Pontifícia Universidade Católica
do Rio de Janeiro

Roberto Bandeira de Mello Morais da Silva

Identificação da localização subcelular de proteínas por meio de técnicas de Deep Learning

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio.

Orientador: Prof. Eduardo Costa da Silva

Rio de Janeiro
Julho de 2019

Pontifícia Universidade Católica
do Rio de Janeiro

Roberto Bandeira de Mello Morais da Silva

Identificação da localização subcelular de proteínas por meio de técnicas de Deep Learning

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Elétrica do Departamento de Engenharia Elétrica da PUC-Rio. Aprovada pela Comissão Examinadora abaixo.

Prof. Eduardo Costa da Silva
Orientador
Departamento de Engenharia Elétrica – PUC-Rio

Prof. Leonardo Alfredo Forero Mendoza
Universidade do Estado do Rio de Janeiro – UERJ

Prof. Carlos Roberto Hall Barbosa
Programa de Pós-Graduação de Metrologia – PUC-Rio

Prof. Elisabeth Costa Monteiro
Programa de Pós-Graduação de Metrologia – PUC-Rio

Rio de Janeiro, 16 de Julho de 2019

Roberto Bandeira de Mello Morais da Silva

Graduou-se em Engenharia Elétrica, com ênfase em Eletrônica e Computadores, pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio). Durante a graduação trabalhou principalmente com eletrônica analógica em projetos de hardware para sensores magnéticos. No mestrado e em sua profissão de *Data Scientist* envolveu-se com a área de *Machine Learning*.

Ficha Catalográfica

Silva, Roberto Bandeira de Mello Morais da

Identificação da localização subcelular de proteínas por meio de técnicas de Deep Learning / Roberto Bandeira de Mello Morais da Silva; orientador: Eduardo Costa da Silva. – 2019.

v., 81 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica .

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Deep learning 3. Redes neurais 4. Multi-classe 5. Human Protein Atlas 6. Classificação I. Silva, Eduardo Costa da. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica . III. Título.

CDD: 621.3

À minha família, por todo o apoio.
Aos meus amigos, por todos os momentos.
À Pontifícia, por todos os ensinamentos.

Agradecimentos

Primeiramente ao meu orientador Eduardo. O Eduardo esteve comigo desde que fiz sua matéria de Eletrônica Analógica na graduação em 2014. Desde então fui seu orientado de Iniciação Científica, monitor da sua disciplina, estagiário de seu laboratório, orientado de TCC da graduação, até agora finalmente a dissertação de mestrado. Mais do que um simples orientador ou um chefe, o professor Eduardo se tornou um verdadeiro amigo e mentor.

À professora Ana Pavani, que desde o início da minha graduação se fez presente como mentora, oferecendo os mais diversos conselhos. Atribuo a ela a decisão de me involver com o mestrado na PUC-Rio e agradeço imensamente por todos esses momentos em 7 anos e meio de universidade.

À PUC-Rio e seus professores, que me ofereceram a estrutura e o conhecimento necessário para que este trabalho pudesse ser realizado.

Aos meus amigos, que em muitos momentos me distraíram do trabalho, mas que na maioria dos momentos me propuseram o lazer necessário para que o estresse não me consumisse. Cito aqui alguns nomes: Samuel Bastos, Tiago Novaes, Guilherme Berger, Felipe Luiz, João Cury, Maurício Pedro, Rebecca Downey, João Augusto, Robson Nascimento, Ulisses Figueiredo, Ian Vasconcellos, Nathalia Valle, Pedro Antonio, entre vários outros nomes. Ênfase também à minha ex-namorada Isabel por me ajudar na motivação inicial do mestrado e à minha namorada atual Gwen pelo suporte no final.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Por fim, à gestão do Transforme o DAAF do Diretório Acadêmico Adhemar Fonseca, que foi responsável por deixar o clima da Engenharia da PUC-Rio mais agradável no último ano da minha graduação e o primeiro ano do meu mestrado.

Resumo

Silva, Roberto Bandeira de Mello Morais da; Silva, Eduardo Costa da. **Identificação da localização subcelular de proteínas por meio de técnicas de Deep Learning**. Rio de Janeiro, 2019. 81p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

As proteínas são macromoléculas biológicas compostas por cadeias de aminoácidos, presentes em praticamente todos os processos celulares, sendo essenciais para o correto funcionamento do organismo humano. Existem diversos estudos em torno do proteoma humano a fim de se identificar quais são as funções de cada proteína nas diferentes células, tecidos e órgãos do corpo humano. A classificação destas proteínas em diferentes formas, como por exemplo a localização subcelular, é importante para diversas aplicações da biomedicina. Com o avanço das tecnologias para obtenção de imagens das proteínas, tem-se que hoje estas são geradas em grande volume e mais rapidamente do que é possível classificá-las manualmente, o que torna importante o desenvolvimento de um classificador automático capaz de realizar esta classificação de maneira eficaz. Dessa forma, esta dissertação buscou desenvolver algoritmos capazes de realizar a classificação automática de padrões mistos de localização subcelular de proteínas, por meio do uso de técnicas de *Deep Learning*. Inicialmente, fez-se uma revisão da literatura em torno de redes neurais, *Deep Learning* e SVMs, e utilizou-se o banco de dados, publicamente disponíve, de imagens de células do *Human Protein Atlas*, para treinamento dos algoritmos de aprendizagem supervisionada. Diversos modelos foram desenvolvidos e avaliados, visando identificar aquele com melhor desempenho na tarefa de classificação. Ao longo do trabalho foram desenvolvidas redes neurais artificiais convolucionais de topologia LeNet, ResNet e um modelo híbrido ResNet-SVM, tendo sido treinadas ao todo 81 redes neurais diferentes, a fim de se identificar o melhor conjunto de hiper-parâmetros. As análises efetuadas permitiram concluir que a rede de melhor desempenho foi uma variante da topologia ResNet, que obteve em suas métricas de desempenho uma acurácia de 0,94 e uma pontuação F1 de 0,44 ao se avaliar o comportamento da rede frente ao conjunto de teste. Os resultados obtidos pela diferentes topologias analisadas foram detalhadamente avaliados e, com base nos resultados alcançados, foram sugeridos trabalhos futuros baseados em possíveis melhorias para as redes de melhor desempenho.

Palavras-chave

Deep Learning; Redes Neurais; Multi-classe; Human Protein Atlas; Classificação.

Abstract

Silva, Roberto Bandeira de Mello Morais da; Silva, Eduardo Costa da (Advisor). **Identification of protein subcellular localization by Deep Learning techniques**. Rio de Janeiro, 2019. 81p. Dissertação de mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Proteins are biological macromolecules composed of aminoacid chains, part of practically all cellular processes, being essential for the correct functioning of the human organism. There are many studies around the human protein aiming to identify the proteins' functions in different cells, tissues and organs in the human body. The protein classification in many forms, such as the subcellular localization, is important for many biomedical applications. With the advance of protein image obtention technology, today these images are generated in large scale and faster than it is possible to manually classify them, which makes crucial the development of a system capable of classifying these images automatically and accurately. In that matter, this dissertation aimed to develop algorithms capable of automatically classifying proteins in mixed patterns of subcellular localization with the use of Deep Learning techniques. Initially, a literature review on neural networks, Deep Learning and SVMs, and a publicly available image database from the Human Protein Atlas was used to train the supervised learning algorithms. Many models were developed seeking the best performance in the classification task. Throughout this work, convolutional artificial neural networks of topologies LeNet, ResNet and a hybrid ResNet-SVM model were developed, with a total of 81 different neural networks trained, aiming to identify the best hyper-parameters. The analysis allowed the conclusion that the network with best performance was a ResNet variation, which obtained in its performance metrics an accuracy of 0.94 and an F1 score of 0.44 when evaluated against the test data. The obtained results of these topologies were detailedly evaluated and, based on the measured results, future studies were suggested based on possible improvements for the neural networks that had the best performances.

Keywords

Deep Learning; Neural Networks; Multi-label; Human Protein Atlas; Classification.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

SVM – *Support Vector Machine*
MLP – *Multi-Layer Perceptron*
RNA – Rede Neural Artificial
ConvNets – Redes Neurais Convolucionais
ReLU – Rectifying Linear Unit
TP – *True Positive*
TN – *True Negative*
FP – *False Positive*
FN – *False Negative*

1
Introdução

1.1
Motivação

Proteínas são responsáveis pela execução de diversas funções na célula humana que juntas tornam possíveis a vida. Estas macromoléculas biológicas constituem-se por uma ou mais cadeias de aminoácidos e estão presentes em praticamente todos os processos celulares, sendo um nutriente que o corpo precisa para crescer e se sustentar [1; 2].

Devido a grande importância das proteínas para o ser humano, existem diversos estudos sobre estas moléculas essenciais e suas funções no corpo humano e de outros seres vivos. Em particular, um dos problemas mais estudados é o da classificação de proteínas por meio do processamento de imagens. Há décadas diversos bancos de dados foram criados pela comunidade científica com este objetivo [3; 4; 5].

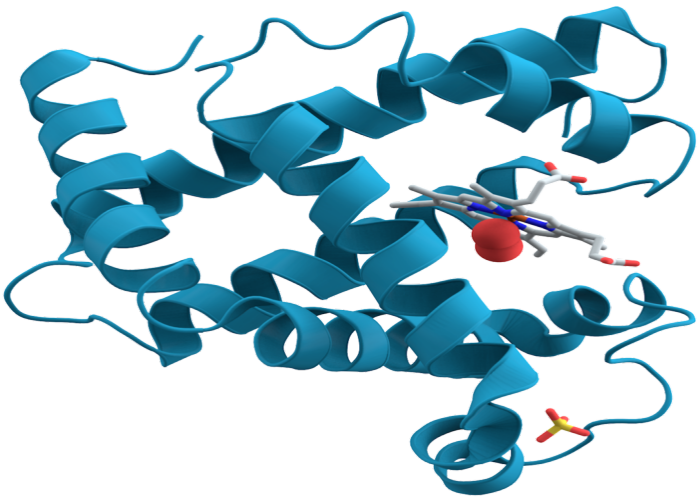

Figura 1.1: Estrutura Molecular de uma Proteína

O *Human Protein Atlas* é um programa sediado na Suécia, fundado em 2013, cujo objetivo é mapear todas as proteínas humanas em células, tecidos e órgãos. O Programa integra diversas tecnologias como imagens baseadas

em anticorpos, espectrometria de massa, transcriptomas e sistemas biológicos [4; 6; 7].

Dentro do espectro de imagens contidas no *Human Protein Atlas*, existem as que permitem a visualização de proteínas dentro das células humanas. Estas costumam ser usadas para pesquisas biomédicas, e talvez possuam a chave para o próximo grande avanço da medicina. Graças às tecnologias modernas de microscopia, é possível gerar tais imagens mais rapidamente do que é possível analisá-las manualmente. Logo, cria-se a necessidade de automatizar o processo de análise destas imagens [7].

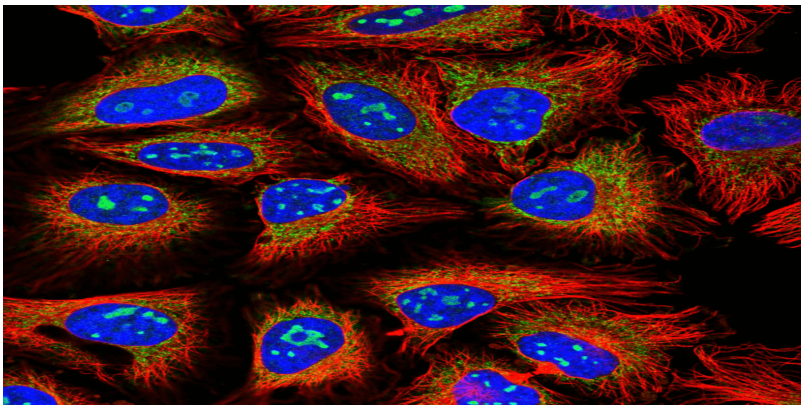

Figura 1.2: Exemplo de imagem do banco de dados do *Human Protein Atlas*

Historicamente, o problema de classificação de imagens de proteínas tem se limitado a padrões únicos em um ou mais tipos de células. Contudo, buscando entender melhor a complexidade de células humanas, os modelos computacionais precisarão ser capazes de classificar múltiplos padrões através de uma gama de diferentes tipos de células humanas [8; 9; 10].

Entre as técnicas mais usadas na bioinformática e classificação de proteínas de forma geral estão as *Support Vector Machines* (SVM) e as Redes Neurais. Em particular com o avanço da tecnologia computacional, hoje consegue-se implementar redes neurais (em particular as redes neurais convolucionais) muito mais densas, que conseguem lidar com tarefas associadas ao processamento de imagens com uma maestria muito maior do que no passado [8; 9; 10]. Este trabalho pretende explorar essas técnicas e propor novas configurações destes algoritmos, buscando uma maior acurácia na classificação das imagens do banco de dados do *Human Protein Atlas* [11].

1.1.1 Inspiração

O trabalho feito nesta dissertação se inspirou no problema originalmente apresentado pela plataforma de competição de ciência dos dados *Kaggle*. O problema é apresentado na competição intitulada "*Human Protein Atlas Image Classification*"e será também abordado aqui, explicado mais detalhadamente no próximo capítulo.

A base de dados, a divisão dos conjuntos de treinamento e teste utilizada, e as classes são todas baseadas nos mesmos selecionados pela organização da competição. Diversas das análises feitas posteriormente neste trabalho, principalmente na análise exploratória da base de dados, são inspiradas em análises vistas por competidores da plataforma *Kaggle* nesta e em outras competições.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver um modelo computacional que seja capaz de identificar de forma automática em imagens de uma dada proteína em células, a localização subcelular, que pode ser de padrões múltiplos, com alto desempenho.

1.2.2 Objetivos Específicos

Além do objetivo principal, este trabalho também busca:

- comparar técnicas de classificação de imagens biomédicas;
- identificar as melhores métricas no que diz respeito a classificação de padrões mistos de proteínas;
- comparar múltiplos modelos de classificação de imagens afim de obter a melhor performance possível.

1.3 Organização da Dissertação

Esta dissertação está dividida em 5 capítulos, que estão organizados da seguinte forma:

– O capítulo 2 apresenta a fundamentação bibliográfica concernente à classificação de proteínas, além de descrever a base de dados utilizada neste trabalho. Também discute a base teórica necessária para o desenvolvimento dos algoritmos utilizados neste trabalho. Em particular serão discutidas as técnicas de *Support Vector Machines*, Redes Neurais e Redes Neurais Profundas.

– O capítulo 3 realiza uma análise exploratória da base de dados, identifica e analisa possíveis métricas de desempenho, e apresenta as topologias desenvolvidas ao longo deste trabalho, baseadas em redes neurais convolucionais, além de apresentar o processo de iteração dos hiper-parâmetros.

– O capítulo 4 apresenta e discute os principais resultados obtidos na metodologia do capítulo 3, realizando uma análise detalhada de como os aprendizados destes resultados podem ser usados para desenvolvimentos nos modelos.

– O capítulo 5 apresenta as conclusões dos estudos realizados neste trabalho, comparando os resultados obtidos com o estado da arte, resultando, por fim, em sugestões para trabalhos futuros.

2
Fundamentação

2.1
Identificação da Localização Subcelular de Proteínas

Um problema importante da bioinformática é a bilogia estrutural, ou seja, a representação das estruturas de diversas macromoléculas biológicas. Em particular, a informação da estrutura de proteínas é um importante agente no combate a diversas doenças, já que muitas delas são causadas por mal funcionamento de proteínas envolvidas em várias funções das células humanas.

Nas últimas décadas, grandes bancos de dados foram criados para o armazenamento e exploração de dados biológicos, devido a decodificação das moléculas de DNA e proteínas. Com a contribuição de técnicas de análise de dados, principalmente de imagens, como nas técnicas de *Deep Learning* e demais métodos de Inteligência Artificial, o tópico é abordado de forma computacional, fornecendo soluções mais flexíveis e automáticas, necessárias para o escalonamento das análises e obtenção de conhecimento. [12; 11]

Em particular, este estudo foca na identificação da localização subcelular de proteínas, levando em conta que esta informação fornece um contexto para determinação da sequência, estrutura e função de uma proteína. Por exemplo, duas proteínas que hipoteticamente possuem estrutura e função similares podem estar localizadas em diferentes regiões da célula e, portanto, estarem envolvidas em processos celulares diferentes. [13; 14; 15]

As imagens do banco de dados explorado contem em um de seus filtros a imagem de um tipo de proteína em específico. Através deste filtro e de outros dois filtros citados mais a frente, este trabalho busca identificar qual a localização subcelular no qual as proteínas da imagem se localizam. Torna-se então um problema de classificação multi-classes.

O mais recente e eficaz estudo associado a classificação de proteínas em sua localização subcelular utilizou algoritmos de *Deep Learning* junto à base de dados *Cell Atlas of the Human Protein Atlas*, para desenvolver um classificador automático de múltiplos padrões. Esse trabalho, publicado na *Nature Biotechnology*, utilizou-se de técnicas de *transfer learning* para treinar uma rede neural na classificação. [16]

2.1.1
A Base de Dados

A base de dados utilizada nesta dissertação é pertencente ao *Human Protein Atlas* e é de livre acesso a qualquer pessoa que deseje explorar o proteoma humano.

Esse banco de dados em particular é composto por 31072 imagens de conjunto de treinamento e 11702 de conjunto de teste. Cada uma dessas imagens tem associada a si uma ou mais classes, subdivididas em 28 classes [17]:

- Nucleoplasma,
- Membrana Nuclear,
- Nucléolos,
- Núcleo Fibrilar,
- Manchas Nucleares,
- Corpos Nucleares,
- Retículo Endoplasmático,
- Aparelho de Golgi,
- Peroxissomas,
- Endossomos,
- Lisossomos,
- Filamentos intermediários,
- Filamentos de actina,
- Locais de adesão focal,
- Microtúbulos,
- Extremidades de microtúbulos,
- Ponte citocinética,
- Fuso Mitótico,
- Centro de organização de microtúbulos,
- Centrossoma,
- Gotículas lipídicas,
- Membrana Plasmática,
- Junções celulares,
- Mitocôndria,

– Agressoma,

– Citosol,

– Corpos citoplasmáticos,

– Hastes e anéis.

A figura 2.1 ilustra algumas das principais organelas celulares, tendo algumas das classes citadas entre elas.

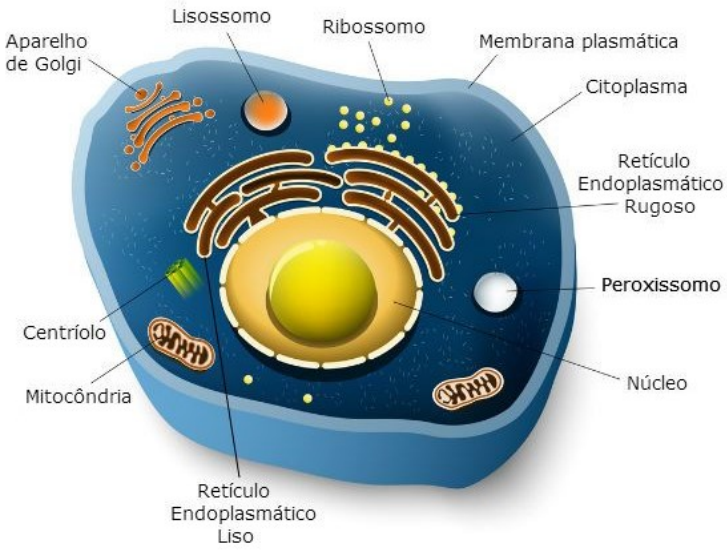

Figura 2.1: Organelas celulares

Por sua vez, cada imagem da base de dados é dividida em quatro sub-arquivos, de tamanho 512 x 512 pixels:

– uma imagem com filtro verde para localizar a estrutura da proteína de interesse;

– um filtro azul para localização de núcleo;

– um filtro vermelho para microtúbulos; e

– um filtro amarelo para o retículo endoplasmático.

Observando por exemplo quatro imagens randômicas da base de dados, todas pertencentes a classe Endossomos, é possível comparar e avaliar o efeito dos diferentes filtros citados, conforme apresentado na figura 2.2.

Por meio da figura 2.2 nota-se que existe grande diferença na intensidade das cores no filtro verde, sendo quase impossível de se ver algo a olho nu na imagem (d). Também, observando-se a figura 2.2(b) nota-se que existe grande diferença morfológica no canal vermelho, o que pode estar associado ao fato das imagens corresponderem a células de tipos diferentes.

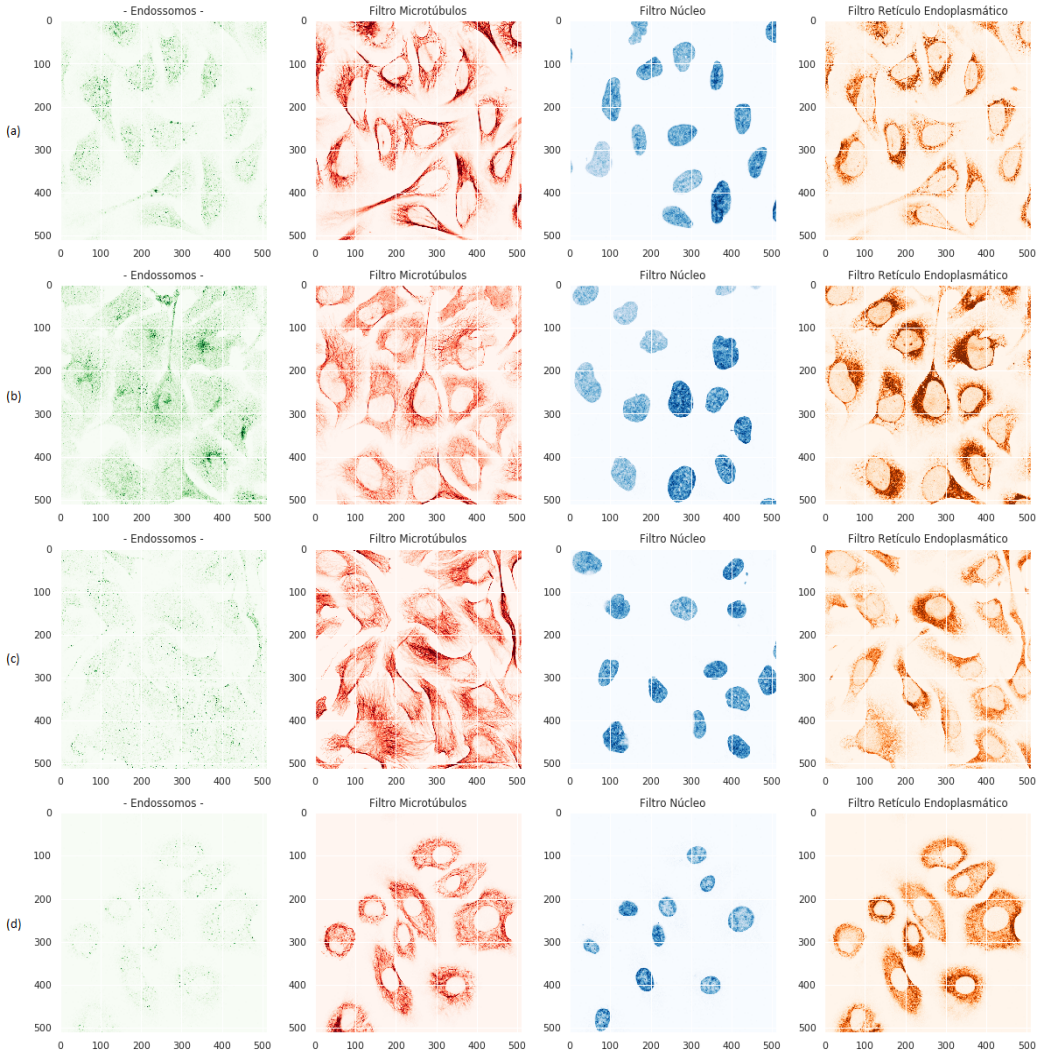

Figura 2.2: Quatro imagens classificadas como Endossomos com seus respectivos filtros

Por sua vez, para efeitos de comparação entre classes diferentes, as figuras 2.3 e 2.4 apresentam as imagens de mais duas classes: Citosol e Peroxissomas, respectivamente. Comparando-se as diferentes imagens, verifica-se que, para um não especialista, ainda é difícil discernir a olho nu as diferentes classes.

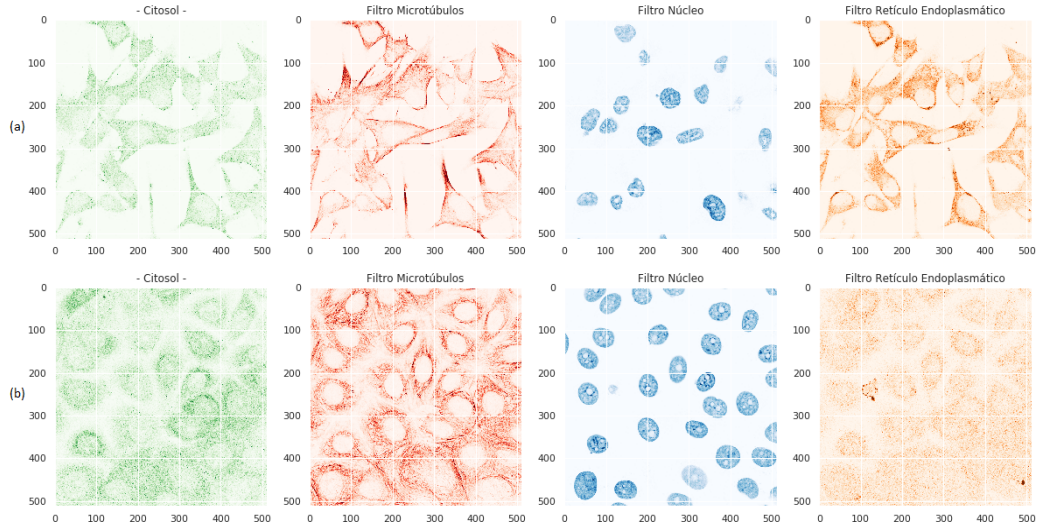

Figura 2.3: Duas imagens classificadas como Citosol com seus respectivos filtros

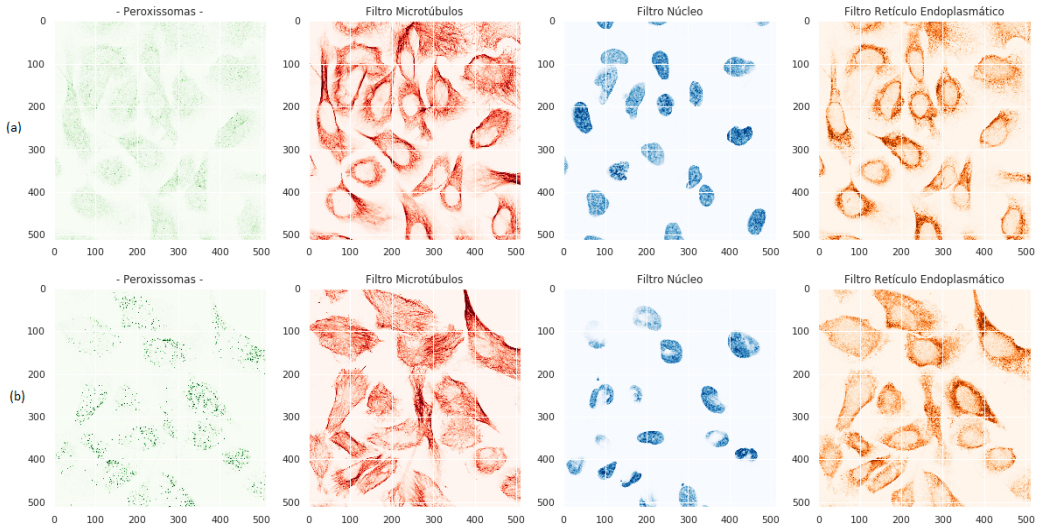

Figura 2.4: Duas imagens classificadas como Peroxissomas com seus respectivos filtros

2.2 Support Vector Machine (SVM)

Support Vector Machines são uma classe de modelo estatístico inicialmente desenvolvido por Vladimir Vapnik, no final de década de 70. O problema que incentivou o desenvolvimento das SVMs se baseia na ideia de que, dada uma tarefa de aprendizado, com uma quantidade finita de dados de treinamento, a melhor generalização ocorrerá com o equilíbrio correto entre a acurácia, atrelada àquele conjunto de treinamento, e a habilidade do algoritmo em

aprender um conjunto de treinamento sem erros, a "capacidade"do algoritmo [18].

Para melhor compreender o conceito de uma SVM, é possível fazer uma analogia com botânica: um SVM com muita capacidade é como um botânico com memória fotográfica que, ao ser apresentado a uma árvore nova, conclui que não é uma árvore pois tem um número de folhas diferente de qualquer outra que viu antes; enquanto que um SVM com pouca capacidade é como o irmão preguiçoso do botânico, que declara que, se é verde, logo deve ser uma árvore. Nenhum dos dois é capaz de apresentar um desempenho satisfatório em relação à generalização [19].

As SVMs evoluíram consideravelmente desde sua criação, tendo se tornado um dos mais flexíveis e efetivos algoritmos de aprendizado de máquina disponíveis. Em resumo, elas são algoritmos de aprendizado supervisionado que podem ser usados na resolução de problemas de classificação e regressão.

O processo de resolução consiste no descobrimento de hiperplanos linearmente separáveis, que implementam limites de decisão capazes de separar diferentes classes. Caso não seja possível definir tais hiperplanos para separação de determinadas classes da base de dados, o algoritmo usa um mapeamento não-linear para transformar os dados do conjunto de treinamento em dados com maior dimensão, em seguida procura novamente por um hiperplano adequado. As SVMs buscam estes hiperplanos por meio do emprego de vetores de suporte e margens [20]. Esta técnica é conhecida por ter alta acurácia e ser comparativamente menos afetada pelo problema de *overfitting*. Existem diversas aplicações nas quais SVMs foram usadas com sucesso, como: reconhecimento de dígitos escritos manualmente, classificação de textos, entre outras [21; 22].

2.2.1 SVMs Lineares

2.2.1.1 Conjunto de Dados Linearmente Separáveis

Inicialmente, assume-se um conjunto de treinamento cujos dados pertencem a duas classes e são linearmente separáveis. Seja o i-ésimo ponto representado por $\{X_i, y_i\}$, onde X_i representa o vetor de características e y_i a classe associada a este vetor, podendo ter valor +1 ou -1.

Na figura 2.5 é possível ver um exemplo disso em uma representação bidimensional. Tem-se a representação desta situação ao se considerar por exemplo as bolas azuis como sendo vetores cuja classe tem valor -1 e as

vermelhas +1. No caso apresentado, nota-se que é possível traçar infinitas retas que separam as bolas de cor azul das bolas de cor vermelha. Consequentemente, é possível afirmar que os dados são linearmente separáveis.

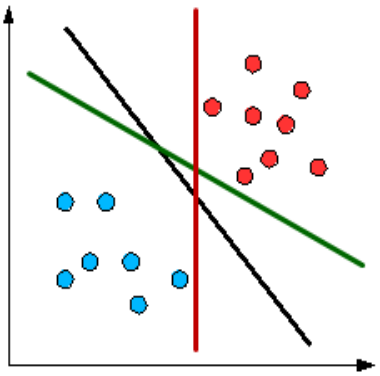

Figura 2.5: Exemplos de dados linearmente separáveis

Extrapolando esta visão para um espaço n-dimensional, a linha da imagem acima torna-se um hiperplano de dimensão n-1. Matematicamente, pode-se escrever o hiperplano como uma combinação linear das dimensões igualadas a 0:

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = 0 \tag{2-1}$$

onde o valor θ_0 é também chamado de *bias*.

No caso linearmente separável, este hiperplano atua como um separador das duas classes. Como indicado na imagem bidimensional, os pontos acima do hiperplano pertencerão a uma classe e os abaixo dele pertencerão à outra.

Após definir a dimensão do hiperplano que separa as classes, o próximo passo é definir os pesos θ_j para $0 < \mathrm{j} < \mathrm{n}$ de forma a encontrar o hiperplano ótimo de separação, tal que este seja o mais distante possível das observações de ambos os lados. A operação do algoritmo de SVM se baseia em achar o hiperplano com maior distância mínima para os pontos do conjunto de treinamento, a margem máxima. Os hiperplanos paralelos ao hiperplano ótimo, que definem o limite de margem, passam por pontos do conjunto de treinamento, os *support vectors*, os pontos mais essenciais do conjunto de treinamento.

Dessa forma, verifica-se que se passa a ter um problema de otimização convexa. Denotando o vetor de pesos como Θ, de [23] tem-se que o tamanho da margem máxima é $\frac{2}{||\Theta||}$. Portanto, de modo a se maximizar a margem, minimiza-se o módulo do vetor de pesos, sujeito à

$$y_i(\theta_0 + \theta_1 x_{1i} + \theta_2 x_{2i} + ... + \theta_n x_{ni}) \geq 1 \quad (2\text{-}2)$$

Uma observação importante é que a complexidade do algoritmo dependerá unicamente do número de *support vectors* e não da dimensão do espaço do vetor de características, razão pela qual este algoritmo tem menos tendência de *overfitting*.

2.2.1.2 Conjunto de Dados Não Linearmente Separáveis

Apesar da técnica descrita em 2.2.1.1 ser eficaz, ela não consegue solucionar problemas onde as classes não sejam linearmente separáveis. É possível evidenciar essa dificuldade por meio da inspeção da imagem bidimensional apresentada na figura 2.6.

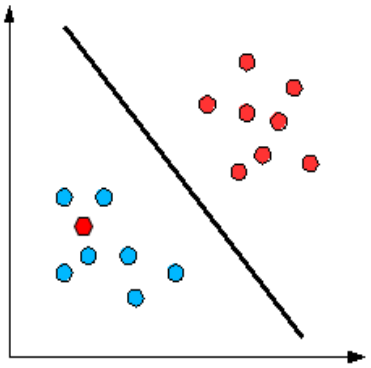

Figura 2.6: Exemplos de dados não linearmente separáveis

Na figura 2.6, é possível identificar que não existe uma reta (ou hiperplano para o caso n-dimensional) que separe as duas classes. De [19; 23], tem-se o resultado de que o problema de otimização convexa acima não possui solução.

2.2.2 SVMs Não Lineares

Visando solucionar este problema [24] propôs a utilização da técnica de aumento da dimensionalidade, também utilizada em [25].

Assim, o primeiro passo para solução do problema consiste na utilização de um mapa não-linear empregado no intuito de aumentar a dimensionalidade dos dados do conjunto de treinamento. Na sequência, encontra-se um hiperplano que separe linearmente os dados neste novo espaço de dimensionalidade

maior que n. Este procedimento gera uma hiper-superfície não-linear que separa os dados do conjunto de treinamento no espaço original.

Utilizando-se de funções *kernel*, como funções de base radial ou sigmoides, no cálculo de produtos internos que surgem do aumento da dimensionalidade, é possível então aplicar o algoritmo SVM para conjuntos de dados não linearmente separáveis. Em [23] os autores empregam tal artifício em diversas aplicações, tais como classificação de textos e detecção de faces. A figura 2.7 mostra uma interpretação geométrica do resultado obtido por [23] ao utilizar a técnica não linear para detecção de faces, e o separador não-linear gerado pelo algoritmo.

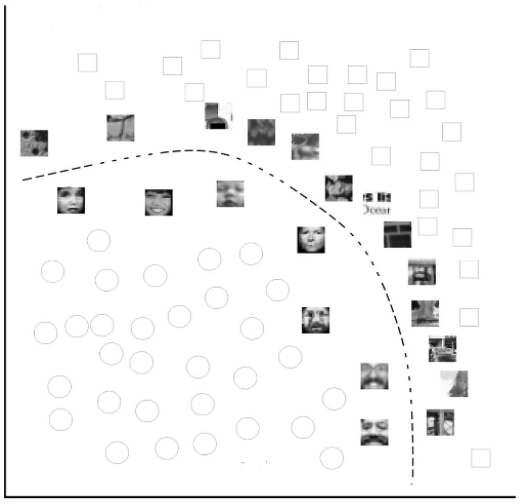

Figura 2.7: Interpretação geométrica de um resultado de SVM não linear

2.2.3 SVM de Múltiplas Classes

Há duas formas principais de abordar o problema quando existem múltiplas classes a serem classificadas nos dados:

1. **One-Versus-All**: Sendo o número de classes C $>$ 2, então C SVMs diferentes de 2 classes são treinados, onde cada classe é comparada com o resto e combinada. Uma nova observação é classificada de acordo com o classificador de maior valor.

2. **One-Versus-One**: Todos as combinações de classes C, dois a dois, são treinadas, e a classificação é decidida como a classe que ganha a maioria dos casos. Este método é mais eficaz, porém costuma ter alta demanda computacional dependendo do tamanho de C [20].

2.3 Redes Neurais

Os primeiros trabalhos relacionados à Redes Neurais Artificiais datam da década de 40. Tendo como base os conhecimentos e premissas à respeito do cérebro humano, neurônios e sinapses, o neurofisiologista Warren McCulloch e o matemático Walter Pitts propuseram um modelo matemático que propunha imitar o comportamento de um neurônio humano [26].

Desde então, a teoria de redes neurais artificiais avançou muito até chegar no que hoje é conhecido como *Deep Learning*, redes neurais com imensa capacidade de aprendizado de padrões, que se tornaram possíveis devido ao avanço das tecnologias de hardware e a grande quantidade de dados hoje disponível para pesquisa e desenvolvimento [27].

2.3.1 Neurônio

2.3.1.1 Modelo McCulloch e Pitts

O modelo de neurônio proposto em [26] pode ser observado de forma simplificada na figura 2.8.

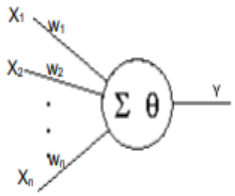

Figura 2.8: Modelo de neurônio de McCullogh e Pitts

O modelo consiste em n valores binários de entrada $x_1, x_2, ...x_n$, inspirados nos dendritos de um neurônio, junto a um terminal de saída y, que seria o equivalente a um axônio. Para efeitos de comparação, a figura 2.9 mostra a representação de um neurônio.

Para ativar as sinapses, as entradas do modelo têm pesos pré-fixados $w_1, w_2, ..., w_n$, com valor igual a -1 ou +1 . O neurônio deste modelo também tem um valor *threshold* θ, que determina se o neurônio foi ou não ativado. Matematicamente temos então:

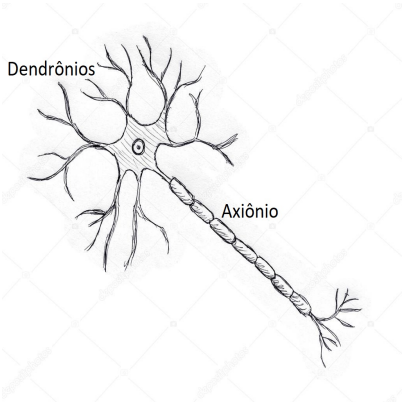

Figura 2.9: Desenho de neurônio

$$\sum_{i=1}^{n} x_i w_i \geq \theta \tag{2-3}$$

Este modelo apresentava duas limitações principais:

– consegue implementar apenas funções linearmente separáveis; e
– possui pesos fixos, sem haver portanto aprendizagem.

2.3.1.2 Perceptron

No final da década de 50 foi proposto o modelo *Perceptron* para mimetizar o comportamento de um neurônio. [28] Com certa similaridade ao modelo de McCullogh e Pitts, também é possível criar um diagrama que lembra o comportamento de um neurônio, conforme indicado na figura 2.10.

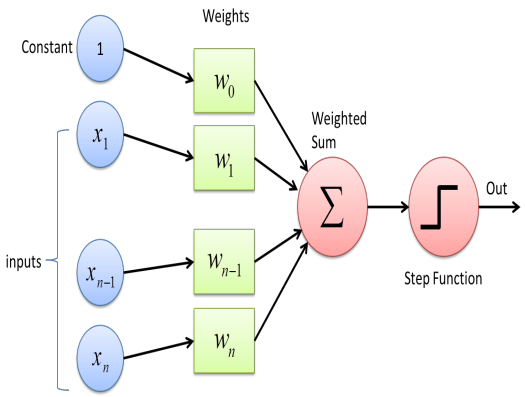

Figura 2.10: Diagrama do *Perceptron*

Matematicamente tem-se que:

$$net = w_0 + \sum_{i=1}^{n} x_i w_i \tag{2-4}$$

$$Out = \begin{cases} 1 \; se \; net \geq 0 \\ \\ -1 \; se \; net < 0 \end{cases} \tag{2-5}$$

Este modelo trouxe uma importante diferença em relação ao anterior. Já existia junto a ele o conceito de treinamento da rede para ajuste dos valores dos pesos. Ou seja, este modelo tinha efetivamente a capacidade de aprender.

De forma simples, o *Perceptron* pode ser usado iterando em todos os padrões e utilizando-se dos erros da previsão de *output*, para ajuste dos pesos sinápticos de acordo com uma taxa de aprendizado. Assim, este modelo era capaz de representar de forma eficaz diversos problemas.

Porém, como posteriormente demonstrado em [29], este modelo também só era capaz de representar funções linearmente separáveis. Entretanto, este mesmo trabalho também demonstrou que, ao utilizar múltiplas camadas de *Perceptrons*, era possível solucionar problema não lineares, contanto que as funções de ativação (degrau no modelo representado em 2.10) fosse não-linear.

2.3.2 Multi-Layer Perceptron

Apesar da descoberta de [29] a respeito do uso de múltiplas camadas possibilitar a solução de problemas linearmente separáveis, as redes neurais artificiais só vieram a ganhar mais popularidade na década de 80.

Até então os métodos de treinamento de redes de múltiplas camadas eram ineficazes em muitos casos e traziam poucos resultados. Em 1986, [30] apresentou na revista da *Nature* o método de treinamento e aprendizado que revolucionaria as redes neurais: o algoritmo de *Back-propagation*.

2.3.3 Arquitetura e Topologia de uma Rede Neural Artificial

A arquitetura de uma rede neural artificial é composta de uma série de neurônios. A maneira que estes neurônios estão organizados e conectados entre si caracteriza diferentes usos e objetivos para uma rede.

Em particular, quando se busca aprendizado supervisionado, a arquitetura tradicional de Rede Neural Artificial é a de *feed forward*, também conhecida como *Multi-Layer Perceptron*, a MLP.

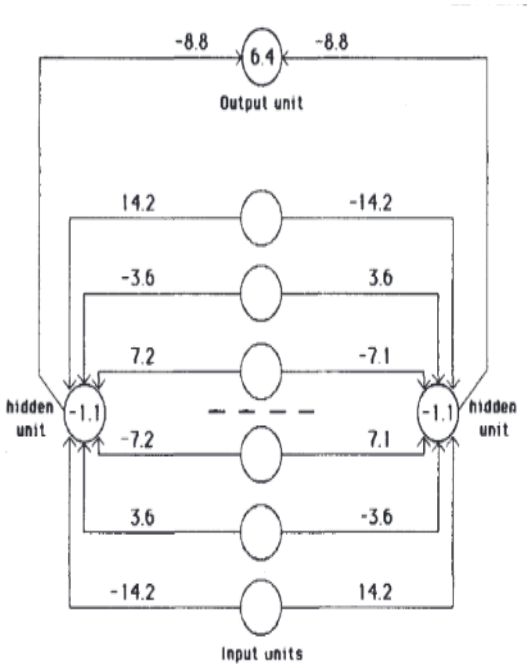

Figura 2.11: Imagem da rede neural do artigo original de Rumelhart

A MLP é uma rede que apresenta seus neurônios organizados em diversas camadas, cujas entradas e saídas são conectadas entre si totalmente ou parcialmente, sem a ocorrência de realimentação. Os valores de entrada são processados por esta sequência de camadas até que se obtenham valores de saída.

É comum dividir suas camadas da seguinte forma:

- **Camada de Entrada**: são apenas os terminais de entrada da rede, não sendo ainda uma camada de neurônios propriamente dita;
- **Camada Escondida**: Uma ou mais camadas intermediárias de neurônios conectados entre si. Esta camada é chamada escondida pois os valores de saída destes neurônios intermediários normalmente não são vistos pelo usuário do algoritmo;
- **Camada de Saída**: Camada de neurônios que faz o último processamento *feed forward*, sendo responsável por entregar os valores de saída da rede.

A figura 2.12 apresenta a topologia de uma rede neural artificial dividida nas camadas mencionadas. Em particular, a rede representada nesta figura é composta por quatro entradas, uma camada escondida com três neurônios e uma camada de saída com dois neurônios.

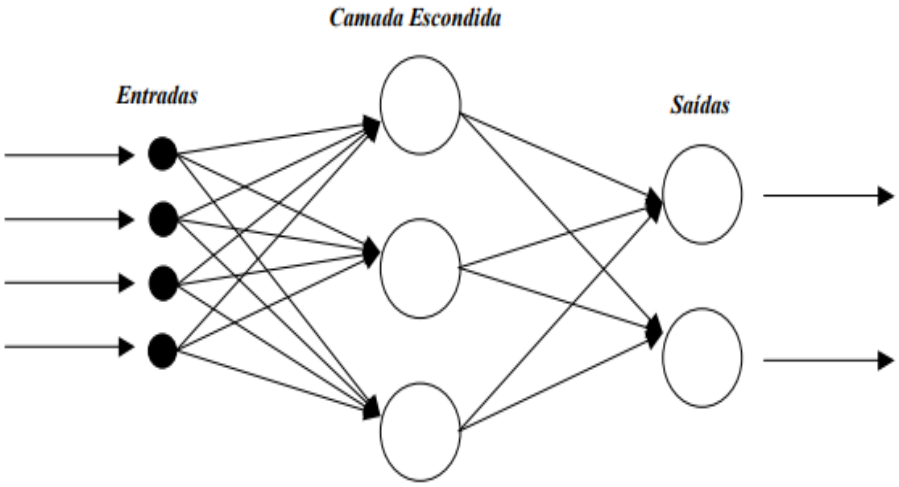

Figura 2.12: Exemplo ilustrativo de rede neural artificial

2.3.4 Aprendizagem

As redes neurais têm a capacidade de aprender padrões através de conjuntos de amostras. Este aprendizado é diretamente representado pelos valores dos pesos sinápticos dos neurônios de suas camadas. Após o treinamento de uma rede, estes pesos passam a ter valores específicos e a rede passa a ser capaz de representar aquele conjunto de amostras.

Em particular, em um problema onde há um conjunto de valores de entrada já mapeado para um conjunto de valores de saída, como no problema abordado nesta dissertação, utiliza-se aprendizado supervisionado.

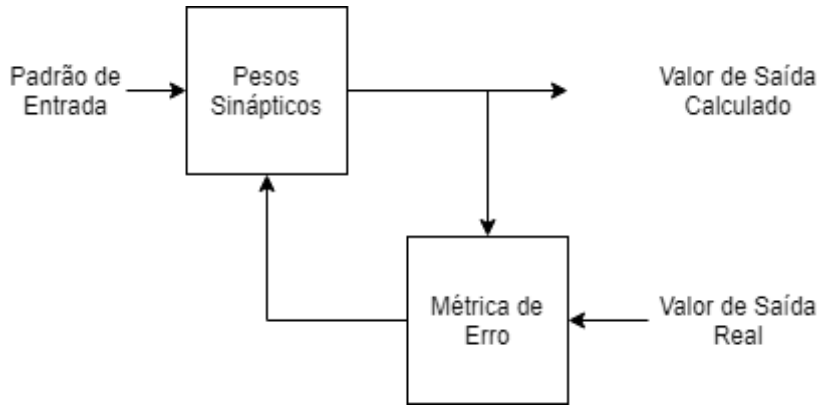

Figura 2.13: Aprendizado Supervisionado

No aprendizado supervisionado, o algoritmo recebe uma série de pares entrada-saída e busca minimizar alguma métrica de erro entre o valor real desses pares e o calculado pelo algoritmo. A figura 2.13 ilustra o processo.

2.3.4.1 Treinamento

Existem diversas formas de realizar o treinamento de uma RNA. A maioria deles consiste na divisão dos dados em dois conjuntos distintos: treinamento e teste.

Os dados do conjunto de treinamento são usados para o aprendizado da rede. Os pares entrada-saída são alimentados para cálculo do valor de saída pela rede, seguido da comparação com o valor real, cálculo da métrica de erro e finalmente recálculo dos pesos sinápticos.

Este processo é repetido até que se encontre um valor de erro considerado aceitável ou até que uma quantidade muito alta de iterações tenha sido feita.

É também comum, durante o treinamento, realizar alguma técnica que busque impedir a rede de perder generalização, isto é, que aconteça *overfitting* com os dados de treinamento. Para tal, normalmente separa-se dentro do conjunto de treinamento um subconjunto,denominado conjunto de validação.

Terminado o treinamento, o conjunto de testes é utilizado de forma se avaliar o desempenho da RNA com dados de entrada que esta não tenha visto durante o treinamento. Este conjunto é utilizado, por exemplo, para gerar métricas de desempenho a respeito da rede treinada.

2.3.4.2 Back-Propagation

O algoritmo de *Back-Propagation* é um método de treinamento de RNAs com aprendizado supervisionado. A forma de atualização dos pesos sinápticos se dá por meio de retro-propagação dos erros, sendo esta a razão do nome do algoritmo.

Através de um algoritmo de gradiente descendente, calcula-se um valor de ajuste dos neurônios baseado na primeira derivada do erro em relação aos pesos sinápticos.

Considerando um neurônio j durante seu treinamento, y_j o valor esperado de saída do neurônio, e $\hat{y}_j$ o valor calculado pelo neurônio, o erro quadrático desse neurônio será:

$$E_j = \frac{1}{2}(\hat{y}_j - y_j)^2 \tag{2-6}$$

então de [31], a atualização do peso sináptico i do neurônio j se dá por

$$\Delta w_{ij} = -\eta \frac{\delta E}{\delta w_{ij}} \tag{2-7}$$

onde o termo η é a taxa de aprendizado, um hiper-parâmetro da rede a ser ajustado posteriormente.

O algoritmo de *Back-Propagation* se tornou a base de técnicas de aprendizado supervisionado de RNAs, sendo que a maioria dos algoritmos de aprendizado atuais se baseiam nele.

2.3.5 *Deep Learning*

2.3.5.1 Redes Neurais Profundas

Nos anos 90, as redes neurais ainda eram deixadas de lado por grande parte da comunidade de *Machine Learning* e quase que completamente ignoradas pela comunidade de visão computacional.

Acreditava-se que era impossível um aprendizado multi-estágio com pouco conhecimento. Um detalhe em especial era o senso comum de que o algoritmo de gradiente descendente ficaria preso em um ótimo local, sendo incapaz de achar os parâmetros ideais para os pesos da rede [32].

O interesse em torno de redes profundas foi revigorado apenas em 2006, quando pesquisadores do *Canadian Institute for Advanced Research* introduziram uma nova e efetiva forma de treinar uma rede neural profunda a partir de pré-treinamentos de cada camada, utilizando o processo de aprendizado não-supervisionado para máquinas de Boltzmann restritas. [33] Este método teve resultados excelentes para reconhecimento de dígitos e detecção de pedestres.

Com a evolução do *hardware* e o advento das GPUs (*Graphical Processing Units*), passou a ser possível treinar redes neurais em uma velocidade muito maior. Em 2009, foi apresentada a primeira grande aplicação de redes profundas submetidas a este tipo de aprendizado, as quais foram aplicadas a reconhecimento de fala. [34] Com o excelente resultado obtido por este trabalho, muitas pesquisas seguiram esta linha se aproveitando do novo poderio tecnológico disponível para treinar redes profundas.

Contanto, o maior sucesso de *Deep Learning* viria posteriormente, com a popularização das redes neurais convolucionais.

2.3.5.2 Redes Neurais Convolucionais

Redes Neurais Convolucionais, conhecidas como *ConvNets*, são redes neurais artificiais projetadas para processar dados na forma de vetores multi-dimensionais. Um exemplo desse tipo de dado seria uma imagem colorida composta por três vetores bi-dimensionais com intensidades de pixels em três cores diferentes.

A figura 2.14 mostra uma estrutura típica de uma *ConvNet*, com uma arquitetura composta por uma série de estágios.

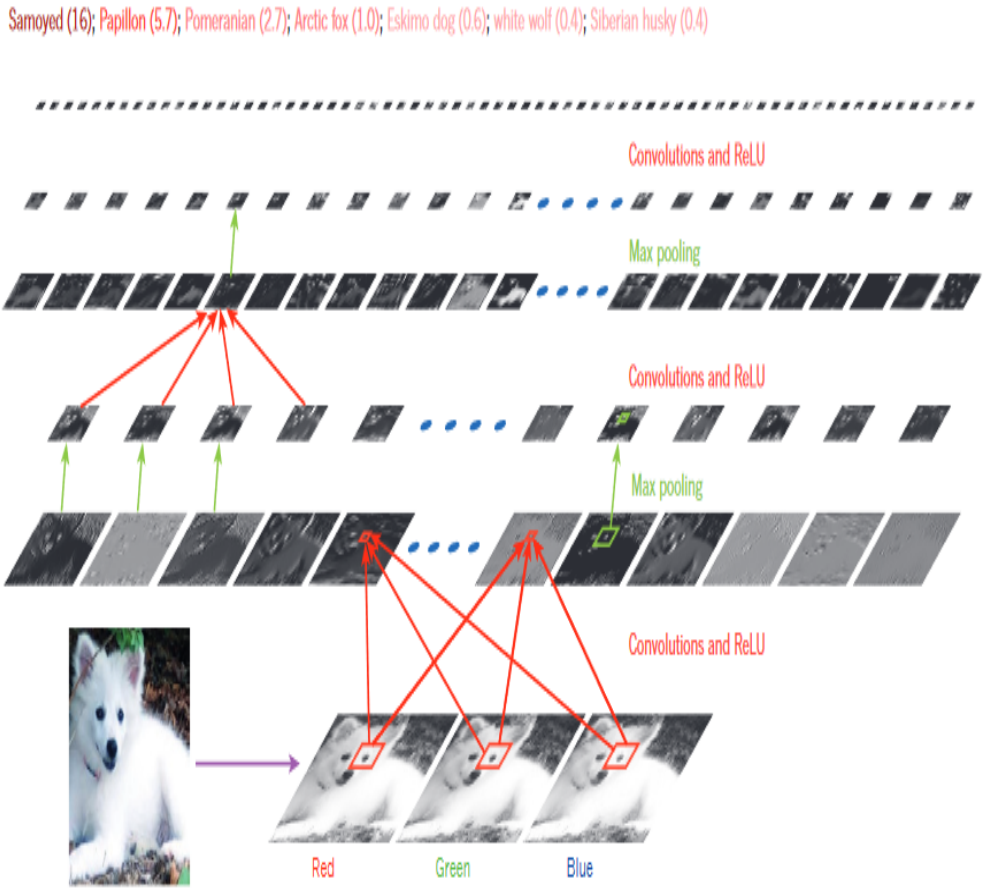

Figura 2.14: Exemplo de *ConvNet*

Na sua camada de entrada, a rede recebe imagens coloridas de três canais, as quais são processadas por camadas de convolução e camadas de *pooling*, até chegarem no final a uma camada totalmente conectada, responsável por apresentar o resultado de saída da rede.

As redes neurais profundas se utilizam do fato de muitos sinais serem compostos de forma hierárquica, de modo que as características de mais alto nível possam ser obtidas pela composição de características mais detalhadas.

Em imagens, por exemplo, a combinação de bordas e cantos formam pequenos padrões, que por sua vez formam partes maiores, que por sua vez formam objetos. Este comportamento é mais fácil de ser compreendido ao se pensar em imagens, mas ocorre de forma similar em texto e discurso também ao quebrá-los em fonemas, sílabas, palavras e frases por exemplo.

O trabalho mais famoso por inicialmente abordar a técnica de *ConvNet*, data de 1999, onde os autores utilizaram a rede para processamento de imagens buscando reconhecer objetos com grande diferença de formato [35].

Porém, as ideias por trás destas redes foram introduzidas em 1980 com o *Neocognitron*, que era baseado na ideia de células simples e células complexas. As células simples seriam responsáveis por processar características mais alto-nível e as complexas por processar as características mais detalhadas [32].

Desde os anos 2000, as *ConvNets* tiveram grande sucesso ao serem aplicadas para detecção e reconhecimento de objetos em imagens. Todas estas são aplicações em que existem enormes quantidades de dados já rotulados, como, por exemplo, reconhecimento de sinais de trânsito.

Apesar deste sucesso, foi apenas em 2012, durante a competição *ImageNet*, que as comunidades de *Machine Learning* e visão computacional deram seu devido valor a este algoritmo. Ao serem aplicadas a um conjunto de dados com aproximadamente um milhão de imagens, contendo mais de mil classes diferentes, obteve-se um resultado cujas taxas de erro foram praticamente a metade das taxas dos algoritmos que ganharam a competição. Isto foi possível devido ao uso eficiente de *hardware*, e de novas técnicas de regularização e funções de ativação que serão apresentadas mais à frente [36].

Hoje, estas técnicas são praticamente unanimidade em aplicações envolvendo reconhecimento e detecção em imagens, com desempenho se aproximando cada vez mais da performance humana. Arquiteturas modernas são compostas de dezenas de milhões de valores de pesos e bilhões de conexões entre neurônios. Consequentemente, verifica-se que a implementação das mesmas só se tornou possível devido aos avanços nas tecnologias de *hardware* e à novas técnicas de paralelização [32].

1. **Camadas de Convolução**

Como o próprio nome sugere, estas realizam a operação mais importante das *ConvNets*, a convolução. Pensando nos dados de entrada como imagens, a operação realizada por tais camadas seria similar a aplicação de uma espécie de filtro nas imagens processadas. A operação consiste em posicionar um filtro de tamanho fxf sobre a entrada e, para cada encaixe do filtro, somar a multiplicação elemento-a-elemento.

A inclusão da operação de convolução no processo da rede neural possibilitou a grande vantagem de permitir que as mesmas sejam implementadas com muito menos parâmetros em sua composição, devido a duas características que traz para o algoritmo:

– Compartilhamento de parâmetros: um parâmetro útil para uma parte da imagem também provavelmente será útil em outra parte; e

– Esparsidade das conexões: cada neurônio está apenas conectado a uma quantidade de neurônios limitada, ao invés de estar conectado a todos os neurônios da camada anterior, o que reduz drasticamente a quantidade de parâmetros.

Para ilustrar como funciona a operação de convolução, a equação (2-8) apresenta o resultado da convolução entre uma matriz **A**, correspondente a uma imagem em escalas cinza de tamanho 6x6, e uma matriz **F**, correspondente a um filtro de tamanho 3x3. Conforme indicado tem-se que a matriz resultante desta operação **R** possui dimensão 4x4.

$$\begin{vmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{vmatrix} \circledast \begin{vmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{vmatrix} = \begin{vmatrix} -5 & -4 & 0 & 8 \\ -10 & -2 & 2 & 3 \\ 0 & -2 & -4 & -7 \\ -3 & -2 & -3 & -16 \end{vmatrix} \tag{2-8}$$

Por exemplo, o primeiro elemento da matriz **R** da equação (2-8) pode ser calculado como sendo a soma do produto elemento-a-elemento da primeira sub-matriz 3x3 de A (destacada em vermelho na equação) pelos respectivos elementos da matriz F que representa o filtro, conforme dado por:

$$\begin{aligned} &3 \cdot 1 + 1 \cdot 1 + 2 \cdot 1 + 0 \cdot 0 + 5 \cdot 0 + ... \\ &... + 7 \cdot 0 + 1 \cdot (-1) + 8 \cdot (-1) + 2 \cdot (-1) = -5 \end{aligned} \tag{2-9}$$

Deslocando o filtro **F** pela matriz **A** e efetuando-se operações análogas a descrita em (2-9) obtêm-se todos os demais elementos de **R**. Um exemplo interessante que ajuda na interpretação da operação de convolução e de como estes filtros funcionam é a detecção de bordas verticais. Na equação (2-10), utilizando o mesmo filtro **F**, temos o seguinte resultado:

$$\begin{vmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{vmatrix} \circledast \begin{vmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{vmatrix} = \begin{vmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{vmatrix} \tag{2-10}$$

Buscando representar as matrizes da equação (2-10) por imagens , teríamos algo similar a figura 2.15. A matriz da esquerda corresponde a uma imagem com brusca mudança de branco para cinza no meio dela, evidenciando a presença de uma borda. Por sua vez, tem-se que a imagem resultante da convolução desta matriz pelo filtro **F** resulta em uma grande ativação da região que contém a borda vertical.

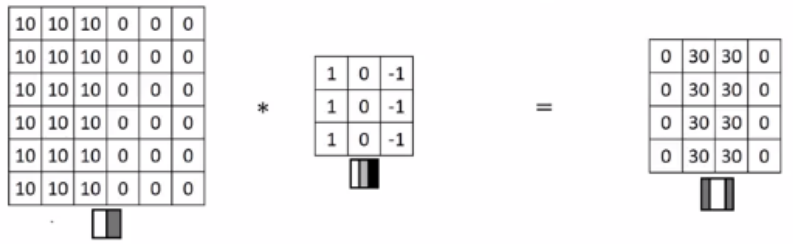

Figura 2.15: Detecção de borda vertical

Os pesos da rede neural para estas camadas correspondem justamente aos parâmetros dos filtros. Após o treinamento da rede, é comum que os filtros ao longo da rede passem a se "especializar", isto é, tornem-se capazes de detectar certas características dos sinais de entrada. É importante ressaltar que o algoritmo de *Back-Propagation* funciona de forma similar nas *ConvNets*, obtendo normalmente os resultados do treinamento [27].

O comportamento destes filtros ilustra muito bem a razão pela qual o compartilhamento de parâmetros funciona: por exemplo um detector de bordas verticais vai ter a mesma utilidade ao longo de toda a imagem.

Contudo, é comum se referir às camadas de convolução não como a aplicação de um único filtro, mas uma série de filtros em múltiplos canais. É comum, por exemplo, receber na entrada de uma destas camada uma imagem com três canais, de tamanho 32x32, e resultar em imagens de, por exemplo, 16x16.

Os principais hiper-parâmetros associados a uma camada de convolução são:

– número de filtros n_c;

– tamanho dos filtros f;

– *padding* p: aumento da dimensão da matriz de entrada, buscando uma maior representatividade dos seus elementos em neurônios;

– *stride* s: passo de evolução das submatrizes na operação de convolução.

Dessa forma, tem-se que, por exemplo, para uma entrada de tamanho $nxnx3$ isso resulta em uma matriz de dimensão isso resulta então numa matriz de dimensão $(\frac{n+2p-f}{s}+1)\text{x}(\frac{n+2p-f}{s}+1)\text{x}n_c$.

A figura 2.16 ilustra a passagem pela camada de convolução.

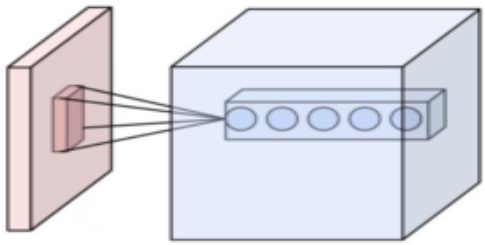

Figura 2.16: Ilustração da camada de convolução

2. **Camadas de *Pooling***

As camadas de *pooling* são utilizadas com os seguintes objetivos:

– reduzir o tamanho das representações;

– acelerar os cálculos computacionais;

– tornar alguns dos parâmetros das convoluções mais robustos.

A ideia por trás do procedimento de *pooling* está em representar uma grande parte dos dados de entrada por uma quantidade menor de dados mais significativa, assumindo que os sinais de uma região podem ser representados por um único valor. [27]

A técnica mais comumente utilizada é a *Max Pooling*, que seleciona o maior valor da matriz presente na sub-região na qual a operação é efetuada.

Os hiper-parâmetros associados a esta operação são:

– tamanho do filtro f;

– *stride* s.

O exemplo apresentado na equação (2-11) ilustra o funcionamento da operação com os hiper-parâmetros $f = 3$ e $s = 1$. Por sua vez, a figura 2.17 apresenta uma ilustração que representa o funcionamento de uma operação de max pooling ($f = 2$ e $s = 2$)em uma matriz 224x224x64.

$$\begin{vmatrix} 1 & 3 & 2 & 1 & 3 \\ 2 & 9 & 1 & 1 & 5 \\ 1 & 3 & 2 & 3 & 2 \\ 8 & 3 & 5 & 1 & 0 \\ 5 & 6 & 1 & 2 & 9 \end{vmatrix} \rightarrow \begin{vmatrix} 9 & 9 & 5 \\ 9 & 9 & 5 \\ 8 & 6 & 9 \end{vmatrix} \tag{2-11}$$

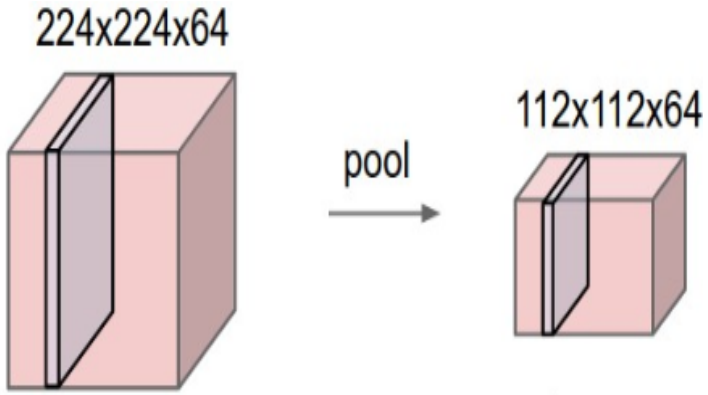

Figura 2.17: Ilustração da camada de *pooling*

3. **Camada Totalmente Conectada**

É uma camada de rede neural artificial padrão, na qual todos os neurônios da camada anterior estão conectados a todos os neurônios desta camada. Estas costumam ser as últimas camadas das *ConvNets*.

A figura 2.18 apresenta a arquitetura de rede LeNet-5, uma das mais tradicionais redes convolucionais. [37]

2.3.5.3 Redes Residuais

Durante o treinamento de redes muito profundas, ocorre muitas vezes o problema de gradientes exponencialmente grandes ou pequenos no processo de *Back-Propagation*. As Redes Residuais surgiram com o objetivo de solucionar este problema [38].

Estas redes implementam o que é conhecido como bloco residual, o qual consiste em algumas camadas de neurônios em sequência nas quais há conexões

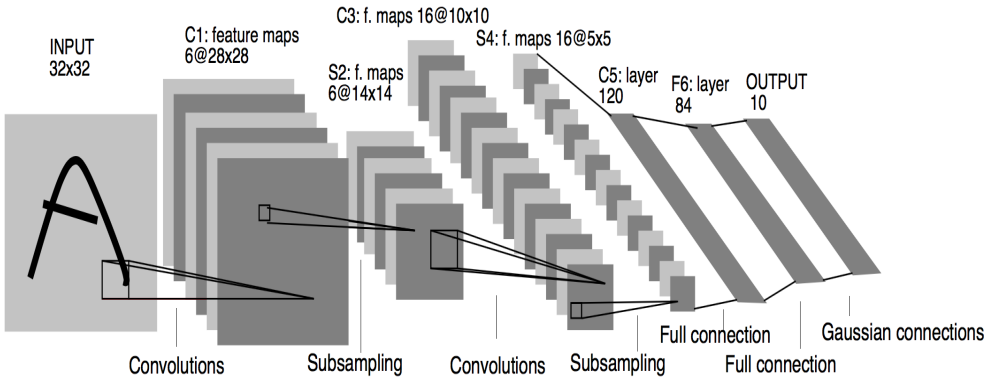

Figura 2.18: Ilustração original da LeNet-5

feedforward entre camadas da rede que não são diretamente em sequência. A figura 2.19 ilustra o funcionamento de um bloco residual.

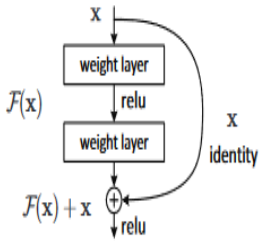

Figura 2.19: Funcionamento do bloco residual

O grande resultado alcançado por este trabalho foi a verificação de que, por meio de redes residuais, pode-se treinar redes mais profundas sem perda de generalização [38]. A figura 2.20 apresenta os resultados obtidos por [38] ao implementar as redes residuais. É possível observar que sem os blocos residuais a rede mais profunda (34 camadas) apresentava um desempenho pior que a de 18 camadas, conforme indicado na figura 2.20(a). Por outro lado, com o uso dos blocos, consegue-se um desempenho consideravelmente melhor, para a rede de 34 camadas, conforme mostrado na figura 2.20(b).

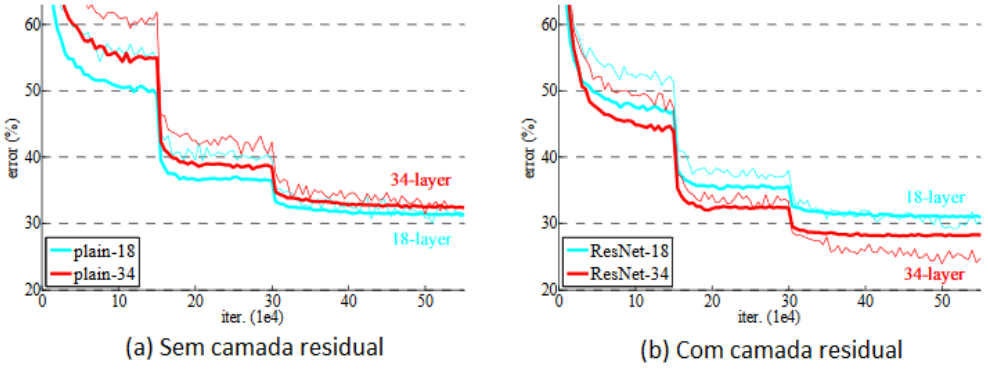

Figura 2.20: Desempenho da rede residual original

Por fim, a figura 2.21 faz uma comparação entre as topologias de uma rede de 34 camadas e uma rede residual de 34 camadas.

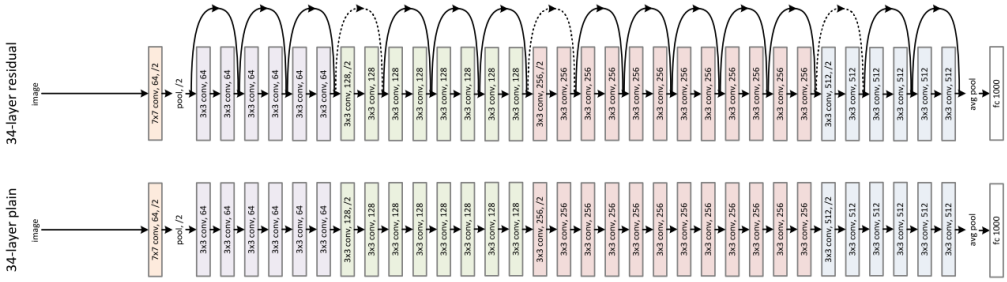

Figura 2.21: Comparação entre topologias de rede com a residual

2.3.5.4 Outras técnicas relevantes de *Deep Learning*

1. **Função de Ativação *ReLU***

Entre as camadas das redes neurais, sempre há uma função de ativação não-linear presente. Hoje, a mais popular destas é a ReLU (*Rectifying Linear Unit*), que apresenta comportamento análogo a um retificador de meia onda, conforme mostrado na figura 2.22.

$$R(z) = max(0, z) \tag{2-12}$$

2. **Regularização *Dropout***

Regularização é uma técnica para reduzir o *overfitting* durante o treinamento de uma rede neural. Dentre as técnicas de regularização existentes,

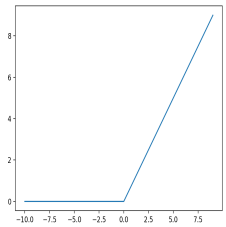

Figura 2.22: Função de Ativação ReLU

a mais famosa é a L2, que consiste em acrescentar à função que queremos minimizar (normalmente a métrica de erro) um valor proporcional a norma euclidiana do vetor de pesos, e traz consigo um novo hiper-parâmetro.

Já a regularização *dropout* consiste em retirar de forma aleatória neurônios da rede nos passos de treinamento. Dentre as técnicas de regularização existentes, esta técnica se mostra mais eficaz em reduzir o *overfitting* que as técnicas mais tradicionais de regularização [32; 39; 27].

3
Metodologia

3.1
Análise Exploratória dos Dados

Buscando avaliar detalhadamente as características da base de dados utilizada neste trabalho, efetuou-se uma análise exploratória dos dados, de forma a se estuar a melhor forma de tratar/processar seus dados, as funções de custo a serem usadas e as melhores topologias de redes a serem empregadas.

3.1.1
Distribuição da frequência das classes

As imagens que compõem a base de dados possuem 28 diferentes classes, não homogeneamente distribuídas. Esse comportamento é evidenciado por meio da análise gráfica da quantidade de classes de localização que ocorrem mais frequentemente, mostrado na figura 3.1

Analisando os resultados explicitados na figura 2.4, é possível observar que a maioria das estruturas proteicas está presente em componentes celulares de granulação grossa, como o nucleoplasma (25,4%), o citosol (16,2%) e a membrana plasmática (7,4%).

Avaliando a distribuição apresentada, associada a frequência de ocorrência de cada classe, também é possível prever que será difícil identificar corretamente as imagens que contenham classes menos frequentes, pois essas possuirão uma menor representatividade no conjunto de dados, aparecendo em quantidade muito menor do que as classes mais frequentes. Consequentemente, a figura de mérito acurácia poderá ter seu desempenho afetado devido a distribuição do conjunto de dados, não sendo isoladamente uma boa forma de medir a qualidade do algoritmo de classificação desenvolvido.

3.1.2
Distribuição da frequência da quantidade de classes

A fim de se avaliar a quantidade típica de classes por imagem, apresenta-se a figura 3.2. Por meio desta figura verifica-se que quase metade das imagens tem apenas uma classe, e quase 90% têm menos de 3 classes, indicando que

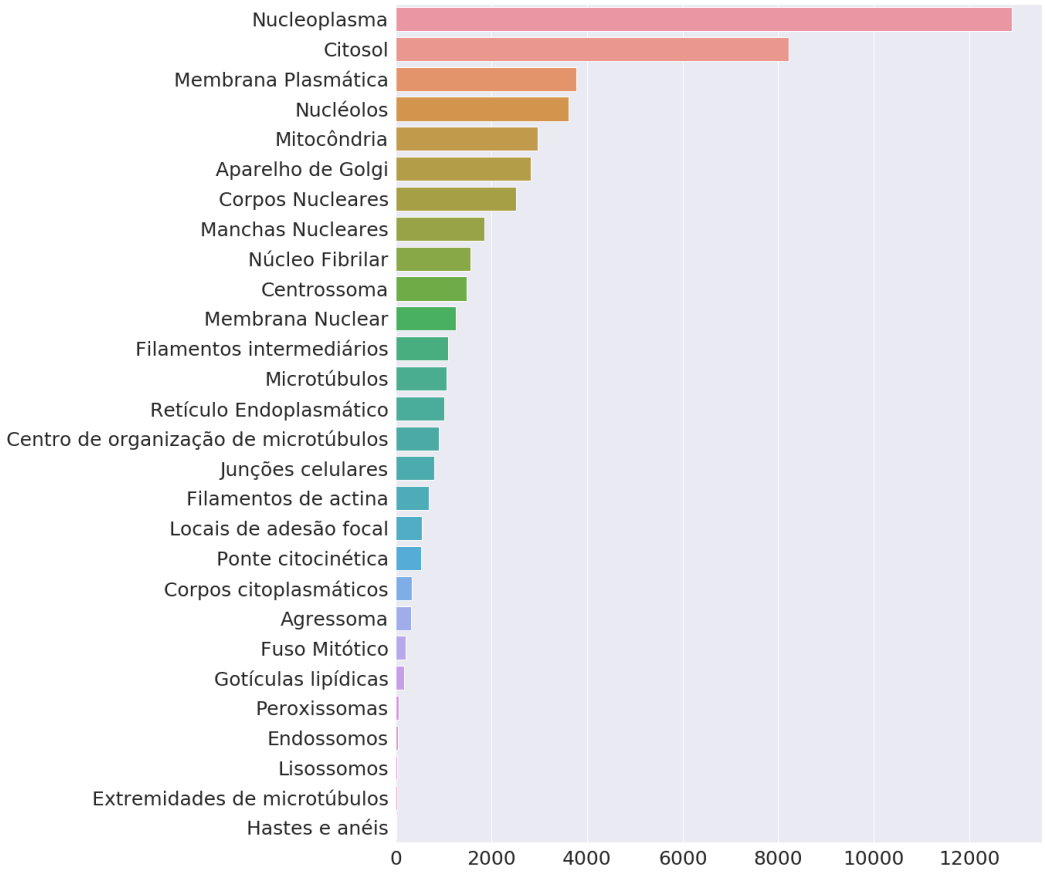

Figura 3.1: Ordem de frequência de classes

são raras as imagens com 3 ou mais classes, as quais representam apenas cerca de 10% do total de casos.

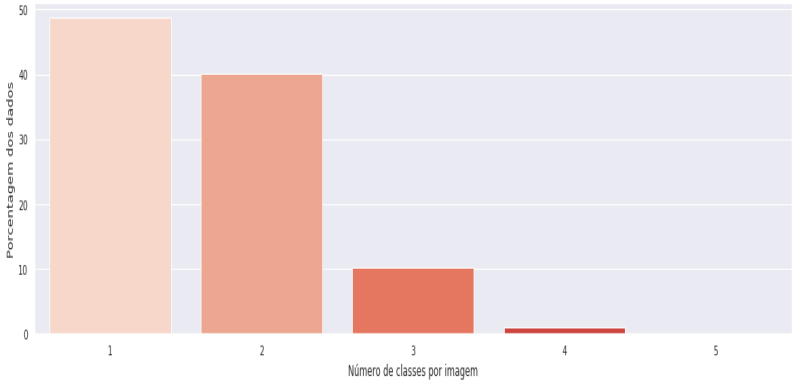

Figura 3.2: Percentual de imagens por quantidade de classes atribuídas

3.1.3
Frequência das classes presentes em conjunto

Outro aspecto relevante de se avaliar na base de dados é se existem classes que costumam aparecer juntas em uma mesma imagem e, consequentemente, se é possível utilizar-se desta informação para realizar uma melhor predição de certas classes.

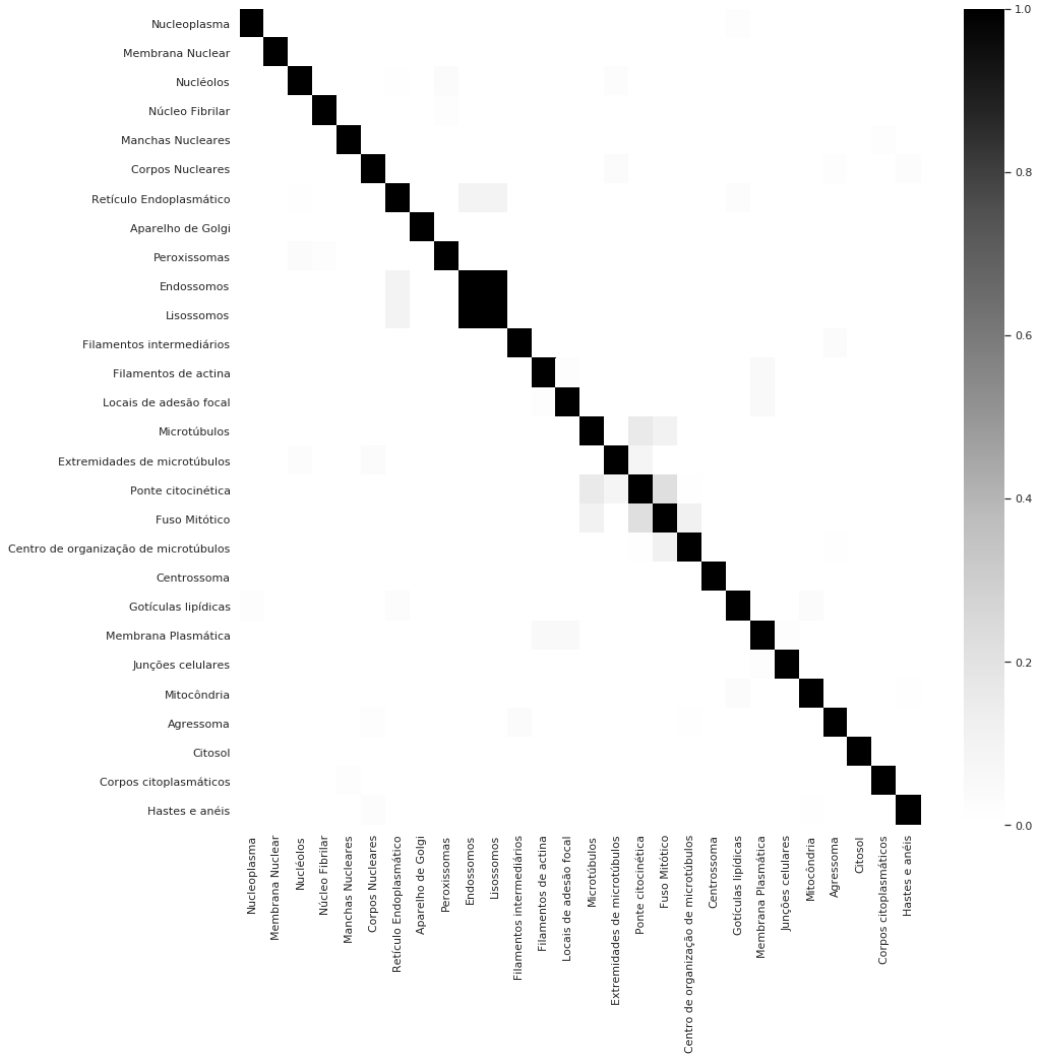

Figura 3.3: Frequência percentual das classes presentes em conjunto

A figura 3.3 apresenta o percentual de presenças conjuntas das classes, por meio da qual percebe-se que a maioria das classes tem pouca ou presença conjunta a outras classes. Entretanto, também é possível verificar que endossomos e lisossomos ocorrem normalmente juntos e algumas vezes localizados próximo ao retículo endoplasmático. Adicionalmente, embora em menor intensidade, também é possível observar uma correlação relevante entre ponte citocinética, fuso mitótico e microtúbulos.

3.1.4
Análise das classes menos presentes

As figuras 3.4 a 3.8 refletem a quantidade de vezes em que as classes menos representativas da base de dados aparecem juntas – Lisossomos (figura 3.4), hastes e anéis (figura 3.5), peroxissomas (figura 3.6), extremidades de microtúbulos (figura 3.7) e manchas nucleares (figura 3.8).

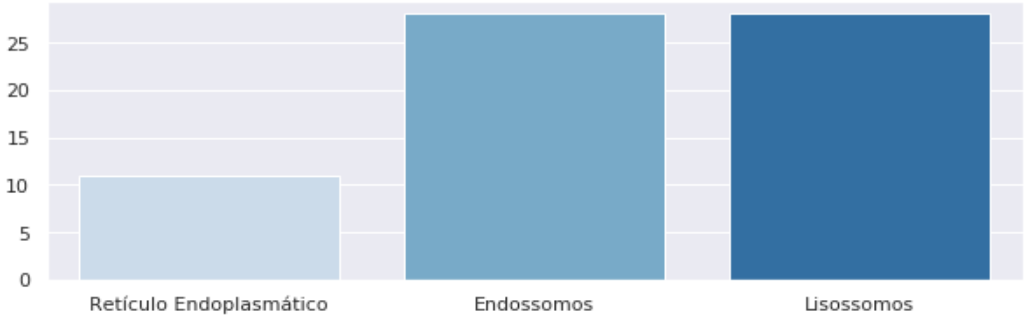

Figura 3.4: Lisossomos presentes com outras classes

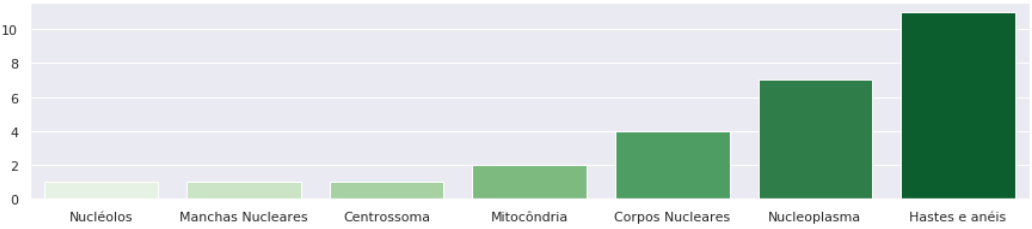

Figura 3.5: Hastes e Anéis presentes com outras classes

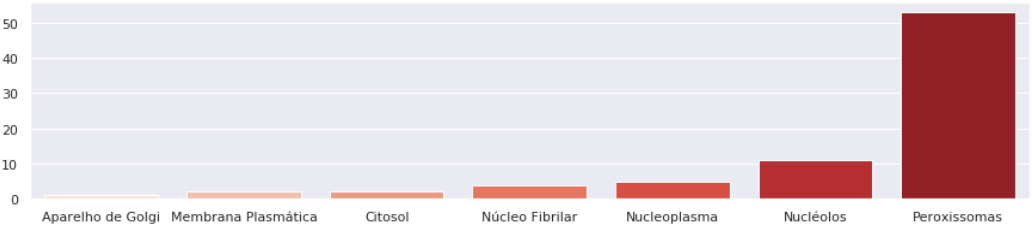

Figura 3.6: Peroxissomas presentes com outras classes

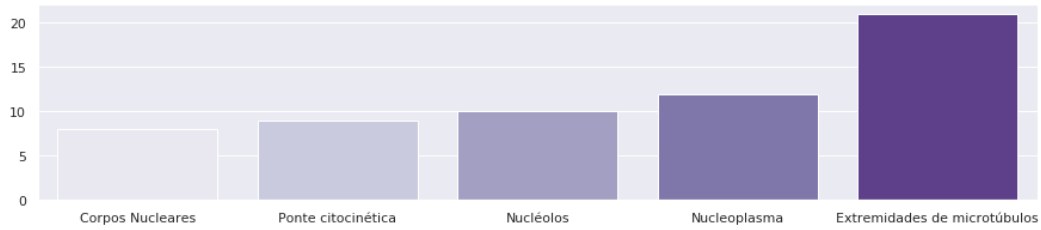

Figura 3.7: Extremidades de Microtúbulos com outras classes

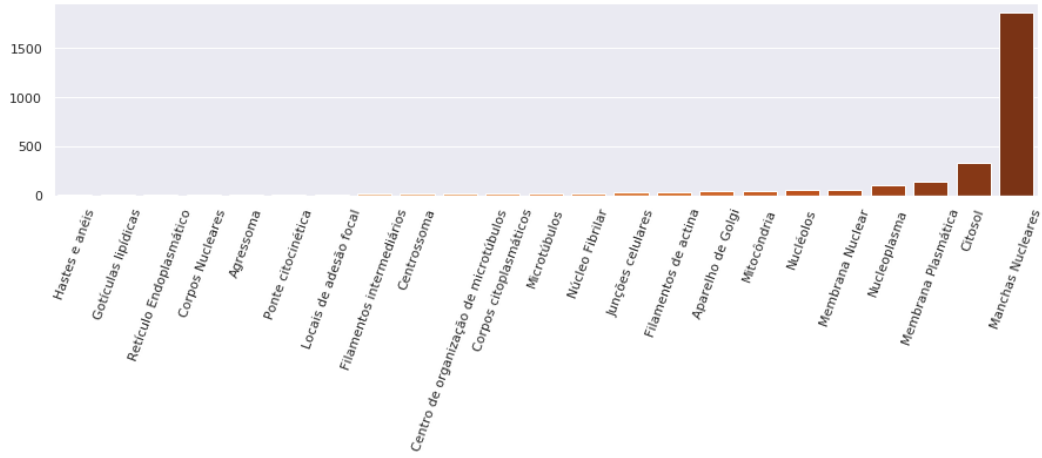

Figura 3.8: Manchas Nucleares com outras classes

Conforme evidenciado na figura 3.3, a figura 3.4 reforça a correlação entre lisossomos e endossomos. Por sua vez, por meio da figura 3.5, nota-se que a classe Hastes e Anéis tem algum tipo de relação com o núcleo celular, enquanto que os peroxissomos parecem ter relação com o núcleo e o citosol, conforme evidenciado na figura 3.6. Da figura 3.7 evidencia-se certa correlação de extremidades de microtúbulos com nucleoplasma e nucléolos, e da figura 3.8 observa-se que de fato as manchas nucleares têm poucos pares, aparecendo em grande parte sozinhas.

3.2 Métricas de Desempenho

3.2.1 Acurácia de Classificação

Conforme definido na equação (3-1), a acurácia é dada pela a razão entre a quantidade de predições corretas e a quantidade total de amostras.

$$Acc = \frac{Quantidade\ de\ Predições\ Corretas}{Quantidade\ de\ Amostras} \tag{3-1}$$

Normalmente, é uma boa métrica de desempenho para conjuntos de dados que possuam quantidades de amostras por classe igualmente distribuídas.

Por exemplo, assumindo uma classificação binária em que 98% das amostras são de uma classe A e apenas 2% de uma classe B, um algoritmo que sempre classifique as amostras como classe A obterá 98% de acurácia. Em contrapartida, se houvesse uma distribuição de 60% para a classe A e 40% para a B, a acurácia desse mesmo algoritmo cairia para 60%.

Como visto no capítulo 2, a análise da base de dados estudada no presente trabalho mostrou que a distribuição das amostras pelas classes é significativamente não uniforme, o que indica que esta métrica não será boa para avaliação do algoritmo de classificação desenvolvido.

3.2.2 *LogLoss*

Esta métrica funciona de forma a penalizar classificações falso positivas. É muito utilizada para classificação multi-classe e pode ser, portanto, uma importante métrica para avaliação do desempenho do algoritmo aqui desenvolvido.

De modo a se utilizar esta métrica, é necessário que o algoritmo tenha como valor de saída a probabilidade de cada amostra pertencer a cada classe.

Assumindo um conjunto de N amostras, com até M classes diferentes, a *LogLoss* pode ser calculada por:

$$LogLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij}), \tag{3-2}$$

onde y_{ij} indica se a amostra i pertence à classe j e p_{ij} corresponde à probabilidade da amostra i pertencer à classe j de acordo com o algoritmo.

Esta métrica retorna valores que podem variar de 0 à infinito, sendo que quanto menor seu valor, melhor a taxa de acertos do algoritmo.

3.2.3 Erro Médio Quadrático

O erro médio quadrático é a média do quadrado da diferença entre o valor correto e o valor previsto pelo modelo. Entre as vantagens desta métrica estão a penalização de erros grandes (atribuída ao valor quadrático) e sua fácil

derivação matemática, sendo muitas vezes usada como a métrica de erro que se tenta otimizar na rede neural. Entretanto, a mesma é adequada para avaliação do erro de redes com saídas analógicas (contínuas), não sendo adequada para avaliação do desempenho de redes destinadas a tarefas de classificação.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2 \tag{3-3}$$

3.2.4 Matriz de Confusão

Esta é uma das métricas mais intuitivas utilizadas para avaliar o desempenho de modelos de *machine learning*. Ela não fornece um valor único diretamente associado ao desempenho em si, mas, ao invés disso, fornece diversos números que permitem um "*deep dive*"nos resultados das demais métricas que estão sendo utilizadas para avaliação da rede.

A fim de interpretar o funcionamento da matriz, considere inicialmente um problema de classificação binária, onde as classes são "Sim"e "Não". No eixo horizontal ficarão as predições de classes e no eixo vertical as classes corretas. Dessa forma, pode-se, por exemplo, construir uma Matriz de Confusão como a apresentada na tabela 3.1.

Tabela 3.1: Exemplo de Matriz de Confusão para Classificação Binária

Classes	**Prevista: Sim**	**Prevista: Não**
Real: Sim	50	10
Real: Não	5	100

Neste exemplo hipotético, é possível analisar por meio da matriz de confusão que o modelo teria classificado um total de 165 amostras. Dentre esses dados, verifica-se pela matriz que 150 foram corretamente classificados, enquanto que ocorreram 5 falsos positivos e 10 falsos negativos. Inspecionando a matriz de confusão é possível identificar qual classe teve pior desempenho por exemplo, permitindo um melhor entendimento do que está dando certo ou errado na tarefa de classificação.

3.2.5 *Precision & Recall*

A precisão é uma métrica que leva em consideração as classes corretamente previstas (*true positive* - TP) e as que foram previstas de forma errônea

(*false positive* - FP), ignorando as classes que não foram previstas nas amostras (*true negatives* - TN - e *false negatives* - FN).

$$Precision = \frac{TP}{TP + FP} \tag{3-4}$$

Por outro lado, *Recall*, também conhecida como sensibilidade, utiliza a quantidade de TP e FN no seu cálculo.

$$Recall = \frac{TP}{TP + FN} \tag{3-5}$$

A decisão de quando usar cada métrica depende do objetivo do trabalho. Quando se deseja minimizar os FN, o *Recall* é uma boa métrica base, enquanto que a Precisão é mais usada nos casos em que se deseja minimizar os FP.

3.2.6 Pontuação F1

A pontuação F1 é uma métrica que busca trazer um equilíbrio entre precisão e *recall*, sendo calculada por meio da média harmônica entre as essas duas métricas.

$$F1 = 2\ \frac{1}{\frac{1}{precision} + \frac{1}{recall}} \tag{3-6}$$

A formulação desta métrica permite que não favoreçamos nem a precisão nem o *recall*, diferentemente do que ocorreria caso simplesmente calculássemos a média aritmética. Ao se efetuar a média harmônica, verifica-se que se qualquer uma das medidas computadas for muito baixa, o resultado da pontuação F1 se torna também muito baixo. Por exemplo, se a precisão fosse 3% (um resultado com muitos FP) e o *recall* 100%, a média aritmética seria 51%, enquanto o F1 seria de 5%.

3.2.6.1 Pontuação F1 Macro

No caso de problemas de classificação multi-classe, amplia-se a definição desta métrica. Em particular calcula-se a Pontuação F1 Macro, que nada mais é do que a média da Pontuação F1 das classes calculada de forma isoladamente como uma espécie de classificação binária para cada classe.

$$MacroF1 = \frac{\sum_{i=1}^{N} F1_i}{N} \tag{3-7}$$

Esta será a principal métrica de avaliação do desempenho do classificador desenvolvido neste trabalho, pois permite uma visão mais precisa do que a Acurácia, e mais completa do que as métricas de Precisão e *Recall* isoladamente, para o caso em que as amostras não são distribuídas igualmente entre as classes. Buscando simplificar a notação, a pontuação F1 Macro será tratada a partir daqui como Pontuação F1 ou apenas F1.

3.3 Conjunto de Validação

Para efeitos de comparação entre topologias, e para uso na iteração de hiper-parâmetros, um terço do conjunto de treinamento foi separado de forma aleatória de modo a se gerar um conjunto de validação.

Buscando ser efetivo na comparação dos modelos, garantiu-se que os conjuntos de validação criados sempre tivessem a mesma distribuição estatística de classes entre suas amostras, não necessariamente de forma agrupada, mas percentualmente em relação ao todo de amostras.

As métricas utilizadas para avaliação do desempenho dos modelos para o conjunto de validação serão base também para avaliar a existência de *overfitting*. É possível fazer tal avaliação ao se comparar o desempenho do modelo no conjunto de treinamento com o apresentado para o conjunto de validação. Caso o conjunto de treinamento tenha um desempenho consideravelmente melhor, é possível concluir que houve *overfitting*.

3.4 Topologia LeNet-5 Adaptada

3.4.1 Topologia

O primeiro modelo desenvolvido visou permitir um aprendizado inicial sobre as características do problema de classificação abordado. Dessa forma, optou-se por empregar uma topologia tradicional de *ConvNet* buscando-se obter resultados que pudessem dar *insights* sobre caminhos a serem adotados para a confecção da topologia final.

Em particular, a topologia escolhida foi a LeNet-5 [37], com algumas modificações. A tabela 3.2 detalha as diferentes camadas dessa rede, descrevendo sua função, hiper-parâmetros e funções de ativação.

Buscando reduzir a demanda computacional do treinamento da rede, a fim de se obter resultados mais rapidamente, para que se pudesse avançar com o desenvolvimento de novas topologias mais robustas, simplificou-se também o problema. Apenas o canal verde foi inicialmente utilizado como entrada da rede, visto que este é o canal responsável por identificar a localização subcelular da proteína de interesse. Além disso, com o mesmo objetivo, reduziu-se o tamanho da imagem para 128x128 pixels.

Tabela 3.2: Características da Topologia da Rede LeNet-5 Adaptada

Camada		**Quantidade de Canais**	**Tamanho da Saída**	**Tamanho do Filtro**	**Stride**	**Função de Ativação**
Entrada	Imagem	1	128x128	-	-	-
1	Convolução	16	28x28	5x5	1	ReLU
2	Max Pooling	16	14x14	2x2	2	ReLU
3	Convolução	32	10x10	5x5	1	ReLU
4	Max Pooling	32	5x5	2x2	2	ReLU
5	Convolução	128	1x1	5x5	1	ReLU
6	Totalmente Conectada	-	84	-	-	ReLU
Saída	Totalmente Conectada	-	28	-	-	Sigmoid

Por sua vez, as modificações feitas na versão original da topologia LeNet foram:

- funções de ativação das camadas escondidas sendo ReLU, ao invés de tangente hiperbólica;
- camada de saída da rede com função de ativação sigmoide, ao invés de *softmax*, já que a classificação pode ser de múltiplas classes para uma mesma amostra;
- substituiu-se a técnica de *pooling* média por *maxpooling*;
- utilizou-se regularização *dropout* com probabilidade 0,5 nas camadas 4 e 5;
- o número de canais das camadas de convolução foi aumentado;
- o tamanho da última camada foi de 29 neurônios ao invés de 10, refletindo o número de classes deste problema em particular.

A figura 3.9 ilustra a topologia final da rede LeNet-5 Adaptada.

3.4.1.1 Iteração de Hiper-parâmetros

Por ser um modelo inicial, que apenas buscava gerar conhecimento sobre as características do problema, não foi feita uma otimização exaustiva dos hiper-parâmetros nesta primeira topologia, utilizando-se valores *default*.

A tabela 3.3 apresenta o resumo dos hiper-parâmetros utilizados para a topologia implementada, cujas definições são apresentadas na sequência:

- *Epochs*: Quantidade de vezes que se treina com todo o conjunto de treinamento;

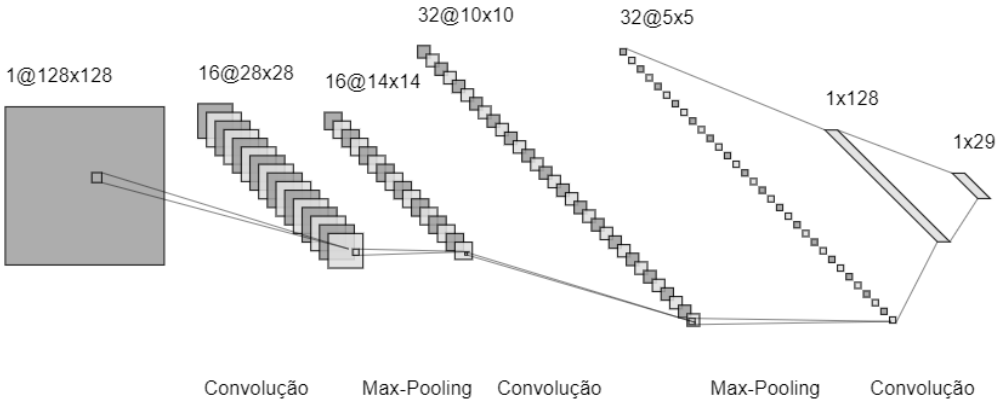

Figura 3.9: Topologia LeNet-5 Adaptada

Tabela 3.3: Hiper-Parâmetros da Rede LeNet-5 Adaptada

Hiper-parâmetro	**Valor**
Número de *Epochs*	10
Tamanho da *batch*	128
Inicialização dos Pesos	*Variance Scaling*
Probabilidade de *Hold*	0,5
Métrica de Erro	*Binary Cross Entropy*
Algoritmo de Otimização	Adadelta

- *Batch*: Conjunto menor do treinamento que é utilizado em um passo de treinamento;
- *Variance Scaling*: Inicialização dos pesos por uma distribuição normal centrada em zero, com desvio padrão $1/n$, onde n é o tamanho da entrada da camada;
- *Binary Cross Entropy*: É uma métrica de erro correspondente a uma adaptação das métricas de *Logloss* e do Erro médio quadrático;
- *Adadelta*: Algoritmo de atualização dos pesos similar ao gradiente descendente no *backpropagation*, porém adaptativo, de forma a não precisar de uma taxa de aprendizado [40].

3.5 Topologia ResNet-34

3.5.1 Topologia

A segunda topologia utilizada para solução do problema de classificação aqui abordado foi a *ResNet*. Assim como discutido no capítulo 2, essa topologia de *ConvNet* foi escolhida por ser capaz de apresentar um aprendizado robusto mesmo com uma grande quantidade de camadas e com um conjunto de treinamento pequeno, devido ao emprego dos blocos residuais. Portanto, inferiu-se que a *ResNet* apresenta grande potencial para resolução do problema de classificação aqui proposto.

Tendo em vista o conhecimento previamente adquirido pela análise dos resultados alcançados pela LeNet, decidiu-se configurar às características do modelo da seguinte forma:

- Camada de Entrada - foram utilizados 4 canais de cores como entrada da rede, porém mantendo as imagens reduzidas a 128x128 pixels;
- Distribuição das imagens nas classes - foram artificialmente geradas imagens adicionais das classes menos presentes, buscando aumentar a representatividade destas na base de dados;
- Regularização - optou-se por manter a *dropout*;
- Métrica de erro - optou-se por empregar uma nova métrica, com uma relação mais direta à pontuação F1;
- Algoritmo de otimização - foram testados o gradiente descendente, além do *Adadelta* e Adam[41];
- Inicialização dos pesos - foram testadas outras formas de inicialização;
- Número de *epochs* - a quantidade de épocas foi aumentada;
- Características das camadas - foi efetuada uma busca pela combinação dos hiper-parâmetros capaz de alcançar o melhor resultado para o conjunto de validação.

3.5.1.1 Blocos residuais

De modo a usufruir dos benefícios associados ao emprego dos blocos residuais das *ResNets*, destacados no capítulo 2, dois blocos residuais foram introduzidos: o de identidade e o convolucional.

O bloco identidade só pode ser utilizado quando as camadas que estão sendo "puladas"pela conexão atalho do bloco residual não modificam a dimensionalidade do tensor de entrada. Neste bloco, o tensor de entrada é somado a outro tensor, resultante de duas camadas de convolução completas e uma

sem ativação, antes de passar pela função de ativação. A figura 3.10 ilustra o funcionamento deste bloco.

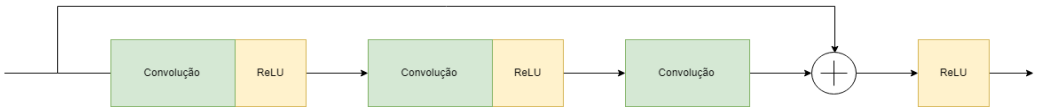

Figura 3.10: Bloco residual identidade

Já o bloco residual convolucional é utilizado quando a dimensionalidade não é mantida ao longo do bloco, sendo necessário que ocorra uma convolução que mude a dimensionalidade do "atalho"também, conforme ilustrado na figura 3.11.

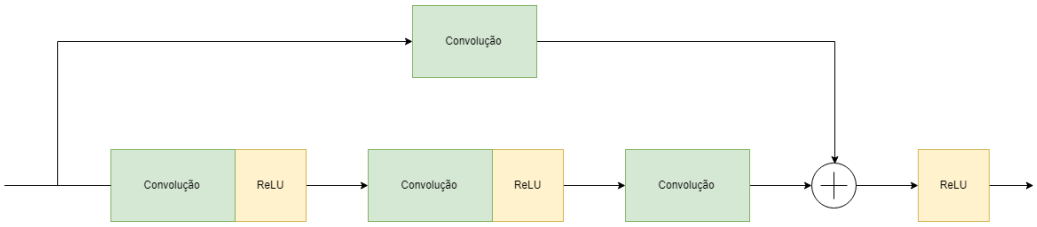

Figura 3.11: Bloco residual de convolução

3.5.1.2 Aumento da Representatividade das classes menos frequentes

Buscando criar um conjunto de treinamento cuja distribuição de classes seja menos heterogênea, de modo a haver um desfavorecimento menor das classes com poucas amostras, fez-se um *oversample* das amostras pertencentes a estas classes.

Em particular, buscando manter verdadeiro o conteúdo das imagens frente às classes representadas por elas, não julgou-se adequado inserir grandes distorções nas imagens, sendo que a técnica de *oversample* utilizada consistiu apenas em rotacionar por 90 graus e 180 graus as imagens originais. Dessa forma, triplicou-se a quantidade de imagens destas classes, possibilitando o aumento e a homogeneização do conjunto de treinamento e, consequentemente, aprimorando a capacidade de aprendizado da rede.

As classes escolhidas para aplicação do processo aqui descrito foram as menos representativas da base de dados, diga-se:

- Hastes e Anéis;
- Peroxissomas;

– Extremidades de Microtúbulos;

– Endossomos; e

– Lisossomos.

3.5.1.3 Métrica de Erro

Buscando melhorar o desempenho da rede com relação a pontuação F1, a principal métrica de desempenho utilizada neste trabalho, é interessante utilizar uma métrica de erro para otimização da rede que tenha uma relação mais direta à F1. Dessa forma, intuitivamente, poder-se-ia tender a usar $1 - F1$ como métrica de otimização, porém esta função é não diferenciável e, consequentemente, acarretaria em problemas na aplicação do *backpropagation*.

Para resolver este problema, usam-se probabilidades (ao invés de 0 ou 1) no momento do cálculo da métrica, de modo que minimizar esta métrica de erro seja equivalente a maximizar o valor da pontuação F1.

3.5.1.4 Iteração dos Hiper-parâmetros

A tabela 3.4 apresenta o conjunto de hiper-parâmetros avaliados e como estes foram variados durante as iterações. Também foram testadas diferentes combinações de blocos residuais de identidade e convolução.

Tabela 3.4: Variação de Hiper-parâmetros da *ResNet*

Hiper-parâmetro	**Variação**
Algoritmo de Otimização	Adadelta x Adam x Gradiente Descendente
Taxa de Aprendizado	0,001, 0,005, 0,01, 0,05
β_1 [a]	0,5, 0,7, 0,9
β_2 [a]	0,9, 0,99, 0,999
Termo de Momento [b]	0,2, 0,4, 0,6, 0,8
Inicialização dos Pesos	Zeros, *Variance Scaling*
Quantidade de *Epochs*	Acima de 20
Métrica de Erro	*Binary Cross Entropy*, Variação de Pontuação F1
Quantidade de Camadas	17, 34, 50

[a] Apenas para o *Adam*
[b] Apenas para o Gradiente Descendente

Após as variações dos hiper-parâmetros da tabela acima, também experimentou-se diferentes combinações de blocos residuais identidades e convolucionais.

Ao fim das iterações pelas diferentes topologias e combinações de hiper-parâmetros, verificou-se que a rede que alcançou o melhor desempenho no

conjunto de validação, dentre todas as avaliadas, é aquela caracterizada pelas tabelas 3.5 e 3.6.

Vale ressaltar que depois de muitas tentativas de aumento do número de *epochs*, este parâmetro foi fixado em 35, não por ser necessariamente o valor associado ao melhor desempenho para as redes, mas devido ao alto tempo de treinamento que o modelo estava chegando para quantidades superiores – da ordem de 5 horas. A topologia final da *ResNet* selecionada pode ser visualizada na figura 3.12.

Tabela 3.5: Valores de Hiper-parâmetros com melhor desempenho para a ResNet

Hiper-parâmetro	**Variação**
Número de Algoritmo de Otimização	Adam
Taxa de Aprendizado	0,01
β_1	0,9
β_2	0,99
Termo de Momento	-
Inicialização dos Pesos	*Variance Scaling*
Quantidade de *Epochs*	35
Métrica de Erro	Variação de Pontuação F1
Quantidade de Camadas	34

Tabela 3.6: Características da Topologia da Rede ResNet-34,

Camadas		**Canais**	**Tamanho do Filtro**	**Stride**	**Função de Ativação**
Entrada	Imagem	4	128x128[a]	-	-
1	Convolução	64	7x7	2	ReLU
2	Max Pooling	64	3x3	2	ReLU
3 - 5	Residual de Convolução	256/camada	3x3	1	ReLU
6 - 8	Residual Identidade	256/camada	3x3	0	ReLU
9 - 11	Residual Identidade	256/camada	3x3	0	ReLU
12 - 14	Residual de Convolução	512/camada	3x3	2	ReLU
15 - 17	Residual Identidade	512/camada	3x3	0	ReLU
18 - 20	Residual Identidade	512/camada	3x3	0	ReLU
21 - 23	Residual Identidade	512/camada	3x3	0	ReLU
24 - 26	Residual de Convolução	1024/camada	3x3	2	ReLU
27 - 29	Residual Identidade	1024/camada	3x3	0	ReLU
30 - 32	Residual Identidade	1024/camada	3x3	0	ReLU
33	Max-Pooling e Totalmente Conectada	-	84[b]	2x2[c]	ReLU
Saída	Totalmente Conectada	-	28	-	Sigmoid

[a]Tamanho da imagem
[b]Número de Neurônios
[c]Tamanho do filtro de pooling

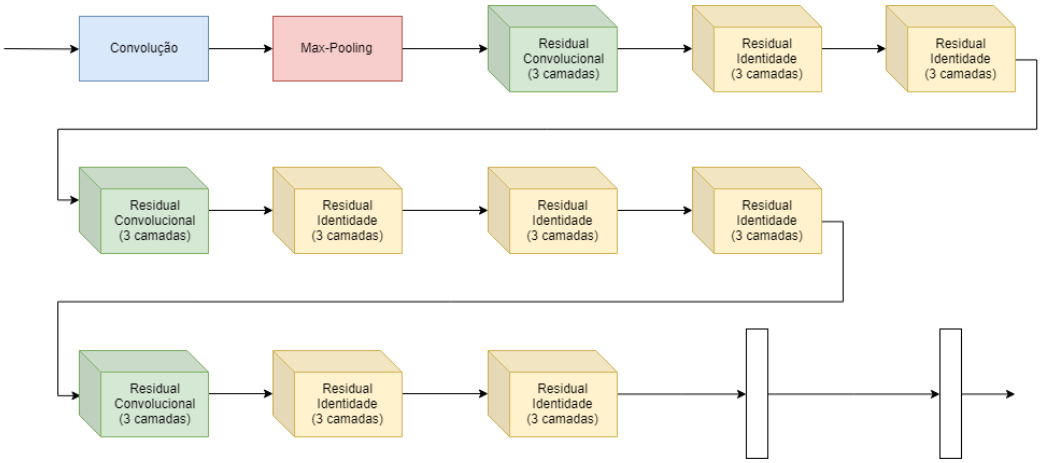

Figura 3.12: Topologia ResNet-34

3.6 Modificações na ResNet Ótima

Buscando melhorar ainda mais o desempenho do modelo desenvolvido, após iteração dos hiper-parâmetros, foram realizadas mais duas de modificações na rede.

Contudo, como estas modificações aumentam ainda mais o já alto tempo de treinamento, não foram realizadas buscas de hiper-parâmetros ótimas para estes modelos. Ou seja, estas variações foram diretamente adicionadas ao conjunto ótimo de hiper-parâmetros identificado na seção 3.4. O principal objetivo destas modificações foi avaliar preliminarmente algumas hipóteses de caminhos para aprimoramentos e, consequentemente, direcionar possíveis futuros estudos.

3.6.1 Topologia Híbrida ResNet-SVM

A primeira alteração realizada na topologia foi a criação de um modelo híbrido ResNet-SVM. Para tal fim, utilizou-se a rede ResNet-34 já treinada e substituiu-se sua última camada por múltiplas SVMs, de acordo com técnica similar a utilizada em [42].

A metodologia utilizada foi a *One-Versus-All*, criando-se assim 28 SVMs, uma para cada classe, cujas entradas correspondem às saídas dos 84 neurônios da penúltima camada da ResNet-34.

Como as 33 camadas da rede já haviam sido previamente treinadas, o treinamento foi realizado desta vez apenas nas SVMs, fixando-se os pesos das camadas mantidas de modo a se evitar um novo treinamento da rede

neural inteira, prática normalmente conhecida na comunidade científica como *Transfer Learning*. Esta ideia permite ganhos na velocidade de treinamento do modelo. A tabela 3.7 apresenta os parâmetros das SVMs utilizadas.

Tabela 3.7: Hiper-Parâmetros das SVMs

Hiper-parâmetro	**Valor**
Função *Kernel*	*Radial Basis*
Coeficiente do *Kernel* (γ)	1,0
Custo do Erro (C)	1,0

3.6.2 Melhorias na Resnet Ótima

Buscando outros caminhos para melhorar o desempenho da classificação, além do modelo híbrido ResNet-SVM proposto na seção 3.5.1, duas outras modificações foram feitas na ResNet-34 original.

3.6.2.1 Implementação de Barreiras de Decisão

Até então, no momento do cálculo da métrica de desempenho, a rede interpretava o resultado da seguinte forma:

$$Out = \begin{cases} 1 \; se \; out \geq 0.5 \\ \\ 0 \; se \; out < 0.5 \end{cases} \tag{3-8}$$

Isto é, para cada amostra, para cada classe, se o valor previsto fosse maior 0,5, o modelo considerava que a amostra avaliada possuía aquela determinada classe, e, caso contrário, não.

Porém, para classes menos presentes, cogitou-se que uma possibilidade para melhorar a efetividade das previsões da rede seria reduzir o valor da barreira de decisão nesses casos, ou seja, ajustá-la para valores menores que 0,5.

Dessa forma, foram alteradas as barreiras de decisão das seguintes classes:

- Hastes e Anéis: 0,05;
- Peroxissomas: 0,1;
- Lisossomos: 0,15.

As figuras 3.13 a 3.15 ilustram a distribuição de probabilidade dos resultados da ResNet-34 para as classes acima. Isto é, para as amostras do

conjunto de validação, como se distribuem os valores de saída antes de serem processados pelas barreiras de decisão. Os valores foram escolhidos por uma análise visual, objetivando-se escolher aproximadamente o valor central da distribuição.

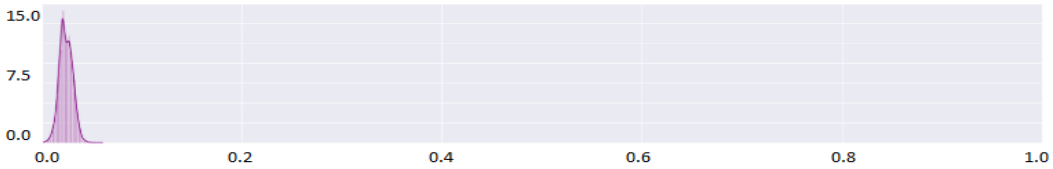

Figura 3.13: Probabilidade de escolha de Hastes e Anéis pelo modelo

Figura 3.14: Probabilidade de escolha de Lisossomos pelo modelo

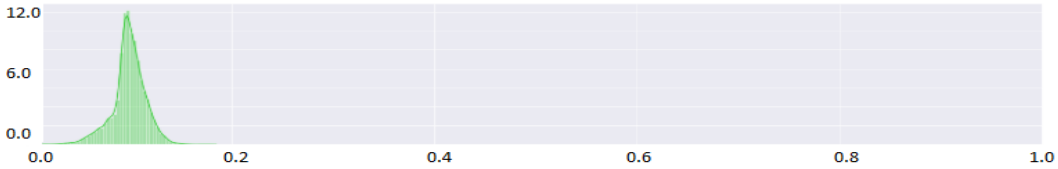

Figura 3.15: Probabilidade de escolha de Peroxissomas pelo modelo

3.6.2.2 Aumento do Tamanho da Imagem de Entrada

Conforme mencionado no início do capítulo 2, as imagens originais do conjunto de treinamento são de tamanho 512x512 pixels. Apesar disso, buscando uma maior velocidade no treinamento, as imagens foram pré-processadas antes do treinamento e tiveram seu tamanho reduzido para 128x128 pixels, de forma a diminuir a quantidade de parâmetros que a rede teria que treinar.

Porém, ao comparar-se imagens com diferentes resoluções, percebeu-se grande diferença. A figura 3.16 apresenta a comparação entre um exemplo aleatório de imagem do conjunto de treinamento no tamanho original (512x512) e a mesma imagem após o pré-processamento (128x128). Dessa forma, nota-se a olho nu uma clara perda de informação e detalhes.

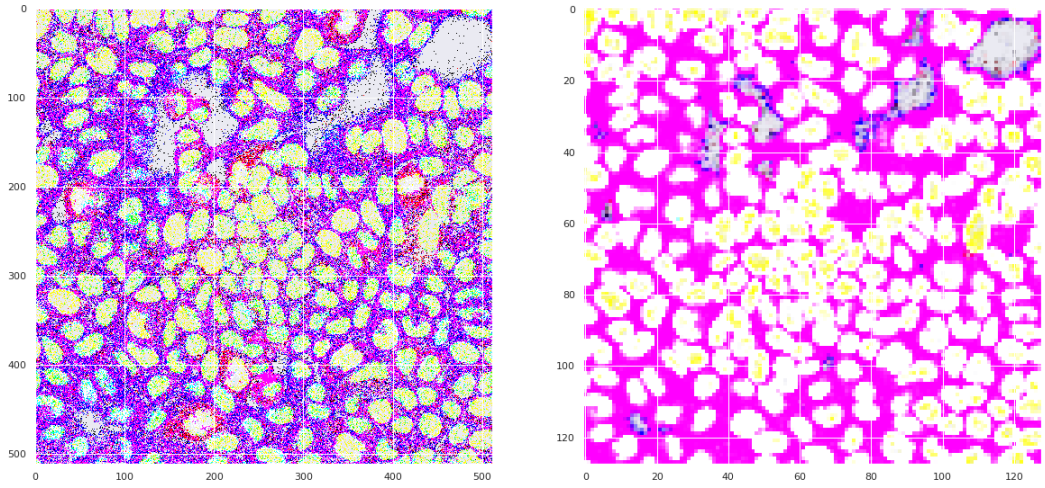

Figura 3.16: A esquerda imagem original e a direita após pré-processamento

Dessa forma, cogitou-se que o aumento da resolução das imagens poderia facilitar a tarefa de classificação e, consequentemente, melhorar o desempenho dos algoritmos desenvolvidos. Consequentemente, de modo a se avaliar essa hipótese, mudou-se o pré-processamento da imagem para reduzir seu tamanho para 256x256 pixels, ao invés de 128x128, buscando reter maior quantidade de informações na imagem e ainda mantendo o treinamento mais rápido do que seria com o conjunto original (512x512).

4
Resultados

4.1
LeNet-5 Adaptada

4.1.1
Resultados

A tabela 4.1 apresenta os resultados obtidos pelas métricas de avaliação de desempenho, para os conjuntos de treinamento e validação, ao fim do treinamento do modelo.

Tabela 4.1: Métricas de Desempenho da Rede

Métrica	**Treinamento Final**	**Validação Final**
Acurácia	0,9123	0,9201
Logloss	0,1694	0,1691
Pontuação F1	0,2430	0,2266

A figura 4.1 apresenta a evolução da métrica *LogLoss* para os conjuntos de treinamento e validação, ao longo das *epochs*. Por sua vez, a figura 4.2 apresenta a evolução da métrica de pontuação F1 para os conjuntos de treinamento e validação, ao longo das *epochs*.

Adicionalmente, as figuras 4.3 e 4.4 apresentam a distribuição de probabilidades encontrada pelo algoritmo para, respectivamente, as classes Nucleoplasma e Citosol.

Finalmente, para análise do efeito da regularização *dropout*, a figura 4.5 apresenta a comparação dos resultados da topologia analisada com e sem o uso desta técnica.

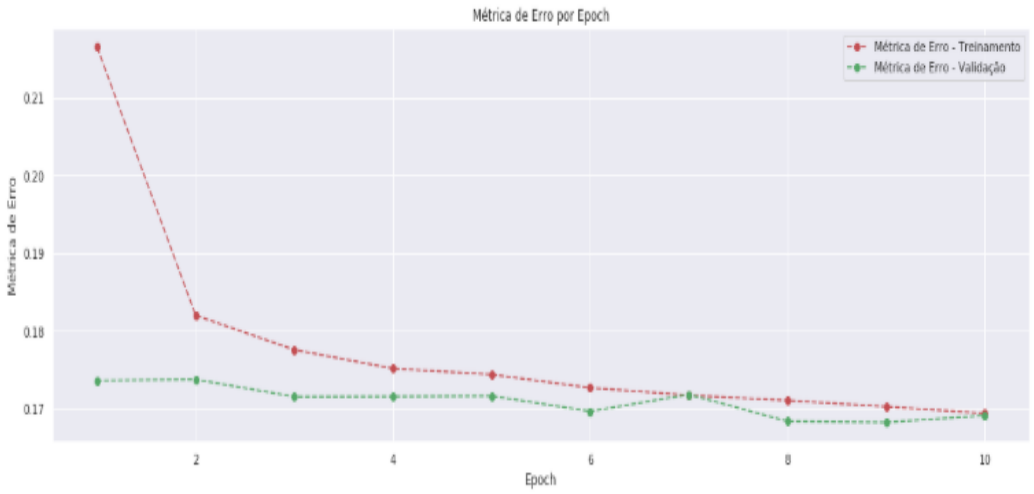

Figura 4.1: Evolução da métrica *LogLoss* para o conjunto de treinamento e validação

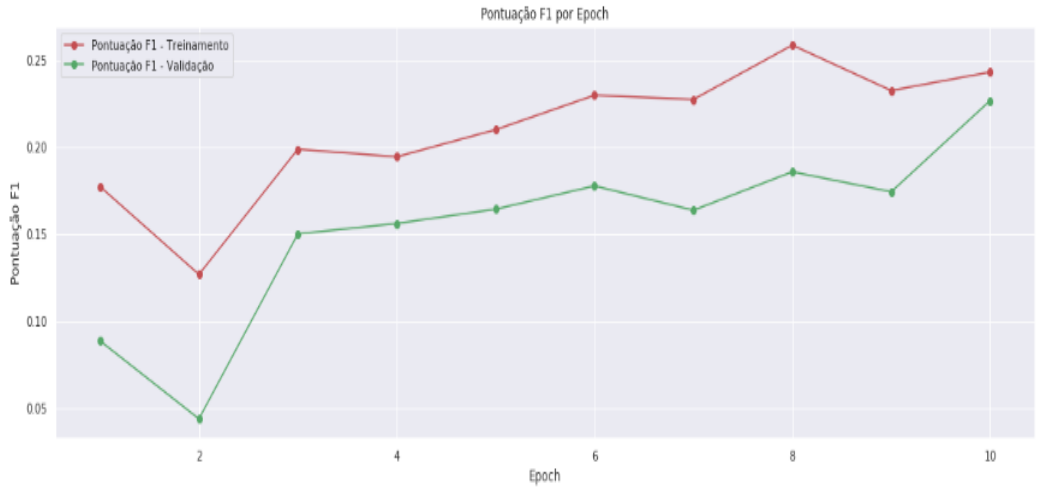

Figura 4.2: Evolução da pontuação F1 para o conjunto de treinamento e validação

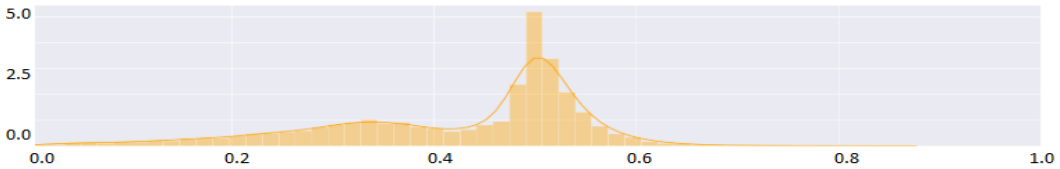

Figura 4.3: Probabilidade de escolha de Nucleoplasma pelo modelo

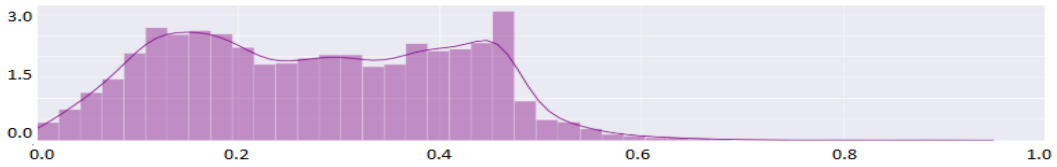

Figura 4.4: Probabilidade de escolha de Citosol pelo modelo

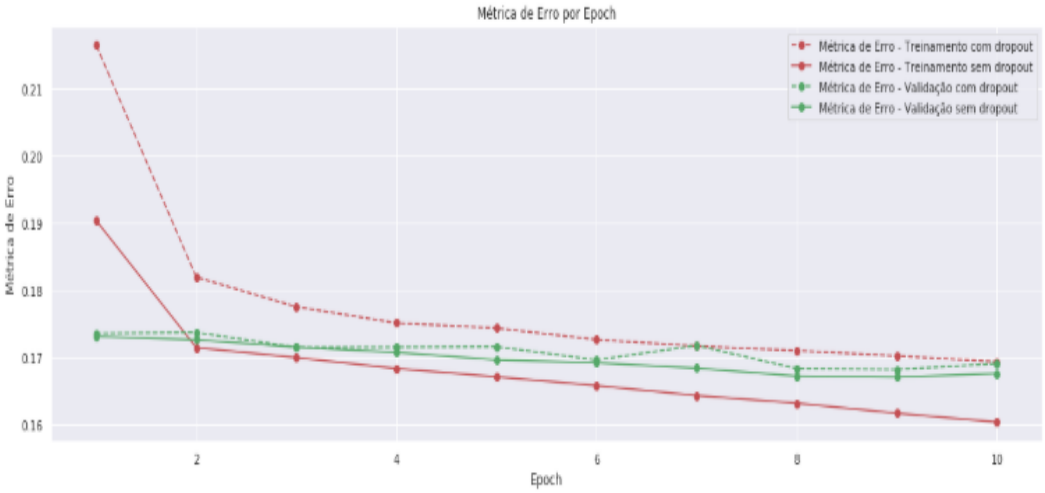

Figura 4.5: Efeito da regularização *Dropout* na LeNet-5

4.1.2 Comentários

Conforme mencionado no início do capítulo 3, esta topologia foi desenvolvida com o objetivo de se obterem mais informações a respeito do problema de classificação e, a partir destas conclusões, criar um novo modelo com melhor desempenho.

O primeiro resultado que chama a atenção é que a acurácia alcançada por este algoritmo já é consideravelmente alta, chegando a mais de 90%. Isto ocorre devido a distribuição das amostras entre as classes ser muito desigual, como visto na figura 2.2. Portanto, a alta acurácia é explicada por alta taxa de acertos associados as classes mais presentes no conjunto, não garantindo, entretanto, que se tenha uma adequada classificação das classes menos representativas.

Este resultado é fortalecido pelos gráficos das figuras 4.3 e 4.4, que mostram que o algoritmo foi capaz de reconhecer as probabilidades distintas da presença das classes nas amostras. Também, é possível inferir que a utilização de valores de *threshold* diferentes para classes diferentes pode ter potencial para melhorar o desempenho do modelo. Isto é, classes menos representativas podem ser consideradas presentes nas imagens quando a probabilidade de presença na amostra for maior que um valor relativamente pequeno (inferior a 0,5), e vice-versa para classes com maior representatividade no conjunto.

É possível observar a evolução da métrica de erro utilizada pelo algoritmo de otimização, a *Logloss*, na figura 4.1. Nota-se que, ao longo das 10 *epochs* de treinamento, o erro diminui tanto no conjunto de treinamento quanto no de validação. A diferença relativamente pequena entre os dois conjuntos indica que o modelo não está sofrendo *overfitting*. Porém, ainda não se observa o erro estabilizar por completo, indicando que um aumento do número de *epochs* pode ser uma maneira eficiente de obter desempenhos ainda melhores.

Além disso, constata-se que o algoritmo variou o erro de forma muito lenta, o que pode ser causado pelo algoritmo adaptativo de taxa de aprendizado do *Adadelta*. Consequentemente, uma forma de acelerar o aprendizado pode ser a utilização de algoritmos de otimização mais eficientes.

Uma análise mais completa a respeito do *overfitting* pode ser feita por meio da inspeção da figura 4.5, que mostra uma comparação entre a rede com e sem a utilização da técnica de regularização *dropout*. No caso em que não se utiliza *dropout*, nota-se que a rede apresenta um desempenho melhor para o conjunto de treinamento, sendo, entretanto, possível observar que o erro começa a aumentar no conjunto de validação, mostrando um claro problema de *overfitting*. A técnica de regularização evitou a ocorrência desse comportamento, se mostrando, portanto, eficiente, e, consequentemente,

também será usada novamente no próximo modelo.

A respeito da métrica de erro escolhida, é difícil inferir pelos resultados obtidos qual sua influência no desempenho do modelo.

Visando estudar a relação entre a métrica de erro *Binary Cross Entropy* e a principal métrica de desempenho usada neste trabalho, a pontuação F1, supõe-se um problema de classificação aleatório. Escolhem-se valores aleatórios que desejam-se prever e valores aleatórios de previsão. Ao se calcular as métricas destas previsões em relação aos valores reais escolhidos, é possível gerar o gráfico da pontuação F1 em função da *Logloss*, conforme ilustrado na figura 4.6.

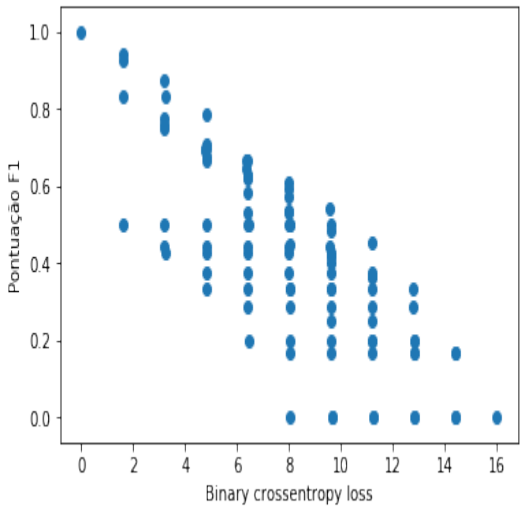

Figura 4.6: Relação entre a Pontuação F1 e *Binary Cross Entropy*

Por meio da figura 4.6, verifica-se que as duas métricas não têm uma relação muito direta, inclusive havendo pontos onde nota-se grande variação da métrica de erro para um mesmo valor de pontuação F1. Portanto, para o próximo modelo desenvolvido, constatou-se que seria interessante definir uma métrica de erro que tenha uma relação mais direta com a métrica de desempenho.

Por fim, avaliando-se o desempenho do modelo como um todo, para o problema aqui abordado, verificou-se que sua pontuação F1 para o conjunto de validação foi de 0,2266, indicando um desempenho insatisfatório do modelo, apesar do alto valor de acurácia obtido. Esta diferença decorre devido aos dados não serem distribuídos uniformemente entre as classes, sendo possível, portanto, conseguir uma acurácia alta para o modelo se este, por exemplo, sempre tentar prever as opções de resultado apenas entre as classes mais frequentes, desprezando as classes menos representativas. Por outro lado, como a pontuação F1 é uma média harmônica entre precisão e *recall*, esta métrica é

mais imune à heterogeneidade de distribuição das classes, só conseguindo atingir valores elevados caso o algoritmo seja capaz de prever corretamente tanto as classes mais representativas quanto as menos representativas. Este resultado será usado para comparação com os outros modelos aqui desenvolvidos e com o estado da arte apresentado no capítulo 5.

4.2 ResNet-34

4.2.1 Resultados

A tabela 4.2 apresenta as métricas de desempenho obtidas para os conjuntos de treinamento e validação. Por sua vez, a figura 3.10 apresenta a evolução da métrica de erro, neste caso a adaptação da pontuação F1, para os conjuntos de treinamento e validação ao longo das *epochs*.

Tabela 4.2: Métricas de Desempenho da Rede

Métrica	**Treinamento Final**	**Validação Final**
Acurácia	0,9335	0,9248
Adaptação de F1 (erro)	0,5772	0,6065
Pontuação F1	0,4228	0,3935

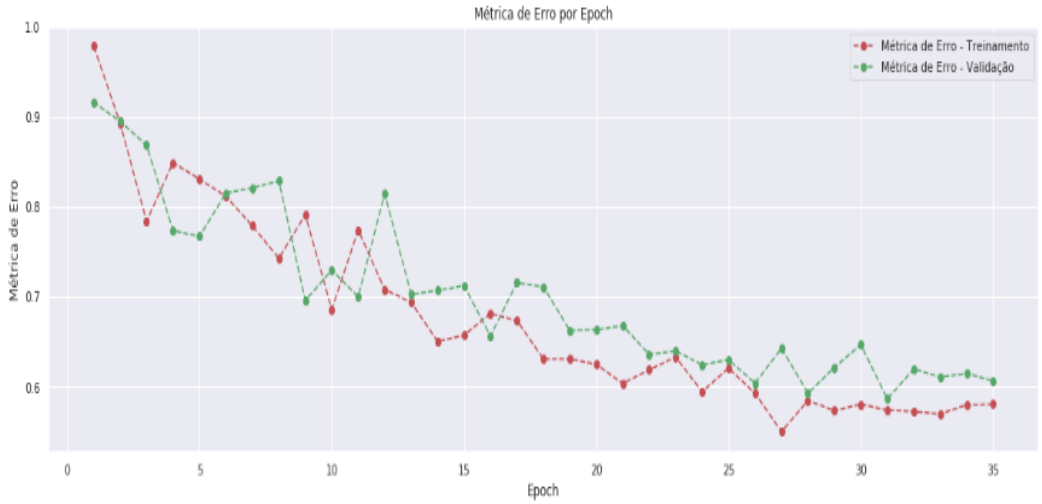

Figura 4.7: Evolução da Métrica de Erro de treinamento e validação durante as épocas

A figura 4.8 apresenta a evolução da métrica da pontuação F1 no conjunto de treinamento e no conjunto de validação, ao longo das *epochs*. Por outro lado, a figura 4.9 apresenta a evolução da métrica de erro para o conjunto de treinamento, alcançada pela melhor seleção de hiper-parâmetros, para cada

um dos três diferentes algoritmos de otimização avaliados. Já a figura 4.10 apresenta a evolução da métrica de erro para o conjunto de treinamento, considerando-se os hiper-parâmetros ótimos apresentados na tabela 3.4, para as três opções de números de camadas testadas (17, 34 e 50).

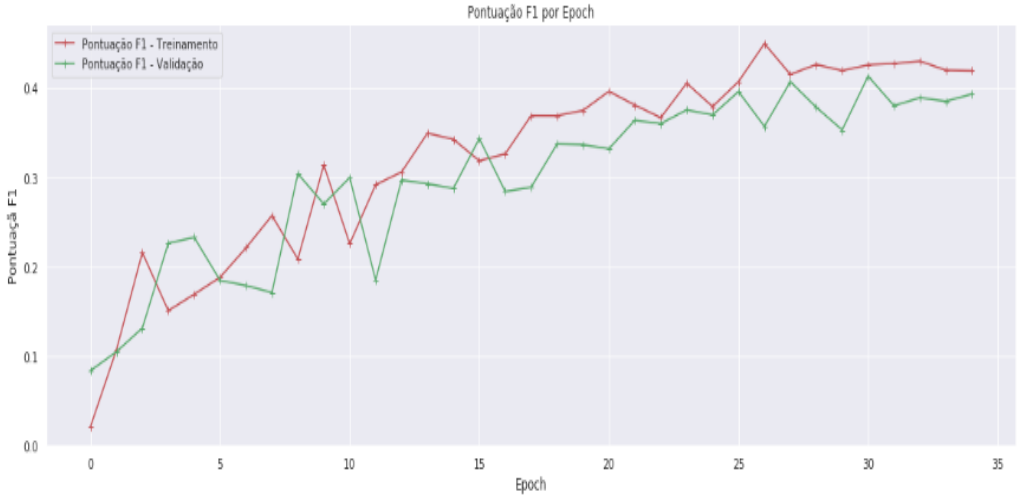

Figura 4.8: Evolução da pontuação F1 dos conjuntos de treinamento e validação ao longo das épocas

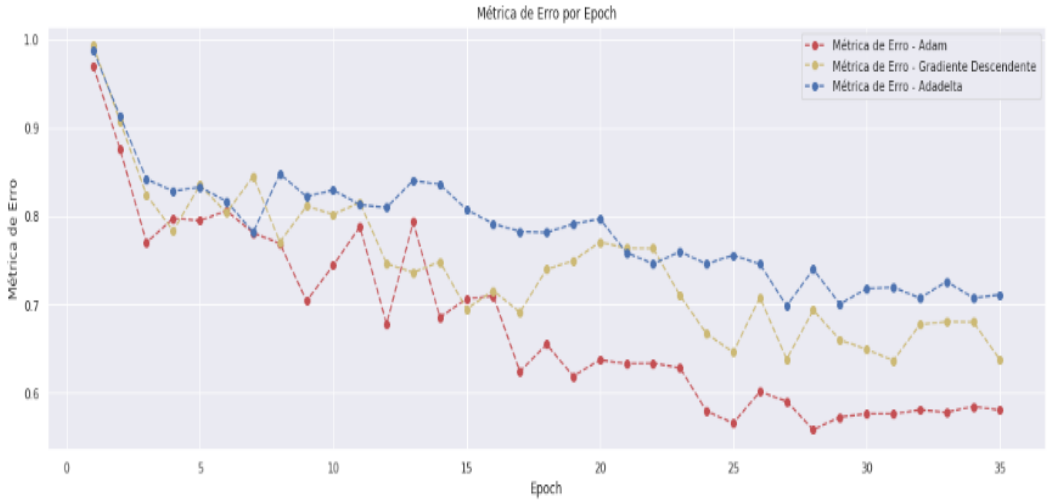

Figura 4.9: Evolução da métrica de erro para diferentes algoritmos de otimização

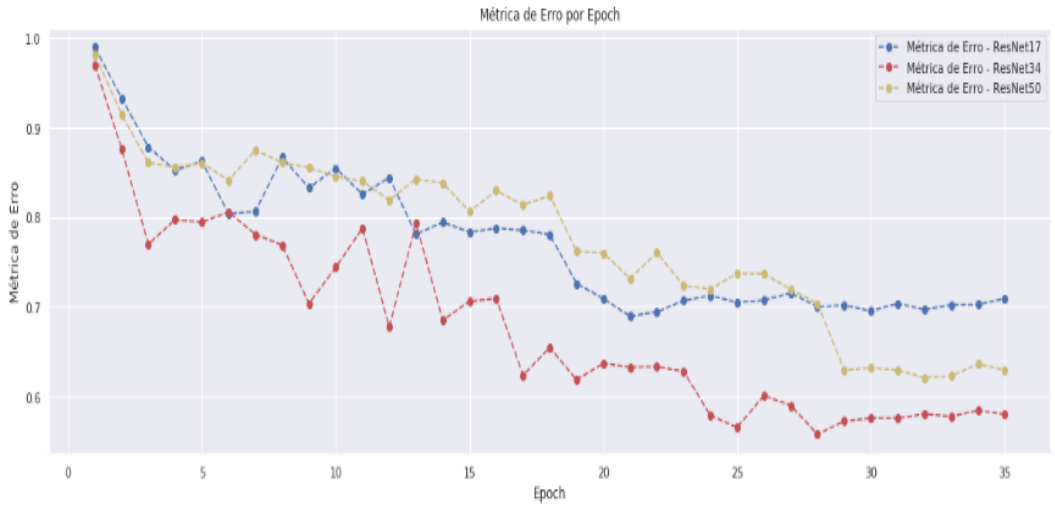

Figura 4.10: Evolução da métrica de erro para diferentes profundidades da rede

4.2.2 Comentários

Analisando-se primeiramente as métricas de desempenho da rede apresentadas na tabela 4.2, nota-se que a acurácia melhora ligeiramente em relação ao modelo LeNet-5, entretanto, o resultado que mais chama a atenção é a grande melhora na pontuação F1, que aumentou de 0,2266 para 0,3935. Conforme discutido anteriormente, a acurácia não é a melhor métrica de desempenho para o conjunto de dados avaliado, podendo levar a conclusões errôneas. Por outro lado, a pontuação F1 é uma métrica mais relevantes para avaliação da real qualidade dos resultados. Dessa forma, o substancial incremento da mesma dá indícios de que o desempenho da ResNet-34 é significativamente superior ao da LeNet-5 na tarefa de classificação.

Por sua vez, por meio das figuras 4.7 e 4.8 observa-se que, novamente, o modelo não teve problemas de *overfitting*, fato aqui diretamente atribuído à utilização da regularização *Dropout*, utilizada no modelo anterior e mantida para este modelo.

Também é possível observar nas mesmas imagens que o desempenho do modelo aparenta convergir por volta da *epoch* 28, visto que a métrica de erro e a pontuação F1 passam a oscilar em torno de um dado valor ao longo das *epochs* seguintes, tanto para o conjunto de treinamento quanto para o de validação. Este comportamento indica uma possível saturação da capacidade de aprendizado desta topologia, para o conjunto de hiperparâmetros selecionado.

Analisando a figura 4.9, observa-se a diferença entre o comportamento do treinamento para diferentes algoritmos de otimização, mantendo-se todos os demais hiper-parâmetros fixos. Nota-se que o algoritmo Adam foi capaz de atingir o melhor desempenho no treinamento, dentre os algoritmos avaliados, para a quantidade de *epochs* escolhida, tendo estabilizado a métrica de erro por volta da *epoch* 28.

O algoritmo de Gradiente Descendente, por outro lado, demonstra forte oscilação na métrica de erro em suas últimas etapas de treinamento, não tendo atingido ainda necessariamente a convergência.

Já o algoritmo *Adadelta* foi aquele com menos oscilação e treinamento mais lento. Apesar disso, neste caso, observam-se oscilações na métrica de erro superiores às verificadas para a topologia LeNet-5. Este fato é possivelmente causado pela mudança na métrica de erro, da função *binary cross entropy* para a alternativa a pontuação F1.

É importante ressaltar também que o algoritmo *Adadelta* costuma causar um treinamento mais lento, devido ao seu comportamento adaptativo. É possível concluir que melhorias futuras poderão ser alcançadas por meio do aumento do número de *epochs* para o *Adadelta* e o Gradiente Descendente, buscando avaliar se estes conseguiriam chegar a valores ainda menores para a métrica de erro, caso tivessem mais tempo de treinamento.

Já no gráfico da figura 4.10, é possível observar as variações no comportamento da rede ao se utilizar diferentes profundidades.

Observa-se que a métrica de erro para a rede de 17 camadas, a menor profundidade avaliada, converge rapidamente, indicando uma saturação na capacidade de aprendizado dessa rede, ao menos para este tamanho de conjunto de treinamento e hiper-parâmetros definidos.

Por sua vez, apesar de ser possível observar que a rede com 34 camadas obteve o melhor desempenho ao fim do treinamento, é difícil garantir que a rede de 50 camadas, a maior delas, tenha efetivamente convergido com 35 *epochs*, visto que a mesma apresentou uma grande variação na métrica de erro por volta da *epoch* 29. Isto é um indicativo de que um número maior de *epochs* poderia propiciar um desempenho ainda melhor para a rede de 50 camadas.

4.3 Modificações na ResNet Ótima

4.3.1
Resultados Topologia Híbrida *ResNet*-SVM

A tabela 4.3 apresenta as métricas de desempenho calculadas para os conjuntos de treinamento e validação.

Tabela 4.3: Métricas de Desempenho da Rede

Métrica	**Treinamento Final**	**Validação Final**
Acurácia	0,9340	0,9281
Adaptação de F1 (erro)	0,5741	0,5977
Pontuação F1	0,4259	0,4023

4.3.2
Resultados Melhorias na *ResNet* Ótima

A tabela 4.4 apresenta as métricas de desempenho calculadas para os conjuntos de treinamento, validação e teste. Como esta foi a topologia que apresentou melhor resultado, ao comparar- se o desempenho dos modelos frente ao conjunto de validação, utilizou-se o conjunto de teste para avaliar seu desempenho ao fim do trabalho.

Tabela 4.4: Métricas de Desempenho da Rede

Métrica	**Treinamento Final**	**Validação Final**	**Teste**
Acurácia	0,9601	0,9493	0,9396
Adaptação de F1 (erro)	0,4879	0,5024	0,5621
Pontuação F1	0,5121	0,4976	0,4379

4.3.3
Comentários

Não foi realizada a busca de hiper-parâmetros nestas duas últimas topologias, pois elas buscavam apenas trazer direcionamentos para estudos futuros afim de melhorar ainda mais o desempenho do modelo, ou tentar sugerir técnicas alternativas.

Conforme explicitado na tabela 4.3, na primeira abordagem, com o uso das SVMs como uma espécie de última camada da rede neural, obtiveram-se resultados ligeiramente melhores, porém muito próximo, aos alcançados pela ResNet-34, discutida na seção 4.2.

Dessa forma, não é possível afirmar com segurança que a adição deste algoritmo, como alternativa à camada de saída, tenha trazido benefícios ao modelo. Entretanto, os resultados deixam em aberto a possibilidade de, com

os ajustes adequados, a adição do SVM possibilitar efetivas melhorarias no desempenho da rede.

Ressalta-se também que não houve uma busca de hiper-parâmetros ótimos para a SVM, outro indicativo de que ainda há grande espaço para melhora do desempenho da rede.

Na segunda abordagem, duas modificações foram feitas:

– aumento da resolução da imagem de entrada da rede; e

– uso de barreiras de decisão.

Estas técnicas possibilitaram consideráveis melhorias no desempenho da rede, tendo sido esta a rede que obteve o melhor desempenho frente ao conjunto de validação – acurácia de 0,9493 e pontuação F1 de 0,4976.

É difícil afirmar qual das duas modificações foi a principal responsável pela melhoria do desempenho do modelo. Há forte indicação de que parte dessa melhora se deve ao aumento da resolução da imagem apresentada a camada de entrada da rede, visto que percebem-se grandes diferenças ao se analisar a olho nu as imagens de qualidades diferentes, conforme mostrado na figura 3.16. Conforme mencionado, aparenta-se ter uma grande perda de informação nas imagens de menor resolução, o que pode impedir a correta detecção de determinadas classes. Além disso, as barreiras de decisão foram adicionadas a uma parte pequena das classes, aquelas menos representativas do conjunto de dados, o que também pode ter contribuído para a melhoria do desempenho, porém, acredita-se que em menor escala.

Vale ressaltar também que, muito provavelmente, os hiper-parâmetros ótimos encontrados para a topologia ResNet-34, discutida na seção 4.2, não serão os ideias para essa versão modificada da ResNet-34. Em outras palavras, destaca-se que é possível que uma nova busca de hiper-parâmetros permita um resultado ainda melhor para o modelo.

5
Conclusão e Trabalhos Futuros

Este trabalho desenvolveu um sistema inteligente, baseado em técnicas de *Deep Learning*, para classificação automática de padrões mistos da localização subcelular de proteínas.

Utilizou-se uma base de dados pública do *Human Protein Atlas* para treinar diversas topologias de rede, de modo a se poder comparar seus desempenhos, buscando identificar aquela capaz de alcançar o melhor desempenho na tarefa de classificação multi-classe aqui abordada.

No total, foram analisadas 81 redes ao longo do desenvolvimento deste trabalho, tendo sido:

- 1 *LeNet*-5;
- 78 *ResNets*;
- 1 *ResNet*-34 + SVM;
- 1 *ResNet*-34 com novas modificações.

Os estudos foram iniciados com o emprego de um modelo base, que já é uma topologia tradicional na comunidade de redes neurais convolucionais, a *LeNet*-5. Por ser uma topologia simples, foi possível treinar esta rede rapidamente. Além disso, para este modelo base, utilizou-se apenas um canal da imagem de entrada, buscando também simplicidade e velocidade no treinamento do modelo.

A partir da análise dos resultados obtidos por este modelo, foi possível identificar possíveis problemas que poderiam comprometer o desempenho de de modelos mais eficazes. Em particular, a análise dos resultados levou aos seguintes direcionamentos:

- a regularização funciona bem e é necessária;
- o algoritmo *Adadelta* teve um treinamento lento, de modo que seria importante avaliar novos algoritmos de treinamento;
- a métrica de erro *Binary Cross Entropy* não é uma boa função para otimizar o classificador em relação à pontuação F1;

– a possibilidade de implementação de diferentes barreiras de decisão para diferentes classes, pode ser útil para a correta classificação das classes menos representativas do conjunto de dados.

Com estes resultados em mente, iniciou-se o desenvolvimento de um modelo mais avançado, bastante usado pela comunidade científica para classificação de imagens, a *ResNet*.

Objetivando encontrar a versão ótima da *ResNet* para o problema em questão, realizou-se uma busca de hiper-parâmetros extensa, cobrindo um total de 78 redes. Foram testadas novas métricas de erro e novos algoritmos de otimização. Além da busca pela melhor combinação de hiper-parâmetros, destaca-se que neste modelo foram utilizados efetivamente todos os canais de cores das imagens de entrada, além da criação de terem sido sintetizadas artificialmente novas imagens para as classes menos representativas do conjunto de treinamento, de modo a se tornar a distribuição das classes mais homogênea. Os resultados alcançados permitem concluir que, ao fim da busca de hiper-parâmetros, a rede ótima encontrada teve desempenho consideravelmente melhor do que a *LeNet*-5, ao se comparar os desempenhos obtidos para o conjunto de validação.

Uma vez definido o modelo ótimo de *ResNet*, duas outras topologias foram treinadas, buscando identificar potenciais caminhos para melhorias a serem implementadas por futuros estudos.

Na primeira versão, manteve-se a rede ótima encontrada e seus pesos, substituindo-se apenas sua última camada por SVMs. Para este arranjo, obteve-se um resultado muito próximo ao da topologia anterior, embora ligeiramente superior. Dessa forma, não é possível afirmar que a pequena melhoria observada seja efetivamente decorrente da adição das SVMs.

Por sua vez, na segunda versão, aumentou-se a resolução da imagem de entrada da rede, além de terem sido implementadas diferentes barreiras de decisão para as classes menos representativas do conjunto. Para esta nova configuração, obteve-se um desempenho substancialmente superior, tendo sido esta a configuração de rede a alcançar o melhor desempenho, dentre todas aquelas avaliadas nesse trabalho.

A tabela 5.1 apresenta os resultados, para os dados do conjunto de validação, dos modelos: *LeNet*-5, *ResNet*-34 (a *ResNet* ótima decorrente da iteração dos hiper-parâmetros), *ResNet*-SVM e *ResNet*-34 com modificações.

O estado da arte para a tarefa de classificação discutida no presente trabalho foi estabelecido em [16], que obteve uma pontuação F1 de 0,72, para o mesmo banco de dados no conjunto de testes, frente a uma pontuação de 0,44 obtida neste trabalho. A princípio, a diferença entre as pontuações F1

Tabela 5.1: Comparação dos Desempenhos das Redes

Métrica	***LeNet*-5**	***ResNet*-34**	***ResNet*-SVM**	***ResNet*-34 Modificada**
Acurácia	0,9201	0,9248	0,9281	0,9493
Pontuação F1	0,2266	0,3935	0,4023	0,4976

alcançadas é principalmente explicada pelas resoluções das imagens utilizadas como entradas da rede. O trabalho publicado na Nature utilizou imagens de 3072x3072 pixels, muito mais detalhadas que as imagens 512x512 deste trabalho, que acabaram sendo ainda mais reduzidas, para 256x256 pixels.

5.1 Trabalhos Futuros

A fim de melhorar o desempenho alcançado pelo modelo aqui desenvolvido, frente aos resultados obtidos ao longo deste trabalho foram identificados alguns possíveis caminhos para trabalhos futuros que podem ser realizados neste intuito.

5.1.1 Aumentar o Número de *Epochs*

Os resultados dos treinamentos feitos neste trabalho não permitem que concluir se o número estabelecido de 35 *epochs* para o treinamento é, de fato, suficiente para garantir a convergência do aprendizado da rede.

Principalmente para as redes de 50 camadas, há indícios de que o desempenho delas pode ser melhorado ao se aumentar a duração do treinamento.

5.1.2 Aumento da Resolução das Imagens de Entrada

Ao aumentar a resolução das imagens de entrada das redes de 128x128 para 256x256 foi possível identificar uma acentuada melhoria de desempenho. Dessa forma, uma possibilidade para melhorar ainda mais o desempenho do modelo é continuar a aumentar a resolução das mesmas, passando a, por exemplo, utilizar imagens de entrada com 512x512 pixels.

5.1.3 Busca de Hiper-parâmetros

As duas últimas topologias aqui propostas não tiveram seus hiper-parâmetros iterados, de modo a se avaliar qual combinação dos mesmos é responsável por maximizar o desempenho. Consequentemente, muito provavelmente seria possível obter resultados ainda melhores para estas redes ao se fazer esta busca.

5.1.4
Transfer Learning

Existem muitos modelos de redes neurais pré-treinadas disponibilizadas pela comunidade científica. Uma opção seria utilizar uma rede pré- treinada para problemas de classificação de padrões mistos, não necessariamente de localização subcelular de proteínas, e retreinar apenas as últimas camadas desta rede. Dessa forma, pode-se acelerar significativamente o tempo e esforço computacional demandados pelo treinamento.

5.1.5
Múltiplas Redes em Paralelo

Outra proposta de trabalho futuro é a utilização de redes separadas para classes específicas. Como visto no capítulo 2, as diferentes cores apresentam diferentes características.

Dessa forma, poder-se-ia usar, por exemplo, uma rede que recebesse apenas o filtro de cor amarela da imagem para classificar apenas se há presença da classe retículo endoplasmático e assim em diante. Por ser uma classificação binária, o problema passa inclusive a ser mais simples e a métrica de erro original de *Binary Cross Entropy* tornaria-se adequada para o problema em questão.

5.1.6
Escolha Ótima dos Valores das Barreiras de Decisão

Uma última proposta de trabalho futuro seria aprofundar a análise e seleção das barreiras de decisão utilizadas no fim do algoritmo. Na última rede desenvolvida neste trabalho, foram selecionados valores de barreira de decisão de forma visual, sem uma análise mais profunda para seleção ótima. É possível que hajam valores que decidam com melhor desempenho a presença daquelas classes menos frequentes.

Referências bibliográficas

[1] VOET, D.; VOET, J. G.. **Biochemistry, Binder Ready Version**. Wiley, 2010.

[2] BROSNAN, J. T.. **Interorgan Amino Acid Transport and its Regulation**. The Journal of Nutrition, 133(6):2068S–2072S, 06 2003.

[3] GILLILAND, G.; BERMAN, H. M.; WEISSIG, H.; SHINDYALOV, I. N.; WESTBROOK, J.; BOURNE, P. E.; BHAT, T. N. ; FENG, Z.. **The Protein Data Bank**. Nucleic Acids Research, 28(1):235–242, 01 2000.

[4] UHLÉN, M.; FAGERBERG, L.; HALLSTRÖM, B. M.; LINDSKOG, C.; OKSVOLD, P.; MARDINOGLU, A.; SIVERTSSON, Å.; KAMPF, C.; SJÖSTEDT, E.; ASPLUND, A.; OLSSON, I.; EDLUND, K.; LUNDBERG, E.; NAVANI, S.; SZIGYARTO, C. A.-K.; ODEBERG, J.; DJUREINOVIC, D.; TAKANEN, J. O.; HOBER, S.; ALM, T.; EDQVIST, P.-H.; BERLING, H.; TEGEL, H.; MULDER, J.; ROCKBERG, J.; NILSSON, P.; SCHWENK, J. M.; HAMSTEN, M.; VON FEILITZEN, K.; FORSBERG, M.; PERSSON, L.; JOHANSSON, F.; ZWAHLEN, M.; VON HEIJNE, G.; NIELSEN, J. ; PONTÉN, F.. **Tissue-based map of the human proteome**. Science, 347(6220), 2015.

[5] MARCHLER-BAUER, A.; SHOEMAKER, B. A.; DEWEESE-SCOTT, C.; LANCZYCKI, C. J.; LIU, C.; LIEBERT, C. A.; ZHANG, D.; HURWITZ, D. I.; LU, F.; MARCHLER, G. H.; SONG, J. S.; YIN, J. J.; ANDERSON, J. B.; JACKSON, J. D.; GEER, L. Y.; GWADZ, M.; MULLOKANDOV, M.; THIESSEN, P. A.; CHERUKURI, P. F.; YAMASHITA, R. A.; HE, S.; BRYANT, S. H.; SIMONYAN, V. ; KE, Z.. **CDD: a Conserved Domain Database for protein classification**. Nucleic Acids Research, 33(suppl1):D192–D196, 01 2005.

[6] UHLÉN, M.; BJÖRLING, E.; AGATON, C.; SZIGYARTO, C. A.-K.; AMINI, B.; ANDERSEN, E.; ANDERSSON, A.-C.; ANGELIDOU, P.; ASPLUND, A.; ASPLUND, C.; BERGLUND, L.; BERGSTRÖM, K.; BRUMER, H.; CERJAN, D.; EKSTRÖM, M.; ELOBEID, A.; ERIKSSON, C.; FAGERBERG, L.; FALK, R.; FALL, J.; FORSBERG, M.; BJÖRKLUND, M. G.; GUMBEL, K.; HALIMI, A.; HALLIN, I.; HAMSTEN, C.; HANSSON, M.; HEDHAMMAR, M.; HERCULES, G.; KAMPF, C.; LARSSON, K.; LINDSKOG, M.; LODEWYCKX,

W.; LUND, J.; LUNDEBERG, J.; MAGNUSSON, K.; MALM, E.; NILSSON, P.; ÖDLING, J.; OKSVOLD, P.; OLSSON, I.; ÖSTER, E.; OTTOSSON, J.; PAAVILAINEN, L.; PERSSON, A.; RIMINI, R.; ROCKBERG, J.; RUNESON, M.; SIVERTSSON, Å.; SKÖLLERMO, A.; STEEN, J.; STENVALL, M.; STERKY, F.; STRÖMBERG, S.; SUNDBERG, M.; TEGEL, H.; TOURLE, S.; WAHLUND, E.; WALDÉN, A.; WAN, J.; WERNÉRUS, H.; WESTBERG, J.; WESTER, K.; WRETHAGEN, U.; XU, L. L.; HOBER, S. ; PONTÉN, F.. **A human protein atlas for normal and cancer tissues based on antibody proteomics**. Molecular & Cellular Proteomics, 4(12):1920–1932, 2005.

[7] UHLEN, M.; OKSVOLD, P.; FAGERBERG, L.; LUNDBERG, E.; VON FEILITZEN, K.; FORSBERG, M.; ZWAHLEN, M.; KAMPF, C.; WESTER, K.; HOBER, S.; WERNÉRUS, H.; BJÖRLING, L. ; PONTÉN, F.. **Towards a knowledge-based human protein atlas**. Nature biotechnology, 28:1248–50, 12 2010.

[8] WU, C.; WHITSON, G.; MCLARTY, J.; ERMONGKONCHAI, A. ; CHANG, T.-C.. **Protein classification artificial neural system**. Protein Science, 1(5):667–677, 1992.

[9] SCHÖLKOPF, B.; SHIN, H. ; TSUDA, K.. **Fast protein classification with multiple networks**. Bioinformatics, 21(suppl2):ii59–ii65, 09 2005.

[10] SWAMIDOSS, I.; PONNUSAMY, P.; SIMONSSON, M.; STRAND, R.; UHLMANN, V.; WÄHLBY, C.; KAMPF, C. ; KÅRSNÄS, A.. **Automated classification of immunostaining patterns in breast tissue from the human protein atlas**. Journal of Pathology Informatics, 4(2):14, 2013.

[11] LITJENS, G.; KOOI, T.; BEJNORDI, B. E.; SETIO, A. A. A.; CIOMPI, F.; GHAFOORIAN, M.; VAN DER LAAK, J. A.; VAN GINNEKEN, B. ; SÁNCHEZ, C. I.. **A survey on deep learning in medical image analysis**. Medical Image Analysis, 42:60–88, Dec. 2017.

[12] DIPLARIS, S.; TSOUMAKAS, G.; MITKAS, P. A. ; VLAHAVAS, I.. **Protein classification with multiple algorithms**. In: ADVANCES IN INFORMATICS, p. 448–456. Springer Berlin Heidelberg, 2005.

[13] BOLAND, M. V.; MURPHY, R. F.. **A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells** . Bioinformatics, 17(12):1213–1223, 12 2001.

[14] XIAO, X.; SHAO, S.; DING, Y.; HUANG, Z. ; CHOU, K.-C.. **Using cellular automata images and pseudo amino acid composition to predict protein subcellular location**. Amino Acids, 30(1):49–54, July 2005.

[15] TAHIR, M.; KHAN, A.. **Protein subcellular localization of fluorescence microscopy images: Employing new statistical and texton based image features and SVM based ensemble classification**. Information Sciences, 345:65–80, June 2016.

[16] SULLIVAN, D. P.; WINSNES, C. F.; ÅKESSON, L.; HJELMARE, M.; WIKING, M.; SCHUTTEN, R.; CAMPBELL, L.; LEIFSSON, H.; RHODES, S.; NORDGREN, A.; SMITH, K.; REVAZ, B.; FINNBOGASON, B.; SZANTNER, A. ; LUNDBERG, E.. **Deep learning is combined with massive-scale citizen science to improve large-scale image classification**. Nature Biotechnology, 36(9):820–828, Aug. 2018.

[17] THUL, P. J.; ÅKESSON, L.; WIKING, M.; MAHDESSIAN, D.; GELADAKI, A.; AIT BLAL, H.; ALM, T.; ASPLUND, A.; BJÖRK, L.; BRECKELS, L. M.; BÄCKSTRÖM, A.; DANIELSSON, F.; FAGERBERG, L.; FALL, J.; GATTO, L.; GNANN, C.; HOBER, S.; HJELMARE, M.; JOHANSSON, F.; LEE, S.; LINDSKOG, C.; MULDER, J.; MULVEY, C. M.; NILSSON, P.; OKSVOLD, P.; ROCKBERG, J.; SCHUTTEN, R.; SCHWENK, J. M.; SIVERTSSON, Å.; SJÖSTEDT, E.; SKOGS, M.; STADLER, C.; SULLIVAN, D. P.; TEGEL, H.; WINSNES, C.; ZHANG, C.; ZWAHLEN, M.; MARDINOGLU, A.; PONTÉN, F.; VON FEILITZEN, K.; LILLEY, K. S.; UHLÉN, M. ; LUNDBERG, E.. **A subcellular map of the human proteome**. Science, 356(6340), 2017.

[18] VAPNIK, V.. **Estimation of Dependences Based on Empirical Data: Empirical Inference Science (Information Science and Statistics)**. Springer, 2006.

[19] BURGES, C. J.. Data Mining and Knowledge Discovery, 2(2):121–167, 1998.

[20] JAMES, G.; WITTEN, D.; HASTIE, T. ; TIBSHIRANI, R.. **An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)**. Springer, 2017.

[21] JOACHIMS, T.. **Text categorization with support vector machines: Learning with many relevant features**. In: MACHINE LEARNING: ECML-98, p. 137–142. Springer Berlin Heidelberg, 1998.

[22] LAUER, F.; SUEN, C. Y. ; BLOCH, G.. **A trainable feature extractor for handwritten digit recognition**. Pattern Recognition, 40(6):1816–1824, June 2007.

[23] HEARST, M.; DUMAIS, S.; OSUNA, E.; PLATT, J. ; SCHOLKOPF, B.. **Support vector machines**. IEEE Intelligent Systems and their Applications, 13(4):18–28, July 1998.

[24] BOSER, B. E.; GUYON, I. M. ; VAPNIK, V. N.. **A training algorithm for optimal margin classifiers**. In: PROCEEDINGS OF THE FIFTH ANNUAL WORKSHOP ON COMPUTATIONAL LEARNING THEORY - COLT '92. ACM Press, 1992.

[25] AIZERMAN, M. A.; BRAVERMAN, E. A. ; ROZONOER, L.. **Theoretical foundations of the potential function method in pattern recognition learning.** In: AUTOMATION AND REMOTE CONTROL„ número 25 em Automation and Remote Control„ p. 821–837, 1964.

[26] MCCULLOCH, W. S.; PITTS, W.. **A logical calculus of the ideas immanent in nervous activity**. The Bulletin of Mathematical Biophysics, 5(4):115–133, Dec. 1943.

[27] GOODFELLOW, I.; BENGIO, Y. ; COURVILLE, A.. **Deep Learning**. MIT Press, 2016. `http://www.deeplearningbook.org`.

[28] ROSENBLATT, F.. **The perceptron: A probabilistic model for information storage and organization in the brain.** Psychological Review, 65(6):386–408, 1958.

[29] MINSKY, M. L.. **Perceptrons: An Introduction to Computational Geometry**. MIT Press, 1969.

[30] RUMELHART, D. E.; HINTON, G. E. ; WILLIAMS, R. J.. **Learning representations by back-propagating errors**. Nature, 323(6088):533–536, Oct. 1986.

[31] HAYKIN, S.. **Neural Networks: A Comprehensive Foundation**. Macmillan Coll Div, 1994.

[32] LECUN, Y.; BENGIO, Y. ; HINTON, G.. **Deep learning**. Nature, 521(7553):436–444, May 2015.

[33] HINTON, G. E.; OSINDERO, S. ; TEH, Y.-W.. **A fast learning algorithm for deep belief nets**. Neural Computation, 18(7):1527–1554, July 2006.

[34] RAINA, R.; MADHAVAN, A. ; NG, A. Y.. **Large-scale deep unsupervised learning using graphics processors**. In: PROCEEDINGS OF THE 26TH ANNUAL INTERNATIONAL CONFERENCE ON MACHINE LEARNING - ICML '09. ACM Press, 2009.

[35] **Shape, Contour and Grouping in Computer Vision (Lecture Notes in Computer Science)**. Springer, 1999.

[36] KRIZHEVSKY, A.; SUTSKEVER, I. ; HINTON, G. E.. **Imagenet classification with deep convolutional neural networks**. In: PROCEEDINGS OF THE 25TH INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS - VOLUME 1, NIPS'12, p. 1097–1105, USA, 2012. Curran Associates Inc.

[37] LECUN, Y.; BOTTOU, L.; BENGIO, Y. ; HAFFNER, P.. **Gradient-based learning applied to document recognition**. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[38] HE, K.; ZHANG, X.; REN, S. ; SUN, J.. **Deep residual learning for image recognition**. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p. 770–778, 2016.

[39] SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I. ; SALAKHUTDINOV, R.. **Dropout: A simple way to prevent neural networks from overfitting**. J. Mach. Learn. Res., 15(1):1929–1958, Jan. 2014.

[40] ZEILER, M. D.. **ADADELTA: an adaptive learning rate method**. CoRR, abs/1212.5701, 2012.

[41] KINGMA, D. P.; BA, J.. **Adam: A method for stochastic optimization**, 2014.

[42] TANG, Y.. **Deep learning using linear support vector machines**. In: IN ICML, 2013.