$, que corresponde ao erro do neurônio $i'j'f'$ na camada $l + 1$, para um l e um neurônio qualquer, esse valor pode ser encontrado pela equação:

$$\delta_{ijf}^l = \sum_{i'j'f'} \delta_{i'j'f'}^{l+1} \frac{\partial z_{i'j'f'}(x, w_{f'})}{\partial x_{ijf}} \quad (2.42)$$

Sabemos que, no membro direito, para um valor qualquer, tal que, $i \neq i' - i'' + 1$ ou $j \neq j' - j'' + 1$ a derivada é x_{ijf} igual a zero. Assim, pode-se rescrever a Equação 2.42 como sendo:

$$\delta_{ijf}^l = \sum_{i'j'f'} \delta_{i'j'f'}^{l+1} w_{i'-i+1, j'-j+1, f', f'} \quad (2.43)$$

Utilizando as equações 2.1.3 (feedforward), 2.41 e 2.43 podemos treinar uma rede com uma camada de convolução. Para camadas de *pooling* o processo é similar, mas varia dependendo do tipo de camada (máximo, média ou mínima). Camadas de sub-amostragem do tipo máximas são mais frequentes em implementações de CNNs e, por tanto, a partir de agora, será omitido o tipo de sub-amostragem utilizada e todas serão subentendida como operações de máxima, a menos que explicitado.

Segundo a definição de uma camada de *pooling* de máxima (Equação 2.36), somente o elemento mais ativado da camada l será mantido na camada $l + 1$. Baseando-se na Equação 2.42, defini-se δ_{ij}^l para as camadas de *pooling* como sendo:

$$\delta_{ij}^l = \sum_{i'j'} \delta_{i'j'}^{l+1} \frac{\partial z_{i'j'}(x)}{\partial x_{ij}} \quad (2.44)$$

Vimos que esse tipo de *pooling* considera somente o maior valor dentre os valores de $\Omega(i, j)$ (campo receptivo) (Figura 9). O valor de δ é retro-propagado de maneira similar, contudo, somente o neurônio $i'_s j'_s$ selecionado receberá o valor de δ da camada subsequente, os outros neurônios não receberão ajuste algum.

Essa metodologia funciona quando deseja-se classificar um modelo, mas deixa a desejar quando o objetivo é o inverso. Dada uma classe, mostrar o modelo. Isso por que as

informações que não foram repassadas para o *pooling* de máxima são ignoradas. Visando resolver esse problema, como já dito outros tipos de *pooling* estão sendo estudados, como o de média.

Voltando ao problema original, agora pode-se redefinir a Equação 2.44 em função desse único neurônio que será ativado e obtém-se:

$$\delta_{ij}^l = \delta_{ij}^{l+1} \mathbb{I}_{j=\operatorname{argmax}_{i'_s j'_s} x_{ij}} \quad (2.45)$$

o elemento $\mathbb{I}_{j=\operatorname{argmax}_{i'_s j'_s} x_{ij}}$ representa a escolha do neurônio ao qual o valor deve ser retro-propagado. Com o que foi mostrado até aqui é possível modelar uma rede neural artificial.

Por fim, para as camadas de ativação do tipo ReLu, a retro-propagação dos seus valores é bem simples. Basta calcular a derivada da Equação 2.38:

$$\frac{\partial J}{\partial u^l} = \begin{cases} 0, & \text{se } u^{l+1} \leq 0 \\ \frac{\partial J}{\partial u^{l+1}}, & \text{caso contrário} \end{cases} \quad (2.46)$$

2.3 Texto e CNNs

Redes Neurais Convolucionais normalmente são associadas com o campo de visão computacional. Contudo, como visto CNNs também são responsáveis por grandes avanços no processamento de linguagem natural (PLN).

Para operar análises textuais com redes neurais convolucionais, se faz necessário expressar esses dados em forma de vetor. Para tanto, existem diversas maneiras, neste trabalho, nos focaremos em utilizar codificadores (*encoders*) juntamente com operações 1D e camadas do tipo *embedding*.

2.3.1 Encoders

Até então, assumiu-se que CNNs aceitam somente imagens como entrada, mas na verdade, imagens não passam de uma representação matricial de um dado. Tendo isto em mente, o primeiro passo para analisar textos utilizando CNNs é codificar seu conteúdo em uma matriz, para isso, indexaremos cada palavra de acordo com a sua frequência na base de dados.

Tomando como exemplo a seguinte frase "Este é um ótimo filme. Recomendo a todos que forem ao cinema.", dependendo do restante da base, a frase de exemplo poderia ser mapeada, levando em conta a frequência de suas palavras, como na Tabela 2, onde a primeira linha representa o índice na codificação e a segunda a palavra codificada.

Com esta abordagem, em textos grandes é possível extrair somente o conteúdo relevante para classificação, ou seja, aquelas palavras que diretamente influenciam no sentido do texto, como "Recomendo", "ótimo" e "filme", no exemplo dado.

Tabela 2 – Codificação da frase "Este é um ótimo filme. Recomendo a todos que forem ao cinema." de acordo com a frequência de palavras.

1	2	3	4	5	6	7	8	9	10	11	12
é	a	que	um	este	ao	Recomendo	ótimo	filme	cinema	todos	forem

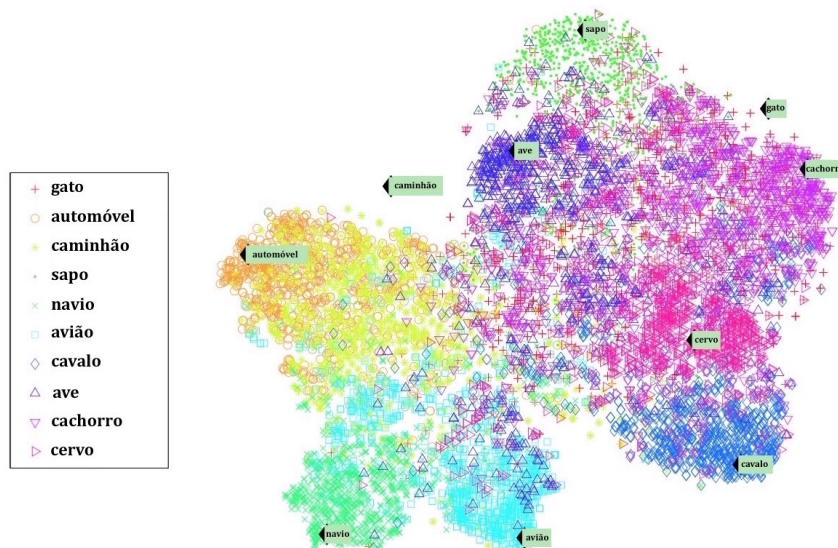
2.3.2 Embedding

Normalmente, redes neurais do tipo *feed-forward* que utilizam palavras de um vocabulário têm como uma de suas primeiras camadas uma do tipo *embedding*. O objetivo é realizar uma transformação linear $W : p \rightarrow \mathbb{R}^n$, tal que p é uma palavra sofrendo a transformação W para um espaço de dimensão n . A função W possui também parâmetros θ que é uma matriz mapeando cada palavra p , tal que $W_{\theta}(p_n) = \theta_n$. A medida que ocorre o aprendizado, W se ajusta para uma tarefa específica.

Vale notar que quando realizada esta transformação sobre todos os elementos de uma frase com k palavras, essa operação resultaria em um volume de dimensão (k, n) que pode ser entendida como a "imagem" de entrada de uma CNN.

Normalmente camadas do tipo *embedding* relacionam palavras de acordo com sua proximidade semântica. Para isso, pode-se utilizar diferentes métodos, o mais básico deles utiliza a hipótese de Firth (FIRTH, 1957) que tenta dar sentido a uma palavra baseando-se nas palavras que aparecem próximas a ela. Suponha um dicionário de N palavras distintas, presentes no corpo de um texto, ainda, considere a operação $C_5(w_1, w_2)$ como a quantidade de vezes que as palavras w_1 e w_2 aparecem a uma distância máxima de 5 palavras no texto. Então, o *embedding* de uma palavra w é um vetor de dimensão N com um índice para cada palavra do dicionário. O conteúdo de cada índice k é o valor de $C_5(w, w_k)$.

Figura 11 – Relação semântica entre um vocabulário.



Fonte: Adaptado de (SOCHER et al., 2013)

Existem também outros métodos mais sofisticados de alcançar tal relação como o *singular value decomposition* (SVD) e os modelos baseados em energia como o *Word2Vec*. A Figura 11 exemplifica o resultado desta relação, onde os objetos que estão mais próximos possuem maior relação semântica como cachorro e gato que são dois animais domésticos ou, ainda, avião e navio que são meios de transporte.

Ainda, em alguns casos, analogias mais sofisticadas podem surgir como $W(\text{"menino"}) - W(\text{"menina"}) \simeq W(\text{"avô"}) - W(\text{"avó"})$. Em outras palavras, operações de soma e subtração dos vetores no espaço \mathbb{R}^n podem resultar em relações extremamente complexas. Aproximações desse tipo, auxiliam em muitos processos de linguagem natural.

2.4 Arquiteturas CNN

Grande parte da popularidade atual das redes neurais convolucionais se deve aos seus resultados obtidos pelos autores [Krizhevsky, Sutskever e Hinton \(2012\)](#) na edição da ImageNet ([DENG et al., 2009](#)) em 2012. Os autores utilizaram uma técnica proposta na década de 1990 por YanLecun ([LECUN et al., 1998](#)). Como já dito anteriormente, outro fator que influenciou e influencia estudos sobre redes neurais convolucionais é o aumento do poder computacional que vem acontecendo desde o início do século. Algumas arquiteturas bem conhecidas são discutidas a seguir.

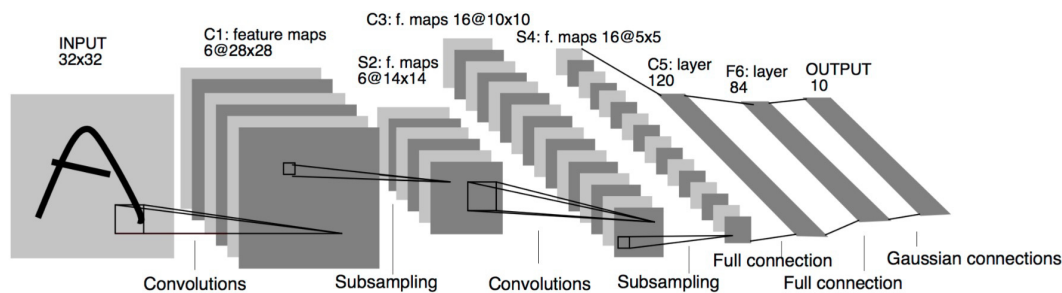
2.4.1 LeNet

Pioneira na área de redes neurais convolucionais, a arquitetura LeNet ([LECUN et al., 1998](#)) foi idealizada por YanLecun, em 1989, com o objetivo de classificar números manuscritos. A base de dados utilizada na época é conhecida como MNIST ([LECUN; CORTES; BURGES, 1998](#)) e é composta por 60 mil imagens de dígitos manuscritos. Nas suas implementações mais robustas, essa arquitetura obtém resultados superiores a 99% de acurácia.

Como discutido na Seção 2.2, geralmente redes neurais são formadas, principalmente, de camadas empilhadas de convolução e *pooling* seguidas de uma MLP para classificação. No caso da arquitetura LeNet, temos duas camadas de convolução seguidas por uma camadas de *pooling* cada totalizando sete camadas, como podemos ver na Figura 12. O número anterior ao símbolo "@" representa a quantidade de filtros presentes na camada e é seguido pelo tamanho (altura x largura) da imagem na camada atual.

Originalmente, a rede foi desenvolvida para classificar imagens em escala de cinza de dimensões 32x32 *pixels*. A primeira convolução (C1) extrai seis mapas de características, uma para cada filtro treinável, cada um destes filtros possui tamanho 5x5 e passo 1x1. Na camada seguinte (S2), um processo de sub-amostragem é realizado com os hiper-parâmetros 2x2 tanto para o tamanho do filtro como para o passo ao qual o filtro se move na imagem.

Figura 12 – Arquitetura da rede neural convolucional LeNet.



Fonte: (LECUN et al., 1998)

Como resultado, temos como saída um volume $14 \times 14 \times 6$ representando as características aprendidas.

Seguido disso, outra camada de convolução é empilhada, mas dessa vez com 16 filtros à serem aprendidos, os outros hiper-parâmetros se mantêm os mesmos da convolução C1. Outra camada de *pooling* similar a camada S2 é adicionada na rede. Por fim, uma rede neural artificial é adicionada ao final da rede para realizar a classificação das características nas classes desejadas.

2.4.2 AlexNet

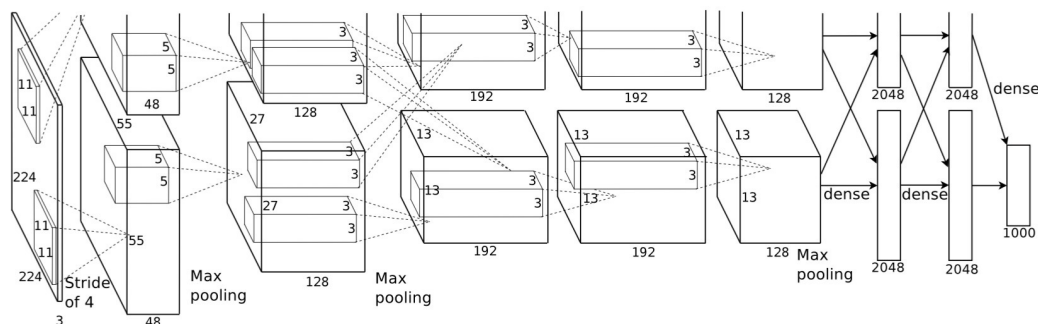
Como dito anteriormente, a arquitetura proposta em 2012 denominada AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) é uma das grandes responsáveis pelo sucesso das redes neurais convolucionais. Originalmente, essa arquitetura foi projetada para a classificação de imagens naturais na competição Imagenet LSVRC-2012, obtendo os menores erros, cerca de 10% a menos que o segundo colocado, na categoria de classificação top-5.

Como mostra a Figura 13 a arquitetura da rede consiste de oito camadas otimizadas para trabalhar em múltiplas GPUs, no caso duas. O modelo desenvolvido pelos autores coloca metade dos neurônios em cada GPU, sendo que as GPUs se comunicam entre si somente em algumas camadas. Isso diminui a sobrecarga em cada GPU, diminuindo o tempo de treino.

Além do uso de múltiplas GPUs, os autores também utilizam *poolings* com sobreposição. Os hiper-parâmetros passo e tamanho de *kernel*, da camada de sub-amostragem, são fixados em, respectivamente 2 e 3 *pixels* fazendo com que o mesmo filtro considere uma mesma área mais de uma vez. Os autores acreditam que realizar operações deste tipo com sobreposição ajudam a diminuir a probabilidade de *overfitting*.

A última camada da rede está ligada a uma função softmax que produz uma distribuição probabilística sobre as 1000 classes presentes na Imagenet. As camadas 2, 4 e

Figura 13 – Arquitetura da rede neural convolucional LeNet.



Fonte: (KRIZHEVSKY; SUTSKEVER; HINTON, 2012)

5 consideram como entrada a saída da camada imediatamente anterior a ela na mesma GPU, enquanto nas outras camadas, as entradas consideram ambas GPUs.

2.4.3 Redes VGG

As arquiteturas VGGNet (SIMONYAN; ZISSERMAN, 2014) são um conjunto de redes da mesma família, que variam de acordo com a quantidade de camadas existentes. Duas configurações são amplamente utilizadas, elas são a VGG-16 e VGG-19. A grande contribuição dessas redes na época foi o aumento de profundidade das redes possibilitado pelo uso de filtros pequenos nas camadas de convolução, geralmente 3x3.

Esse tipo de arquitetura foi inicialmente projetada para o concurso da Imagenet LSVRC-2014 alcançando o primeiro e segundo lugar, respectivamente, nas tarefas de localização e classificação. Desde então, vem sendo utilizada nas mais diversas aplicações como na segmentação e análise de fundo de olho (MANINIS et al., 2016), análise de dados volumétricos (SU et al., 2015), reconhecimento facial (PARKHI; VEDALDI; ZISSERMAN, 2015) e transferência de estilo (LI; WAND, 2016).

A Tabela 3 mostra a configurações das duas principais instâncias das arquiteturas VGG. As camadas totalmente conectadas ou *fully connected* (FC) são camadas de uma rede neural artificial e a camada **softmax** uma função de ativação. Os hiper-parâmetros das camadas de convolução são denotados como "*conv*<campo receptivo>-<numero de canais>".

Essas arquiteturas priorizam o uso de filtros pequenos no lugar de filtros de tamanho grande, por exemplo, três filtros 3x3 no lugar de um único 7x7. O objetivo é utilizar mais camadas ReLu transformando a função de decisão mais discriminativa e diminuir a quantidade de pesos à serem aprendidos. Considerando o exemplo anterior, essa redução é de, aproximadamente, 19%. O uso de filtros de dimensão pequena é, também utilizado em outros trabalhos com essa mesma finalidade (SZEGEDY et al., 2015) e (HE et al., 2015) como veremos na seção a seguir.

Tabela 3 – Estrutura das redes VGG-16 e VGG-19.

Arquiteturas	
VGG-16	VGG-19
conv3-64	conv3-64
conv3-64	conv3-64
maxpool	
conv3-128	conv3-128
conv3-128	conv3-128
maxpool	
conv3-256	conv3-256
conv3-256	conv3-256
conv3-256	conv3-256
maxpool	
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
maxpool	
conv3-512	conv3-512
conv3-512	conv3-512
conv3-512	conv3-512
maxpool	
FC-4096	
FC-4096	
FC-1000	
soft-max	

Fonte: Adaptado de: (SIMONYAN; ZISSERMAN, 2014)

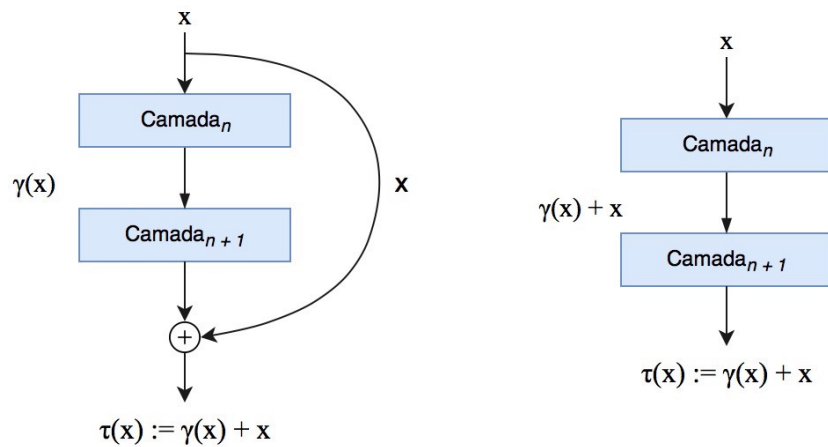
De maneira geral, as camadas de convoluções variam somente a quantidade de filtros à serem aprendidos em cada camada. Os hiper-parâmetros passo e tamanho de filtro, são fixados em 1 *pixel* em janelas de *pixels* com dimensão 3x3, respectivamente. Ainda, cada convolução possui um *padding* de 1 *pixel* em cada dimensão. Já nas camadas de sub-amostragem que totalizam cinco ao todo, o processo é realizado sobre áreas 2x2 com passo do mesmo tamanho, ou seja, sem sobreposição.

2.4.4 Redes Neurais Residuais

O conceito de Redes Neurais Convolucionais Residuais (ResNet) (HE et al., 2015) possibilitam desenvolver redes neurais mais profundas do que CNNs tradicionais. A ideia foi utilizada nas competições Imagenet LSVRC-2015 e COCO 2015 (LIN et al., 2014) obtendo o primeiro lugar em, pelo menos 4 categorias, e implementando redes com mais de 100 camadas, mas com complexidade inferior à redes mais rasas propostas até então.

Observando somente os resultados obtidos pelas CNNs propostas para a competição Imagenet nos anos de 2012 até 2014, pode-se erroneamente deduzir que o simples empilhamento e aumento de camadas implica em redes com maior capacidade de generalização e aprendizagem. A ideia é que uma rede Y com $x + y$ camadas não deveria gerar erros superiores à rede X com x camadas, considerando que y camadas são convoluções de identidade (*kernel* 1x1). Contudo, como mostrado em (HE; SUN, 2015; SRIVASTAVA; GREFF; SCHMIDHUBER, 2015; HE et al., 2015) o aumento de camadas pode gerar uma degradação na acurácia do treino.

Figura 14 – Aprendizado residual, onde $\tau(x)$ é o valor que se deseja obter, x a entrada e $\gamma(x)$ o resíduo.



(a) Aprendizado residual.

(b) Aprendizado convencional.

Fonte: Adaptado de: (HE et al., 2015)

A solução encontrada se baseia em aprendizado residual, onde as camadas de uma dada rede têm de aprender resíduos adicionais ao conteúdo de entrada da rede. Consideremos o esquema representado na Figura 14, onde x é a entrada da rede e $\tau(x) = \gamma(x) + x$ o valor esperado como saída. Em uma arquitetura CNN não residual, a rede teria de adaptar seus parâmetros a todo o conteúdo de $\tau(x)$, já em uma arquitetura residual o valor de x é adicionado ao resultado por meio de operações identidade, assim, a rede tem de ajustar seus parâmetros aprendendo somente o conteúdo de $\gamma(x)$. Formalmente, podemos definir aprendizado residual como sendo:

$$\tau(x) = \gamma(x, \omega_{ij}) + x, \quad (2.47)$$

onde ω representa os pesos associados às camadas de i até j que à serem aprendidos.

O aprendizado residual pode ser subdividido em dois: identidade e projeção,

descritos matematicamente segundo as Equações 2.47 e 2.48, respectivamente.

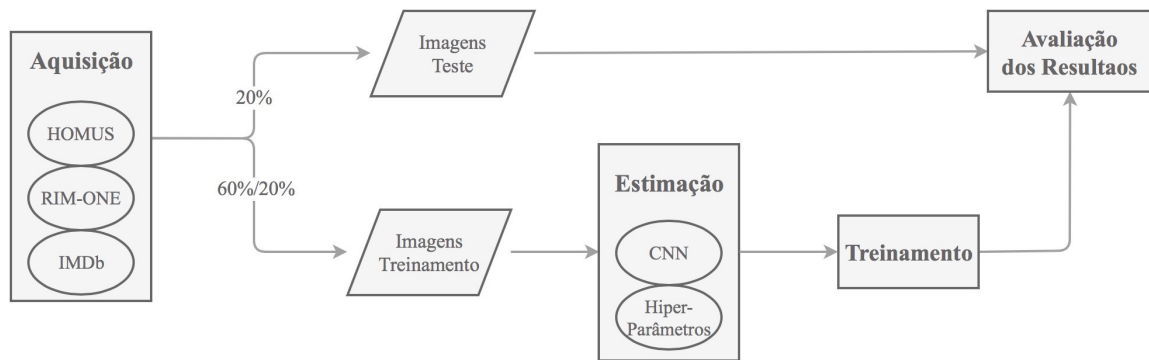
$$\tau(x) = \gamma(x, \omega_{ij}) + \omega_s x \quad (2.48)$$

A diferença consiste em ω_s que representa os parâmetros de uma projeção linear, necessária quando as camadas da sub-rede que contem o atalho alteram a quantidade de canais do volume de entrada. Na prática essa operação pode ser implementada por convoluções 1x1. Experimentos realizados em (HE et al., 2015) mostram que o uso de redes neurais residuais profundas convergem mais rapidamente do que redes neurais planas (convencionais) e exibem erros inferiores durante as etapas de treino e a validação.

3 Metodologia

A metodologia proposta para este trabalho envolve 4 etapas: aquisição de dados, estimação de hiper-parâmetros, treinamento e avaliação dos resultados. A Figura 15 representa estas etapas que estão descritas nas seções a seguir. A estimação de hiper-parâmetros é realizada sobre uma CNN para uma base de dados. O modelo é treinado com 60% e validado com 20% da base de dados escolhida e, então, avaliado sobre 20% desta base.

Figura 15 – Metodologia proposta.



Fonte: Autor

3.1 Aquisição

Dedicamos a primeira etapa desta metodologia a aquisição das imagens estudadas neste trabalho. Três bases foram utilizadas: a base de dados HOMUS, RIM-ONE (versão 2) e IMDb para, respectivamente, o reconhecimento de símbolos musicais, diagnóstico de glaucoma e análise de sentimento em textos.

Como discutido nas seções a seguir, a escolha destas bases está relacionada com o fato de possuírem elementos que as diferenciam por suas formas, textura e organização semântica. A seguir, descreve-se cada uma destas bases juntamente com seus problemas relacionados.

3.1.1 HOMUS: Classificação de Símbolos Musicais

Classificar símbolos musicais é uma das etapas fundamentais no desenvolvimento de um sistema de reconhecimento óptico musical (*Optical Music Recognition - OMR*). Sistemas desse tipo tentam converter partituras manuscritas em formatos passíveis de

serem reproduzidos por computadores. Atualmente, grande parte das partituras existentes estão disponíveis somente em suas versões manuscritas originais. O desenvolvimento de um *software* do tipo OMR facilitaria a acessibilidade sobre este conteúdo musical, assim como também auxiliaria na preservação da herança musical. Contudo, um sistema OMR robusto e preciso ainda não existe.

Notas musicais manuscritas são difíceis de serem processadas e reconhecidas. A maioria dos autores possuem seu próprio estilo de escrever símbolos musicais, conforme é apresentado na Figura 16. Como os símbolos não são impressos por uma máquina, podem existir variações de forma, tamanho e angulação. Objetivamos utilizar CNNs para implementar um processo robusto e automático para a seleção dessas e de outras características utilizadas na classificação de símbolos musicais.

Figura 16 – Nota musical idêntica (Mínima) escrita por diferentes músicos.



Fonte: Autor

Neste trabalho, a base HOMUS (CALVO-ZARAGOZA; ONCINA, 2014) é utilizada para a aquisição de símbolos musicais manuscritos. Com um total de 15.200 amostras, a base possui notas e marcações musicais produzidas por 100 músicos distintos, cada um transcrevendo o símbolo musical de acordo com sua própria caligrafia e maneira de representar. As amostras são divididas em 32 classes distintas, cada uma representando um símbolo musical diferente. Cada amostra é composta por traços, a identificação do autor e a sua classificação.

Sabendo que redes neurais convolucionais utilizam, principalmente, imagens como entrada, para dar continuidade ao trabalho, primeiro se fez necessário a representação das amostras da base HOMUS em imagens. Para tanto, assumiu-se uma espessura de 2 *pixels* para cada traço, assim, gerando da amostra original (Figura 17a) uma imagem de tamanho fixo (Figura 17b). As imagens geradas possuem resolução 64x64 e são binárias (preto e branco).

Cada músico possui sua caligrafia única, isso reflete na forma como o autor escreve um símbolo musical. Assim como uma letra, um mesmo símbolo musical pode ser representado de diferentes maneiras o que gera a necessidade de um modelo computacional robusto capaz de identificar tais variações, mas ainda eficiente em diferenciar um símbolo de outro.

Figura 17 – Exemplo de geração de imagem da base de dados HOMUS.



Fonte: Autor

A base de dados HOMUS foi escolhida por ser uma representação de formas e contornos. Possibilitando, assim, a análise dos modelos estudados na tarefa de classificação de imagens com estes padrões de características.

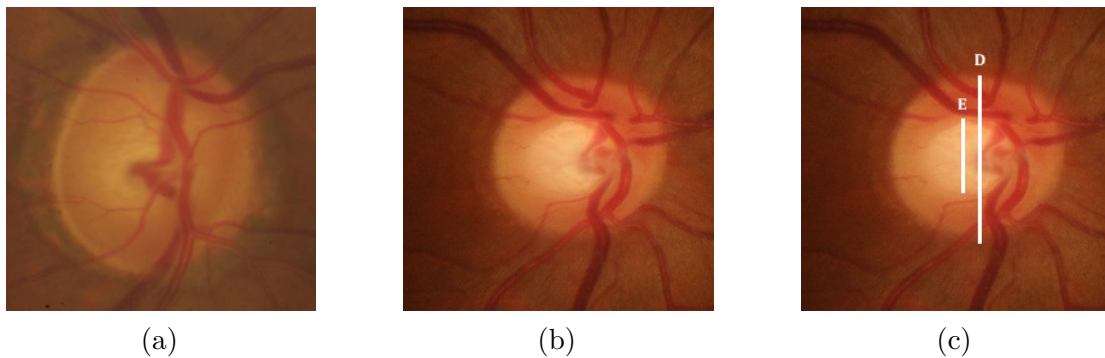
3.1.2 RIM-ONE: Diagnóstico de Glaucoma

De acordo com [Hattenhauer et al. \(1998\)](#), o glaucoma é uma doença capaz de causar cegueira quando não diagnosticada e tratada a tempo. O nervo óptico é aquele que transmite as informações visuais recebidas pelos cones e bastonetes, presentes retina, até o cérebro. E o glaucoma refere-se a um grupo de doenças oculares que provocam danos irreparáveis neste nervo. A doença é considerada crônica e não tem cura, contudo, pode ser controlada com tratamento contínuo.

A estimativa atual é de que a doença atinge cerca de 70 milhões de pessoas em todo o mundo, isto é, de 2 a 3% da população mundial, e que, em 2020, esse número cresça para 80 milhões ([GONÇALVES et al., 2013](#)). Na sua forma mais comum, o glaucoma de ângulo aberto, cerca de 80% dos casos não apresentam sintomas nos estágios iniciais ([ABRAG, 2014](#)). Na maioria dos casos, a pressão interna é gradualmente elevada e a cegueira só ocorre nos últimos estágios da doença. Assim, se faz necessário a detecção precoce desse tipo de doença por meio de exames de rotina junto a um oftalmologista.

Durante o exame, o especialista analisa a pressão intraocular e o disco óptico do paciente. A Figura 18 mostra duas imagens de retinografias, na Figura 18a tem-se um olho glaucomatoso e na Figura 18b um olho saudável. Para examinar o disco óptico, o especialista analisa a forma do disco que é, normalmente, oval com 1,5 mm de diâmetro e 5 a 10% maior na horizontal. Uma métrica bem conhecida é o *Cup-to-disc ratio* (CDR) que mede a razão entre a escavação (E) e o disco óptico (D), essa razão é utilizada para indicar o fator de risco glaucomatoso ([PARANHOS-JUNIOR; OMI; PRATA-JUNIOR, 2009](#)).

Figura 18 – Retinografias de olho glaucomatoso e saudável.



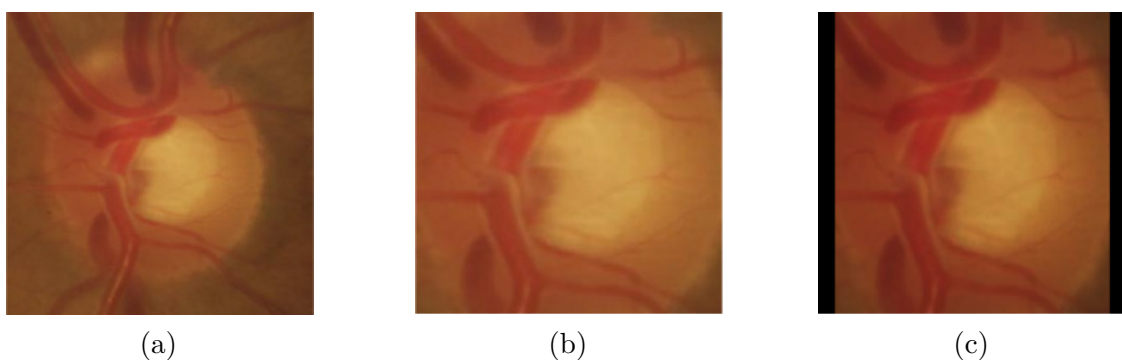
Fonte: (FUMERO et al., 2011)

A base de dados RIM-ONE (*An Open Retinal Image Database for Optic Nerve Evaluation*) consiste de imagens digitais do nervo óptico. Um total de 455 imagens estão disponíveis, sendo 200 dessas com glaucoma em diferentes estágios e as 255 restantes são imagens de nervo óptico saudáveis. Segundo os autores da base em (FUMERO et al., 2011) as imagens foram obtidas de diferentes regiões da Espanha e a classificação realizada por 5 especialistas. A base também disponibiliza a marcação do nervo óptico em cada imagem.

Tais marcações foram utilizadas para gerar a região de interesse (ROI) em cada imagem. É importante ressaltar que cada imagem possui um tamanho distinto e não quadrático, sendo assim, para utilizar as imagens com as redes neurais convolucionais, um enquadramento se fez necessário. A Figura 19 mostra um exemplo do resultado obtido, na Figura 19a tem-se uma imagem original do nervo óptico e, em 19b, a ROI de acordo com a marcação do especialista para a mesma imagem.

Com o enquadramento da ROI (Figura 19c), todas as imagens resultantes possuem a mesma resolução fixa de 64x64 *pixels*. A região preta adicionada não interfere nos resultados, já que em algum momento estas áreas serão removidas pelas camadas de *poolings* de máxima.

Figura 19 – Exemplo de imagem resultado da base RIM-ONE.



Fonte: Adaptado de: (FUMERO et al., 2011)

Após alguns experimentos, percebeu-se que o uso dos canais RGB resultavam em *overfitting* na maioria dos modelos testados e que a utilização dos canais HSV por si só proporcionava poucas informações distintas entre os dados. Assim, o primeiro passo foi separar o canal vermelho das imagens RGB geradas e combinar com os canais *hue* e *saturation* da escala de cor HSV. O canal vermelho foi escolhido por destacar o disco óptico sobre outras estruturas (AQUINO; GEGÚNDEZ-ARIAS; MARÍN, 2010) e o canal *value* da escala de cor HSV removido por que, durante os experimentos, diminuiu a capacidade de classificação dos modelos estudados.

Além do fator social associado ao diagnóstico de glaucoma, outros fatores influenciaram na escolha deste tipo de imagens médicas. Para o diagnóstico de glaucoma, o modelo computacional tem de ser capaz de analisar, além de aspectos como textura e formas, proporções entre diferentes elementos da imagem.

3.1.3 IMDb: Análise de Sentimentos em Avaliações de Filmes

Redes sociais são uma plataforma de comunicação importante no mundo atual. Aplicativos como *Facebook*, *Snapchat* e *Twitter* agrupam opiniões relevantes para diversas áreas de interesse público, como política, tendência de mercado, moda e até mesmo segurança. Já a indústria de filmes movimenta uma quantidade de dinheiro inimaginável com propagandas e estudos de mercado. Muito desses recursos poderiam ser economizados, para isso, a construção de um sistema automático, capaz de analisar as avaliações de clientes de diferentes *websites*, pode se mostrar útil e eficiente.

Quando deseja-se realizar a análise de sentimentos em textos, podemos classificar cada texto em três grupos: o de textos positivos, textos negativos ou textos objetivos (Tabela 4). O primeiro é formado pelos textos que avaliam positivamente um determinado tema contido no texto, e de maneira contrária os textos negativos são aqueles que expressam opiniões contrárias sobre um certo tema, ou seja, avaliam algo negativamente. Por sua vez, os textos objetivos são aqueles que não realizam classificação alguma, estes textos afirmam, explicam ou relatam algo.

A base de dados referente a análise de sentimentos de texto foi extraída da *Internet Movie Database - IMDb*. O *website* possui mais de 3 milhões de filmes cadastrados no seu banco de dados cada um com vários comentários associados. A base de dados utilizada neste trabalho possui 25 mil avaliações rotuladas em positivas e negativas. Cada elemento do conjunto representa a opinião de um usuário sobre determinado filme.

A base de dados é disponibilizada e pré-processada por (CHOLLET, 2015). O pré-processamento consiste em uma simples codificação de acordo com a frequência das palavras na base de dados. Ou seja, os termos que mais aparecem são indexados com valores mais baixos e os que não aparecem com tanta frequência possuem índices mais

Tabela 4 – Exemplo - Análise de sentimento em texto.

Sentimento	Texto
Negativo	The screen-play is very bad, but there are some action sequences that i really liked. I think the image is good, better than other romanian movies. I liked also how the actors did their jobs.
Positivo	Looking for a REAL super bad movie? If you wanna have great fun, don't hesitate and check this one! Ferrigno is incredibly bad but is also the best of this mediocrity.
Objetivo	In the future, Jake, a paraplegic war veteran, is brought to a moon, Pandora, which is inhabited by the Na'vi, a humanoid race with their own language and culture.

elevados. Assim, utilizaremos as 5000 palavras mais frequentes, removendo os termos que raramente aparecem, como palavras eruditas e deixando as de maior frequência como adjetivos.

Para obter volumes do mesmo tamanho, foi utilizada a técnica de preenchimento ou *padding*. Fixou-se o tamanho de 400 termos por sequência, considerando que comentários possuem em média uma quantidade de palavras similar. O procedimento consiste em remover todos os termos com índices superiores a 400 e, caso o comentário possua menos que 400 palavras, preencher com nulo os índices vazios. Após esta etapa de pré-processamento, tem-se um volume fixo de entrada equivalente a 400 por amostra.

3.2 Escolha das Redes Neurais Convolucionais

Para realizar este trabalho utilizou-se três arquiteturas CNN: LeNet (LECUN et al., 1998), VGG-16 (SIMONYAN; ZISSERMAN, 2014), e ResNet (HE et al., 2015). A escolha foi feita baseada na maneira como as camadas de cada rede estão dispostas e na profundidade destas redes.

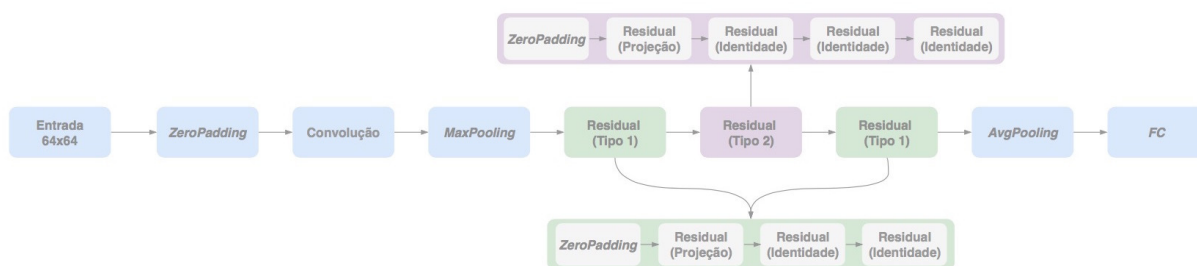
Como discutido na Seção 2.4.1, a rede LeNet é constituída de duas camadas de convolução, cada uma seguida de uma camada de *pooling*, além da camada de classificação (RNA). Classificar dígitos, problema para o qual a rede foi originalmente desenvolvida, se resume em identificar formas e suas variações, ou seja, é uma tarefa similar a classificação de símbolos musicais. Além disso, o fato desta rede ser constituída de poucas camadas,

somente sete, facilita o processo de treinamento com poucas amostras, como é o caso da base de dados RIM-ONE.

As redes da família VGGNet, por outro lado, não possuem essa característica, são redes bem profundas e, geralmente, treinadas com mais de mil amostras por classe. Assim, nosso objetivo é estudar o comportamento dessas redes utilizando bases bem menores do que o usual. Dentre as redes VGG, foi escolhida a rede VGG-16 pela sua alta capacidade de extração de características. Originalmente, como discutido na Seção 2.4.3, essa arquitetura foi projetada para classificar imagens naturais o que requer a habilidade de distinguir além de, formas e texturas, outras características mais complexas.

Por sua vez, as Redes Neurais Residuais (HE et al., 2015) possuem características peculiares. Apesar de serem redes extremamente profundas, possuem complexidade inferior às arquiteturas planas equivalentes o que facilita o treino com poucas amostras. Durante os testes, utilizaremos uma rede neural residual com 30 camadas, que chamaremos, a partir de agora, ResNet-30. A Figura 20 ilustra a organização das camadas da ResNet-30, em cada conjunto de blocos residuais, representados em verde e roxo, os hiper-parâmetros são compartilhados entre as camadas de convolução que não são utilizadas para projeções. Já as "camadas" de *zero-padding*, na verdade, são inseridas na camada seguinte as mesmas.

Figura 20 – Rede Neural Residual utilizada durante o trabalho.



Fonte: Autor

Todos os blocos residuais são constituídos de três camadas de convolução empilhadas, todas as três possuem os mesmos hiper-parâmetros. Lembrando do que foi discutido na Seção 2.4.4, a diferença entre os blocos residuais de projeção e os de identidade é a existência ou ausência de aprendizado no caminho de atalho. No nosso caso, os atalhos de projeção são implementados utilizando uma camada de convolução com *kernel* 1x1 pixels. A quantidade de filtros nas projeções é a mesma das outras camadas de convolução do mesmo bloco.

Alguns fatores influenciaram na escolha da rede residual, entre eles a complexidade. Redes neurais extremamente profundas, como por exemplo a ResNet-50 (HE et al., 2015), são mais suscetíveis a sofrer de super ajuste, quando tem-se poucas amostras, do que redes

mais rasas. Outro fator impactante é a quantidade de combinações de hiper-parâmetros possíveis. Em Redes neurais convolucionais extremamente profundas, quanto maior o tamanho do passo e do filtro, menor o volume de saída, o que pode limitar as possibilidades de combinações destes hiper-parâmetros. Redes mais rasas, são mais flexíveis e, logo, possibilitam um maior número de combinações.

Para utilizar tais redes com a base IMDb, inseriu-se uma camada de *embedding* (Seção 2.3.2) após a entrada de cada CNN. O parâmetro dimensão de saída (n) é fixado em 50, assim, os elementos de entrada são mapeados em um volume de tamanho fixo: tamanho da sequência pre-processada x dimensão de saída.

3.3 Estimação de Hiper-Parâmetros

A estimação de hiper-parâmetros é uma das etapas fundamentais deste estudo. Os hiper-parâmetros selecionados levam em consideração a diversidade das redes utilizadas e os principais fatores que permitem as redes extraírem e treinarem características. A seleção de hiper-parâmetros é feita de acordo com a camada às quais pertencem.

Para as camadas de convolução, os hiper-parâmetros selecionados foram: quantidade e tamanho de filtros (*kernel*), *stride* e *zero-padding*. Estes parâmetros influenciam diretamente na extração das características, a quantidade de filtros em uma camada de convolução pode ser entendida como a quantidade de *kernels* a serem treinados naquela camada. O tamanho do filtro e de *zero-padding* refletem diretamente na área de busca, é o campo receptivo de uma camada. Por último, o *stride* indica o tamanho do passo da janela do filtro sobre a entrada da camada atual.

Nas camadas de *pooling* os hiper-parâmetros selecionados foram: *kernel* e *stride* do filtro. Esses hiper-parâmetros possuem a mesma finalidade nas camadas de sub-amostragem e de convolução. Contudo, enquanto camadas de convolução têm como objetivo extrair características, as camadas do tipo *pooling* têm como objetivo resumir essas características com o auxílio de uma operação matemática, no nosso caso, esta operação, é a função de máxima.

Os hiper-parâmetros utilizados variam de acordo com a arquitetura e a base escolhidas, para cada par (arquitetura/base) um conjunto de hiper-parâmetros são escolhidos para as camadas de convolução e *pooling*. Estes hiper-parâmetros foram selecionados baseando-se em outros trabalhos que utilizam redes neurais convolucionais como (LECUN et al., 1998), (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) e (BERGSTRA; BENGIO, 2012).

Normalmente, o tamanho do *kernel* e o tamanho do passo é um valor primo, como 1x1, 2x2, 3x3, 5x5 ou 7x7 tentando extrair características de diferentes tamanhos. Quando

o tamanho do passo é menor que o tamanho do *kernel* diz-se que existe sobreposição. Já para a quantidade de filtros, durante este trabalho, sempre será utilizado quantidade de filtros múltiplos de 2, como 16, 32, 64, 128 e 256.

A seguir, discutiremos cada uma das combinações propostas neste trabalho para as arquiteturas LeNet, VGG-16 e ResNet-30.

3.3.1 LeNet: Hiper-Parâmetros

A arquitetura LeNet possui duas camadas de convolução, cada uma seguida por uma camada de sub-amostragem. Os hiper-parâmetros *stride* e *kernel* destas camadas utilizados durante os experimentos estão apresentados, respectivamente, nas Tabelas 6 e 5. Nestas, cada coluna representa uma possível combinação de um hiper-parâmetro (*stride* ou *kernel*) para as diferentes camadas da rede, representadas pelas linhas das tabelas.

Tabela 5 – Combinação de campos receptivos para rede LeNet, com total de 35 possibilidades.

Pooling						Convolução							
1	2x2	3x3	2x2	5x5	3x3	1	3x3	5x5	1x1	5x5	3x3	1x1	1x1
2	3x3	2x2	5x5	2x2	5x5	2	3x3	5x5	5x5	1x1	5x5	3x3	1x1
	(a)						(b)						

Para camadas do mesmo tipo, convolucional, por exemplo, deseja-se estudar como a organização, crescente ou decrescente, dos hiper-parâmetros destas camadas influencia no modelo treinado. Duas camadas do mesmo tipo também podem possuir os mesmos valores de hiper-parâmetros. Em algumas situações, camadas de *pooling* podem ser substituídas por camadas de convolução (SPRINGENBERG et al., 2014), sendo assim, não serão utilizadas duas camadas de *pooling* com tamanho de *kernels* iguais.

Tabela 6 – Conjuntos de hiper-parâmetros *stride* para rede LeNet. Cada coluna representa uma possível combinação e cada linha uma camada da CNN.

LeNet - <i>Stride</i>						
1 Convolução	1x1	2x2	1x1	3x3	1x1	2x2
1 Pooling	2x2	1x1	3x3	1x1	1x1	2x2
2 Convolução	1x1	2x2	1x1	3x3	1x1	2x2
2 Pooling	2x2	1x1	3x3	1x1	1x1	2x2

Todas as instâncias válidas dentre as 210 combinações possíveis - 6 opções de *stride*, 5 de *kernel* para *pooling* e 7 de *kernel* para convolução - foram avaliadas com treino de 20 épocas e com a quantidade de filtros pré-definida da rede LeNet, que podem ser encontradas na Seção 2.4.1. Para viabilizar este trabalho, com a base de dados HOMUS, escolheu-se as três melhores instâncias e duas outras com acurácia menor que 70%. Com a base de dados RIM-ONE, essa relação totalizou 10 instâncias, sendo as 6 melhores e 4

com acurácia menor que 70%. E, por fim, para a base IMDb filtrou-se 20 instâncias, dessa vez as 14 melhores e 6 com acurácia abaixo de 70%.

Esta seleção resultou nas instâncias apresentadas nas Tabelas 7, 8 e 9, para as bases RIM-ONE, HOMUS e IMDb, respectivamente. Nestas tabelas, cada linha representa uma instância contendo os hiper-parâmetros das camadas de convolução e sub-amostragem. Para cada célula, têm-se o tamanho do filtro e o passo do mesmo representados por $\langle \text{tamanho em } x \rangle \times \langle \text{tamanho em } y \rangle + \langle \text{passo} \rangle$.

A seleção somente das melhores arquiteturas poderia resultar em buscas locais. Assim, a seleção de outras instâncias (abaixo de 70%) também se fez necessária já que posteriormente serão selecionadas combinações de quantidade de filtros.

Tabela 7 – Instâncias da rede LeNet para a base RIM-ONE.

Instância	Convolução 1	Pooling 1	Convolução 2	Pooling 2
1	5x5+1	5x5+3	1x1+1	3x3+3
2	5x5+1	2x2+2	5x5+1	5x5+2
3	3x3+1	2x2+3	5x5+1	3x3+3
4	3x3+2	2x2+1	3x3+2	5x5+1
5	5x5+1	2x2+3	5x5+1	3x3+3
6	3x3+2	3x3+1	1x1+2	5x5+1
7	5x5+2	2x2+1	5x5+2	3x3+1
8	5x5+1	3x3+3	5x5+1	2x2+3
9	5x5+1	5x5+3	5x5+1	2x2+3
10	1x1+1	2x2+3	5x5+1	5x5+3

Tabela 8 – Instâncias da rede LeNet para a base HOMUS. Cada linha representa uma instância.

Instância	Convolução 1	Pooling 1	Convolução 2	Pooling 2
1	3x3+1	1x1+1	2x2+2	3x3+2
2	3x3+2	5x5+2	3x3+1	2x2+1
3	3x3+1	3x3+1	3x3+2	2x2+2
4	2x2+1	5x5+1	2x2+1	2x2+1
5	5x5+1	2x2+1	3x3+1	2x2+1

Tendo delimitado os hiper-parâmetros de passo, tamanho de filtro e considerando que não há *zero-padding*, define-se a quantidade de filtros nas camadas de convolução que estão demonstradas na Tabela 10, onde cada linha representa um par de hiper-parâmetros, o primeiro para a camada de convolução 1, a mais rasa e o segundo para a camada de convolução mais profunda.

Tais combinações foram escolhidas de maneira que permitam analisar a relação entre o crescimento ou decréscimo da quantidade de filtros com a capacidade de generalização dos modelos selecionados.

Tabela 9 – Instâncias da rede LeNet para a base IMDb. Cada linha representa uma instância.

Instância	Convolução 1	Pooling 1	Convolução 2	Pooling 2
1	3x3+1	2x2+2	5x5+1	3x3+2
2	3x3+1	3x3+1	5x5+1	2x2+1
3	5x5+1	3x3+3	5x5+1	5x5+3
4	5x5+1	3x3+1	5x5+1	2x2+1
5	5x5+1	5x5+1	1x1+1	3x3+1
6	5x5+1	3x3+1	1x1+1	2x2+1
7	3x3+1	2x2+2	3x3+1	3x3+2
8	5x5+1	3x3+3	5x5+1	2x2+3
9	1x1+1	3x3+1	3x3+1	2x2+1
10	3x3+1	2x2+1	5x5+1	5x5+1
11	3x3+1	2x2+1	1x1+1	3x3+1
12	5x5+1	2x2+1	5x5+1	5x5+1
13	3x3+1	3x3+2	3x3+1	2x2+2
14	1x1+1	2x2+1	5x5+1	3x3+1
15	1x1+1	2x2+1	3x3+1	5x5+1
16	1x1+1	5x5+3	3x3+1	2x2+3
17	3x3+1	3x3+2	1x1+1	2x2+2
18	1x1+1	5x5+1	3x3+1	3x3+1
19	3x3+2	5x5+1	3x3+2	2x2+1
20	1x1+1	3x3+2	3x3+1	5x5+2

Tabela 10 – LeNet: Quantidade de filtros nas camadas de convolução.

Combinação	Qtd. Filtros (Convolução 1)	Qtd. Filtros (Convolução 2)
1	16	256
2	64	32
3	64	64
4	64	256
5	128	32
6	256	128

Ao final, foram obtidas 30 instâncias para a base de dados HOMUS, 60 instâncias para base de dados RIM-ONE e 120 para a base de dados IMDb. Cada uma destas foram avaliadas e seus resultados discutidos na Seção 4.2.

3.3.2 VGG: Hiper-Parâmetros

Como visto na Seção 2.4.3, as arquiteturas VGGNet possuem sequências de blocos de convoluções separados por camadas de *pooling* idênticas. Os hiper-parâmetros variam mantendo essa estrutura, ou seja, todas as camadas de *pooling* compartilham os mesmos hiper-parâmetros e as camadas de convolução são agrupadas para compartilharem, entre si, os mesmos hiper-parâmetros.

Para realizar esse estudo, considerou-se o campo receptivo das camadas de *pooling* como sendo 1x1, 2x2 ou 3x3 *pixels* e para as camadas de convolução esse hiper-parâmetro pode ser uma das combinações da Tabela 11, onde cada conjunto representa uma série de convoluções que compartilham os mesmos hiper-parâmetros.

Tabela 11 – VGG-16: Campo receptivo para as camadas de convolução. Cada combinação é representada por uma linha.

Combinação	Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4
1	3x3	3x3	3x3	3x3
2	5x5	5x5	5x5	5x5
3	7x7	7x7	7x7	7x7
4	2x2	2x2	2x2	2x2
5	5x5	5x5	2x2	2x2
6	2x2	2x2	3x3	5x5

Considerando a camada de entrada como 64x64 *pixels* para as bases HOMUS e RIM-ONE, temos somente 9 combinações válidas, aquelas que possuem um volume de saída positivo. Estas combinações estão representadas na Tabela 12. Já para a base IMDB, temos cerca de 50 possibilidades válidas. Cada escolha válida foi combinada com todas as 5 alternativas presentes na Tabela 13 e, então, pré-avaliadas. Cada linha da tabela representa uma combinação de quantidade de filtros, e as colunas os 4 conjuntos de convolução.

Tabela 12 – Combinações válidas, para rede VGG-16 para a base HOMUS e RIM-ONE. Cada Linha representa uma combinação válida.

Combinações	Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4	Pooling
1	3x3	3x3	3x3	3x3	1x1
2	5x5	5x5	5x5	5x5	1x1
3	7x7	7x7	7x7	7x7	1x1
4	2x2	2x2	2x2	2x2	1x1
5	5x5	5x5	3x3	2x3	1x1
6	2x2	2x2	3x3	5x5	1x1
7	3x3	3x3	3x3	3x3	2x2
8	2x2	2x2	2x2	2x2	2x2
9	5x5	5x5	3x3	2x2	2x2

De maneira similar a realizada com a rede LeNet, treinou-se cada par, campo receptivo e combinação de quantidade de filtros, por 20 épocas para a base HOMUS e RIM-ONE. Para a base IMDB, as arquiteturas válidas foram treinadas por 5 épocas. O objetivo foi, novamente, filtrar algumas das arquiteturas que não foram satisfatórias. Contudo, todas as redes avaliadas resultaram em modelos com *underfitting* nas três bases estudadas, modelos em *underfitting* são aqueles que não têm a capacidade de determinar a diferença entre classes ou que o faz com uma taxa de erro inaceitável, abaixo de 55% em nosso caso. No próximo capítulo, analisaremos melhor os motivos de tais resultados.

Tabela 13 – VGG-16: Quantidade de filtros nas camadas de convolução.

Combinações	Conjunto 1	Conjunto 2	Conjunto 3	Conjunto 4
1	64	128	256	512
2	32	64	128	256
3	16	32	64	128
4	64	64	128	64
5	16	16	128	16

3.3.3 ResNet: Hiper-Parâmetros

Como discutido na Seção 3.2 utilizaremos a rede neural ResNet-30. Com o objetivo de analisar a influência dos hiper-parâmetros nos blocos residuais, somente as camadas pertencentes à blocos residuais serão afetadas com mudanças de hiper-parâmetros. Ainda, seguindo a definição discutida anteriormente, todas as convoluções de um mesmo conjunto de blocos possuirão os mesmos hiper-parâmetros, uma exceção se faz para as convoluções utilizadas para realizar projeções que possuirão sempre *stride* e tamanho de *kernel* fixos em 1x1 e 1 pixels, respectivamente. A partir de agora, nos referiremos somente aos hiper-parâmetros das camadas alteradas.

Tabela 14 – ResNet: Quantidade de canais nas camadas de convolução. Cada linha representa uma combinação de 3 hiper-parâmetros, que vão das camadas mais rasas (bloco 1) até as camadas mais profundas (bloco 3). Cada bloco é constituído por um conjunto de camadas residuais.

Combinação	Bloco 1	Bloco 2	Bloco 3
1	32	64	128
2	64	128	256
3	256	128	64
4	128	64	32

O *stride* destas camadas seguem o valor padrão, ou seja, o mesmo tamanho do filtro. Os *kernels* das camadas de convolução assumem um dos valores 2, 3 ou 5. Já a organização dos filtros nos conjuntos de blocos, camadas verde e roxa da Figura 20, seguem de acordo com a Tabela 14, totalizando 12 combinações testadas sobre a base de dados HOMUS. Para a base de dados RIM-ONE, adicionou-se a possibilidade de se ter *zero-padding* de 1 ou 3 *pixels* antes de cada bloco residual de projeção o que resulta no dobro de possibilidades, resultando em 24 possibilidades. Já para a base de dados IMDB, foram testadas 48 possibilidades, acrescentou-se para cada combinação escolhida para a base HOMUS a possibilidade de *stride* 1 ou o padrão e a existência ou não *zero-padding* de tamanho 1. Considerando que a quantidade de combinações não foram muitas, todas as combinações foram testadas.

3.4 Treino e Validação

Após a etapa de seleção dos hiper-parâmetros escolhidos, devemos agora treinar todos as combinações escolhidas. Primeiramente dividimos as bases de dados em 3: treino, validação e teste contendo, respectivamente, 60%, 20% e 20% dos elementos. Os elementos pertencentes a cada grupo são definidos aleatoriamente a cada modelo treinado.

O método de otimização escolhido durante os treinos quando utilizado as bases HOMUS e RIM-ONE, em todas as arquiteturas é o SGD, descrito na Seção 2.1.3. A taxa de aprendizagem foi fixada em 0,01. A quantidade de épocas para cada treinos também é constante, em 100 interações. Em especial, para a base de dados de texto IMDb o método de otimização foi o adam (KINGMA; BA, 2014) que utiliza o mesmo princípio do SGD. Ao final de cada interação o modelo é reavaliado com o conjunto de validação. A mesma arquitetura foi treinada cinco vezes, sendo cada treino independente do outro e com seleção aleatória dos conjuntos de treino, validação e teste.

Por último, durante o processo de treino e validação utilizamos duas funções de custo, definidas de acordo com as Equações 2.11 e 2.12 que representam, respectivamente, a entropia-cruzada e sua variação inserindo a sensibilidade. O objetivo desta última é diminuir o efeito de bases não balanceadas sobre os modelos treinados. Os resultados obtidos seguindo essa metodologia são discutidos no capítulo seguinte.

4 Resultados

Neste capítulo são apresentados os resultados obtidos seguindo a metodologia proposta neste estudo. O principal objetivo é relacionar a variação dos hiper-parâmetros de uma CNN com o seu desempenho no reconhecimento de símbolos musicais, diagnóstico de glaucoma e análise de sentimento em textos.

A variabilidade dos resultados, entre os cinco treinos, utilizando uma mesma arquitetura permite analisar se os modelos treinados super ajustaram-se ou não aos dados de treino. Quando o desvio padrão de diferentes treinos é elevado, acima de 5%, neste trabalho, considera-se que estes modelos possuem alta probabilidade de *overfitting*. Ainda, quando estes modelos possuem acurácia média inferior a 55%, considerou-se que estão em *underfitting*. Com isso, considera-se somente aqueles modelos que possuem alto poder de generalização.

Como medida de avaliação utilizaremos a acurácia das arquiteturas em cada um dos problemas. A distribuição das cinco interações em cada um dos cenários são resumidas em diagramas de caixa ou *box plot*. A seguir, são discutidos os resultados obtidos com as redes VGG-16, LeNet e ResNet-30.

4.1 Resultados: VGG-16

A rede neural VGG-16 mostrou o pior desempenho dentre as três bases estudadas, todos os modelos analisados resultaram em *underfitting*, como pode ser visto na Tabela 15. Os modelos se especializaram somente em uma classe, o que resultou em acurácias baixíssimas. Sendo assim, esta seção tem como objetivo estudar as possíveis causas que levaram a tal resultado e métodos de reamostragem que poderiam ajudar a contornar essa situação. As principais causas que levaram a modelos incapazes de distinguir as diferentes classes dos problemas foram pequena quantidade de amostras em cada problema e a profundidade da rede VGG-16.

Tabela 15 – Resultados obtidos com os modelos baseados na VGG-16.

Arquitetura	Base	Acurácia
Todas	HOMUS	3.5% ~ 5.46%
Todas	RIM-ONE	50%
Todas	IMDb	50%

Problemas como os estudados nesse trabalho, são geralmente considerados de difícil classificação, já que as características relevantes nem sempre são fáceis de serem detectadas

até mesmo por especialistas. Isso gera a necessidade de modelos mais robustos na tarefa de classificação.

Arquiteturas da família VGGNet foram originalmente projetadas para bases com diversas classes e com várias amostras. Embora não seja possível definir uma quantidade mínima de amostras necessárias para treinar uma CNN rasa, um valor aceitável é de 1.000 imagens para cada classe dependendo da complexidade do problema como acontece em (SIMONYAN; ZISSERMAN, 2014) e (MANINIS et al., 2016), no caso de processamento de linguagem natural esse número pode aumentar consideravelmente. Esse requisito não é satisfeito por nenhuma das bases estudados neste trabalho.

Contudo, vale ressaltar que diversos estudos já obtiveram o estado da arte utilizando redes baseadas na VGGNet. Tais estudos geralmente possuem mais amostras para cada classe ou utilizam técnicas de reamostragem para obtê-las. Existem diferentes abordagens de reamostragem em imagens, as mais comuns consistem em rotacionar, cortar e transladar as amostras originais gerando novas com conteúdo similar. Ainda, outros trabalhos inicializam os pesos das primeiras camadas da rede VGG com os pesos pré-treinados em outras bases maiores como a ImageNet, essa transferência de pesos é conhecida como *transfer learning* (OQUAB et al., 2014).

Ainda, é importante notar, que este trabalho também tem como objetivo analisar o desempenho de diversas arquiteturas convolucionais treinadas com poucas amostras. E, este entre outros motivos, como o fator social ligado aos três problemas, foi fundamental na escolha das bases a serem analisadas. Como é possível perceber, a rede VGG-16 não conseguiu gerar nenhum modelo com capacidade de generalização quando treinado com poucas amostras. Por outro lado, como veremos nas seções a seguir, redes mais rasas ou residuais foram capazes de extrair características relevantes para a classificação das bases selecionadas.

4.2 Resultados: LeNet

Em geral, os resultados obtidos com a rede LeNet possuem pequena taxa de variabilidade. Os melhores resultados obtidos para o diagnóstico de glaucoma e a análise de sentimento foram obtidos utilizando esta arquitetura.

4.2.1 Reconhecimento de Símbolos Musicais

Para o problema de reconhecimento de símbolos musicais, representado pela base de dados HOMUS, os resultados utilizando a rede LeNet estão contidos nas Figuras 21 e 22 para, respectivamente, as funções de custo de entropia cruzada e a adaptada. Em ambos os casos, a maioria dos resultados possui pequena variabilidade o que sugere modelos robustos.

Observando-se as Tabelas 16 e 17 que descrevem as arquiteturas dos gráficos em questão, perceber-se que todas as arquiteturas possuem sobreposição tanto nas camadas de convolução quanto nas de *pooling*, ou seja, o *stride* dessas camadas são inferiores ao tamanho do *kernel* nas mesmas. Cerca de 80% das convoluções possuem o hiper-parâmetro *stride* igual a 1x1 e o mesmo ocorre em 60% das camadas de sub-amostragem.

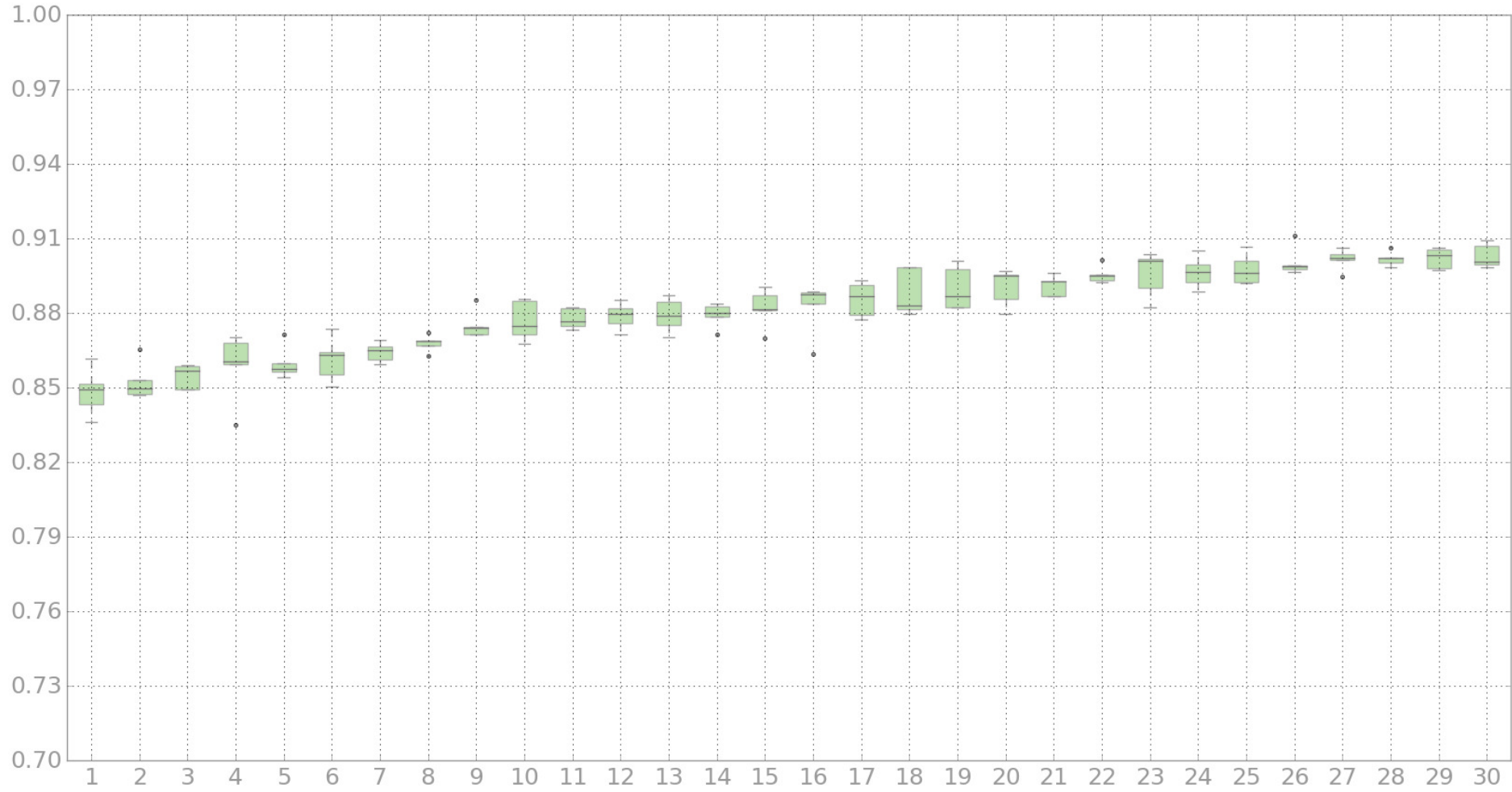
Normalmente, o tamanho do campo receptivo na primeira camada de convolução é mediano (3x3) e, o mesmo hiper-parâmetro na segunda camada de convolução, geralmente, tem área de 5x5 *pixels*. Os piores modelos ocorrem quando há pouca quantidade de filtros treináveis nas primeiras camadas de convolução, 16 e 32.

Por outro lado, com ambas funções de custo estudadas, geralmente, os melhores modelos possuem uma quantidade maior ou igual de filtros na primeira camada de convolução do que na segunda camada do mesmo tipo. Isso sugere que uma quantidade maior de características simples é extraída e, então, essas características são unidas em um volume menor de características mais complexas, como formas e contornos.

Em especial, variando somente a combinação de quantidade de filtros, o pior modelo válido analisado (Combinação 5 da Tabela 8 da Seção 3.3.1) resultou em modelos mais eficientes (Modelos 16 e 17 da Tabela 17 e 14 da Tabela 16). Tais arquiteturas possuem acurácia melhor do que os modelos 11 da Tabela 16 e 14 da Tabela 17 que são variações do melhor modelo inicialmente conjecturado (Combinação 1 da Tabela 8). Essa variação confirma que a combinação de quantidade de filtros é fundamental na geração de modelos eficientes.

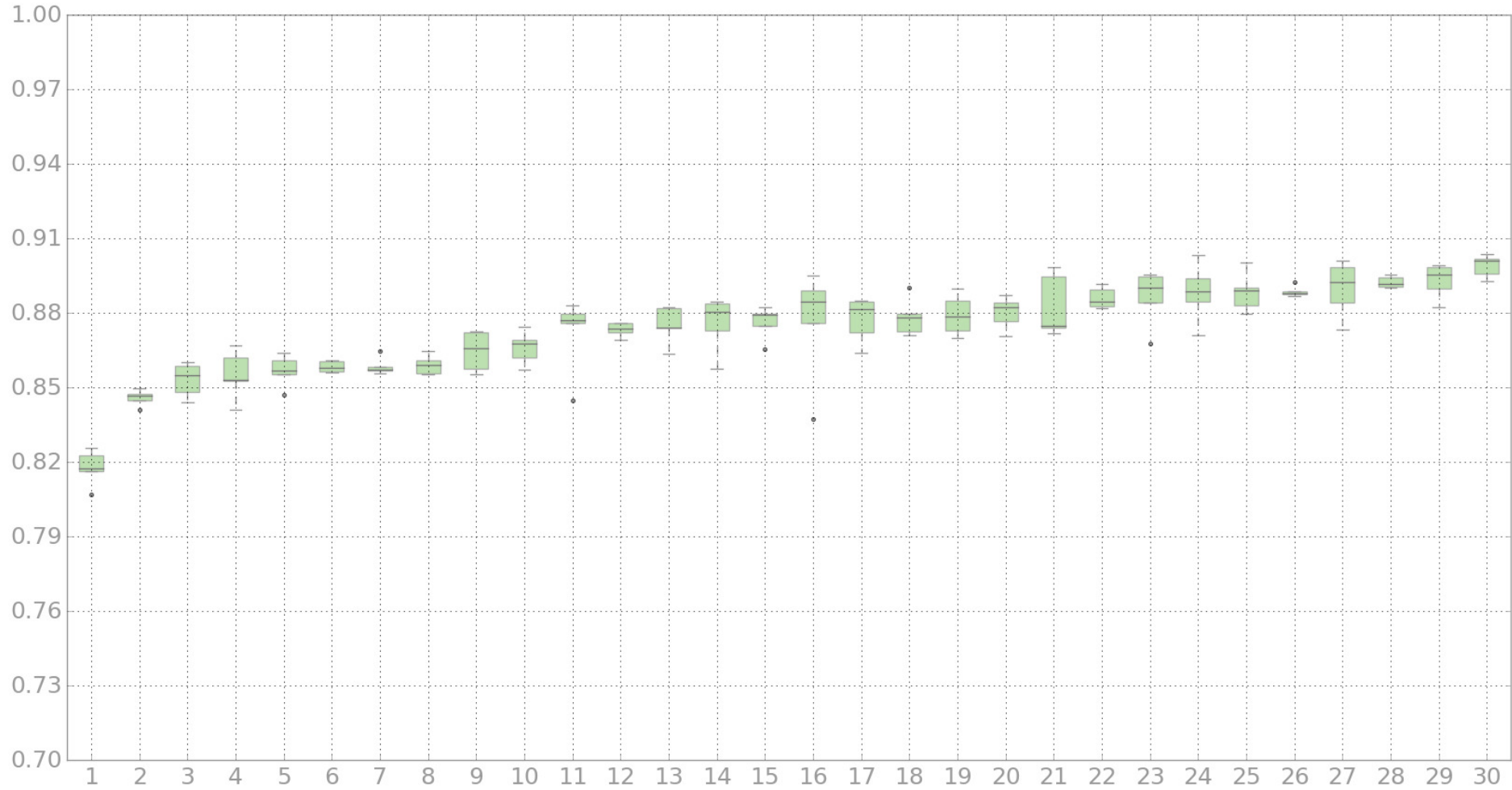
Os dois melhores resultados, com as duas funções de custo, possuem hiper-parâmetros similares. Em ambos, a primeira camada de convolução possui campo receptivo 3x3 e passo 1 e a seguinte *kernel* 1x1 com o mesmo passo. As camadas de *pooling* são, respectivamente, 2x2+2 e 3x3+2 para a primeira e segunda camada do tipo. A diferença consiste na quantidade de filtros que é de 64 e 256 quando utilizado a entropia cruzada, resultando na melhor acurácia média (90.22%), e de 256 e 128 para a função adaptada com acurácia média de 89.92%.

Figura 21 – Gráfico - Resultados com a base HOMUS e a arquitetura LeNet utilizando a função de custo entropia cruzada. A arquitetura de cada modelo é descrita em detalhes na Tabela 16.



Fonte: Autor

Figura 22 – Gráfico - Resultados com a base HOMUS e a arquitetura LeNet utilizando a função de custo adaptada. A arquitetura de cada modelo é descrita em detalhes na Tabela 17.



Fonte: Autor

Tabela 16 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo entropia cruzada para o reconhecimento de símbolos musicais.

Modelo	Convolução 1	Pooling 1	Convolução 2	Pooling 2	Acurácia Média (%)	Desvio Padrão
1	3x3+1@256	2x2+1	5x5+1@128	2x2+1	84.85	0.86
2	3x3+2@16	3x3+1	5x5+2@256	2x2+1	85.26	0.68
3	3x3+1@128	2x2+1	5x5+1@32	2x2+1	85.47	0.45
4	3x3+1@64	2x2+1	5x5+1@32	2x2+1	85.99	0.60
5	3x3+1@64	2x2+1	5x5+1@256	2x2+1	85.87	1.25
6	3x3+1@64	2x2+1	5x5+1@64	2x2+1	86.14	0.80
7	3x3+1@16	2x2+1	5x5+1@256	2x2+1	86.45	0.36
8	5x5+1@256	3x3+1	2x2+1@128	2x2+1	86.80	0.31
9	3x3+1@16	2x2+2	1x1+1@256	3x3+2	87.55	0.52
10	5x5+1@16	3x3+1	2x2+1@256	2x2+1	87.70	0.72
11	3x3+1@16	3x3+2	3x3+1@256	2x2+2	87.79	0.36
12	5x5+1@128	3x3+1	2x2+1@32	2x2+1	87.93	0.61
13	5x5+1@64	3x3+1	2x2+1@64	2x2+1	87.89	0.49
14	5x5+1@64	3x3+1	2x2+1@32	2x2+1	87.95	0.43
15	3x3+2@64	3x3+1	5x5+2@32	2x2+1	88.22	0.70
16	3x3+1@64	3x3+2	3x3+1@32	2x2+2	88.84	0.84
17	3x3+2@64	3x3+1	5x5+2@64	2x2+1	88.57	0.63
18	3x3+2@128	3x3+1	5x5+2@32	2x2+1	89.01	0.78
19	5x5+1@64	3x3+1	2x2+1@256	2x2+1	88.24	0.96
20	3x3+2@64	3x3+1	5x5+2@256	2x2+1	89.12	0.38
21	3x3+1@128	3x3+2	3x3+1@32	2x2+2	89.56	0.32
22	3x3+2@256	3x3+1	5x5+2@128	2x2+1	89.07	0.66
23	3x3+1@64	2x2+2	1x1+1@64	3x3+2	89.78	0.56
24	3x3+1@64	3x3+2	3x3+1@256	2x2+2	89.66	0.57
25	3x3+1@256	2x2+2	1x1+1@128	3x3+2	90.08	0.53
26	3x3+1@128	2x2+2	1x1+1@32	3x3+2	90.32	0.44
27	3x3+1@64	3x3+2	3x3+1@64	2x2+2	89.61	0.82
28	3x3+1@64	2x2+2	1x1+1@32	3x3+2	90.20	0.27
29	3x3+1@256	3x3+2	3x3+1@128	2x2+2	90.18	0.39
30	3x3+1@64	2x2+2	1x1+1@256	3x3+2	90.22	0.38

Tabela 17 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo adaptada para o reconhecimento de símbolos musicais.

Modelo	Convolução 1	Pooling 1	Convolução 2	Pooling 2	Acurácia Média	Desvio Padrão
1	3x3+2@16	3x3+1	5x5+2@256	2x2+1	81.78	0.64
2	3x3+1@256	2x2+1	5x5+1@128	2x2+1	84.60	0.28
3	3x3+1@16	3x3+2	3x3+1@256	2x2+2	85.53	0.89
4	3x3+1@16	2x2+2	1x1+1@256	3x3+2	85.33	0.62
5	3x3+1@128	2x2+1	5x5+1@32	2x2+1	85.70	0.58
6	3x3+1@64	2x2+1	5x5+1@32	2x2+1	85.87	0.32
7	3x3+1@64	2x2+1	5x5+1@256	2x2+1	85.86	0.20
8	3x3+1@64	2x2+1	5x5+1@64	2x2+1	85.93	0.35
9	3x3+1@16	2x2+1	5x5+1@256	2x2+1	86.48	0.72
10	5x5+1@256	3x3+1	2x2+1@128	2x2+1	86.63	0.60
11	5x5+1@64	3x3+1	2x2+1@64	2x2+1	87.35	0.25
12	5x5+1@16	3x3+1	2x2+1@256	2x2+1	87.53	0.68
13	3x3+1@256	3x3+2	3x3+1@128	2x2+2	88.29	1.14
14	3x3+2@64	3x3+1	5x5+2@256	2x2+1	87.22	1.39
15	3x3+1@64	3x3+2	3x3+1@32	2x2+2	87.84	0.67
16	5x5+1@64	3x3+1	2x2+1@32	2x2+1	87.93	0.73
17	5x5+1@128	3x3+1	2x2+1@32	2x2+1	87.64	0.60
18	5x5+1@64	3x3+1	2x2+1@256	2x2+1	87.59	1.00
19	3x3+2@64	3x3+1	5x5+2@32	2x2+1	87.75	0.81
20	3x3+2@64	3x3+1	5x5+2@64	2x2+1	88.02	0.59
21	3x3+1@128	3x3+2	3x3+1@32	2x2+2	87.64	2.04
22	3x3+1@64	3x3+2	3x3+1@256	2x2+2	88.61	0.38
23	3x3+1@64	3x3+2	3x3+1@64	2x2+2	88.88	0.19
24	3x3+2@256	3x3+1	5x5+2@128	2x2+1	88.84	1.07
25	3x3+1@64	2x2+2	1x1+1@256	3x3+2	88.86	0.70
26	3x3+2@128	3x3+1	5x5+2@32	2x2+1	88.65	1.02
27	3x3+1@64	2x2+2	1x1+1@64	3x3+2	89.26	0.21
28	3x3+1@128	2x2+2	1x1+1@32	3x3+2	88.99	1.01
29	3x3+1@64	2x2+2	1x1+1@32	3x3+2	89.32	0.62
30	3x3+1@256	2x2+2	1x1+1@128	3x3+2	89.92	0.40

4.2.2 Diagnóstico de Glaucoma

Quando utilizada com a base RIM-ONE para o reconhecimento de glaucoma, cerca de 20% dos modelos treinados com a arquitetura LeNet possuíam alta variabilidade (acima de 5%), esses modelos estão em vermelho nas Figuras 24 e 25. Isso se deve a pequena quantidade de amostras durante o treino. Para facilitar a visualização, somente 1 em cada 2 modelos serão mostrados, gerando 30 resultados.

O melhor resultado foi obtido utilizando a função de custo adaptada (Tabela 20) com acurácia média de 88.57% e desvio padrão 1.79%. Seu modelo respectivo, utilizando a função de custo *categorical cross-entropy*, indicado pelo elemento 29 da Tabela 19 possui

desvio padrão superior (3.83%) e acurácia média inferior (87.69%) o que sugere um modelo menos confiável e com menor capacidade de predição.

De maneira geral, como descrito nas Tabelas 19 e 20, as arquiteturas que utilizaram *kernels* 1x1 na primeira convolução obtiveram os piores resultados. Isso por que campos receptivos 1x1 não permitem que haja extração real de características, mas servem principalmente para transformar a dimensão do volume tratado.

Por outro lado, a medida que o tamanho do campo receptivo nas camadas de convolução aumentam, a acurácia também tende a aumentar. Os modelos com melhores resultados possuem *kernels* 5x5 com passo 1x1 permitindo que características próximas sejam agrupadas.

Assim como ocorre no reconhecimento de símbolos musicais, quando a quantidade de filtros na primeira camada é inferior a da segunda, esses modelos tendem a ter o desvio padrão entre os testes próximos a 5%. Já quando a quantidade de filtros nas primeiras camadas é grande (64, 128 e 256) os modelos possuem maior poder de generalização. A razão é mantida a mesma, existe uma quantidade maior de as características simples a serem extraídas.

Novamente, nas camadas de sub-amostragem ocorreu sobreposição de filtros, contudo, em algumas arquiteturas o tamanho do filtro é menor do que o passo, ou que sugere perda de informação. Essa perda de informação pode ser associada às bordas adicionadas para o enquadramento das ROIs.

Os modelos que foram treinados com a função de custo entropia cruzada, na maioria dos casos, obtiveram um maior desvio padrão. Já aqueles treinados com a função de custo adaptada, normalmente possuem um desvio padrão menor o que sugere uma melhora nos resultados.

4.2.3 Análise de Sentimentos em Avaliações de Filmes

Por fim, os resultados obtidos com a LeNet para análise de sentimento em textos estão detalhados nas Tabelas 21 e 22 e resumidos nos gráficos das Figuras 26 e 27. Novamente, se fez necessária a condensação dos dados para facilitar a visualização, assim, 1 em cada 4 resultados serão mostrados.

O melhor modelo possui acurácia média de 88.30% e desvio padrão de 0.22% sobre as cinco execuções. A arquitetura possui convoluções com *kernels* 5x5 e tem sobreposição de informação em todas suas camadas. Nas camadas de sub-amostragem, os *kernels* foram 3x3 e 5x5 para a camada mais rasa e mais profunda da rede, respectivamente. Já a quantidade de filtros decaiu de 64 para 32 da primeira para a segunda convolução.

Mais uma vez, os piores resultados foram obtidos por modelos que possuem

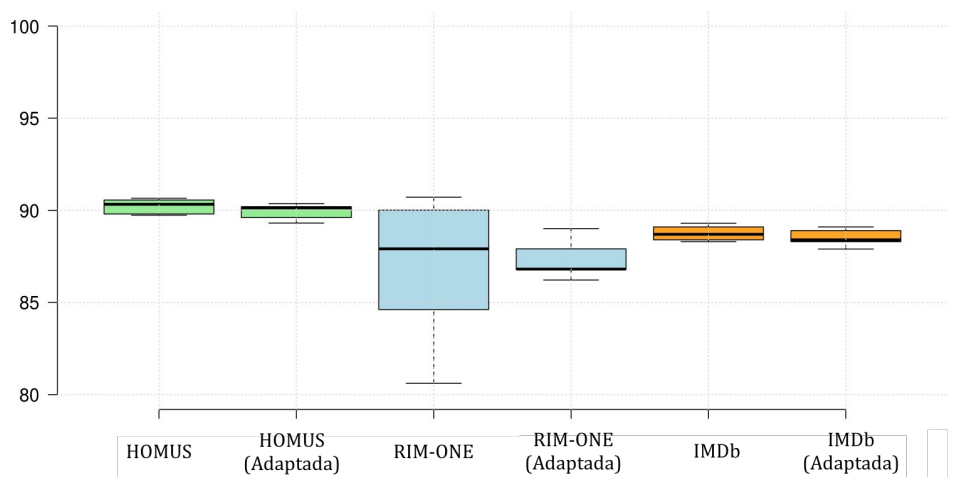
convoluções com poucos filtros treináveis, 16 e 32, tanto na camada mais rasa quanto na mais profunda. Os resultados não possuem altas taxas de variabilidade, sendo a maior delas encontrada no modelo 7 da Tabela 22 com desvio padrão de 2.37%.

Tabela 18 – Resumo das melhores arquiteturas baseadas na LeNet.

	HOMUS	RIM-ONE	IMDb
Convolução 1	Campo receptivo médio 3x3 com sobreposição	Campo receptivo grande 5x5 com muita sobreposição (stride 1 ou 2).	Campo receptivo grande 5x5 com muita sobreposição (stride 1).
Pooling 1	Campo receptivo médio 2x2 sem sobreposição ou 3x3 com passo 2.	Campo receptivo médio 2x2.	Campo receptivo grande 3x3 sem sobreposição com passo 3.
Convolução 2	Campo receptivo pequeno 1x1 com passo 1x1.	Campo receptivo grande 5x5 com muita sobreposição (stride 1 ou 2).	Campo receptivo grande 5x5 com muita sobreposição (stride 1).
Pooling 2	Campo receptivo médio 2x2 sem sobreposição ou 3x3 com passo 2.	Campo receptivo médio 3x3.	Campo receptivo grade 5x5 com passo 3 ou 2x2 com passo 1.

A Tabela 18 faz um resumo dos hiper-parâmetros que geraram os melhores modelos para cada problema. Em todas as arquiteturas as convoluções foram realizadas utilizando o passo menor que o tamanho do *kernel*, ou seja, com sobreposição. Nos tês casos, convoluções com poucos filtros (16 e 32) tendem a gerar os piores resultados. E a medida que a quantidade de filtros foi maior, principalmente na primeira convolução, a acurácia dos modelos tendem a subir. d

Figura 23 – Resumo dos resultados obtidos com a arquitetura LeNet.



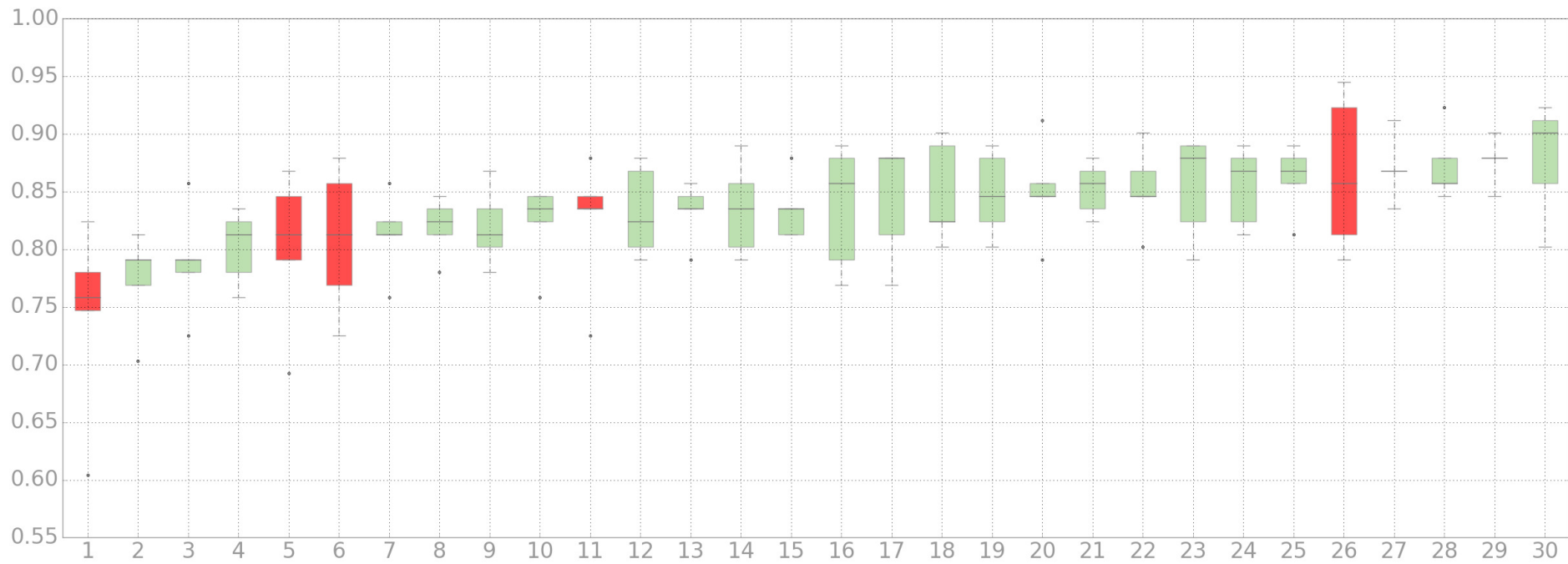
Fonte: Autor

A Figura 23 resume os melhores resultados obtidos na classificação dos diferentes problemas. Em verde estão descritos por meio de gráficos de caixa os melhores modelos treinados com a base de dados HOMUS, em azul para a base RIM-ONE e em laranja para base IMDB.

Baseado nos gráficos desta figura, é possível perceber que a função de custo adaptada mostrou resultados que se equiparam ou superam os resultados obtidos com a função de custo original. Em especial, para o diagnóstico de glaucoma com a base RIM-ONE, uma base desbalanceada, o melhor resultado obteve desvio padrão menor, cerca de 2.5%, utilizando a função de custo adaptada do que seu respectivo modelo treinado com a função de custo entropia cruzada.

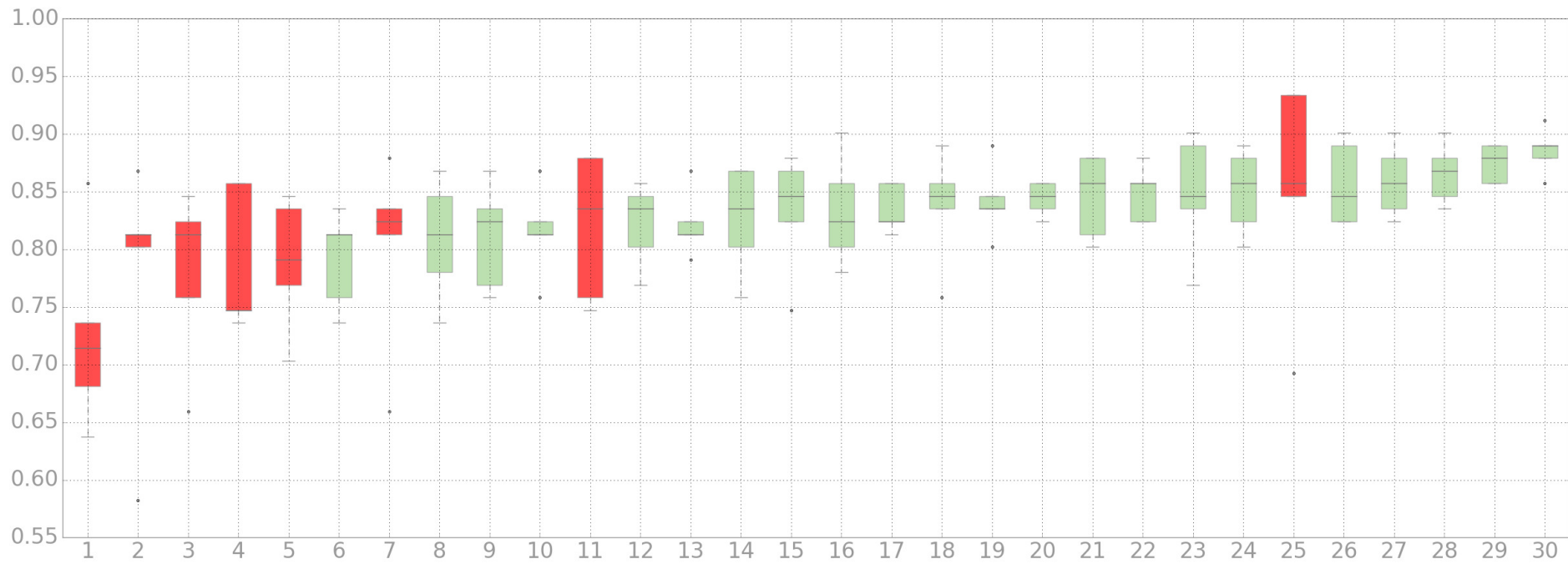
Contudo, frequentemente, quando se usa a função adaptada é possível perceber uma queda na acurácia média. Isso ocorre, normalmente, quando o modelo está tentando equilibrar a taxa de acerto em diferentes classes.

Figura 24 – Gráfico - Resultados com a base RIM-ONE e a arquitetura LeNet utilizando a função de custo entropia cruzada. Em vermelho estão representados os modelos com desvio padrão maior que 5%, consideradas em overfitting. A arquitetura de cada modelo é descrita em detalhes na Tabela 19.



Fonte: Autor

Figura 25 – Gráfico - Resultados com a base RIM-ONE e a arquitetura LeNet utilizando a função de custo adaptada. Em vermelho estão representados os modelos com desvio padrão maior que 5%, consideradas em overfitting. A arquitetura de cada modelo é descrita em detalhes na Tabela 20.



Fonte: Autor

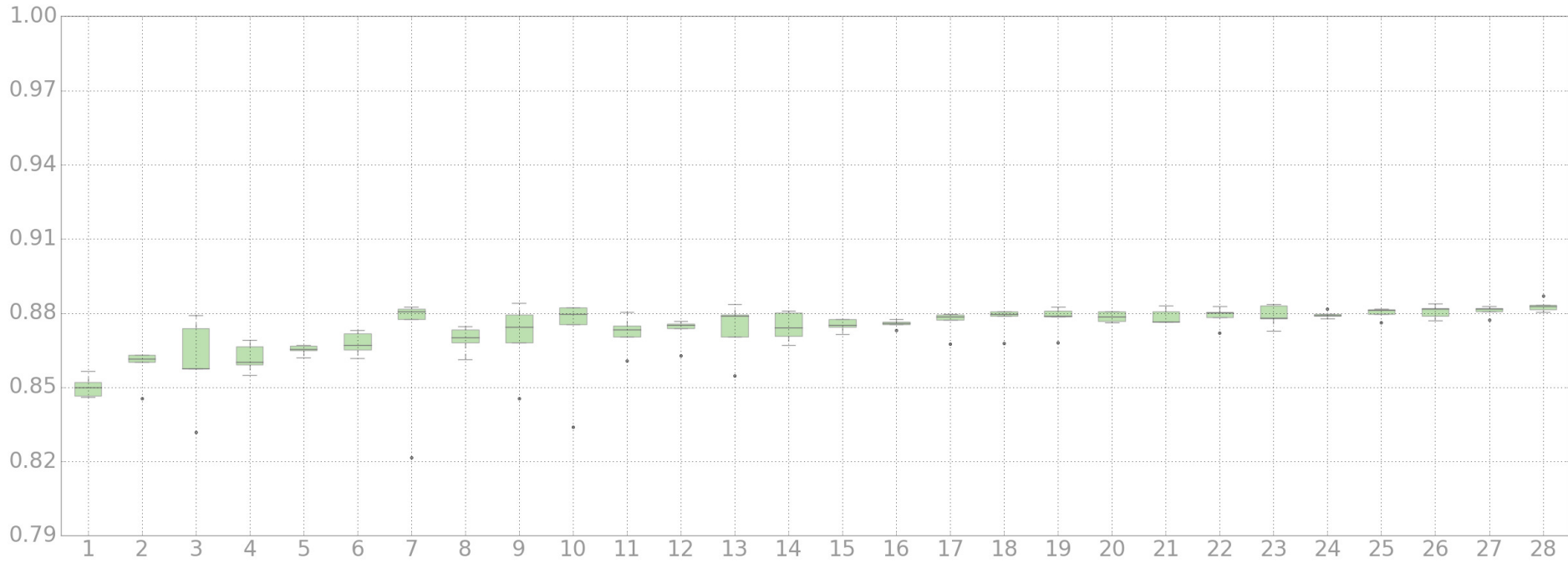
Tabela 19 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo entropia cruzada para o diagnóstico de glaucoma.

Modelo	Convolução 1	Pooling 1	Convolução 2	Pooling 2	Acurácia Média (%)	Desvio Padrão (%)
1	1x1+1@64	2x2+3	5x5+1@32	5x5+3	74.29	7.41
2	1x1+1@128	2x2+3	5x5+1@32	5x5+3	77.36	3.78
3	5x5+1@16	5x5+3	5x5+1@256	2x2+3	78.90	4.19
4	3x3+2@16	2x2+1	3x3+2@256	5x5+1	80.22	6.10
5	1x1+1@256	2x2+3	5x5+1@128	5x5+3	80.22	2.87
6	3x3+2@64	2x2+1	3x3+2@256	5x5+1	80.88	5.63
7	3x3+2@64	2x2+1	3x3+2@32	5x5+1	81.32	3.18
8	5x5+1@64	5x5+3	5x5+1@64	2x2+3	81.98	3.00
9	5x5+1@16	2x2+2	5x5+1@256	5x5+2	81.98	2.26
10	3x3+2@256	3x3+1	1x1+2@128	5x5+1	82.20	3.29
11	5x5+1@64	3x3+3	5x5+1@32	2x2+3	82.42	5.20
12	5x5+1@64	3x3+3	5x5+1@256	2x2+3	83.30	3.50
13	3x3+2@256	2x2+1	3x3+2@128	5x5+1	83.30	6.50
14	5x5+1@16	5x5+3	1x1+1@256	3x3+3	83.52	5.69
15	5x5+1@256	2x2+2	5x5+1@128	5x5+2	83.52	1.84
16	5x5+1@64	2x2+2	5x5+1@64	5x5+2	83.74	4.84
17	5x5+1@128	2x2+2	5x5+1@32	5x5+2	84.40	4.53
18	3x3+2@64	3x3+1	1x1+2@256	5x5+1	84.84	3.96
19	5x5+1@16	2x2+3	5x5+1@256	3x3+3	84.84	4.42
20	3x3+2@128	3x3+1	1x1+2@32	5x5+1	85.05	4.59
21	5x5+1@128	2x2+3	5x5+1@32	3x3+3	85.27	4.54
22	5x5+1@128	5x5+3	5x5+1@32	2x2+3	85.27	2.15
23	5x5+1@64	2x2+2	5x5+1@32	5x5+2	85.49	4.02
24	3x3+1@256	2x2+3	5x5+1@128	3x3+3	85.49	1.62
25	3x3+1@128	2x2+3	5x5+1@32	3x3+3	86.15	2.66
26	5x5+1@64	3x3+3	5x5+1@64	2x2+3	86.59	5.99
27	3x3+1@64	2x2+3	5x5+1@256	3x3+3	87.03	2.45
28	5x5+1@256	5x5+3	1x1+1@128	3x3+3	87.25	2.75
29	5x5+1@64	2x2+3	5x5+1@32	3x3+3	87.69	3.83
30	5x5+1@256	2x2+3	5x5+1@128	3x3+3	87.91	4.45

Tabela 20 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo adaptada para o diagnóstico de glaucoma.

Modelo	Convolução 1	Pooling 1	Convolução 2	Pooling 2	Acurácia Média (%)	Desvio Padrão (%)
1	1x1+1@64	2x2+3	5x5+1@256	5x5+3	72.53	7.39
2	5x5+1@128	3x3+3	5x5+1@32	2x2+3	77.58	9.94
3	5x5+1@16	2x2+3	5x5+1@256	3x3+3	78.02	6.70
4	1x1+1@64	2x2+3	5x5+1@64	5x5+3	78.90	5.58
5	1x1+1@128	2x2+3	5x5+1@32	5x5+3	78.90	4.68
6	5x5+1@64	3x3+3	5x5+1@256	2x2+3	79.12	3.74
7	5x5+2@256	2x2+1	5x5+2@128	3x3+1	80.22	7.49
8	3x3+1@16	2x2+3	5x5+1@256	3x3+3	80.88	4.69
9	3x3+2@256	2x2+1	3x3+2@128	5x5+1	81.10	2.81
10	5x5+2@16	2x2+1	5x5+2@256	3x3+1	81.54	3.50
11	3x3+1@128	2x2+3	5x5+1@32	3x3+3	81.98	5.88
12	5x5+1@64	5x5+3	5x5+1@32	2x2+3	82.20	3.22
13	3x3+2@64	3x3+1	1x1+2@64	5x5+1	82.20	2.54
14	5x5+1@64	5x5+3	1x1+1@32	3x3+3	82.64	4.19
15	5x5+2@128	2x2+1	5x5+2@32	3x3+1	83.30	2.24
16	5x5+1@16	5x5+3	5x5+1@256	2x2+3	83.30	4.25
17	3x3+2@256	3x3+1	1x1+2@128	5x5+1	83.52	1.84
18	5x5+1@64	5x5+3	1x1+1@256	3x3+3	83.74	4.36
19	5x5+1@256	3x3+3	5x5+1@128	2x2+3	84.18	3.97
20	5x5+1@16	2x2+2	5x5+1@256	5x5+2	84.40	2.98
21	5x5+1@64	3x3+3	5x5+1@32	2x2+3	84.62	3.48
22	3x3+2@64	2x2+1	3x3+2@256	5x5+1	84.84	2.13
23	5x5+1@128	5x5+3	1x1+1@32	3x3+3	84.84	4.68
24	5x5+2@64	2x2+1	5x5+2@64	3x3+1	85.05	3.30
25	5x5+1@256	2x2+3	5x5+1@128	3x3+3	85.27	8.84
26	5x5+2@64	2x2+1	5x5+2@32	3x3+1	85.71	3.26
27	3x3+1@256	2x2+3	5x5+1@128	3x3+3	85.93	2.81
28	5x5+1@64	2x2+2	5x5+1@64	5x5+2	86.59	2.35
29	5x5+1@64	2x2+2	5x5+1@32	5x5+2	87.47	1.49
30	5x5+1@64	2x2+3	5x5+1@32	3x3+3	88.57	1.79

Figura 26 – Gráfico - Resultados com a base IMDb e a arquitetura LeNet utilizando a função de custo entropia cruzada. A arquitetura de cada modelo é descrita em detalhes na Tabela 21.



Fonte: Autor

Fonte: Autor

Figura 27 – Gráfico - Resultados com a base IMDb e a arquitetura LeNet utilizando a função de custo adaptada. A arquitetura de cada modelo é descrita em detalhes na Tabela 22.

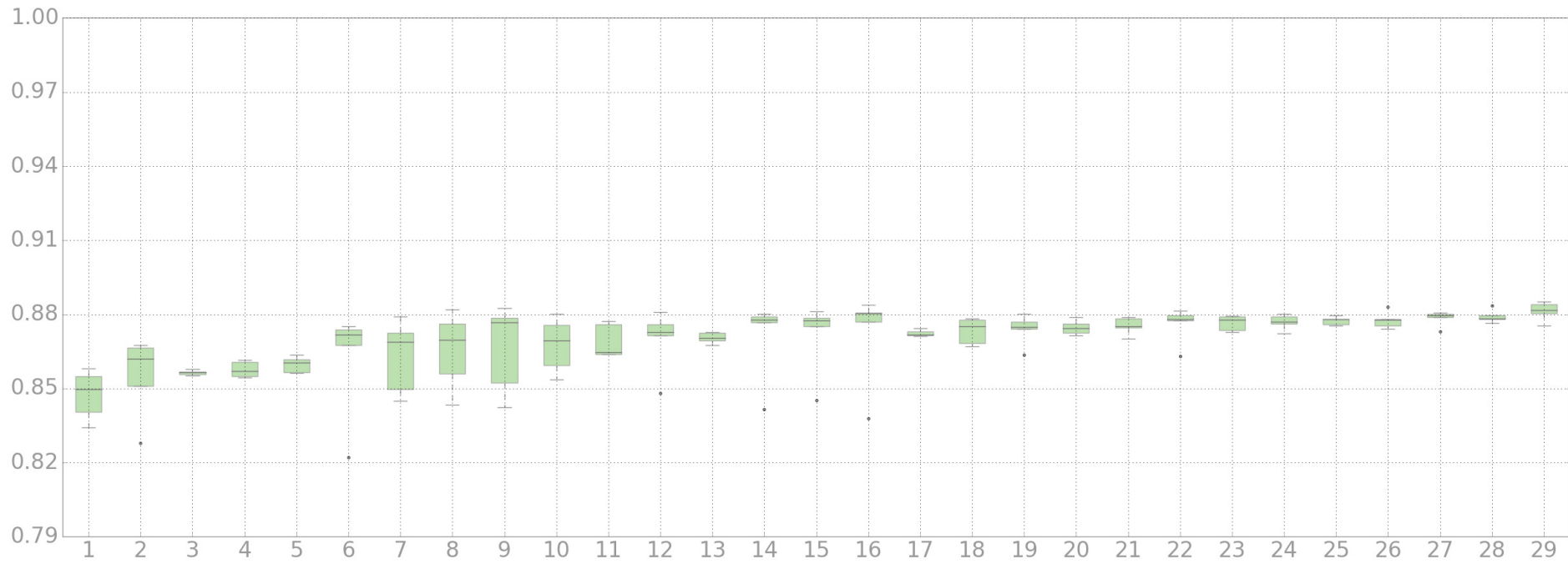


Tabela 21 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo entropia cruzada para a análise de sentimentos em texto.

Modelo	Convolução 1	Pooling 1	Convolução 2	Pooling 2	Acurácia Média (%)	Desvio Padrão (%)
1	1x1+1@16	5x5+3	3x3+1@256	2x2+3	85.02	0.39
2	3x3+2@64	5x5+1	3x3+2@64	2x2+1	85.86	0.66
3	5x5+1@16	2x2+1	5x5+1@256	5x5+1	86.00	1.65
4	3x3+1@16	3x3+2	1x1+1@256	2x2+2	86.20	0.51
5	3x3+2@256	5x5+1	3x3+2@128	2x2+1	86.52	0.17
6	1x1+1@16	5x5+1	3x3+1@256	3x3+1	86.78	0.42
7	3x3+1@128	2x2+1	5x5+1@32	5x5+1	86.88	2.37
8	3x3+1@128	3x3+2	1x1+1@32	2x2+2	86.95	0.47
9	3x3+1@128	3x3+1	5x5+1@32	2x2+1	87.03	1.35
10	5x5+1@128	3x3+1	1x1+1@32	2x2+1	87.07	1.86
11	5x5+1@16	3x3+3	5x5+1@256	2x2+3	87.19	0.65
12	3x3+1@16	2x2+1	5x5+1@256	5x5+1	87.29	0.52
13	3x3+1@64	3x3+1	5x5+1@256	2x2+1	87.34	1.03
14	3x3+1@64	2x2+2	3x3+1@64	3x3+2	87.46	0.54
15	3x3+1@64	2x2+2	3x3+1@32	3x3+2	87.52	0.22
16	5x5+1@64	3x3+3	5x5+1@64	5x5+3	87.57	0.14
17	3x3+1@64	2x2+1	5x5+1@32	5x5+1	87.65	0.45
18	5x5+1@64	3x3+1	1x1+1@64	2x2+1	87.75	0.49
19	5x5+1@128	3x3+3	5x5+1@32	2x2+3	87.78	0.51
20	1x1+1@256	5x5+1	3x3+1@128	3x3+1	87.86	0.19
21	1x1+1@64	5x5+1	3x3+1@32	3x3+1	87.87	0.27
22	5x5+1@64	2x2+1	5x5+1@64	5x5+1	87.88	0.37
23	1x1+1@64	2x2+1	5x5+1@64	3x3+1	87.91	0.39
24	1x1+1@64	2x2+1	5x5+1@256	3x3+1	87.94	0.13
25	1x1+1@64	2x2+1	5x5+1@32	3x3+1	88.00	0.20
26	5x5+1@64	3x3+3	5x5+1@256	2x2+3	88.07	0.24
27	1x1+1@256	3x3+1	3x3+1@128	2x2+1	88.09	0.20
28	5x5+1@64	3x3+3	5x5+1@32	5x5+3	88.30	0.22

Tabela 22 – Descrição dos modelos e resultados obtidos utilizando a arquitetura LeNet e a função de custo adaptada para a análise de sentimentos em texto.

Modelo	Convolução 1	Pooling 1	Convolução 2	Pooling 2	Acurácia Média (%)	Desvio Padrão (%)
1	3x3+2@16	5x5+1	3x3+2@256	2x2+1	84.76	0.89
2	3x3+1@128	3x3+2	1x1+1@32	2x2+2	85.50	1.48
3	1x1+1@256	5x5+3	3x3+1@128	2x2+3	85.64	0.09
4	1x1+1@128	5x5+3	3x3+1@32	2x2+3	85.77	0.29
5	3x3+2@64	5x5+1	3x3+2@32	2x2+1	85.97	0.29
6	3x3+1@64	2x2+1	1x1+1@64	3x3+1	86.21	2.01
7	5x5+1@128	2x2+1	5x5+1@32	5x5+1	86.30	1.33
8	5x5+1@128	5x5+1	1x1+1@32	3x3+1	86.54	1.41
9	3x3+1@128	2x2+1	5x5+1@32	5x5+1	86.65	1.61
10	5x5+1@16	5x5+1	1x1+1@256	3x3+1	86.76	0.99
11	3x3+1@64	3x3+2	3x3+1@256	2x2+2	86.91	0.62
12	3x3+1@64	2x2+2	3x3+1@64	3x3+2	86.99	1.14
13	1x1+1@256	3x3+2	3x3+1@128	5x5+2	87.06	0.19
14	1x1+1@128	2x2+1	3x3+1@32	5x5+1	87.11	1.48
15	5x5+1@64	3x3+1	1x1+1@32	2x2+1	87.15	1.33
16	3x3+1@64	2x2+1	5x5+1@64	5x5+1	87.19	1.72
17	1x1+1@64	3x3+2	3x3+1@256	5x5+2	87.25	0.12
18	5x5+1@16	3x3+3	5x5+1@256	5x5+3	87.34	0.47
19	3x3+1@256	2x2+2	3x3+1@128	3x3+2	87.39	0.56
20	1x1+1@64	5x5+1	3x3+1@256	3x3+1	87.47	0.27
21	1x1+1@64	5x5+1	3x3+1@64	3x3+1	87.54	0.31
22	3x3+1@64	2x2+2	3x3+1@32	3x3+2	87.60	0.43
23	3x3+1@16	3x3+1	5x5+1@256	2x2+1	87.65	0.27
24	1x1+1@64	3x3+1	3x3+1@64	2x2+1	87.70	0.27
25	1x1+1@256	2x2+1	3x3+1@128	5x5+1	87.74	0.15
26	5x5+1@128	3x3+3	5x5+1@32	2x2+3	87.77	0.30
27	5x5+1@64	3x3+3	5x5+1@64	5x5+3	87.85	0.28
28	5x5+1@256	3x3+3	5x5+1@128	5x5+3	87.92	0.24
29	3x3+1@256	3x3+1	5x5+1@128	2x2+1	88.14	0.34

4.3 Resultados: ResNet-30

Nesta seção, serão discutidos os resultados obtidos utilizando a ResNet-30 seguindo a escolha de hiper-parâmetros proposta na Seção 3.3.3. Essa arquitetura superou todas as outras no reconhecimento de símbolos musicais. Já no diagnóstico de glaucoma e na análise de textos, os modelos gerados possuíram altas taxas de variabilidade e não superaram os modelos obtidos com a rede LeNet.

4.3.1 Reconhecimento de Símbolos Musicais

A arquitetura ResNet-30 gerou o melhor na tarefa de classificação de notas musicais. As Tabelas 23 e 24 descrevem as arquiteturas e resultados obtidos para esta tarefa e os

gráficos da Figura 28 resume estes resultados. Com acurácia média de 95.16% o melhor modelo possui baixa taxa de variabilidade, com desvio padrão de 0.33% entre as 5 interações. A arquitetura possui *kernel* 5x5 com os filtros 128, 64 e 32 (Combinação 4 da Tabela 14).

Em geral, utilizar convoluções com campo receptivo 2x2 gerou os modelos com piores poder de classificação (80% ~ 81.7%). Contudo, nenhum modelo demonstrou está em *overfitting* ou *underfitting*. Apesar de não ocorrer com o melhor modelo, em geral, a função de custo adaptada possui menor desvio padrão, como no modelo 5 de ambos casos. Isso ocorre em cerca de 66% dos modelos treinados.

Assim como ocorreu com a rede LeNet, novamente, os melhores resultados foram aqueles que iniciam com filtros grandes e vão diminuindo, ou seja, a combinação 3 e 4 da Tabela 14 para, respectivamente, os modelos com a função de custo entropia cruzada e a adaptada.

Tabela 23 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo entropia cruzada para o reconhecimento de símbolos musicais.

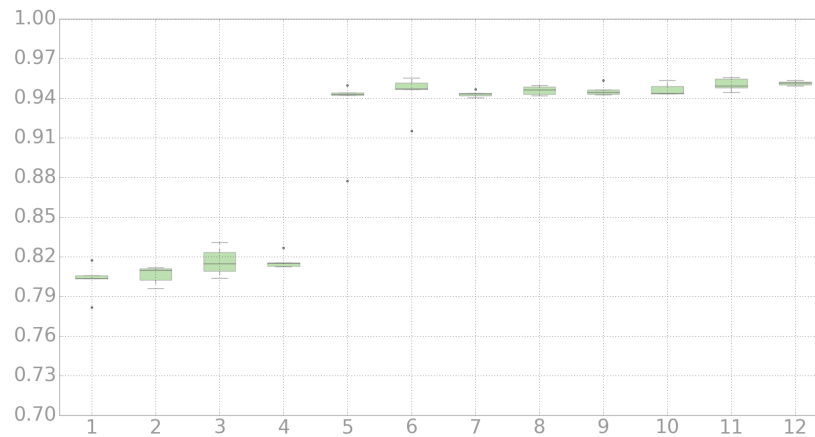
Modelo	Kernel	Filtros 1	Filtros 2	Filtros 3	Acurácia Média (%)	Desvio Padrão (%)
1	2	32	64	128	80.25	1.16
2	2	256	128	64	80.63	0.60
3	2	128	64	32	81.66	0.97
4	2	64	128	256	81.67	0.53
5	3	64	128	256	93.14	2.71
6	5	32	64	128	94.34	1.44
7	3	32	64	128	94.34	0.21
8	3	256	128	64	94.61	0.31
9	5	64	128	256	94.61	0.40
10	3	128	64	32	94.68	0.41
11	5	256	128	64	95.06	0.42
12	5	128	64	32	95.16	0.15

4.3.2 Diagnóstico de Glaucoma

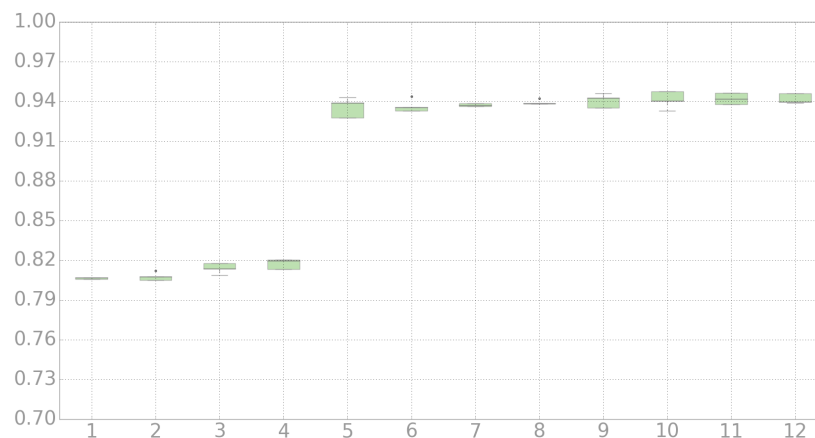
Como esperado, devido a quantidade de amostras, novamente os modelos gerados para a classificação de retinografias possuem altas taxas de variabilidade. Somente 7 das 24 arquiteturas avaliadas têm índices aceitáveis, com desvio padrão menor que 5% entre os cinco experimentos realizados. Assim, serão apresentados nessa seção somente estes modelos com baixa variabilidade. Estes estão apresentados pelos gráficos na Figura 29 e descritos nas Tabelas 26 e 25.

Dessas 7 arquiteturas, 2 estão presentes utilizando ambas funções de custo, 3 está presente nos resultados obtidos utilizando a função de entropia cruzada e as outras 2 nos

Figura 28 – Gráfico - Resultados com a base HOMUS e a arquitetura ResNet-30. **Em (a):** estão contidos os resultados obtidos com a função de custo entropia cruzada. **Em (b):** estão contidos os resultados obtidos com a função de custo adaptada.



(a)



(b)

Fonte: Autor

resultados gerados com a função de custo adaptada. Estes modelos estão contidos nas Figura 29 para ambas funções de custo. Quando comparada com suas equivalentes, as arquiteturas utilizando a função de custo adaptada possuem menor desvio padrão nos cinco treinos. Em especial, no modelo 21 da Figura 29a a acurácia média subiu em cerca de 1.1% quando utilizado a nova função de custo e diminui cerca de 2.77% no desvio padrão, sendo representado pelo modelo 20 na Figura 29b.

Os melhores modelos, utilizando ambas funções de custo, possuem organização similares. Tamanho de *kernel* 5, *zero-padding* 1 e filtros com tamanho 128, 64 e 32 sendo representados pela combinação 4. Utilizando a função de custo adaptada, o modelo teve desvio padrão de 1.8% com 82% de acurácia e utilizando a função original 4.2% e 84.17%.

Pode-se perceber, por fim, que nenhum modelo com quantidade decrescente de

Tabela 24 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo adaptada para o reconhecimento de símbolos musicais.

Modelo	Kernel	Filtros 1	Filtros 2	Filtros 3	Acurácia Média (%)	Desvio Padrão (%)
1	2	256	128	64	80.72	0.16
2	2	64	128	256	80.91	0.31
3	2	32	64	128	81.18	0.45
4	2	128	64	32	81.35	0.86
5	3	64	128	256	93.70	0.40
6	3	128	64	32	93.82	0.17
7	5	32	64	128	93.99	0.45
8	3	32	64	128	93.82	0.63
9	5	256	128	64	94.10	0.42
10	5	64	128	256	94.17	0.57
11	3	256	128	64	94.16	0.40
12	5	128	64	32	94.29	0.33

Tabela 25 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo adaptada para o diagnóstico de glaucoma.

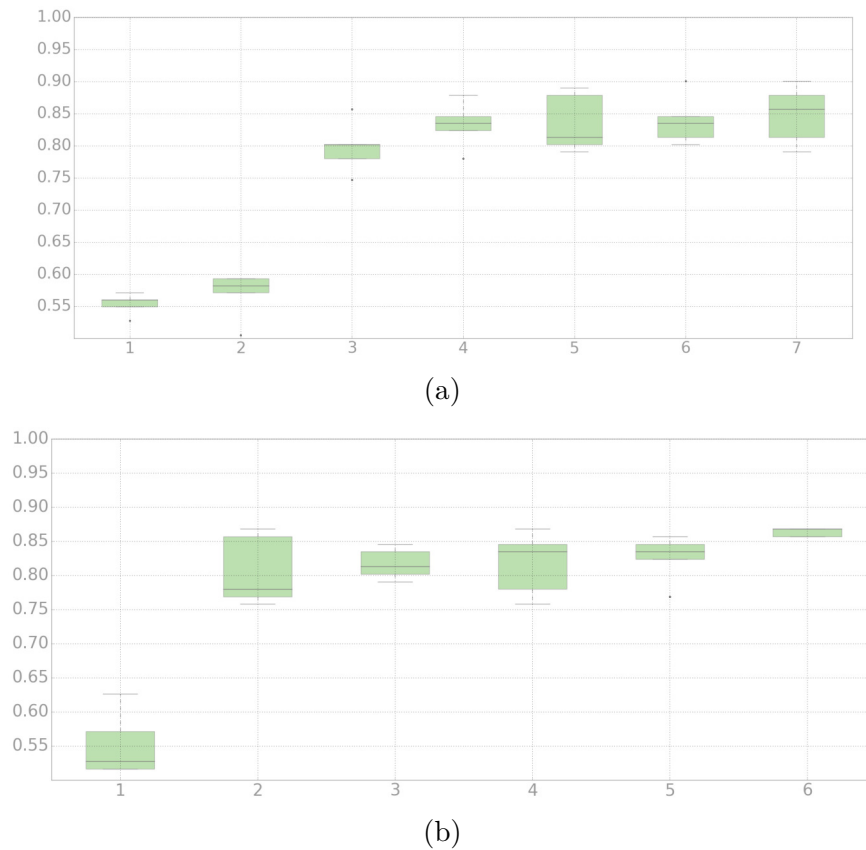
Modelo	Kernel	Filtros 1	Filtros 2	Filtros 3	Padding	Acurácia Média (%)	Desvio Padrão
1	5	32	64	128	3	55.16	4.25
2	3	256	128	64	1	80.66	4.64
3	3	128	64	32	1	81.76	2.04
4	5	128	64	32	1	82.64	3.06
5	2	256	128	64	1	81.76	4.15
6	2	128	64	32	3	86.37	0.54

filtros obteve resultados satisfatórios. Os modelos 1 e 2 da Tabela 26 e o modelo 1 da Tabela 25 que possuem essa combinação do hiper-parâmetro estão em *underfitting*.

Tabela 26 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo entropia cruzada para o diagnóstico de glaucoma.

Modelo	Kernel	Filtros 1	Filtros 2	Filtros 3	Padding	Acurácia Média (%)	Desvio Padrão
1	2	64	128	256	1	55.38	1.49
2	3	64	128	256	3	56.92	3.29
3	2	128	64	32	3	79.78	3.58
4	3	256	128	64	1	83.52	4.11
5	3	128	64	32	1	83.30	3.22
6	5	256	128	64	1	83.96	3.45
7	5	128	64	32	1	84.84	4.08

Figura 29 – Gráfico - Resultados com a base RIM-ONE e a arquitetura ResNet-30. **Em (a)**: estão contidos os resultados obtidos com a função de custo entropia cruzada. **Em (b)**: estão contidos os resultados obtidos com a função de custo adaptada.



Fonte: Autor

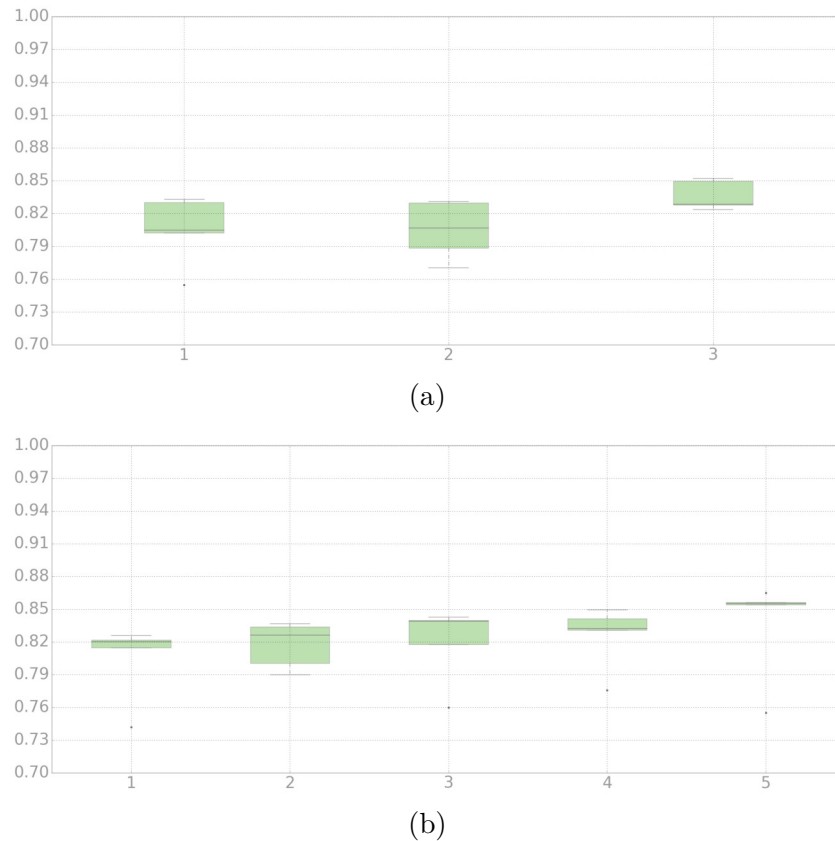
4.3.3 Análise de Sentimentos em Avaliações de Filmes

Com a base de dados IMDb, os modelos residuais também não superaram os resultados obtidos pelos modelos baseados na LeNet. A maioria das arquiteturas testadas resultaram em *underfitting* em pelo menos um dos cinco experimentos, assim, nas Figuras 30a e 30a estão contidos somente os resultados que não obtiveram *underfitting* em nenhum dos experimentos.

A função de custo adaptada obteve 5 modelos que não apresentaram *underfitting* enquanto a função de custo entropia cruzada obteve somente 3. Estes modelos estão descritos nas Tabelas 27 e 28, o valor de *stride* "X" representa os modelos que utilizam, nas camadas de convolução, os valores padrões desse hiper-parâmetro, ou seja, o mesmo valor do *kernel*.

Em geral, os modelos com *stride* 1 tiveram os melhores resultados. O melhor resultado foi obtido utilizando a função de custo adaptada. O modelo com *kernel* de

Figura 30 – Gráfico - Resultados com a base IMDb e a arquitetura ResNet-30. **Em (a):** estão contidos os resultados obtidos com a função de custo entropia cruzada. **Em (b):** estão contidos os resultados obtidos com a função de custo adaptada.



Fonte: Autor

Tabela 27 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo entropia cruzada para a análise de sentimentos em texto.

Modelo	Kernel	Stride	Filtros 1	Filtros 2	Filtros 3	Padding	Acc. Média (%)	Desvio Padrão (%)
1	3	X	64	128	256	3	80.52	2.81
2	3	X	32	64	128	3	80.56	2.35
3	5	1	64	128	256	0	83.65	1.20

tamanho 5, stride 1, sem *zero-padding* e com filtros decaindo 256, 128, 64 obteve acurácia média de 83.73% e 4.11% de desvio padrão. Contudo, este resultado não supera o resultado obtido com a arquitetura LeNet.

A Tabela 29 resume os resultados obtidos com a rede residual utilizada. Para todos os problemas, arquiteturas com kernel 5x5 e filtros decrescendo inversamente proporcional a profundidade da camada de convolução ao qual pertencem, obtiveram os melhores resultados.

Tabela 28 – Descrição dos modelos e resultados obtidos utilizando a arquitetura ResNet-30 e a função de custo adaptada para a análise de sentimentos em texto.

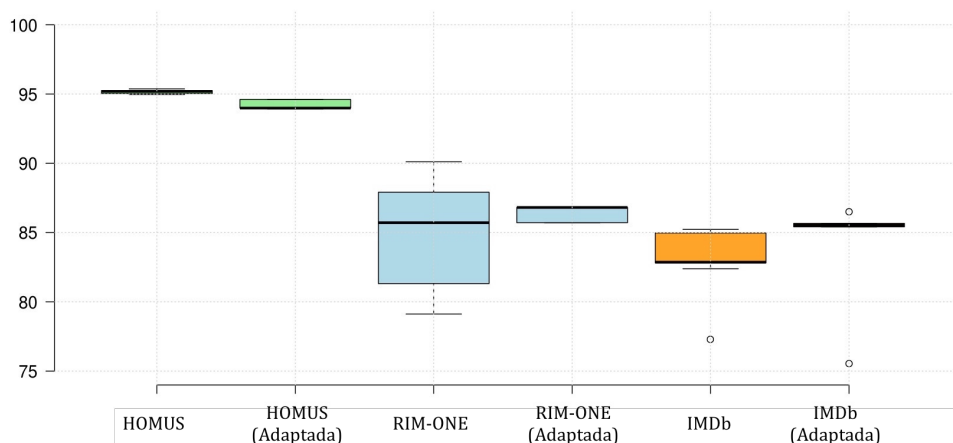
Modelo	Kernel	Stride	Filtros 1	Filtros 2	Filtros 3	Padding	Acc. Média (%)	Desvio Padrão (%)
1	2	X	128	64	32	1	80.51	3.17
2	2	X	128	64	32	0	81.76	1.89
3	2	X	256	128	64	1	82.01	3.14
4	5	1	128	64	32	1	82.62	2.59
5	5	1	256	128	64	0	83.73	4.11

Tabela 29 – Resumo das melhores arquiteturas baseadas na ResNet-30

	HOMUS	RIM-ONE	IMDb
Kernel	Campo receptivo grande (5x5).	Campo receptivo 5x5 com zero-padding 1x1.	Campo receptivo 5x5 com stride 1x1.
Bloco 1	Maior quantidade de filtros.	Maior quantidade de filtros.	Maior quantidade de filtros.
Bloco 2	Quantidade de filtros mediana.	Quantidade de filtros mediana.	Quantidade de filtros mediana.
Bloco 3	Menor quantidade de filtros.	Menor quantidade de filtros.	Menor quantidade de filtros.

Para o diagnóstico de imagens glaucomatosas e para a análise de sentimento em textos as redes residuais não ultrapassaram os modelos gerados utilizando arquiteturas mais simples baseadas na LeNet. Contudo, os melhores modelos em todos os casos possuem as mesmas características, campo receptivo 5x5 e quantidade de filtros decaindo a medida que as camadas vão ficando mais profundas. Por outro lado, como visto, para a classificação de símbolos musicais, os melhores modelos foram obtidos com as redes residuais com acurácia acima de 90% e baixo desvio padrão.

Figura 31 – Resumo dos resultados obtidos com a arquitetura ResNet-30.



A Figura 31 faz um resumo dos melhores resultados obtidos utilizando a arquitetura ResNet-30. Novamente, treinando modelos com a base de dados RIM-ONE e a função adaptada gerou modelos mais precisos. Por outro os modelos treinados com a base IMDB e a função adaptada não geraram bons resultados.

5 Publicações

Durante o processo de desenvolvimento desta monografia, os seguintes trabalhos foram publicados:

- PEREIRA, R. M. P.; MAIA, L. B.; SILVA, T. A.; PESSOA, A. C. P.; JUNIOR, G. B. Um estudo sobre diferentes tipos de funções de custo para redes neurais convolucionais. Jornada de Informática do Maranhão, v. 6, São Luís, Maranhão, 2016.
- PEREIRA, R. M. P.; MAIA, L. B.; SILVA, T. A.; PESSOA, A. C. P.; JUNIOR, G. B.; ALMEIDA, J. D. Uma Abordagem Genética Para Redes Neurais Convolucionais. Jornada de Informática do Maranhão, v. 6, São Luís, Maranhão, 2016.
- PEREIRA, R. M. P.; MATOS, C.; DINIZ, J.; JUNIOR, G. B.; ALMEIDA, J. D. D.; SILVA, A. C.; PAIVA, A. C. de. Abordagem deep learning para classificação de lesões mamárias. WIM-XVI Workshop de Informática Médica, Porto Alegre, 2016.
- PEREIRA, R. M. P.; MATOS, C. E.; JUNIOR, G. B.; ALMEIDA, J. D.; PAIVA, A. C. de. A deep approach for handwritten musical symbols recognition. In: ACM. Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web. Teresina, Piauí, 2016. p. 191–194.

6 Conclusão

Este trabalho apresentou o uso de Redes Neurais Convolucionais no reconhecimento de símbolos musicais, glaucoma e processamento de linguagem natural. Ainda, discutiu-se a variação dos hiper-parâmetros das redes LeNet, VGG-16 e ResNet e sua implicação no desempenho de cada uma das tarefas, além de estudar uma função de custo adaptada que utiliza a sensibilidade média das classes para gerar modelos com menor taxa de variabilidade.

Com exceção da rede VGG-16 que falhou em todas as tarefas, o uso de CNN mostrou resultados promissores. Com 88,57% de acurácia o modelo baseado na rede LeNet com filtros de convolução grandes ($5 \times 5 + 1$) e camadas de *pooling* ($2 \times 2 + 3$ e $3 \times 3 + 3$) obteve o melhor resultado para o reconhecimento de glaucoma. Já para a tarefa de reconhecimento de símbolos musicais, a rede residual foi a que teve o melhor desempenho com 95,16% de acurácia e desvio padrão de 0,15% entre os testes. Por fim, a análise de sentimentos em textos teve seu melhor modelo com acurácia 88,30% também baseado na arquitetura LeNet.

Os resultados obtidos utilizando a rede LeNet para o reconhecimento de símbolos musicais superaram obtidos anteriormente com a mesma arquitetura. Neste trabalho, o melhor modelo encontrado com esta arquitetura possui 9,78% de erro, já em (PEREIRA et al., 2016c) obtivemos erros superiores a 21% para esta arquitetura e base.

Já no diagnóstico de glaucomas com a base RIM-ONE Haleem et al. (2016) obteve acurácia de 94,4% utilizando extratores de características regionais. E, para a base IMDb (JOHNSON; ZHANG, 2014) obteve 92,33% de acurácia. Ambos métodos utilizam extratores de características ou métodos similares. Neste trabalho, estudamos a extração completamente automática sem qualquer engenharia ou seleção prévia de características obtendo bons resultados com um mesmo método.

A quantidade reduzida de amostras foi o principal obstáculo a ser vencido pelas arquiteturas estudadas. Como dito anteriormente, CNNs necessitam de uma quantidade de amostras grande, cerca de 1.000 por classe, para auxiliar na etapa de treino. Neste trabalho, nenhuma das bases estudadas possuíam essa quantidade de amostras. Contudo, isso não inviabiliza os resultados. Na verdade, pode-se perceber que as bases com pequena quantidade de dados obtiveram resultados melhores com arquiteturas mais rasas e a base com maior quantidade de dados obteve melhor resultado com a arquitetura residual que é mais profunda.

A função de custo adaptada que utiliza a sensibilidade média entre as classes para contornar o treino sobre bases desbalanceadas se mostrou eficiente. Os melhores resultados

obtidos para o reconhecimento de imagens glaucomatosas, utilizando as redes LeNet e ResNet-30, foram obtidos com os modelos treinados com a função de custo adaptada. Além do maior poder de predição, estes modelos também possuem maior generalização.

Por fim, acreditamos ser possível utilizar os modelos obtidos neste trabalho para outros problemas similares. Percebeu-se que os melhores modelos extraem uma maior quantidade de características simples e esse número deve ser convertido em uma quantidade menor de características complexas. Esses modelos, geralmente, possuem sobreposição quando possuem filtros grandes (5×5 *pixels*).

6.1 Trabalhos Futuros

Com resultados positivos, a metodologia abre espaço para a evolução dos trabalhos nestas três áreas. Em especial, o uso de um classificador de símbolos musicais robusto pode integrar um OMR completo o que viabilizaria tanto a conservação do acervo histórico musical do maranhão como também possibilitaria a tradução de matérias desse tipo para braile e outras notações.

Os resultados promissores obtidos na classificação das imagens de retinografia servem de base para o desenvolvimento de novas arquiteturas. Ainda, técnicas de reamostragem podem ser utilizadas para aumentar a quantidade de amostras e, assim, auxiliar no treino de modelos mais precisos e robustos.

O mesmo se aplica aos resultados obtidos com o processamento de linguagem natural com CNNs. O desenvolvimento de novos modelos baseados nos resultados deste trabalho podem auxiliar em diferentes áreas de pesquisa socioeconômicas como o estudo de mercado, previsão de votações e tarefas como a descrição de cenas.

Referências

- ABRAG. *Glaucoma Crônico de Ângulo Aberto*. 2014. Acessado em 28 de Novembro de 2016. Disponível em: <http://www.abrag.org.br/tipos_glaucoma.php>. Citado na página 46.
- AQUINO, A.; GEGÚNDEZ-ARIAS, M. E.; MARÍN, D. Detecting the optic disc boundary in digital fundus images using morphological, edge detection, and feature extraction techniques. *IEEE transactions on medical imaging*, IEEE, v. 29, n. 11, p. 1860–1869, 2010. Citado na página 48.
- BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, v. 13, n. Feb, p. 281–305, 2012. Citado na página 51.
- CALVO-ZARAGOZA, J.; ONCINA, J. Recognition of pen-based music notation: the homus dataset. In: IEEE. *2014 22nd International Conference on Pattern Recognition (ICPR)*. Sweden, 2014. p. 3038–3043. Citado 2 vezes nas páginas 17 e 45.
- CHEN, H.; DOU, Q.; YU, L.; HENG, P.-A. Voxresnet: Deep voxelwise residual networks for volumetric brain segmentation. *arXiv preprint arXiv:1608.05895*, 2016. Citado 2 vezes nas páginas 16 e 18.
- CHOLLET, F. *Keras*. USA: GitHub, 2015. <<https://github.com/fchollet/keras>>. Citado na página 48.
- DAI, J.; HE, K.; SUN, J. Instance-aware semantic segmentation via multi-task network cascades. *arXiv preprint arXiv:1512.04412*, 2015. Citado na página 16.
- DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009*. Florida, USA: IEEE, 2009. p. 248–255. Citado 2 vezes nas páginas 14 e 38.
- DOSOVITSKIY, A.; SPRINGENBERG, J. T.; BROX, T. Learning to generate chairs with convolutional neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Boston: IEEE, 2015. p. 1538–1546. Citado na página 16.
- FARFADE, S. S.; SABERIAN, M. J.; LI, L.-J. Multi-view face detection using deep convolutional neural networks. In: ACM. *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. Shangai, 2015. p. 643–650. Citado na página 16.
- FIRTH, J. R. *A synopsis of linguistic theory, 1930-1955*. Blackwell, 1957. Citado na página 37.
- FUMERO, F.; ALAYÓN, S.; SANCHEZ, J.; SIGUT, J.; GONZALEZ-HERNANDEZ, M. Rim-one: An open retinal image database for optic nerve evaluation. In: IEEE. *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*. Bristol, United Kingdom, 2011. p. 1–6. Citado na página 47.
- GATYS, L. A.; ECKER, A. S.; BETHGE, M. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015. Citado na página 16.

GENG, M.; WANG, Y.; TIAN, Y.; HUANG, T. Cnusvm: Hybrid cnn-uneven svm model for imbalanced visual learning. In: IEEE. *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*. California, USA, 2016. p. 186–193. Citado na página 34.

GINNEKEN, B. van; SETIO, A. A.; JACOBS, C.; CIOMPI, F. Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans. In: *12th International Symposium on Biomedical Imaging (ISBI)*. Brooklyn Bridge, NY: IEEE, 2015. p. 286–289. Citado 2 vezes nas páginas 17 e 18.

GIRSHICK, R.; DONAHUE, J.; DARRELL, T.; MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Columbus, Ohio: IEEE, 2014. p. 580–587. Citado na página 16.

GONÇALVES, M. R.; GUEDES, M. d. M. R.; CHAVES, M. A. P. D.; PEREIRA, C. C. de L.; OTTON, R. Análise dos fatores de risco e epidemiologia em campanha de prevenção da cegueira pelo glaucoma em João Pessoa, Paraíba. *Rev Bras Oftalmol*, SciELO Brasil, v. 72, n. 6, p. 396–9, 2013. Citado na página 46.

HAFEMANN, L. G. An analysis of deep neural networks for texture classification. 2014. Citado 3 vezes nas páginas 17, 18 e 24.

HALEEM, M. S.; HAN, L.; HEMERT, J. v.; FLEMING, A.; PASQUALE, L. R.; SILVA, P. S.; SONG, B. J.; AIELLO, L. P. Regional image features model for automatic classification between normal and glaucoma in fundus and scanning laser ophthalmoscopy (slo) images. *Journal of Medical Systems*, v. 40, n. 6, p. 132, 2016. ISSN 1573-689X. Disponível em: <<http://dx.doi.org/10.1007/s10916-016-0482-9>>. Citado 2 vezes nas páginas 18 e 84.

HATTENHAUER, M. G.; JOHNSON, D. H.; ING, H. H.; HERMAN, D. C.; HODGE, D. O.; YAWN, B. P.; BUTTERFIELD, L. C.; GRAY, D. T. The probability of blindness from open-angle glaucoma. *Ophthalmology*, Elsevier, v. 105, n. 11, p. 2099–2104, 1998. Citado na página 46.

HE, K.; SUN, J. Convolutional neural networks at constrained time cost. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Boston: IEEE, 2015. p. 5353–5360. Citado na página 42.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. Citado 10 vezes nas páginas 14, 15, 16, 18, 40, 41, 42, 43, 49 e 50.

HUBEL, D. H.; WIESEL, T. N. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, Wiley Online Library, v. 195, n. 1, p. 215–243, 1968. Citado na página 28.

JADERBERG, M.; SIMONYAN, K.; VEDALDI, A.; ZISSERMAN, A. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, Springer, v. 116, n. 1, p. 1–20, 2016. Citado na página 16.

- JOHNSON, R.; ZHANG, T. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014. Citado na página 84.
- KIM, Y. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. Citado na página 18.
- KINGMA, D.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado na página 57.
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. Nevada, USA: Curran Associates, Inc., 2012. p. 1097–1105. Citado 7 vezes nas páginas 14, 16, 29, 38, 39, 40 e 51.
- LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, IEEE, v. 86, n. 11, p. 2278–2324, 1998. Citado 5 vezes nas páginas 15, 38, 39, 49 e 51.
- LECUN, Y.; CORTES, C.; BURGES, C. J. The mnist database. *URL* <http://yann.lecun.com/exdb/mnist>, 1998. Citado 2 vezes nas páginas 14 e 38.
- LI, C.; WAND, M. Combining markov random fields and convolutional neural networks for image synthesis. *arXiv preprint arXiv:1601.04589*, 2016. Citado na página 40.
- LIN, T.-Y.; MAIRE, M.; BELONGIE, S.; HAYS, J.; PERONA, P.; RAMANAN, D.; DOLLÁR, P.; ZITNICK, C. L. Microsoft coco: Common objects in context. In: SPRINGER. *European Conference on Computer Vision*. Zurich, 2014. p. 740–755. Citado na página 41.
- MANINIS, K.-K.; PONT-TUSET, J.; ARBELÁEZ, P.; GOOL, L. V. Deep retinal image understanding. In: SPRINGER. *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Athenes, Greece, 2016. p. 140–148. Citado 4 vezes nas páginas 17, 18, 40 e 59.
- MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Citado na página 20.
- MICHAEL, N. Artificial intelligence a guide to intelligent systems. *ISBN*, v. 321204662, p. 1–18, 2005. Citado na página 20.
- MINSKY, M. *Paper, S.: Perceptrons: An introduction to Computational Geometry*. Cambridge: MIT Press, 1969. Citado na página 22.
- MIYAO, H.; MARUYAMA, M. An online handwritten music score recognition system. In: IEEE. *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*. UK, 2004. v. 1, p. 461–464. Citado 2 vezes nas páginas 17 e 18.
- MOOKIAH, M. R. K.; ACHARYA, U. R.; LIM, C. M.; PETZNICK, A.; SURI, J. S. Data mining technique for automated diagnosis of glaucoma using higher order spectra and wavelet energy features. *Knowledge-Based Systems*, Elsevier, v. 33, p. 73–82, 2012. Citado na página 18.

- NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. Haifa, Israel: Omnipress, 2010. p. 807–814. Citado na página 33.
- NAYAK, J.; ACHARYA, R.; BHAT, P. S.; SHETTY, N.; LIM, T.-C. Automated diagnosis of glaucoma using digital fundus images. *Journal of medical systems*, Springer, v. 33, n. 5, p. 337–346, 2009. Citado na página 17.
- NOH, H.; SEO, P. H.; HAN, B. Image question answering using convolutional neural network with dynamic parameter prediction. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA: IEEE, 2016. Citado 2 vezes nas páginas 16 e 18.
- OQUAB, M.; BOTTOU, L.; LAPTEV, I.; SIVIC, J. Learning and transferring mid-level image representations using convolutional neural networks. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Columbus, Ohio: IEEE, 2014. Citado na página 59.
- PARANHOS-JUNIOR, A.; OMI, C.; PRATA-JUNIOR, J. Sociedade brasileira de glaucoma: Iii consenso brasileiro de glaucoma primário de ângulo aberto. *Sao Paulo: BestPoint*, p. 77–96, 2009. Citado na página 46.
- PARKHI, O. M.; VEDALDI, A.; ZISSERMAN, A. Deep face recognition. In: *British Machine Vision Conference*. Swansea, UK: BMVA Press, 2015. v. 1, n. 3, p. 6. Citado na página 40.
- PEREIRA, R. M.; MATOS, C.; DINIZ, J.; JUNIOR, G. B.; ALMEIDA, J. D. D.; SILVA, A. C.; PAIVA, A. C. de. Abordagem deep learning para classificação ao de lesões mamárias. *WIM-XVI Workshop de Informática Médica*, Porto Alegre, 2016. Citado 2 vezes nas páginas 17 e 18.
- PEREIRA, R. M. P.; MAIA, L. B.; SILVA, T. A.; PESSOA, A. C. P.; JUNIOR, G. B. Um estudo sobre diferentes tipos de funções de custo para redes neurais convolucionais. *Jornada de Informática do Maranhão*, v. 6, 2016. Citado na página 24.
- PEREIRA, R. M. P.; MATOS, C. E.; JUNIOR, G. B.; ALMEIDA, J. D. de; PAIVA, A. C. de. A deep approach for handwritten musical symbols recognition. In: *ACM. Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. Teresina, Piauí, 2016. p. 191–194. Citado 3 vezes nas páginas 17, 18 e 84.
- REBELO, A.; CAPELA, G.; CARDOSO, J. S. Optical recognition of music symbols. *International Journal on Document Analysis and Recognition (IJ DAR)*, Springer, v. 13, n. 1, p. 19–31, 2010. Citado na página 17.
- REED, S.; AKATA, Z.; LEE, H.; SCHIELE, B. Learning deep representations of fine-grained visual descriptions. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA: IEEE, 2016. Citado 2 vezes nas páginas 16 e 18.
- RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Cognitive modeling*, v. 5, n. 3, p. 1, 1988. Citado 2 vezes nas páginas 26 e 28.

- SANTOS, C. dos; GATTI, M. Deep convolutional neural networks for sentiment analysis of short texts. In: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, 2014. p. 69–78. Citado na página 18.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Citado 6 vezes nas páginas 14, 15, 40, 41, 49 e 59.
- SOCHER, R.; GANJOO, M.; MANNING, C. D.; NG, A. Zero-shot learning through cross-modal transfer. In: *Advances in neural information processing systems*. Nevada, USA: Citeseer, 2013. p. 935–943. Citado na página 37.
- SPRINGENBERG, J. T.; DOSOVITSKIY, A.; BROX, T.; RIEDMILLER, M. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. Citado na página 52.
- SRIVASTAVA, N.; HINTON, G. E.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014. Citado na página 34.
- SRIVASTAVA, R. K.; GREFF, K.; SCHMIDHUBER, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015. Citado na página 42.
- SU, H.; QI, C. R.; LI, Y.; GUIBAS, L. J. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In: *The IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, 2015. Citado na página 40.
- SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S.; ANGUELOV, D.; ERHAN, D.; VANHOUCHE, V.; RABINOVICH, A. Going deeper with convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Boston: IEEE, 2015. p. 1–9. Citado 3 vezes nas páginas 14, 16 e 40.
- VALERO-MAS, J. J.; CALVO-ZARAGOZA, J.; RICO-JUAN, J. R.; IÑESTA, J. M. An experimental study on rank methods for prototype selection. *Soft Computing*, Springer, p. 1–13, 2016. Citado na página 17.
- ZILLY, J. G.; BUHMANN, J. M.; MAHAPATRA, D. Boosting convolutional filters with entropy sampling for optic cup and disc image segmentation from fundus images. In: SPRINGER. *International Workshop on Machine Learning in Medical Imaging*. Munich, Germany, 2015. p. 136–143. Citado na página 17.