Rodrigo Cezar Silveira

Análise de Arquiteturas de Redes Neurais Convolucionais para Classificação de Sinais de Trânsito

Formiga - MG

2019

Rodrigo Cezar Silveira

Análise de Arquiteturas de Redes Neurais Convolucionais para Classificação de Sinais de Trânsito

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais

Campus Formiga

Ciência da Computação

Orientador: Everthon Valadão dos Santos

Formiga - MG

2019

Silveira, Rodrigo Cezar.
004 Análise de Arquiteturas de Redes Neurais Convolucionais para Classificação de Sinais de Trânsito / Rodrigo Cézar Silveira. -- Formiga : IFMG, 2019.
87p. : il.

Orientador: Prof. Msc. Everthon Valadão dos Santos
Trabalho de Conclusão de Curso – Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Inteligência Artificial. 2. Redes Neurais Convolucionais. 3 . Classificação de Sinais de Trânsito. 4. Arquiteturas de Redes Neurais Convolucionais. 5. Campeonatos de Classificação de Sinais e Trânsito. I. Título.

CDD 004

Ficha catalográfica elaborada pela Bibliotecária Msc. Naliana Dias Leandro CRB6-1347

Rodrigo Cezar Silveira

Análise de Arquiteturas de Redes Neurais Convolucionais para Classificação de Sinais de Trânsito

Monografia do trabalho de conclusão de curso apresentado ao Instituto Federal Minas Gerais - Campus Formiga, como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado em 14 de junho de 2019.

BANCA EXAMINADORA

M.e Everthon Valadão dos Santos
Professor no IFMG *Campus* Formiga
(Orientador)

M.e Diego Mello da Silva
Professor no IFMG *Campus* Formiga

M.e Fernando Paim Lima
Professor no IFMG *Campus* Formiga

Formiga - MG
2019

Por todas as ações que nos precederam, as que acompanham nosso tempo, e aquelas que ainda estão por vir... dedico esta obra a nossa alma e corações e aos esforços que prosseguem a marcha da evolução pelo nosso bem e o de toda a humanidade.

Agradecimentos

De forma alguma eu deveria aqui mencionar aqueles sobre os quais recai minha gratidão, pois eles já bem a conhecem ainda que eu não a declare publicamente. De mim, tendes não apenas gratidão, mas todo o amor com o qual guardo vosso nome reverentemente.

Do mais, agradeço a todos os colegas, técnicos administrativos e professores os quais estiveram junto a mim durante esta jornada, destas interações, recebemos não apenas o ensinamento que nos formara profisionais, mas valiosas lições e verdadeiros exemplos de vida que nos acompanharão pelo decorrer de toda a vida.

Agradeço ainda, a todos aqueles que de alguma forma colaboraram para a conclusão deste trabalho, especialmente ao meu orientador, pela apoio, atenção, paciência e confiança dedicados a mim durante todo este processo.

“ As letras e a ciência só tomarão o seu verdadeiro lugar na obra do desenvolvimento humano no dia em que, livres de toda a servidão mercenária, forem exclusivamente cultivadas pelos que as amam e para os que as amam. ” (Piotr Kropotkin)

Resumo

O presente trabalho inspirou-se sobre alguns resultados obtidos durante campeonatos de reconhecimento de sinais de trânsito ocorridos ao redor do mundo, dentre eles, o campeonato alemão e o campeonato belga, os quais evidenciam o sucesso das redes neurais convolucionais para o reconhecimento de imagens. Em uma primeira abordagem, teve-se a intenção de explorar a forma como a arquitetura de uma rede neural convolucional afeta seu desempenho, e para isso, desenvolveu-se uma série de arquiteturas baseadas em diferentes parâmetros de configuração topológica; o experimento dividiu-se em quatro etapas, cada qual explora um determinado elemento da rede: camada convolucional, camada totalmente conectada, características das imagens de entrada, desempenho sobre outros datasets. Como contribuição, este trabalho demonstrou como foi possível obter uma acurácia de 100% e um desempenho de 99,97% sobre o *German Traffic Sign Recognition*, além de demonstrar que as dimensões das máscaras de convolução de uma rede neural convolucional podem ser maiores do que as dimensões da própria imagem de entrada, tendo inclusive, produzido modelos de arquitetura com os quais se obteve bons resultados.

Palavras-chave: Inteligência Artificial, Redes Neurais Convolucionais, Classificação de Sinais de Trânsito.

Abstract

The present work was inspired by some results obtained during traffic sign recognition championships that took place around the world, including the German championship and the Belgian championship, which highlight the success of convolutional neural networks for image recognition. First, it was intended to explore how the architecture of a convolutional neural network affects its performance, therefore a series of architectures based on different parameters of topological configuration was developed; the experiment was divided into four stages, each of which explores a particular network element: convolutional layer, fully connected layer, characteristics of input images, performance over other datasets. As a contribution, this work demonstrated how it was possible to obtain 100% accuracy and a 99.97% performance on the German Traffic Sign Recognition, in addition, it demonstrates the fact that the dimensions of CNN's convolutional masks may be larger than the dimensions of the input image itself, having even produced architectural models by which good results have been obtained.

Keywords: Artificial intelligence, Convolutional Neural Networks, Traffic Sign Recognition.

Sumário

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABS	*Anti-lock Braking System*
IA	*Inteligência Artificial*
TSR	*Traffic Sign Recognition*
CNN	*Convolutional Neural Network*
ANN	*Artificial Neural Network*
TDS	*Traffic Sign*
R-CNN	*Region Based CNN*
YOLO	*You Only Look Once*
SSP	*Spatial Pyramid pooling*
ML	*MachineLearning*
HoG	*Histogram of Gradients*
ROI	*Region of Interest*
SVM	*Support Vector Machine*
UPNs	*Unsupervised Pretrained Network*
ReLU	*rectified-linear-units*
RPM	*Region proposal methods*
RPN	*Region Proposal Network*

1 Introdução

Avanços no campo da inteligência artificial (IA) possibilitaram a automação e o desenvolvimento de tecnologias assistivas, capazes de auxiliar ou até mesmo substituir as ações humanas nos mais diversos setores e atividades, como os sistemas especialistas no auxílio de diagnósticos médicos (RUTA; LI; LIU, 2010), lógica *fuzzy* para o monitoramento e tomada de decisões, geoprocessamento, redes neurais aplicadas à sistemas de concessão de créditos (MARCOS; RIBEIRO; MERLO, 2005), dentre outros.

Dentre tais aplicações, a indústria automotiva presencia enormes contribuições deste campo, sobre a qual este último tem desempenhado papel de suma importância, desenvolvendo-se conforme a evolução do próprio automóvel, de forma que, muitos recursos computacionais estão presentes em um veículo moderno, desde a central de comandos e injeção eletrônica, unidades as quais, além de manterem o funcionamento adequado do motor, utilizam dados como nível de oxigênio, gás carbônico e posição do pedal para determinar o volume de combustível a ser lançado na câmara de combustão, sistema de freios *Anti-lock Braking System* (ABS), sistema automáticos de diagnóstico, até modo de navegação, reconhecimento de voz e de gestos, sistema de assistência a estacionamento, integração veículo/dispositivos móveis e, por último, o carro autônomo surge como um maior expoente, elevando o emprego de dispositivos computacionais e inteligência artificial a nível de tecnologia de ponta (PURDY, 2017; GADAM, 2018)

Deste fato, infere-se que a IA torna-se um ponto crucial para o desenvolvimento do veículo autônomo, haja em vista a necessidade de atribuir-lhe a capacidade de tomada de decisões automáticas, assim reduzindo sua dependência de um intermediário humano; portanto dotá-lo de um sistema computacional integrado capaz de corretamente perceber e interpretar o ambiente em seu entorno e com base nestes critérios tomar decisões relativas ao conjunto de ações as quais ele deve desempenhar em relação ao seu percurso e à via, é requisito fundamental para atribuir-lhe tal nível de autonomia.

Para tanto, faz-se necessário o emprego de técnicas de visão computacional, esta última, definida como o campo da Inteligência Artificial responsável por simular a capacidade de percepção do mundo físico através de imagens, característica típica dos entes biológicos, por intermédio de dispositivo computacionais, através de um conjunto de técnicas para a extração automatizada de características de imagens, empregando programação e modelagem matemática e estatística (SOLEM, 2012). Neste sentido, para fins deste trabalho, considera-se reconhecimento de sinais de trânsito (*Traffic Sign Recognition* TSR) um problema pertencente à recognição de imagens, sub-área da visão computacional, a qual tem por objetivo o desenvolvimento de sistemas inteligentes capazes de automatica-

mente reconhecer, distinguir e interpretar imagens contendo diferentes tipos de elementos sinalizadores comumente encontrados em vias de tráfego, tais como placas, semáforos, faixas de pedestre, faixas de pista, dentre outros.

Segundo Salti et. al. (SALTI et al., 2015), o estudo das técnicas de TSR dividide-se entre problemas de detecção e classificação, sendo os primeiros equivalentes à correta localização de um sinal de trânsito em uma imagem arbitrária; enquanto os últimos objetivam-se a determinar uma rotulação apropriada da área detectada, segundo tipos e categorias específicas bem definidas. Em (MATHIAS et al., 2013) os autores conduzem seu trabalho demonstrando como foi possível, utilizando os métodos presentes na época, alcançar um nível de acurácia em torno de 95% e 99% em problemas relativos à detecção e classificação de pedestres, escrita manual e reconhecimento facial.

Considerando os avanços no arsenal de técnicas atualmente disponíveis, seria possível conseguir desempenhos superiores na resolução de problemas de outros domínios; neste trabalho desenvolver-se-á uma abordagem para a classificação de sinais de trânsito, utilizando para tal objetivo, redes neurais artificiais. Espera-se obter um desempenho próximo do ideal (acima de 99,99%), adicionalmente, o grande foco deste trabalho situa-se em torno ao estudo sobre a forma como os elementos, parâmetros e arquitetura de uma rede neural convolucional influenciam seu desempenho.

1.1 Justificativa

Nas últimas décadas, o avanço das redes neurais artificiais têm revolucionado tanto o campo de aplicações científicas quanto mercadológicas, atualmente tornando possível a construção de aplicações as quais mostravam-se impossíveis algumas décadas atrás. O termo *Deep Learning* designa a nova tendência de construção de redes neurais compostas por múltiplas camadas de algoritmos, acopladas ao classificador com o intuíto de extrair diferentes características da informação armazenada em um volume de dados brutos (BROWNLEE, 2016b); esta técnica tem possibilitado o aumento significativo do potencial das redes neurais, estando dela dependente o avanço da inteligência artificial como um todo (FOOTE, 2017). Dentre suas principais contribuições, encontram-se a proposição de quatro variantes das redes neurais artificiais tradicionais, cada qual desenvolvida para lidar com uma classe de problemas específicos de forma eficaz, são elas: *Unsupervised Pretrained Network* (UPNs), *Recurrent Neural Network*, *Recursive Neural Network*, *Convolutional Network* (CNNs) (CHANDRAYAN, 2017a).

Estas últimas destacaram-se como técnicas eficientes em aplicações de classificação de imagens e crescem em importância conforme o aumento de pesquisas relacionadas ao desenvolvimento de veículos autônomos, no que concerne ao reconhecimento e à capacidade de interpretação da via, requisitos necessários para o desenvolvimento de funcionalidades

mais poderosas e inteligentes, como por exemplo, a possibilidade de um veículo capturar imagens do ambiente ao seu entorno, pelo intermédio de uma câmera e, nelas reconhecer diferentes elementos gráficos, como placas, semáforos, marcas de solo, entre outras.

A correta interpretação desses elementos possibilitaria ao automóvel tomar decisões preventivas, como emitir um alerta sonoro para indicar situações específicas, como a presença de uma curva acentuada à direita ou advertir o piloto de que este está trafegando na direção contrária à via; ou corretivas, como parar o carro ao detectar um sinal vermelho adiante ou a presença de pessoas cruzando a faixa de pedestres, por exemplo. Embora se encontre avançado o atual estado da arte, o grande desafio reside na possibilidade de desenvolver um sistema confiável de classificação de imagens capaz de operar em situações reais.

1.2 Objetivos

1.2.1 Objetivo Geral

Destacam-se, dentre os objetivos gerais deste trabalho, a análise da influência dos principais parâmetros e seus impactos sobre o desempenho de uma rede neural convolucional e a identificação e seleção das melhores configurações para a obtenção de um bom desempenho na classificação de placas de *datasets* internacionais.

1.2.2 Objetivos Específicos

1. Estudar os principais parâmetros envolvidos no bom desempenho de uma rede neural convolucional (CNN) durante a classificação de imagens;
2. Construir diferentes arquiteturas de redes neurais e para cada uma delas, aferir o desempenho alcançado conforme a variação de parâmetros predeterminados;
3. Selecionar dentre os melhores resultados, as redes com maior potencial de acurácia;
4. Com base nos modelos anteriormente selecionados, modelar uma rede neural artificial (ANN) para identificar sinais de trânsito;
5. Treinar a ANN com bases de dados de imagens de trânsito, publicamente disponíveis;
6. Alterar a estrutura e os parâmetros desta rede para alcançar a maior precisão possível na decisão da ANN, dentro das restrições de prazo para realização deste TCC.

1.3 Organização do texto

Este texto está dividido em seis capítulos: o capítulo 2 traz uma introdução e revisão teórica acerca dos temas relacionados à inteligência artificial, redes neurais convolucionais e veículos autônomos; os capítulos 3 e 4 abordam a metodologia e o desenvolvimento deste trabalho, descrevendo as diferentes etapas de construção deste experimento e os resultados obtidos conforme a condução de suas etapas são analisados e comentados no capítulo 5; por último, o capítulo 6 traz as considerações finais e ideias para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo traz o levantamento bibliográfico e os principais conceitos no tocante ao atual estado da arte relacionado à recognição de imagens e veículos autônomos, visto a importância com a qual este primeiro se vincula ao último. As seções dividem-se entre inteligencia artificial, redes neurais, aprendizagem profunda, redes neurais convolucionais recognição de sinais de trânsito e técnicas de detecção e classificação.

2.1 Veículos Autônomos

O veículo autônomo advém da ideia de um automóvel capaz de autopilotar-se em um ambiente não controlado sem qualquer tipo de intervenção humana, porém, o atual estado da arte ainda não alcançou tal cenário ideal (BIMBRAW, 2015). Esta tecnologia baseia-se na presença de uma série de sensores, entre câmeras, radares, lidars, sensores ultrassônicos e sistemas embarcados embutidos em diferentes partes do automóvel, interligados a um computador de bordo, o qual integra os módulos do sistema e processa os dados por eles capturados, assim oferecendo diferentes tipos de funcionalidade ao usuário (CEA, 2017).

Segundo o critério de classificação europeia Reese (2016), a autonomia veicular divide-se em seis níveis: veículos totalmente controlados pelo motorista (nível 0); veículos nos quais se implementam funções específicas de forma automática, capazes de assistir o motorista, mas não substituir suas ações (nível 1); veículos nos quais se implementa ao menos um sistema assistivo de forma automática, como por exemplo modo cruzeiro, capazes de substituir ao menos uma das ações do motorista, de forma que este apenas supervisione o veículo (nível 2); veículos capazes de desempenhar funções pré-definidas sobre determinadas situações específicas (nível 3); veículos capaz de desempenhar funções críticas sobre determinadas situações, por exemplo, ser capaz de estacionar sozinho em uma garagem ou oferecer a opção de piloto automático e retornar o controle da direção para o motorista quando não puder fazê-lo por si mesmo (nível 4); veículos totalmente automáticos capazes de autopilotar-se sobre qualquer situação (nível 5).

2.2 Inteligência Artificial

O conceito de inteligência artificial abarca a capacidade de simular o comportamento inteligente, tipicamente humano, pelo intermédio de dispositivos computacionais, comportamento este concernente à tomada de decisões ou à propositura de soluções para uma determinada sorte de problemas, atribuindo à máquina autonomia necessária para a execução de tarefas, as quais até então só poderiam ser desempenhadas por um ente

humano (HAMMOND, 2015). Jogos eletrônicos são típicas aplicações de IA, considerando o fato de personagens e elementos atuarem programaticamente de tal forma como se estivessem a ser operados por um jogador; ainda, jogos clássicos como Xadrez ou Go podem ser jogados por computadores tão bem quanto o são por humanos, conforme esses foram capazes de derrotar campeões mundiais em situações como em 1997, o computador *Deep Blue*, da IBM, derrotou o campeão mundial Kasparov em uma partida de xadrez e, recentemente Lee Sedol (LARSON,), o campeão mundial de Go, foi derrotado pelo algoritmo AlphaGo (KURENKOV, 2016).

A inteligência artificial divide-se entre fraca [1] e forte [2], sendo à esta primeira, pertencentes todas as aplicações relacionadas à área de IA criadas até o presente momento; pesquisadores desta linha objetivam-se a desenvolver algoritmos que permitam aos computadores reagirem programaticamente a uma série de estímulos (entradas) para desempenharem determinadas tarefas, porém não acreditam que a IA seja capaz de recriar todas as faculdades inerentes ao espírito humano, sendo esta, nada além de uma mera simulação programada para reproduzir as ações humanas e, dessa forma, as máquinas nunca seriam capazes de desenvolver consciência ou uma maior compreensão de mundo (HAMMOND, 2015).

A segunda vertente é composta por adeptos os quais acreditam que conforme os avanços da IA, os cientistas serão capazes não apenas de programar máquinas para que imitem ações e reações, mas recriar o próprio cérebro humano, em todas as suas atribuições e funções, assim o cérebro artificial seria capaz de desenvolver todas as faculdades humanas, inclusive, sentimentos, emoções e consciência. Turing em seu artigo *Computing Machinery and Intelligence*, propôs o questionamento "Poderia uma máquina pensar?"desenvolvendo seu raciocínio sobre uma situação hipotética onde um dispositivo computacional estaria a simular o comportamento de um ente humano tão fielmente, que seria praticamente impossível distingui-lo deste. Hodiernamente, a linha forte da IA é mais um conceito filosófico do que uma linha de pesquisa e consequentemente, ainda não produziu aplicações práticas (HAMMOND, 2015).

Dentre as aplicações, a inteligência artificial atualmente ganha destaque em áreas como: reconhecimento de linguagens naturais, reconhecimento de imagens, automação, sistemas especialistas, aprendizado de máquinas, otimização, jogos eletrônicos, planejamento [3]; e dentre seu arcabouço de técnicas e seus principais campos de pesquisa, apontam-se:

- Algoritmos de Busca: são estratégias utilizadas para tratar problemas cuja a solução se baseia em explorar um determinado conjunto de possibilidades e escolher, dentre

[1]
[2] TechnopediaStrongIA
[3] TutorialsPointIA

estas, aquela que melhor se ajusta a um critério específico (função objetivo) [4] (RUSSELL; NORVIG, 1995a). Tal classe de problemas admite a representação na forma de conjuntos enumeráveis, através da qual é possível construir um espaço de busca contendo todos os elementos que compõem o seu domínio, ou todas as possíveis permutações entre estes. Poderiam citar-se dentre os problemas clássicos pertencentes a esta classe: caminho mínimo, o problema da mochila, o problema de coloração de grafos, o problema das oito rainhas, entre outros (RUSSELL; NORVIG, 1995a).

Geralmente, o espaço de busca pode ser descrito através de um grafo, onde estão representados todos os possíveis estados de seus elementos e, basicamente, uma vez estando o problema modelado sobre tal estrutura, pode-se encontrar uma solução, na pior das hipóteses, pela verificação de cada nó, percorrendo-se o grafo por força bruta; os algoritmos de busca desenvolvem-se no intuito de oferecer maior eficiência no processo de encontrar uma solução sobre tais estruturas, citando-se como algoritmos comuns a este propósito: busca em largura, busca em profundidade, poda alfa-beta, minimax, busca local, dentre outros (RUSSELL; NORVIG, 1995a).

Não raro, em situações reais, o número de possibilidades tende a ser exponencial, e portanto torna-se inviável tratá-los através dos métodos de busca clássicos; neste cenário a meta-heurística surge como estratégia valiosa, sendo possível empregá-la sobre contextos nos quais se saiba de antemão a viabilidade de uma solução, através de algum critério que especifique o quão bem esta se ajusta a resolução do problema em questão (busca informada) e, que a solução ótima seja prescindível, sendo suficiente encontrar dentre o conjunto de boas soluções, uma solução aceitável capaz de atender algum critério de aceitação específico (RUSSELL; NORVIG, 1995a). Dessa forma não se faz necessário percorrer todo o espaço de busca, mas apenas uma porção deste e ainda assim encontrar soluções satisfatórias, citam-se como exemplos de meta-heurística: *hill climbing*, busca gulosa, *simulated annealing*, algoritmos genéticos (RUSSELL; NORVIG, 1995a).

- Sistemas Especialistas: são aplicações desenvolvidas para executar tarefas as quais requerem um alto nível de conhecimento, sendo capazes de substituir ou auxiliar especialistas humanos em diferentes campos do saber [5]. Baseiam-se na existência de grandes bases de dados e emprego de lógica para a construção de métodos de inferência, através dos quais estes sistemas são capazes de propor soluções a partir de uma entrada específica (RUSSELL; NORVIG, 1995b).

  Conforme Russell e Norvig (1995b), seu desenvolvimento inicia-se com a construção de uma base de dados envolvendo um domínio em questão, contando com a ajuda

[4] https://www.tutorialspoint.com/artificial$_i ntelligence/artificial_i ntelligence_p opular_s earch_a lgorithms.htm$
[5] https://www.tutorialspoint.com/artificial$_i ntelligence/artificial_i ntelligence_e xpert_s ystems.htm$

de vários especialistas da área; sobre este volume de conhecimento, constrói-se um motor de inferências, responsável por determinar para uma determinada entrada, um conjunto de prováveis soluções. Estas aplicações são utilizadas em diversas áreas, como sistemas de diagnóstico médico, sistemas comerciais, sistemas de monitoramento, dentre outros.

- Processamento de Linguagens Naturais: este campo engloba o reconhecimento de linguagem falada ou escrita, assim como a produção/conversão entre linguagem falada e escrita, é uma área multidisciplinar a qual envolve teoria da linguagem e teoria da computação, gramáticas livre de contexto, a representação dos elementos da linguagem e os significados através deles, análise léxica, análise sintática, análise semântica, análise de contexto e geração de linguagens naturais [6] (INDURKYA; DAMERAU, 2010).

- Lógica nebulosa (*Fuzzy*): a lógica clássica, baseada na existência de apenas dois níveis de sinal (verdadeiro ou falso, alto ou baixo), encontra vários problemas decorrentes de sua representação binária torná-la extremamente rígida e, por consequência incapaz de trabalhar com espectros mais amplos de valores; por esta razão técnicas de nebulização foram desenvolvidas para transcender as limitações impostas pelos níveis discretos de sinal, possibilitando a existência de níveis intermediários e, dessa forma, gerar um espectro contínuos de valores (RUSSELL; NORVIG, 1995c).

  Como descreve Russell e Norvig (1995c), um sistema nebuloso é capaz de dar respostas acuradas ainda que as entradas sejam imprecisas, ambíguas ou distorcidas. Seu funcionamento consiste em duas etapas: nebulização, na qual são mapeados, para cada variável, diferentes classes cada qual pertencente a determinados intervalos de valores; e posteriormente o cálculo do valor e a "desnebulização", na qual tendo valores de entrada, estabelece-se a quais classes estes melhor se enquadram e com base neste determina-se a saída correspondente [7] (RUSSELL; NORVIG, 1995c).

- Redes Neurais Artificiais: estas serão descritas na próxima seção.

2.3 Redes Neurais Artificiais

Segundo Caccia (2018), em sua origem, as redes neurais artificiais desenvolveram-se como abstrações matemáticas inspiradas no funcionamento do córtex cerebral humano, com o intuito de oferecer um modelo paralelo de computação baseado em aprendizado. Segundo este princípio, uma rede neural poderia computar valores de entrada com base em uma "experiência prévia", adquirida durante o processo de treinamento.

[6] https://www.tutorialspoint.com/artificial$_i ntelligence/artificial_i ntelligence_n atural_l anguage_p rocessing.htm$
[7] https://www.tutorialspoint.com/artificial$_i ntelligence/artificial_i ntelligence_f uzzy_l ogic_s ystems.htm$

Semelhante à um sistema nervoso natural, em escala reduzida, uma rede neural artificial é composta por um aglomerado de neurônios, sendo estes sua menor unidade funcional; diferentes tipos de estruturas foram desenvolvidos, desde simples redes neurais formadas por um único neurônio, até estruturas multicamadas ou totalmente conectadas (BERGER, 2016).

2.3.1 Perceptron

Conforme descreve Berger (2016), perceptrons são considerados o modelo mais simples de neurônio, os quais simulam o comportamento de uma única célula nervosa. Sua estrutura consiste em uma função de ativação (*threshold*), um conjunto de terminais de entrada (dendritos) e saída (axônios), cada qual associado a um respectivo peso de ligação (pesos sinápticos) e, um valor de pequena magnitude denominado viés ou *bias*.

Seu funcionamento consiste na recepção e processamento dos valores presentes em seus terminais de entrada, prosseguindo com a transmissão de seu resultado adiante por meio dos terminais de saída. Uma vez tendo disponível os dados de entrada, o *perceptron* realiza o somatório de cada um destes multiplicado pelo valor do respectivo peso associado ao terminal de entrada. Em seguida, utiliza este valor como parâmetro para a função de ativação a qual definirá o valor a ser repassado para seu terminal de saída (BERGER, 2016).

O segredo por trás de sua capacidade de aprendizado reside no processo de treinamento por meio do qual o neurônio aprenderá pela minimização do erro total de saída, reforçando ou enfraquecendo as conexões entre seus terminais, ajustando gradativamente seus pesos a cada iteração sobre os valores de entrada fornecidos. Este processo pode estar ou não, baseado no conhecimento prévio dos valores de saída esperados para cada entrada, podendo ser conhecido como treinamento supervisionado, não-supervisionado ou recorrente (BERGER, 2016).

2.3.2 Perceptron Multi Camada

Um único *perceptron* apresenta-se demasiadamente limitado, sendo na verdade um bloco fundamental para a composição de estruturas mais poderosas, assim redes neurais são construídas pela junção de vários *perceptrons*, dispostos em estruturas semelhantes à uma matriz: cada coluna contém uma coleção de *perceptrons* e, uma rede é composta por uma ou mais colunas, sendo, por esta razão, chamada rede multicamada (*Multi Layer Network*). Neste sentido uma rede multicamada é composta basicamente por uma camada de entrada (*input layer*), um determinado número de camadas ocultas (*hidden layer*) e uma de saída (*output layer*); estas estruturas apresentam forte capacidade de aprendizado. A profundidade de uma rede é determinada pelo número de camadas ocultas o qual esta

possui; podendo ser composta por uma única camada (rede monocamada) e, neste caso apresenta capacidade de aprendizado linear, ou várias camadas apresentando capacidade de aprendizado não linear. Teoricamente, quanto maior o número de camadas, maior será sua capacidade de generalização (BERGER, 2016).

2.3.3 Treinamento de Redes Neurais Artificiais

2.3.3.1 *Feedfoward*

O algoritmo *feedfoward*, é responsável por computar o comportamento de uma rede do tipo multicamada, sem laços (*loops*) entre os neurônios, dessa forma o sinal recebido nas entradas será transmitido de camada em camada, conforme os valores de saída de cada neurônio de uma dada camada são repassados aos neurônios da camada posterior e o somatório destas saídas intermediárias, é propagado através das camadas até o resultante da saída final da rede. A rede não possui controle sobre os valores recebidos nem sobre sua função de ativação, nem sobre os valores de outros neurônios, portanto um neurônio possui apenas a alteração dos pesos de seus terminais de entrada e o seu valor de *bias*, como forma de influenciar a sua saída, sendo estes os únicos parâmetros variáveis durante o processo de treinamento. (AHIRE, 2018)

2.3.3.2 Função Custo (*Loss*

Este termo se refere à função a qual determina a qualidade de uma solução, o que em outras palavras pode ser descrito como a estratégia de cálculo do valor referente à taxa de acerto da rede. Esta pode assumir várias formas diferentes, mas basicamente consiste em calcular a diferença entre o valor de saída da rede para uma dada entrada comparado ao valor esperado para a mesma, a partir de então pode-se considerar o valor absoluto desta diferença, o seu quadrado ou ainda associar-lhe o somatório dos pesos de cada neurônio, dentre outras informações. Atrelado à função custo, o algoritmo de treinamento da rede executará uma série de ajustes em cada peso da mesma, na tentativa de ajustar o seu valor de saída ao esperado para uma determinada entrada (AHIRE, 2018).

2.3.3.3 *Backpropagation*

Backpropagation: Existem vários métodos para realizar o treinamento de *perceptron* multicamada, *Backpropagation* apresenta-se como um dos algoritmos mais utilizados devido a sua simplicidade e eficiência, sendo empregado como método de aprendizagem supervisionada tanto para problemas de classificação quanto de regressão e pode ser facilmente implementado através de simples operações de matrizes, como soma de matrizes, produto escalar e produto vetorial.

Seu funcionamento consiste no cálculo do gradiente descendente, de tal forma a reduzir o erro total garantindo o melhor ajuste a cada época, este é estabelecido a partir da função de ativação utilizada durante a etapa de *feedfoward*, o cálculo do gradiente envolve a derivada desta, por esta razão é desejável o uso de uma função facilmente derivável, sendo a função sigmoide amplamente utilizada para este propósito. A taxa de aprendizado (*learning rate*) determina quanto o valor de cada peso alterar-se-á a cada iteração e reflete na velocidade/magnitude com a qual a rede altera sua estrutura de pesos (MAZUR, 2015).

2.3.4 Métricas de Desempenho e Problemas Decorrentes

Para aferir o quão bem uma rede neural artificial se ajusta a um problema, avalia-se a sua taxa de acerto durante o treinamento e também durante o teste, que neste são descritas pelos termos acurácia e desempenho, respectivamente. Esta subseção trata sobre os principais problemas relacionados a estes indicadores e as estratégias utilizadas para evitá-los.

2.3.5 *Overfitting* e *Underfitting*

Conforme afirma Dieterle (2019), a complexidade de uma rede neural e o volume de dados para treino (observações) são dois parâmetros os quais devem balizar a escolha da arquitetura; para isto, não existe uma fórmula exata e o verdadeiro contrapeso envolvido em seu processo de construção é encontrar a relação adequada entre complexidade e *underffiting*/*overffiting*, o que ditará o desempenho final da mesma. *Underfitting* é percebido quando uma arquitetura é incapaz de obter uma boa acurácia, mesmo conforme lhe são adicionadas maior número de épocas durante o treinamento; *overfitting* refere-se ao fenômeno de uma arquitetura obter desempenho insatisfatório após haver apresentado um alto valor de acurácia.

Tais fenômenos ocorrem por algumas razões: o primeiro basicamente devido a uma arquitetura muito simples, com número insuficiente de máscaras de convolução, camadas e neurônios, de modo que não seja capaz de identificar características suficientes para discriminar os dados satisfatoriamente, ou também devido ao uso incorreto das operações de *pooling* (estas serão descritas na subseção sobre *pooling*), o que pode levar a uma simplificação exagerada da informação transferida entre as máscaras convolucionais, descartando parte relevante da informação e ocasionando a perda de desempenho do classificador. O segundo, devido a uma arquitetura demasiadamente complexa submetida a um volume reduzido de observações, onde exista a presença de ruídos (considerando-se a imagem como um sinal); neste cenário quanto mais complexa a arquitetura e menor o volume de dados, maior será a possibilidade de que ocorra o *overfitting*, pela maior predisposição da rede em aprender não as características distintivas do *dataset*, mas o

ruído e, dessa forma se tornar dependente do conjunto de dados de entrada (DIETERLE, 2019).

Dessa forma *underfitting* consiste na incapacidade de uma rede neural alcançar grau suficiente de generalização para o conjunto de observações fornecido e portanto apresentará resultados insatisfatórios durante ambas as etapas, treinamento e validação. *Overfitting* refere-se ao fenômeno de uma arquitetura aprender o ruído juntamente com o sinal de entrada, obtendo bom rendimento durante o treino, mas apresentar um desempenho relativamente baixo durante a validação, sendo ineficaz quando exposta a novas observações as quais não estavam inclusas no *dataset* de treinamento (DIETERLE, 2019).

O *overfitting* agrava-se conforme o aumento do grau de complexidade de uma rede neural, ou quando pela redução do volume de observações; por esta razão existe um contrapeso o qual o arquiteto de redes neurais deve sempre resguardar: a complexidade da rede e o *overfitting*. Ainda segundo Dieterle (2019), quando se está a diante de um volume de dados pequeno, a melhor opção será um modelo simples de rede neural; e conforme o volume de dados aumenta e, consequentemente reduz-se o ruído, o modelo deve também crescer em complexidade; dessa forma a calibração do modelo está fortemente atrelada à natureza do *dataset* de entrada e, requer um ajuste empírico para cada caso. Uma rede muito simples (poucas camadas) pode ser ineficiente por não conseguir extrair dos dados de entrada todas as características necessárias para bem classificá-los, por outro lado conforme a complexidade da rede cresce, maior torna-se a possibilidade de ocasionar *overfitting* devido a perda de generalização (DIETERLE, 2019).

Esta última ocorre devido a supervalorização de certos neurônios e a supressão de outros, dessa forma a saída final da rede torna-se dependente de um determinado conjunto desses primeiros, os quais tendem a possuir um peso relativamente superior aos demais. Um único neurônio com pesos demasiadamente elevados assumiria importância desproporcional em relação aos outros, proporcionando um impacto de grande magnitude na saída final; por outro lado um conjunto de muitos neurônios dotados de pequenos valores de peso tornaria a saída dependente da ação conjunta destes, este equilíbrio garante maior poder de abstração à rede. A literatura trata de alguns métodos para contornar a perda de generalização, o mais simples deles é a redução da complexidade da rede neural, calibrando a complexidade para que corresponda a devida natureza do *dataset*, além deste, outros métodos implicam em regularização L1 e L2, *dropout* e *batch normalization* (DIETERLE, 2019).

2.3.5.1 Regularização L1 e L2

Conforme afirma Nagpal (2017b), ambas as técnicas de regularização consistem em estender a função de custo, adicionando-lhe elementos os quais possibilitam penalizar a presença de neurônios com pesos de magnitude elevada durante o cálculo do erro (*loss*). Tal

estratégia vincula à eficiência da rede não apenas a taxa de acerto em relação à resposta esperada, mas também à escolha de pequenos valores para cada peso da rede, uma vez que estes se incorporam ao erro, assim obrigando o algoritmo de treinamento a reduzir a magnitude de cada peso conforme o ajusta. Dessa forma, a saída final é composta pela ação conjunta de diversos neurônios atuando complementarmente, cada qual oferecendo uma pequena parcela de contribuição dentro da rede. Segundo Nagpal (2017a), existem dois tipos de regressão: a regressão L1 *Lasso Regression*, a qual adiciona à função custo (loss) o valor absoluto da magnitude de cada coeficiente; regressão L2 (Ridge Regression) a qual adiciona à função custo o quadrado da magnitude de cada coeficiente.

2.3.5.2 Dropout

Consiste em reduzir a dependência de neurônios específico para a saída final da rede, adotando-se a estratégia de suprimir aleatoriamente um determinado conjunto de entradas, assim estimulando a rede a desempenhar um bom resultado mesmo sob a inoperância destas, estando parcialmente exposta à informação. Isto a obriga a descentralizar-se e criar redundâncias, contando com o maior número possível de neurônios para gerar o valor final de saída, assim como diversos caminhos para se obter o mesmo resultado (BUDUMA, 2019).

2.3.5.3 Batch Normalization

Esta estratégia permite uma maior independência entre as camadas durante o treinamento através do ajuste de escala, assumindo-se que as representações de dados estejam dispostas em escalas diversas, estas serão normalizadas em um único padrão, de forma que todas as representações estejam uniformes. Tal estratégia reduz a covariância interclasse, problema este que, conforme explica Doukkali (2017), pode ser entendido através do exemplo: desejava-se construir uma rede neural para o reconhecimento de gatos, utilizou-se para isso um *dataset* o qual continha imagens de gatos brancos e pretos durante o treinamento, a rede obteve boa acurácia, mas ao submetê-la a um teste de validação no qual se encontravam imagens de gatos com colorações distintas, pardos, amarelos, malhados... esta pequena diferença de amostragem entre os dados de treino e validação acarretou na perda de desempenho da rede como um todo, resultando um *overfitting*.

Este pode estar relacionado não à arquitetura da rede, mas sim à variância da representação de dados; a estratégia de normalização é capaz de reduzir a diferença entre os gatos de uma ou outra coloração e por consequência evitar o *overfitting*, além de permitir o uso de maiores valores para a taxa de aprendizado (DOUKKALI, 2017).

2.3.6 Aprendizagem Profunda

Como define Voinigescu (2017), aprendizagem profunda (*Deep learning*) como uma subcategoria de redes neurais artificiais inspiradas na estrutura do neocórtex, as quais empregam técnicas adequadas para a construção de redes neurais de grandes dimensões capazes de processar grandes volumes de dados de forma eficiente. O termo "profunda" refere-se ao fato de estas serem construídas com o emprego de um maior número de camadas em relação às redes neurais tradicionais; estas camadas adicionais são responsáveis por extrair/representar a informação a partir de diferentes graus de abstração, provendo um maior volume de informação ao classificador. Segundo Brownlee (2016b), tal abordagem trouxe grandes avanços no campo da inteligência artificial, possibilitando aplicações tais como: colorização artificial de filmes em preto e branco, reconhecimento de linguagem natural, classificação de objetos em fotografias, geração automática de escrita manual, adição de áudio em filmes mudos, entre outras (BROWNLEE, 2016a). De acordo com Chandrayan (2017b), existem quatro tipos de redes neurais pertencentes à categoria de *deep learning*: *Unsupervised Pretrained Network* (UPNs), *Recurrent Neural Network*, *Recursive Neural Network*, *Convolutional Network* (CNNs) (CHANDRAYAN, 2017b)

2.3.7 Redes Neurais Convolucionais (CNN)

Redes neurais convolucionais são uma variante das redes neurais artificiais, as quais se mostraram eficientes no reconhecimento e classificação de imagens, sendo uma ferramenta útil, em especial quando aliadas à visão computacional, para o desenvolvimento de aplicações embutidas em robôs e carros autônomos. Segundo Geitgey (2016), estas são capazes de reduzir a complexidade no tratamento de imagens utilizando camadas de convolução e filtros, as quais simplificam a imagem através de operações de convolução e agrupamento.

2.3.7.1 Operações Convolucionais

Segundo Geitgey (2016) estas seguem diferentes etapas de refinamento: primeiramente, uma camada de convolução a qual, assim como uma janela deslizante, desloca-se sobre um passo específico (*stride*) [8], seccionando a imagem enquanto gera um conjunto de sub-regiões, as quais se sobrepõem-se umas as outras; sobre estas são aplicados filtros matriciais os quais tratam esta imagem e retiram dela alguma característica. Como descrito por Sermanet e Lecun (2011) a convolução acontece em três estágios: no primeiro operações lineares de convolução extraem algum tipo de informação representada por valores positivos e negativos; no segundo convertem-se estes valores em não-lineares (ReLU), preservando somente os valores positivos; e por último a etapa de *pooling*, simplifica a imagem resultante, preservando de forma seletiva apenas a informação essencial, assim

[8] número de células a serem deslocadas a cada iteração

gerando os mapas de convolução, os quais serão posteriormente fornecidos como entrada a um classificador.

2.3.7.2 Deslinearização (ReLU)

Esta é realizada pela função de retificação (ReLU), a qual preserva inalterados todos os valores positivos, enquanto substitui os negativos por zero. Esta estratégia fora proposta como uma forma de substituir a arquitetura de pesos compartilhados ao criar uma matriz esparsa de dados, diferente de uma matriz densa. Seu uso traz duas recompensas: primeiramente ganha-se eficiência computacional, através da redução do conjunto de operações realizadas durante o processamento, uma vez que os cálculos necessitariam apenas multiplicações e somas para serem realizados, além disso através desta abordagem, estaria resolvido o problema de *gradient vanishing* uma vez que a derivada da função linear apresenta apenas dois valores, zero para valores negativos e um para os positivos, o que facilita o cálculo do gradiente descendente (BECKER, 2018)

2.3.7.3 *Pooling*

As operações de convolução extraem mapas de características presentes na imagem e uma vez detectada determinada característica, sua relação espacial relativa à sua vizinhança é desconsiderável, sendo portanto descartada enquanto apenas a informação relevante quanto à característica local é preservada. Esta função é desempenhada pela camada de *pooling* através da aplicação de funções específicas, as quais estabelecem quais valores serão transmitidos à camada subsequente, atuando como uma espécie de filtro, o qual retém uma parcela da informação menos relevante. A função de *pooling* mais famosa é o *Max Pooling*, a qual tomando uma matriz quadrada de dimensão **n**, tem como saída o maior de seus valores. Por meio desta técnica é possível otimizar o tempo de processamento da rede durante a etapa de convolução, uma vez que o volume de dados trocados entre as camadas convolucionais reduz-se; em contrapartida o uso indiscriminado da mesma pode levar a problemas de perda de informação e, consequentemente, à incapacidade de atingir altos valores de acurácia, uma vez que parte das informações relevantes pode ser descartada e portanto menor número de características são fornecidas ao classificador (GEITGEY, 2016).

2.4 Redes Neurais Convolucionais e Reconhecimento de Sinais de Trânsito

Placas de trânsito, do inglês *Traffic Signs* (TS), são projetadas e construídas segundo uma regulamentação a qual estabelece critérios básicos de padronização para facilitar sua visualização e reconhecimento por parte de agentes humanos (motoristas). Tais

padrões utilizam como indicadores de discrepância o formato (geometria) e a coloração (tonalidade, intensidade), este último possui características apropriadas para possibilitar uma observação facilitada, mesmo sob diferentes condições climáticas e de luminosidade, como por exemplo diferentes horas do dia, luz do amanhecer/entardecer, escuridão à noite, ou sobre forte chuva; em contrapartida existe a degradação estrutural da placa por parte de elementos naturais ou antrópicos, tais como a danificação da pintura pela própria ação do tempo/clima por variações de temperatura, ação do sol, chuva, ferrugem; ou pela prática de vandalismo ou acidentes (MATHIAS et al., 2013; SALTI et al., 2015).

Como descrito em Mathias et al. (2013), o padrão europeu (Alemanha e Bélgica) apresenta três categorias: *mandatory*, distinguível pela forma redonda e cor de fundo azul; *danger*, formato triangular e cor de fundo vermelha; *prohibition*, formato circular, fundo branco e bordas vermelhas. De maneira similar, placas de trânsito brasileiras dividem-se entre duas principais categorias: placas de direção, contendo formato circular, borda vermelha, escrita negra e fundo branco; e placas de advertência, caracterizadas pelo fundo amarelo, listra de borda e escrita negras. Esta padronização permite que tais características discriminantes sejam exploradas por um algoritmo de reconhecimento durante a etapa de detecção, enquanto tenta determinar a presença de placas e localizar suas posições na imagem, para uma posterior classificação.

- Detecção: consiste em identificar a correta localização de um sinal de trânsito em uma imagem arbitrária, para tanto, originalmente foram desenvolvidas técnicas divididas entre duas vertentes de padrões distintivos: aquelas baseadas em geometria e aquelas baseadas em segmentos; posteriormente, estratégias híbridas viriam a beneficiar-se dos pontos fortes de ambos e, ainda, pesquisas recentes apresentaram abordagens alternativas, baseadas em particionamento da imagem e aprendizado de máquina (SALTI et al., 2015). Atualmente existem estratégias baseadas em classificação para confrontar o problema da detecção, realizando ambos os passos simultaneamente conforme a rede detecta e classifica o objeto desejado (WED, 2017)

  Embora existam vários algoritmos de detecção, um algoritmo de classificação seria suficiente para determinar a posição de um dado objeto na imagem, podendo-se utilizar de uma estratégia baseada em força bruta, ao se particionar a imagem em n sub-regiões e verificar, para cada uma destas, se é possível classificar corretamente algum padrão conhecido (um exemplo, seria um algoritmo de *sliding window* ou janela deslizante). Esta abordagem construída sobre tentativa e erro mostra-se ineficiente uma vez que para uma simples classificação fariam-se necessárias muitas tentativas, inviabilizando o seu uso em sistemas de tempo real, e portanto, sendo necessário a existência de técnicas mais eficientes.

- Classificação: consiste em determinar uma rotulação apropriada para a imagem

presente em uma área detectada, segundo tipos e categorias bem definidos. Recentemente, o campo de redes neurais convolucionais, converge para a tendência de unificar ambas as etapas em um único processo de detecção e classificação, uma vez estabelecida a possibilidade de redução de problemas de detecção a problemas de classificação (RUTA; LI; LIU, 2010).

- Região de Interesse (ROI): O trabalho desenvolvido em (SALTI et al., 2015) demonstra o quanto a detecção será facilitada quanto mais eficiente for o método utilizado para a extração das denominadas regiões de interesse. Estas consistem em conjuntos de sub-regiões da imagem, as quais possuem alta similaridade com um determinado padrão característico do objeto procurado. A importância deste conceito encontra-se na redução do espaço de busca através da extração de porções da imagem as quais apresentam maior probabilidade de conter o objeto alvo, com isso reduzindo consideravelmente a complexidade computacional exigida no processo de reconhecimento.

2.4.1 Técnicas relacionadas ao Reconhecimento de Placas de Trânsito

Esta seção aborda um levantamento de algumas técnicas de representação de dados, de detecção e de classificação de imagens presentes em outros trabalhos sobre o mesmo tema. Dentre estas:

- Características Pseudo-HAAR: *Haar-Like Features* tornou-se um método amplamente conhecido pela abordagem de separação de canais e a variação Haar em Cascatas descrita por Viola and Jones em seu trabalho *Robust Real-Time Face Detection* (VIOLA; JONES, 2004), tendo se baseado no trabalho proposto por Papageorgiou, Oren e Poggio (2002) para o desenvolvimento de um algoritmo de detecção de faces; embora tenha aprimorado o trabalho deste, tornando-o mais eficiente, a abordagem de Viola apresentava altas taxas de falsos-positivo.

  Esta técnica é desenvolvida sobre o conceito de classificadores fracos, cada um destes é obtido a partir da aplicação de uma máscara retangular sobre uma dada região da imagem e a esta é atribuído um peso. Após a geração de um conjunto de n classificadores fracos, o fator de característica de cada máscara é calculado multiplicando cada peso pela intensidade (níveis de cinza) média da região coberta por cada máscara e, por fim o fator característico final é obtido através da soma destes. Uma vez calculado, o fator de característica é utilizado para identificar regiões em uma outra imagem para as quais seja possível reconhecer a mesma característica (VIOLA; JONES, 2004).

  Trabalhos realizados na tentativa de aprimoramento desta técnica empregam abordagens baseadas em estratégias evolutivas para determinar o número, tamanho e posição

de cada região retangular, assim como, o valor de seu respectivo peso, entre estes encontra-se (VIOLA; JONES,), um algoritmo *Adaboost* é utilizado para minimizar a quantidade de características *HAAR* como uma espécie de *pooling*, selecionando aquelas mais importantes e descartando as menos significativas, criando assim um algoritmo de seleção de características em cada etapa do Algoritmo *Adaboost* um novo classificador fraco.

- Color Enhancement: Color Enhancement é uma técnica proposta por Ruta, Li e Liu (2010) em *Real Time Traffic Recognition in Three Stages*, fora utilizada por Zaklouta e Stanciulescu (2014) no primeiro estágio de detecção. Seu funcionamento se baseia sobre o conceito de Regiões de Interesse o que em outras palavras equivale a identificar as regiões de uma imagem para as quais existe um alto grau de semelhança em relação à tonalidade alvo (tonalidade do objeto a ser identificado), dessa forma uma vez identificadas áreas para as quais a tonalidade se assemelha às características da tonalidade característica para determinado objeto, estas são intensificadas. Por este meio é capaz reduzir o espaço de busca para uma subsequente classificação, ao destacar as regiões de interesse.

  Este procedimento é conduzido através da aplicação de filtros simples e um fator de *threshold* a partir do qual regiões com valor acima deste serão destacadas pela cor branca, enquanto as outras permanecerão negras. Embora seja capaz de oferecer bons resultados de uma forma relativamente fácil, seu aspecto negativo reside na dependência tonal, na qual, sob determinadas circunstâncias de iluminação e condições climáticas, como chuva, neblina ou fumaça, a eficiência método pode ser comprometida. Além do mais, para imagens nas quais existem demais elementos de tonalidade semelhante à característica desejada, esta técnica se torna praticamente irrelevante visto que não ocasionaria uma redução o espaço de busca considerável (RUTA; LI; LIU, 2010).

- Histograma de Gradientes Orientados Conforme descrito por Dalal e Triggs (2005) em *Histogram of Oriented Gradients for Human Detection*, esta técnica se baseia na aplicação de filtros específicos em porções segmentadas de uma matriz densa da imagem (particionando-a em células), os quais calculam a direção para a qual existe uma maior variação de contraste (gradientes). Uma sequência de vetores são gerados sobre cada célula da matriz densa e, o conjunto das resultantes (soma dos vetores) de cada uma formam uma espécie de contorno sobre os elementos da imagem. Esta imagem contornada é então, submetida como entrada para um classificador devidamente treinado, tendo muitos autores optado por utilizar uma *support vector machine*.

2.4.2 Detecção Baseada em Redes Neurais

Esta subseção traz algumas técnicas baseadas no uso de redes neuais convolucionais para a detecção e a possibilidade de converção entre problemas de detecção em problemas de classificação o que tem permitido o desenvolvimento de tais técnica, dentre elas R-CNN Faster CNN, YOLO.

A possibilidade de conversão entre problemas de detecção em problemas de classificação fora o precursor para a atual tendência com que os métodos baseados em segmentação de imagem e aprendizado de máquina estão ganhando espaço entre as estratégias mais promissoras. Tal abordagem porém não é uma descoberta recente, uma vez que operações de segmentação de imagem já eram utilizadas nos algoritmos de força bruta. Entretanto este processo demonstra-se ineficiente, ainda mais, quando atrelado ao custo envolvido nas operações de convolução e, por isso, várias abordagens foram propostas como formas de contornar tal limitação através de melhorias aplicadas sobre o método de proposição de regiões de interesse,

Assim, a eficiência da detecção por redes neurais convolucionais está determinada pela eficiência do algoritmo de redução do espaço de busca e identificação de regiões de interesse, enquanto a precisão dependeria tão somente da acurácia na classificação. Na prática observa-se a necessidade de incluir no dataset de treinamento, imagens de contra exemplos dos objetos que se deseja classificar, para que desta forma a rede neural seja capaz de distinguir satisfatoriamente quando um elemento representa o fundo e não o objeto em questão (reduzir a ocorrência de falsos-positivos).

- R-CNN: *Region based CNN* utiliza um método para a redução do espaço de busca através de um algoritmo denominado busca seletiva, o qual propõe uma pré-seleção de sub-regiões candidatas, as quais possivelmente representam um objeto, segundo Parthasarathy esta busca é estabelecida sobre alguns critérios básicos de discriminação, como cor, textura e iluminação (PARTHASARATHY, 2017). Uma vez gerado o conjunto de sub-regiões, elas então serão submetidas à uma seleção mais fina pelo emprego de uma CNN. Exemplos de algoritmos recentes para a busca seletiva podem ser encontrados em (ZAGORUYKO et al., 2016).

  Conforme descrito por Parthasarathy (2017), os módulos de uma rede R-CNN são implementados através de uma CNN para extração de características, uma *Vector Support Machine* para a detecção do objeto e posteriormente um algoritmo de *bounding box* para a delimitação correta da região onde se encontra o objeto. Este método apresenta uma grande desvantagem visto seu tempo de processamento durante a extração de características e regiões, o que pode ser contornado ao optar por armazená-las em disco rígido, gerando em contrapartida, um grande consumo de espaço de memória; todavia é capaz de atingir alta acurácia.

- *Spacial Pyramyd Pooling* (SPP):

  Segundo Jia, Huang e Darrell (2012), esta técnica surgiu como uma tentativa de contornar as limitações apresentadas pela técnica R-CNN e, além de reduzir o número de imagens candidatas geradas e, consequentemente, necessitar um menor consumo de memória, proporciona a capacidade de percepção espacial da imagem, isto é, a relação entre os objetos presentes na paisagem, estabelecendo uma relação de hierarquia entre estes.

  Isto é realizado particionando-se a imagem em sub-regiões, através da chamada operação de *pooling* espacial, cujo conceito se baseia na representação esparsa da informação pelo córtex cerebral; as imagens locais são extraídas e armazenadas como patches codificados por um valor de ativação, as características extraídas pelo *pooling* são geradas aplicando-se determinadas operações, como cálculo do valor médio ou máximo, sobre este conjunto de valores de ativação. Uma forma de meta-heurística é implementada para obter as possíveis melhores regiões da imagem, uma vez que o número de soluções tende a um valor exponencial (JIA; HUANG; DARRELL, 2012).

  Em outras palavras, ela realiza a divisão da imagem em fragmentos menores, preservando a característica da imagem como um todo, enquanto analisa o contexto em torno uma r egião específica da imagem, dessa forma estabelecendo relação de profundidade entre os seus elementos através da relação entre estes e sua vizinhança. Além de proporcionar maior eficiência em relação ao consumo de memória, a principal contribuição desta abordagem está em permitir uma maior flexibilidade em relação ao tamanho das imagens de entrada, visto que as estratégias de redes neurais convolucionais abordadas até o presente momento estão rigidamente limitada à trabalhar com imagens de mesmo tamanho e, embora este fato possa ser facilmente contornado, através de bibliotecas de manipulação de imagens, é necessário considerar que o redimensionamento da imagem ocasiona perda de dados (TRICKS, 2018).

- *Fast-CNN*: A ineficiência encontrada em R-CNN está relacionada ao não compartilhamento de computação durante as diferentes fases de busca seletiva, classificação e delimitação da *bounding box*; *Fast R-CNN* traz a implementação destas melhorias promovendo compartilhamento de processamento entre as fases. Isto pode ser feito por meio da propagação dos gradientes através das operações de *pooling*, por exemplo, utilizando *backpropagation*, onde cada célula pode sobrepor-se sobre outras, assim calculando gradientes de múltiplas regiões como operações de convolução. Também, a etapa de delimitação da *bounding box* pode ser atrelada à etapa de treinamento da rede convolucional, operando em conjunto ambas as operações de classificação e localização/delimitação, atacando o problema sob duas frentes simultaneamente (GANDHI, 2018a).

- *Faster R-CNN*:Conforme mencionado anteriormente, a eficiência temporal do emprego de métodos baseados em particionamento e aprendizado de máquina estão intrinsecamente relacionados à eficiência do algoritmo de seleção de regiões de interesse. Isto se evidencia em *Faster R-CNN*, o qual é capaz de obter um desempenho em média cinco vezes superior ao *Fast R-CNN* através da simples substituição do algoritmo de busca seletiva (GANDHI, 2018a).

  Métodos comuns de sugestão de região de interesse (*Region proposal methods*, RPM) são busca seletiva e *EdgeBox* (GANDHI, 2018a). Segundo Gandhi, *Region Proposal Network* (RPN) se baseia em uma rede neural convolucional pura para executar ambas as operações de *bounding box* e atribuição de scores. Esta rede possui duas camadas adicionais de convolução, uma para gerar um mapa de características e outra para gerar um mapeamento de valores de objetos e sua correspondente delimitação. Em outras palavras, *Faster R-CNN* consiste em associar uma (RPN) à *Fast R-CNN*, desta forma, otimizando o tempo de busca por região de interesse (GANDHI, 2018a).

- *Single Shot Detector* (SSD): é uma melhoria ainda maior sobre este algoritmo, de acordo com Tsang (2018), esta técnica consiste na execução de ambos os passos em uma única etapa, mesclando a etapa de proposta de regiões de interesse e a detecção do objeto em cada uma destas regiões.

- *You Only Look Once* (YOLO): Segundo Gandhi (2018a) uma das técnicas atuais de melhor custo benefício em relação a qualidade de detecção e processamento, ela consiste em particionar a imagem em n sub-imagens dispostas em células de uma matriz quadrada de dimensão **n**. Uma rede neural opera sobre cada célula, atribuindo um score o qual indica a probabilidade de que aquela célula pertença a uma determinada classe, após pré-classificar as n células é preciso então estabelecer os limites delimitados por cada objeto (*bounding box*), mesclando células vizinhas de mesma classificação (GANDHI, 2018a).

  Assim como as famílias de *R-CNN*, YOLO integra operações de classificação e detecção em uma única etapa, utilizando informações da imagem como um todo e também encontrando as *bounding box* simultaneamente, como aborda Redmon et al. (2015). A desvantagem na proximidade na qual os objetos se encontram uns aos outros, pode impossibilitar a classificação correta, pois só pode classificar no máximo um objeto por caixa (*box*) (GANDHI, 2018a).

2.4.3 Pré-Processamento

A coloração da imagem é uma característica de extrema importância para a detecção, entretanto a mesma não expressa equivalente relevância para a classificação, sendo até mesmo dispensável, de acordo com Mathias, o elemento essencial para a extração de

características de uma placa é seu padrão interno e seu formato, a cor é um elemento secundário; além do mais, a representação de imagens baseadas em cores exige o uso de três canais, seja o mapa de cores baseado em modelos RGB ou CMYK. Em contrapartida escalas de cinza reduzem a representação de dados a um único canal e, por consequência além de otimizar o custo de armazenamento das imagens, também o faz sobre o número de computações necessárias para a manipulação das mesmas. De tal forma, escalas de cinza são preferíveis, não apenas devido às vantagens anteriormente citadas, mas também, por apresentarem, em alguns casos, resultados mais satisfatórios do que os obtidos a partir de abordagens baseadas em cores. (MATHIAS et al., 2013)

Deve-se levar em conta não apenas as características de imagens tomadas isoladamente, mas sua relação com o contexto das outras imagens presentes no dataset, uma vez que estas podem apresentar diferenças quanto a nível de contraste, saturação, brilho, resolução... e esta discrepância pode trazer consequências negativas sobre o processo de treinamento, sendo portanto, necessário normalizá-las através de técnicas específicas (DOUKKALI, 2017).

Ainda, em situações nas quais existe um volume reduzido de amostragem, é desejável utilizar-se de certos artifícios para obter um bom desempenho, artifícios estes, o quais consistem em aumentar o volume de amostragem através da síntese de imagens, isto é, expandir o volume de imagens iniciais a partir de si próprio, através de simples operações sobre suas imagens.

Dessa forma, a primeira etapa no processo de treinamento de uma rede neural para classificação, consiste na redução da complexidade dos dados de entrada assim como um pré-processamento cujo objetivo é extrair informações importantes do dataset, as quais podem facilitar o processo de treinamento e classificação. Para tal podem ser utilizadas diferentes técnicas, cada qual se mostra mais ou menos apropriada para determinado contexto e, cabe ao projetista de redes neurais selecionar o conjunto de técnicas que mais se adapta e favorece o desempenho do método de classificação empregado.

Por esta razão, existe uma etapa opcional, anterior à classificação, na qual são utilizadas diferentes técnicas de tratamento de imagens a fim de simplificar a representação de dados e destacar as características essenciais para um bom desempenho, assim preparando os dados brutos do *dataset* para serem submetidos ao classificador; esta etapa denomina-se pré-processamento.

Conforme descrito em Stallkamp et al. (2012), as quatro equipes, as quais obtiveram melhores resultados durante o campeonato alemão de classificação de sinais de trânsito, realizaram o pré-processamento e a redução da complexidade por conta própria, de forma a favorecer o método escolhido para a classificação. De certa forma os métodos empregados variaram entre redimensionamento da imagem para um tamanho específico, a redução tonal à escala de cinza, normalização de contraste... houveram também equipes as quais

dispensaram tal etapa, utilizando os dados brutos propriamente.

A equipe IDSIA efetuou um procedimento mais rigoroso, através da extração de regiões de interesse, selecionando apenas o aspecto relevante da imagem e descartando as porções menos importantes, o resultante então, fora redimensionado num tamanho de 48x48 pixel (STALLKAMP et al., 2011) e, posteriormente, submetido a um pré-processamento onde se utilizou um conjunto de quatro técnicas diferentes: alongamento e equalização de histograma, equalização adaptativa e equalização de contraste (STALLKAMP et al., 2011). Por fim, converteu-se cada imagem em escala de cinza e redimensionou-lhes ao tamanho de 28x28, posteriormente fora aplicado uma pirâmide de histograma de gradientes orientados (STALLKAMP et al., 2012).

Linear Discriminant Analisys: Em seu trabalho *"how far we are from solution"*, Mathias descreve como foram utilizadas várias técnicas de pré-processamento. Primeiramente a conversão em escala de cinza e seu redimensionamento para 32x32 píxeis, a extração de características através de histogramas de gradientes orientados HOG1 HOG2 e HOG3 e a redução da dimensionalidade atraveś de *Linear Discriminant Analysis*, Representação esparsa baseada em projeção linear, *Sparse Representation Based Linear Projection*, uma variante de *Locality Preserving Projections*; *Iterative Nearest Neighborhood Based Linear Projection*, (MATHIAS et al., 2013).

Segundo Mathias et al. (2013), o emprego de *Linear Discriminant Analisys*, além de proporcionar a redução da complexidade, apresenta como segundo efeito positivo, elevar a afinidade entre as classes de imagem, o que aumenta a variância interclasse, enquanto diminui a variância intraclasse, em outras palavras isto equivale a dizer que imagens pertencentes a uma mesma classe se tornam mais semelhantes, enquanto imagens de classes distintas tornam-se mais diferentes. Neste mesmo trabalho, Mathias et al. (2013) verifica ainda a influncia dos métodos de pré-processamento e redução da complexidade de dados sobre o desempenho durante a classificação, para isso realizando uma série de combinações e variações destas. Entre outras técnicas se encontram, Representação Esparsa Baseada em Projeção Linear SRLP e Iteração pelo Vizinho mais Próximo Baseada em Projeção Linear INNPL.

O volume de dados disponível para conduzir o treinamento é fator de grande relevância para o desempenho final da rede, visto que a quantidade de observações (instâncias de entrada durante o treinamento) ditará a facilidade em atingir um grau suficiente de generalização. Eventualmente pode-se deparar com *datasets* de volume reduzidos contendo poucas instâncias; diante destas situações é necessário utilizar-se de artifícios para contornar tais limitações e, uma estratégia importante para tal, é a técnica de *data augmentation*, através da qual é possível aumentar consideravelmente o volume de dados disponíveis, pela simples ação de alterar levemente as imagens presentes no volume inicial, através de transformações como, translação, rotação, inversão horizontal/vertical,

aplicação de níveis variados de borrões (*blur*).

2.4.4 A Etapa de Classificação

Na seção anterior foram descritas algumas técnicas as quais podem ser utilizadas para a de detecção de imagens, visto que esta constitui etapa importante no reconhecimento de imagens, uma vez que identifica a correta posição de um objeto nas mesmas; a classificação consiste em uma etapa posterior à detecção, na qual, uma vez determinado a área onde se encontra o objeto, ser-lhe-á atribuído um rótulo, informando sobre a qual classe o objeto identificado pertence. Esta seção cobre os principais conceitos e técnicas relacionados à etapa de classificação, entre elas: *random forest*, *linear discriminant analisis*, árvores de decisão, *k nearest neighborhood* e redes neurais, além da análise de algumas técnicas adicionais estudadas no decorrer da leitura dos artigos.

2.4.4.1 Classificação

Nesta seção descrevem-se diferentes algoritmos de classificação, os quais foram encontrados no decorrer da leitura dos trabalhos relacionados; embora nem todos se baseiem em redes neurais, considerou-se citá-los para fins de documentação. Seguem os principais:

- *Random Forest*:

  Segundo Eulogio (2017), esta é uma estratégia derivada de *bootstrap bagging*, criada com a intensão de minimizar o *overfitting* experimentado por árvores de decisão. Esta última buscou solucionar o problema através de uma estrutura composta por diversas árvores de decisão, de forma que o resultado final dependa da saída das N árvores, obtido através de votação, isto é, a moda da saída de cada uma das árvores que compõem a floresta. Todas as árvores são treinadas sobre um mesmo conjunto de dados; *Random Forest* divide o espectro de características sobre a floresta, de forma que cada subárvore tome apenas um subespaço de alternativas e se especialize sobre uma porção específica de classes, isto é capaz de fazer tal estratégia consideravelmente melhor do que ambas as precedentes. *Random Forest* está entre as técnicas as quais obtiveram piores resultados no campeonato alemão.

- *K Nearest Neighborhood*:

  estratégia baseada em aprendizado supervisionado, simples porém robusta, apresentando desempenho compatível com outras técnicas de classificação mais complexas. Seu funcionamento consiste no cálculo da distância (seja esta uma distância euclidiana, de manhatan, entre outros) entre as observações, esta representa o fator de similaridade, na qual dada uma instância **t** calcula-se a distancia entre esta e todas

as outras instancias, sendo este o fator de similaridade e, com base no valor da constante **K**, estabelecem-se os k pontos mais próximos de **t**, os quais são adicionados em **A**; posteriormente, de acordo com a classe dos pontos que estão em **A**, calcula-se a probabilidade de que **t** pertença a cada uma destas classes, com base no número de vizinhos mais próximos presentes na mesma (ZAKKA, 2016).

Segundo Patel (2017) *Support Vector Machine* é um método de aprendizagem de máquina supervisionada o qual basicamente consiste em um problema de regressão, no qual estabelece-se um hiperplano de separação entre classes de dados. A dinâmica de seu funcionamento pode ser entendida como, tendo um conjunto de pontos dispersos em um espaço multidimensional, os quais representam valores típicos pertencentes a determinadas categorias (estes dados devem estar rotulados), o problema consiste em identificar o hiperplano de maior distância (melhor margem) entre os pontos pertencentes a esta classe, de forma a delimitar o hiper espaço entre pontos pertencentes à classe C1, dos pertencentes a classe C2 até a classe Cn, dessa forma determinando o melhor intervalo de confiança durante a predição.

Assim como outros métodos de regressão linear, o hiperplano então separa o espaço multidimensional entre diferentes classes de tal forma que valores que estejam posicionados além dos limites do hiperplano para uma determinada região são considerados como pertencentes a uma classe e, os que estão aquém desse limite são pertencentes a outra. O grande diferencial deste método em relação a outras abordagens de regressão linear, é que *Support Vector Machine* estabelece o hiperplano de forma a separar o hiperespaço com a maior distância entre pontos de classes distintas, dessa forma determinando um melhor intervalo de confiança para a predição (PATEL, 2017).

(GANDHI, 2018b)

- Redes Neurais Convolucionais e Comitês de Redes Neurais: CNN foram tratadas nos tópicos anteriores, sendo o tema central deste trabalho; vale mencionar aqui o fato de que muitas redes diferentes podem ser treinadas sobre um mesmo *dataset* e posteriormente, agrupadas na forma de um comitê (conjunto de redes), do qual a saída será tomada através da moda entre o valor de saída de cada uma das redes que o compõe. Dessa forma, através de um conjunto de arquiteturas cuja acurácia de treino e validação atinjam valores bons valores, serão capazes de produzir um desempenho muito próximo ao ideal.

2.4.4.2 Validação

O classificador é responsável por identificar a qual classe pertence uma determinada imagem, conforme descrito nas seções precedentes, este deve ser treinado através de um processo no qual uma série de ajustes tornarão sua saída mais acurada. Após o processo

de treinamento, o classificador necessita ter seu desempenho avaliado em relação à taxa de acerto quando exposto a novas amostragens contendo elementos (imagens) para os quais este fora treinado para classificar; esta etapa de validação, também conhecida como etapa de teste, é a etapa a qual ditará o quão bem ajustado um classificador está em relação a um determinado domínio de classificação.

2.5 Trabalhos Relacionados

2.5.1 *Man vs Computer*

Desenvolvido por Stallkamp et al. (2012), este é assim como *How Far are we from the Solution?* (MATHIAS et al., 2013), um trabalho o qual pode ser tomado como base para determinação de qual técnica empregar em um primeiro momento. Neste os autores realizam uma análise sobre a metodologia empregada pelas quatro equipes, as quais obtiveram os melhores resultados no campeonato de reconhecimento de sinais de trânsito alemão, promovido pelo instituto *Institut Für Neuroinformatik* (INI).

Os experimentos foram conduzidos sobre o *dataset* GTSRB, o qual será detalhado mais adiante e a análise da eficiência de cada algoritmo é expressa através da função de *loss*, a qual equivale à razão entre o número de classificações corretas sobre o de classificações errôneas; além da comparação de desempenho entre cada uma das técnicas, elas são comparadas também em relação ao desempenho humano e, ainda, os autores abordam a eficiência quanto ao tempo de resposta, visto que técnicas precisas porém computacionalmente custosas, não são uma solução efetiva para aplicações relacionadas ao reconhecimento de imagens em tempo real. A seguir é descrito brevemente as equipes com melhores resultados no campeonato e os métodos utilizados:

- *Baseline* (LDA): a equipe utilizou um classificador linear aplicado sobre os histogramas de gradientes orientados fornecidos pelo GTSRB, este fora treinado através da técnica *linear discriminant analysis* e implementado conforme a Stark Machine Learning Library (Igel, Glasmachers, Heidrich-Meisner, 2008).

- *Sermanet*: utilizou uma CNN multiescala, composta de vários estágios de extração de informações, cada qual desempenhado por uma camada convolucional, uma camada de transformação não linear e uma camada de *pooling* espacial; seu principal diferencial em relação a uma CNN tradicional consiste em que todos os estágios de extração de características são fornecidos ao classificador.

- IDSIA: comitê de CNNs, um conjunto de redes neurais convolucionais e uma rede de *perceptrons* multi camada configurada por tentativa e erro, treinadas sobre a extração de regiões de interesse; cada MPL e CNN é treinada individualmente e

Classificador	**Score**
o melhor humano	99,22
media humana	98,84
Committee of CNNs (IDSIA)	99,46
Multi scale CNN (Sermanet)	98,31
LDA e random forests (INI-RTCV and CAOR)	< 96.15

Tabela 1 – Analise comparativa do desempenho alcançado pelas equipes em Human vs Computer

após isto, são combinadas, de forma que a melhor saída de cada um destes é tomada como resultado parcial e a moda destes valores é tomada como resultado final.

- CAOR: utilizando a técnica de random forest contendo 500 árvores, técnica semelhante ao comitê de redes neurais uma vez que a saída depende da votação por maioria das árvores. O time utilizou o *dataset* oficial de características HOG2, disponibilizado durante o campeonato.

Conforme demonstra a Tabela 1, *linear discriminant analisis* e *random forest* obtiveram resultados consideravelmente inferiores à performance humana, com pontuações abaixo de 96.15; já a CNN, obteve um desempenho próximo a esta, chegando à 98.31 de precisão; comitê de redes neurais superou a performance humana atingindo 99.46. Tomando por base os resultados encontrados neste trabalho, pode-se considerar redes neurais convolucionais o mais eficaz dentre os principais métodos conhecidos para a classificação de imagens, embora traga como consequência um maior custo computacional.

- *Detection at Tree Stages*:

  Neste trabalho os autores desenvolvem um reconhecedor de placas de trânsito baseado em três estágios, primeiramente, utiliza-se segmentação para detectar a posição da imagem através filtros cromáticos e acromáticos, extraindo regiões de interesse baseadas nas características das placas e, a classificação é realizada por um método baseado em formato, sendo para isto verificado desempenho utilizando-se diferentes classificadores, dentre estes *k-d trees*, *random forest* e *support vector machines*.

O site oficial do campeonato alemão [9] disponibiliza dois *datasets*: GTSDB e GTSRB, destinados à detecção e classificação de sinais de trânsito, respectivamente, totalizando um montante de 144,769 imagens de sinais de trânsito divididas entre 70 classes referentes à detecção e 43 à classificação. Segundo o site oficial GTSRB contém uma média de 30 imagens de sinais de trânsito por classe, cada qual tomada a partir de três câmeras dispostas em angulações diferentes do veículo.

[9] http://benchmark.ini.rub.de/

Ambos os *datasets* trazem instâncias já divididas entre imagens de treino e de teste, de forma a não ser necessário separá-las para a efetuação das etapas de treino/validação. Além do mais as imagens estão divididas em pastas, cada qual referente a uma determinada categoria sendo por esta razão, desnecessário a utilização de anotações, sendo suficiente a rotulação das pastas.]

2.5.2 A Committee of Neural Networks for Traffic Sign Classification

Desenvolvido por Ciresan et al. (2011), neste trabalho os autores avaliam o ganho de desempenho proporcionado pelo uso de comitês de redes neurais convolucionais, avaliando seu desempenho sobre o *German Trafic Sign Recognition Datase*. Após a avaliação de desempenho de várias arquiteturas submetidas a diferentes modelos de cores, imagens coloridas (três canais) e escala de cinza e, diferentes padrões de características, HOG e HAAR, a melhor das CNNs alcançado uma precisão de 98,73%, construiu-se um comitê capaz de atingir 99.15%.

Contrariamente ao descrito na seção sobre o preprocessamento, os resultados obtidos neste, demonstraram que a estratégia baseada em cores resultou melhores valores comparada aos resultados obtidos sobre imagens em escalas de cinza, embora esta última também tenha apresentado bons resultados; tal fato levanta a dúvida quanto ao fato do uso de três canais de cores para a representação da imagem resultar desempenho superior à representação monocanal.

3 Metodologia

Neste capítulo são descritos os materiais e métodos envolvidos na construção do experimento, a primeira seção aborda o *software* e o *hardware*, e a segunda as linguagens de programação, *frameworks* e *datasets*.

3.1 Materiais

3.1.1 *Software*

Os códigos concernentes à implementação da rede neural desenvolveram-se nas linguagens Python 2.7 e Python 3.5, auxiliados pelas bibliotecas TFLearn, Mathplot, Pandas, Seaborn. Bash script foram utilizados a fim de automatizar os experimentos tanto durante o treinamento e validação, no disparo de programas, produção de configurações e variações de parâmetros, quanto na coleta, tratamento e análise de dados , na organização e produção dos experimentos e extração de dados dos arquivos de log.

3.1.2 *Hardware*

Utilizaram-se três dispositivos para este trabalho, dentre os quais, um *notebook*, Processador Intel(R) Core(TM) i5-4200U CPU @ 1.60GHz, disco rígido 1 TB, RAM 8 GiB e sistema operacional Linux Mint; um *desktop* Processador Intel(R) Core(TM) i3-4200U CPU @ 1.60GHz, disco rígido 500 GiB, RAM 4 GiB e sistema operacional Linux Mint; e uma *workstation*, Processador Intel(R) Xeon(R) CPU E3-1240 V2 @ 3.40GHz, disco rígido RAM: 8GiB, e sistema operacional Ubuntu.

3.2 Métodos

3.2.1 Estudo dos *Frameworks*

Como extensão à linguagem Python utilizaram-se diferentes bibliotecas de inteligência artificial e tratamento de dados, tanto para a implementação dos códigos durante os experimentos relativos à construção das redes neurais, quanto para a análise e visualização de dados; esta seção explana brevemente os principais *frameworks* estudados durante a execução deste trabalho.

- PyBrain: Segundo (SCHAUL JUSTIN BAYER, 2010), PyBrain é uma biblioteca modular de aprendizado de máquina desenvolvida para a linguagem Python, com o

objetivo de prover algoritmos flexíveis e fáceis de usar, mas ainda assim poderosos. Oferece recursos de aprendizagem supervisionada, não-supervisionada e evolutiva.

- Caffe: Segundo (CAFFE,) , Caffe é um framework para aprendizado de máquina baseado em expressividade, velocidade e extensibilidade, desenvolvido pelo Berkeley AI Research.

- TensorFlow: Segundo (GOOGLE, 2015), TensorFlow é uma biblioteca de computação numérica de código aberto desenvolvida para propósitos de aprendizado de máquina, a qual permite a programação paralela sobre CPU ou GPU, oferecendo como principal funcionalidade a representação de dados através de grafos, onde cada nó representa uma operação e cada aresta representa um tensor — matriz contendo informação trocada entre os nós do grafo.

  O básico do *framework* TensorFlow foi estudado através do site Tutorials Point [1]; um exemplo de rede neural aplicada ao benchmark MNIST está disponível no endereço [2]

  TensorFlow fora o primeiro framework a ser estudado neste trabalho, posteriormente, com a descoberta do TFLearn, verificou-se que o desenvolvimento das arquiteturas de redes neurais ocorreria de forma mais fácil através deste, e por esta razão, optou-se por utilizá-lo em detrimento do TensorFlow.

- Keras: De acordo com seu site oficial [3], Keras é uma API para a construção de redes neurais, capaz de executar sobre TensorFlow, Theano e CNTK, desenvolvida em Python e voltada para prototipagem. A estrutura principal de um código Keras é construída sobre o objeto modelo, adicionando-lhe camadas as quais estabelecem a estrutura da rede. Cada camada é definida quanto à forma e a função de ativação. o mesmo exemplo desenvolvido para TensorFlow sobre o *dataset* MNIST em Keras pode ser encontrado está disponível em *Elite Data Science* [4].

  Um modelo básico para a construção de uma rede neural arbitrária é a forma Sequential(), a partir da qual a rede é construída, adicionando-lhe camadas do tipo Dense() e funções de ativação, geralmente do tipo ReLU e *Softmax*. Além da facilidade de construção de modelos neurais, oferecendo suporte até mesmo a arquiteturas convolucionais, recorrentes ou mistas, a principal vantagem deste *framework* reside na facilidade de adaptar uma rede treinada previamente sobre imagens para a classificação de vídeos. Além do mais esta é facilmente integrável com o próprio framework TensorFlow, sendo utilizada para a construção de modelos os quais posteriormente serão utilizados junto a este último [5].

[1] https://www.tutorialspoint.com/tensorflow/tensorflow$_q uick_g uide.htm$
[2] https://github.com/aymericdamien/TensorFlow-Examples/blob/master/notebooks/3_NeuralNetworks/convolutiona
[3] https://keras.io/
[4] https://elitedatascience.com/keras-tutorial-deep-learning-in-python
[5] https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html

- TFLearn: Assim como Keras, TFLearn é uma API de alto nível para o desenvolvimento de redes neurais, capaz de executar sobre TensorFlow ou Theano, porém, esta biblioteca proporciona alta expressividade, simplificando profundamente o trabalho do programador. Com poucas linhas de código é possível construir aplicações que em outros *frameworks* seriam consideravelmente mais extensas. Por esta mesma razão TFLearn foi utilizado para a construção de todas as arquiteturas presentes neste trabalho. A construção do algoritmo de classificação foi segmentada entre vários módulos, dentre eles, os responsáveis por: obtenção dos dados, construção da arquitetura da rede neural, treinamento, teste/validação. Realizou-se o estudo da sintaxe do TFlearn através dos tutoriais de exemplo disponíveis em seu próprio site oficial [6].

- Matplotlib: Matplotlib é uma biblioteca Python concebida para produzir gráficos 2D dos mais variados tipos para aplicações acadêmicas e científicas. Sua principal vantagem é possibilitar a construção de *scripts* e dessa forma facilitar a elaboração de gráficos. Também pode ser integrado com IPython, Jupyter e aplicações *web*. Oferece dentre as possibilidades de gráficos: gráficos linerares, histogramas, *power spectra*, gráficos de barra, gráficos de pizza, *scatterplots*, entre outros.

- Pyplot: Matplotlib.pyplot é uma biblioteca interna ao Matplotlib concebida para conferir a este funçoes que o assemelham ao MATLAB, possibilitando a customização de gráficos quanto aos seus sobre eixos, escalas, títulos e legendas, além de possibilitar alterações sobre a figura como, criar diferentes áreas para plotar vários gráficos em uma mesma figura.

- Pandas: é um *framework* destinado à análise de dados, trazendo amplas funcionalidades relativas à importação dos dados, seu armazenamento em estruturas de dados, a manipulação destas, e a obtenção de valores como média e desvio padrão, além da plotagem de gráficos e tabelas. De acordo com o produtor, Pandas é um pacote de alto nível, desenvolvido a partir da biblioteca numpy, para análise de dados em Python e foi concebida com o propósito de ser simples, rápida, flexível e facilmente integrável a outras bibliotecas Python. Esta ferramenta oferece recursos para leitura e manipulação de dados de forma facilitada, como seu próprio autor descreve, esta se tornará um *framework* para análise de dados. Dentre os principais recursos estão: leitura de arquivos tabulares como uma *table* SQL ou uma planilha de Excel; as estruturas de dados apresentam duas formas: *series* e *dataframes*, sendo designados respectivamente para dados em 1-D e 2-D; integrabilidade com o Pyplot e outras bibliotecas de análise de dados.

[6] http://tflearn.org/tutorials/

- Seaborn: Seaborn é uma biblioteca Python concebida com o propósito de possibilitar uma melhor qualidade na apresentação e visualização de dados, esta foi projetada tendo como base o matplotlib e portanto, pode ser utilizado conjuntamente com o *framework* Pyplot e Pandas, seu principal objetivo é promover a construção de gráficos mais bem apresentáveis e bonitos.

3.2.2 *Datasets*

Através de uma busca sobre os principais *benchmarks* mundiais para a classificação de sinais de trânsito, encontraram-se três apropriados aos objetivos deste trabalho, sendo eles:

- *Dataset* Alemão: *German Traffic Sign Recognition Benchmarks* (GTSRB): disponibilizado através do site oficial do *Institut Für Neuroinformatik* (INI) [7], este fora construído para propósitos do campeonato alemão a partir de mais de 10 horas de filmagem, apresentando uma grande variação em termos de qualidade e legibilidade de imagens. Segundo (MATHIAS et al., 2013) as gravações foram tomadas sob velocidades diferentes, isto faz com que existam diferentes números de recursos disponíveis para cada um deles, favorecendo as classes de imagem tomadas à baixa velocidade, uma vez que ha um maior número de quadros gravados para estas. Além do mais imagens pequenas, obtidas de uma longa distância, apresentam valores pequenos de resolução, enquanto imagens muito grandes foram obtidas a partir de uma alta proximidade entre o veículo e a placa, provocando borrões e fantasmas. Isto acarreta a presença de classes para as quais existe um número muito pequeno de amostras, as quais serão desfavorecidas em relação àquelas contendo um número maior, dificultando o processo de treinamento da rede neural. De igual modo, imagens distorcidas ou de baixa resolução podem comprometer o desempenho da CNN. O *dataset* apresenta um total de 43 classes, somando aproximadamente 50 mil imagens, já divididas entre imagens de treino e de teste, de forma a não ser necessário separá-las para a efetuação de cada etapa de treino/validação. Além do mais as imagens estão divididas em pastas, cada qual referente a uma determinada categoria sendo por esta razão, desnecessário a utilização de anotações, bastando a rotulação das pastas.

- *Dataset* Belga: disponibilizado no endereço [8], conta com uma série de arquivos de imagens e anotações. Utilizou-se o pacote definitivo, o mesmo utilizado durante as experimentações do VISICS, contendo o conjunto reduzido de 7219 imagens dispostas em 62 classes. Uma característica importante é que este apresenta uma amostragem

[7] http://benchmark.ini.rub.de/?section=gtsrbsubsection=dataset
[8] https://btsd.ethz.ch/shareddata/

reduzida de observações por classe, sendo ideal para avaliar a robustez de uma rede quanto ao overfitting.

- *Dataset* Chines: disponibilizado no endereço [9], de acordo com o site disponibilizador, este inclui 6164 imagens de sinais de trânsito divididas entre 58 categorias, ainda foram divididos entre imagens de treino e validação, sendo que cada um apresenta 4170 e 1994 amostras respectivamente, as anotações das imagens contém também as coordenadas onde se encontra a placa na imagem na imagem e a categoria. Cada classe possui em média 80 amostragens.

3.2.3 Modelagem de arquiteturas de CNN

Criaram-se os *scripts* de automação sobre os quais as redes neurais seriam construídas segundo algum dos quatro modelos de arquiteturas de camada convolucional, os quais serão explicados na próxima seção. As variáveis de controle dos parâmetros da rede neural foram modeladas através de arquivos de entrada tabular (csv), estratégia a qual permitiu um maior nível de versatilidade na prototipagem das diferentes configurações de arquiteturas avaliados neste trabalho.

3.2.4 Avaliação e ajuste dos modelos

A avaliação sucedera-se sobre os arquivos de log resultantes do treinamento, os quais trouxeram valores importantes como a evolução da acurácia, o tempo de processamento, a taxa de *loss*; e também sobre os valores de desempenho obtidos durante a validação. Sobre tais indicadores, geraram-se planilhas e gráficos através das bibliotecas matplot, pandas e seaborn.

3.2.5 Validação do melhor modelo

Utilizando os três *datasets* de teste disponíveis, validaram-se os cinco melhores modelos desenvolvidos nas fases anteriores, considerando-se como critérios para a eleição dos melhores modelos, o valor médio e máximo do valor de acurácia e o desempenho alcançado durante a etapa de validação. Dessa forma, valorizaram-se os modelos que obtiveram melhor desempenho durante o treinamento, verificando sua capacidade de esquivar-se do *overfitting*, assim preservando desempenho equivalente durante a validação, por esta metodologia esperava-se selecionar os modelos, além de mais eficazes, os mais robustos.

[9] http://www.nlpr.ia.ac.cn/pal/trafficdata/recognition.html

4 Desenvolvimento

Este capítulo aborda as etapas de desenvolvimento deste trabalho, sendo dividido entre obtenção dos *datasets*, delimitação do escopo, construção e validação das redes neurais e uma última seção que aborda as definições de nomenclatura para cada um dos modelos.

4.1 Obtenção do Dataset

Após consultar o site do CONTRAN [1], obteve-se a regulamentação oficial dos padrões de placas brasileiras, e a partir destes, construiu-se uma biblioteca de imagens de placas brasileiras, contendo os modelos desejados para este trabalho, atraveś dos recursos disponíveis no site Aimore [2]. Uma vez tendo em mãos os modelos de referência, seria necessário obter um *dataset* final composto por um grande número de amostras de fotografias de placas em situações reais para cada modelo de placa. Para a execução desta tarefa, havia duas possibilidades: encontrar alguma base de dados já existente e disponível livremente, ou construir um *dataset* especificamente para este trabalho, a partir da coleta de imagens em fontes abertas como o Google Imagens; esta tarefa poderia ser automatizada com o uso de técnicas de *crawling* [3].

Embora num primeiro instante, este trabalho esteve centrado em torno do objetivo de desenvolver um classificador de placas de trânsito para sinalização brasileira, no decorrer do mesmo surgiram alguns problemas quanto à obtenção do *dataset*. Primeiramente a base de dados disponibilizada pela Universidade Federal do Rio Grande do Sul [4] apresentava imagens destinadas à detecção e não à classificação (as placas ocupavam porções relativamente pequenas em relação ao todo da imagem), logo tais imagens deveriam ser recortadas a fim de contemplar apenas a porção da imagem onde estava a placa. Em contrapartida, a tentativa de construção de um *dataset* a partir de fontes livres, além de trazer o mesmo problema presente na abordagem anterior, apresentou uma quantidade reduzida de amostras válidas (visto que muitas das imagens obtidas através de *crawling* do Google não continham imagens de placas reais ou apresentavam baixa qualidade).

Por fim, a etapa de construção do *dataset* tornava-se demasiadamente trabalhosa, tendo consumido grande parte deste trabalho, atrelada à ineficiência em se encontrar boas imagens para classificação de placas de trânsito nacionais, o que levou à conclusão de

[1] http://new.denatran.gov.br/publicacoes/download/MANUAL_VOL_I.pdf, http://new.denatran.gov.br/publicacoes/download/MANUAL_VOL_II.pdf
[2] https://aimore.net/placas/
[3] como por exemplo, a biblioteca Beautiful Soup para Python
[4] http://lapsi.eletro.ufrgs.br/Download/BRTSD/

que seria inviável o emprego de um *dataset* nacional para a condução dos experimentos: consumir-se-ia um enorme volume de tempo com o trabalho de recortar as imagens, ou do contrário, haveria um conjunto pequeno de imagens de placas reais disponível. Portanto, considerou-se como opção mais viável utilizar bases de dados estrangeiras.

Neste sentido, após uma pesquisa das principais bases de dados mundiais, foram encontrados três *datasets* compatíveis às necessidades deste trabalho: Alemão, Belga e Chinês. Cada *dataset* foi obtido a partir do site de seu respectivo disponibilizador, através de arquivos compactados, cada qual dividido entre duas pastas correspondentes ao de treino e teste, e suas imagens divididas em pastas nomeadas por uma contagem numérica crescente, cada qual representando uma classe. Por fim, obteve-se três pastas contendo três *datasets*, cada qual teve suas subpastas renomeadas através de uma numeração formada por uma sequência de 5 dígitos de 00000 até o número final de classes do respectivo *dataset*; esta etapa de padronização foi importante para a automatização dos testes.

4.2 Delimitação do tema e escolha do método

Primeiramente, o tema deste trabalho fora delimitado em torno ao problema de classificação, sendo a detecção deixada em segundo plano por se demonstrar uma tarefa mais complexa e incompatível com o prazo disponível. Como método de classificação optou-se pelo uso de redes neurais convolucionais, não apenas por ser este o tema principal deste trabalho, mas, tendo em vista seu poder de categorização e relativa simplicidade em relação a outras técnicas. Dentres as possibilidades ainda se encontra o emprego de comitês de redes neurais, os quais apresentam uma acurácia ainda melhor em relação a uma única rede neural.

Como *framework* foi decidido prosseguir este trabalho através do TFLearn, pois sua curva de aprendizado demonstrou-se mais acentuada do que qualquer outra, contribuindo para o tempo de implementação e também para a facilidade de alteração da arquitetura da rede. A construção de um modelo em TFLearn é relativamente simples e conta com o emprego dos seguintes módulos: camada para os dados de entrada input_data, camada convolucional conv2d, camada de max_*pooling*, camadas totalmente conectadas fully_connected e drop_out (a qual elimina determinados de acordo com uma probabilidade). Além disso uma das grandes facilidades deste *framework* está em sua forma de representação das imagens através de estruturas do tipo *arrays* da biblioteca Numpy aliadas à biblioteca de manipulação de imagens Skimage.

Construção da rede neural

O site oficial do TFLearn disponibiliza tutoriais [5] acerca das diferentes técnicas as

[5] http://tflearn.org/tutorials/

quais podem ser desenvolvidas através do *framework*. Cada tipo de problema envolve o uso de técnicas adequadas para a solução do mesmo, por exemplo, regressão linear não seria uma técnica eficiente na classificação de imagens como o são redes neurais convolucionais. Durante a etapa de aprendizado do *framework*, a fim de se adaptar com a sintaxe do mesmo, seguiu-se progressivamente os tutoriais de exemplo envolvendo diferentes problemas e suas técnicas, entre eles: Iris, MNIST, e CIFAR; embasado sobre este conhecimento implementou-se um código genérico para a construção das arquiteturas descritas neste trabalho.

A estratégia básica de construção de uma rede neural utilizando TFLearn conduz-se de forma facilitada, concentrando-se sobre a construção do modelo da rede neural, abstraindo-se os detalhes. Primeiramente efetua-se o tratamento e extrapolação do *dataset* através do pré-processamento, realizado pelas funções nas linhas 3 à 5 do Código 4.2, e também pelo *data augmentation* nas linhas 8 à 9 posteriormente, na linha 13 é declarada a rede e sobre ela constrói-se a arquitetura, adicionando-lhe camadas de convolução (como na linha 18) e *pooling* (linha 19), por fim, é determinada a camada de classificação, escolhendo sua dimensão e o tipo de função de ativação (linhas 23 à 25), e o método de regressão, um exemplo de código segue abaixo (linhas 27 à 29).

Descrevendo brevemente sua sintaxe tem-se: *img_prep* (linha 3) corresponde ao pré processamento, nesta etapa calcula-se a média e o desvio padrão das características das imagens do *dataset*; *img_aug* (linha **8**) corresponde as técnicas de extrapolação da informação inicial contida no *dataset* original, esta extrapolação consiste na produção de distorções como rotações, reescala, *flip*, produção de borrões... através do comando *tflearn.input_data* inicializa-se uma rede neural, e através de *conv_2d* (linha 18) e *max_pool* (linha 20) constrói-se a camada de convolução, deve-se escolher os valores para os parâmetros da dimensão da máscara de convolução e sua função de ativação, e a dimensão da máscara de *pooling*; através de fully_connected (linha 23) e *dropout* (linha 24) constrói-se a camada totalemnte conectada, também chamada classificador, onde escolhe-se o número de neurônios em cada camada totalmente conectada e a função de ativação para a mesma, além de escolher a probabilidade de permanência para o *dropout*.

Código-fonte com a arquitetura da CNN no TFLearn

```
1
2 # definindo o preprocessamento
3 img_prep = ImagePreprocessing()
4 img_prep.add_featurewise_zero_center()
5 img_prep.add_featurewise_stdnorm()
6
7 # definindo o data augmentation
8 img_aug = ImageAugmentation()
```

```
9  img_aug.add_random_flip_leftright()
10 img_aug.add_random_rotation(max_angle=25.)
11
12 #modelando a CNN
13 network = tflearn.input_data(shape,
14 data_preprocessing, data_augmentation)
15
16 # m scara convolucional
17 # deve-se definir a dimens o e a fun   o de ativa   o
18 network = conv_2d(network, dimension, activation)
19 # m scara de pooling
20 network = max_pool_2d(network, 2)
21
22 # definindo a camada totalmente conectada
23 network = fully_connected(network, 1024, activation='tanh')
24 network = textit{dropout}(network, 0.5)
25 network = fully_connected(network, 105, activation='softmax')
26
27 network = regression(network, optimizer='adam',
28         learning_rate=0.005,    loss='categorical_crossentropy',
29         name='target')
```

4.3 A construção das arquiteturas

A proposta deste trabalho é construir um classificador de imagens de sinalização de trânsito e identificar a arquitetura da rede neural mais eficiente para tal tarefa. Existe consenso entre os pesquisadores de que não existe uma técnica exata para a configuração e ajuste fino de uma rede neural, sobretudo devido ao fato de que a arquitetura ideal varia conforme se altera a natureza do *dataset*, e portanto faz-se necessário ajustá-la, empiricamente, por tentativa e erro, conforme se modificam seus parâmetros e observa-se a ocorrência de melhorias. Assim, a cada acerto, seleciona-se dentre as melhores arquiteturas aquelas que servirão de base para a produção de outras melhores, e determinar a estrutura mais eficiente de uma rede neural consiste na melhora progressiva de uma arquitetura previamente identificada como dotada de um alto potencial; alterando sua configuração, buscar-se-ia a obtenção de variantes mais eficientes.

Partindo deste pressuposto, fora desenvolvido um experimento através do qual deduzir-se-iam quais parâmetros e de que forma eles influenciariam o desempenho da rede neural convolucional, conhecimento este que embasaria a posterior construção de

arquiteturas mais promissoras. Assim a construção das arquiteturas partira de configurações básicas e, gradualmente, adicionaram-se-lhes elementos até a produção de configurações mais complexas, simultaneamente variando seus parâmetros. O experimento dividiu-se em fases que se subdividiram em seções, permitindo a análise da influência de tais parâmetros isoladamente, isto é, a cada fase e a cada seção apenas um parâmetro específico foi alterado, permanecendo estático todo o restante da estrutura da rede.

Dentre os parâmetros analisados encontram-se: (i) a influência da disposição das máscaras convolucionais na camada de convolução, (ii) suas dimensões, (iii) seu sequenciamento, (iv) a existência ou não de camadas intermediárias de *pooling* ou *dropout*, (v) a influência do tamanho das imagens de entrada, (vi) a arquitetura da camada totalmente conectada, (vii) uso ou não de camada intermediária de *dropout*, entre outras. Esperar-se-ia obter, ao fim deste procedimento, um amplo espectro de configurações, das quais as mais promissoras forneceriam substrato para a construção de variantes possivelmente melhores.

Num primeiro momento, este trabalho utilizou como critérios para determinar a eficiência de uma rede os seguinte indicadores: a acurácia final obtida durante o treinamento (ACC), desempenho durante teste (SCR). Observou-se, posteriormente, que uma vez tendo a rede neural se estabilizado durante o processo de treinamento, esta apresentava uma leve oscilação em sua acurácia e, por este motivo, tomar a acurácia final (último valor da acurácia) da fase de treinamento seria uma forma injusta de determinar a eficiência de uma arquitetura em relação à outra. Considerando-se duas arquiteturas arbitrárias (Arquitetura I e Arquitetura II), ainda que em etapas intermediárias do treinamento a arquitetura I tenha alcançado maiores valores de acurácia em relação à arquitetura II, apenas o valor final seria considerado e, sendo este menor que o valor final da Arquitetura II, esta última seria tomada como mais eficiente, ainda que tenha apresentado um histórico de acurácia consideravelmente pior do que a primeira, sendo esta portanto, injustamente superestimada.

Consequentemente, definiu-se como critério mais justo, incluir a média da acurácia durante treinamento (μACC), uma vez que seriam levados em consideração todos os seus valores intermediários, e dessa forma, arquiteturas as quais apresentassem uma maior progressão em sua curva de aprendizado e maior capacidade de sustentar sua taxa de acurácia (robustez) seriam favorecidas em relação àquelas arquiteturas com uma curva de aprendizado pouco acentuada ou que apresentassem alta taxa de oscilação em sua acurácia. Ainda como uma métrica para o overfitting, utilizar-se-ia o valor de μACC multiplicado pelo valor de SCR.

4.4 Modelos de Arquitetura da camada de Convolução

Nesta seção, apresentar-se-ão os diferentes modelos utilizados para a geração automática de redes neurais convolucionais a partir do *framework* TFLearn. Esta etapa consistiu em verificar os efeitos sobre o desempenho para os diferentes tipos de arquitetura da camada de convolução, quanto à intercalação de camadas de mesma dimensão ou dimensões diferentes, além do modo de intercalação. Desenvolveram-se ao todo quatro modelos batizados de: A, B, FC, FD.

- **Modelo A**: conjunto de arquiteturas caracterizadas pela presença de uma sequência de máscaras de convolução de dimensões constantes, conforme a Figura 1. O objetivo desta arquitetura é determinar a forma como o tamanho da máscara da camada de convolução (número de células) interfere na eficiência.

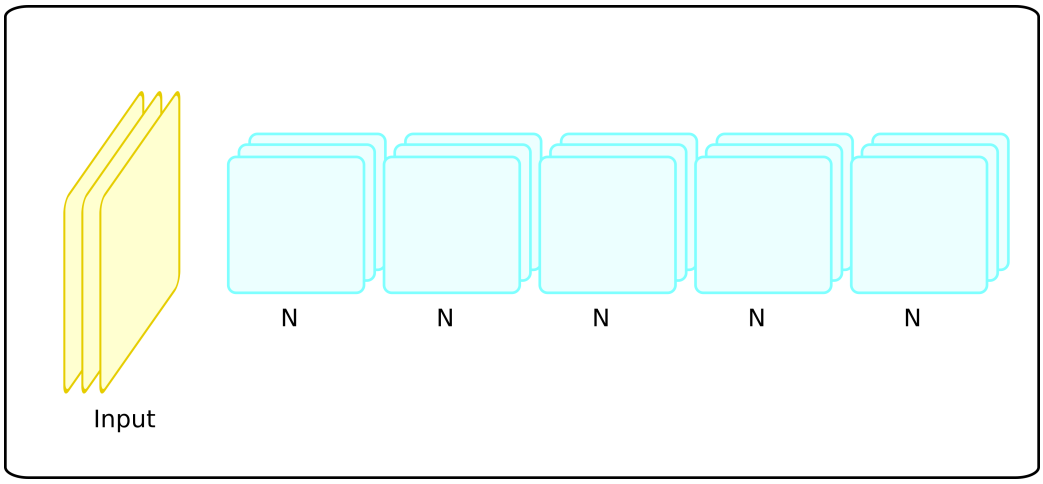

Figura 1 – Representação do Modelo de Arquitetura A

- **Modelo B**: conjunto de arquiteturas caracterizadas pela intercalação de máscaras de convolução de dimensões discrepantes intercaladas entre si, conforme a Figura 2. Este se objetiva a determinar qual é o efeito da intercalação entre camadas maiores e menores e o efeito da variação na diferença entre elas.

- **Modelo FC/FD**: conjunto de arquiteturas caracterizadas pelo afunilamento da máscara de convolução, isto é, as máscaras estão dispostas em uma sequência crescente ou decrescente em relação à dimensão da máscara de convolução, confome as Figuras 3 e 4.

4.4.1 Nomenclatura dos Modelos

Criou-se uma nomenclatura padrão a fim de facilitar o desenvolvimento, teste e validação das diversas arquiteturas. Neste tópico descrever-se-ão os critérios utilizados para

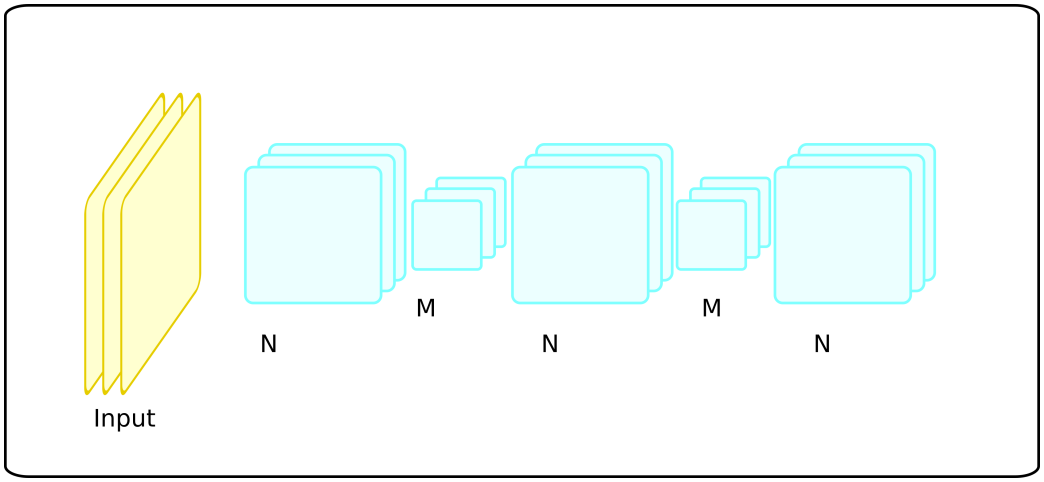

Figura 2 – Representação do Modelo de Arquitetura B

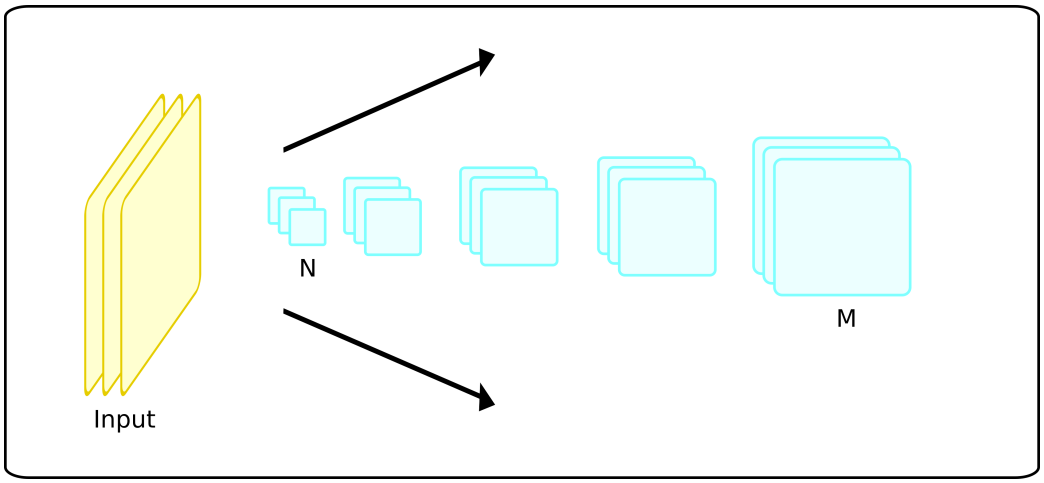

Figura 3 – Representação do Modelo de Arquitetura FC

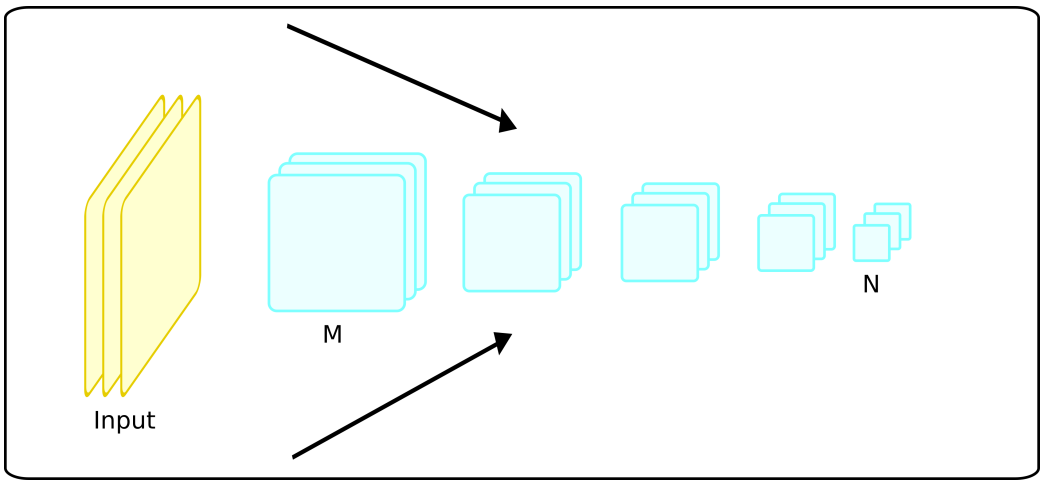

Figura 4 – Representação do Modelo de Arquitetura FD

definir a nomenclatura. Partindo do ponto de que todo treinamento ou teste efetuar-se-ia sobre um *dataset*, faz-se necessário definir o tamanho de imagem de entrada, o modelo de

Parâmetro	Sigla	Espectro de Valores
Dataset	T	[GERMAN, BELGIAN E CHINESE]
Dimensão da imagem de entrada	I	[8, 16, 24, 32]
Modelo de cores	C	[escala de cinza, RGB]
Modelo camada convolucional	P	[A, B, FC, FD]
Supressão de *Pooling*	W	A sigla em questão
Presença de *Dropout*	Q	A sigla em questão
Comprimento da camada convolucional	L	[5, 10, 15, 20]
Dimensão da camada convolucional	D	8, 16,32,64, 128]
Arquitetura da camada totalmente conectada	M	[8, 16, 32, 64, 128]
Taxa de aprendizado	Y	[0.01, 0.001, 0.0001]

Tabela 2 – Nomenclatura dos parâmetros de configuração para as arquiteturas

cores, o modelo da camada de convolução, assim como o comprimento e a largura de suas máscaras, a configuração da camada totalmente conectada, a taxa de aprendizado e por último a presença ou ausência de camadas de max_*pooling* e/ou *dropout* intercaladas às camadas convolucionais e as camadas totalmente conectadas, respectivamente; a seguir são descritas as siglas e as possíveis variações criadas para cada um destes parâmetros.

Dessa forma a nomenclatura é dada por T.IpCx.P[W,Q]_LxDxMxYx, ou em palavras: [Dataset].[Dimensões da imagem de entrada e modelo de cores]. [Modelo][Supressão de *Pooling*, Presença de *Dropout*]_[comprimento e largura da camada de convolução][arquitetura da camada totalmente conectada][taxa de aprendizado]. Sendo "x"equivalente às lacunas, as quais serão substituídas por variáveis referentes à configuração do parâmetro, representando um número inteiro ou real, ou uma lista de números precedidos por x na forma xN_0 xN_1 xN_2 ... até xN_n.

4.5 Organização do Experimento

Nesta seção, descrever-se-ão as fases e etapas do experimento relativo ao estudo dos parâmetros de construção da rede, entre eles: tamanhos de máscara de convolução, podendo ser constantes, alternados, ou que variam gradativamente de forma crescentes ou decrescentes (afunilamento); presença ou ausência de camadas de *pooling* e *dropout*; arquitetura da camada totalmente conectada, características da imagem de entrada e desempenho sobre outros *datasets*.

Inicialmente, todas as arquiteturas seguiram uma configuração padrão, conforme pode ser visto na tabela 3, possuindo a camada de convolução composta por exatamente cinco máscaras convolucionais, cada qual sucedida de uma camada de *pooling*; como classificador uma rede neural perceptron monocamada, totalmente conectada, com função de ativação *softmax*, e função *loss* do tipo *categorical crossentropy*.

Camada	**Dimensão**
Imagem de Entrada	24x24
Convolução	$N_0 x N_0$
Pooling	2x2
Convolução	$N_1 x N_1$
Pooling	2x2
Convolução	$N_2 x N_2$
Pooling	2x2
Convolução	$N_3 x N_3$
Pooling	2x2
Convolução	$N_4 x N_4$
Pooling	2x2
Totalmente Conectada	43

Tabela 3 – Arquitetura padrão Fase I

4.5.1 Fase I - Variação da Camada Convolucional

A Fase I objetiva-se a avaliar os efeitos da camada de convolução sobre o desempenho da rede, para tanto esta foi dividida em 2 seções, sendo a primeira relativa às arquiteturas construídas com base em camadas de *pooling* intercaladas às camadas convolucionais sem o uso de *dropout*, enquanto na segunda, foram substituídas as camadas de *pooling* por camadas de *dropout*.

4.5.1.1 Etapa I

Avaliaram-se os efeitos da variação das dimensões das máscaras de convolução, sendo que cada uma destas possui uma dimensão quadricular, e a variável relaciona-se com o tamanho do lado do quadrado; para tanto, o parâmetro D determina as dimensões da sequência de máscaras, variando entre o espectro de Dx8 a Dx128, sendo seus valores derivados de potências de dois.

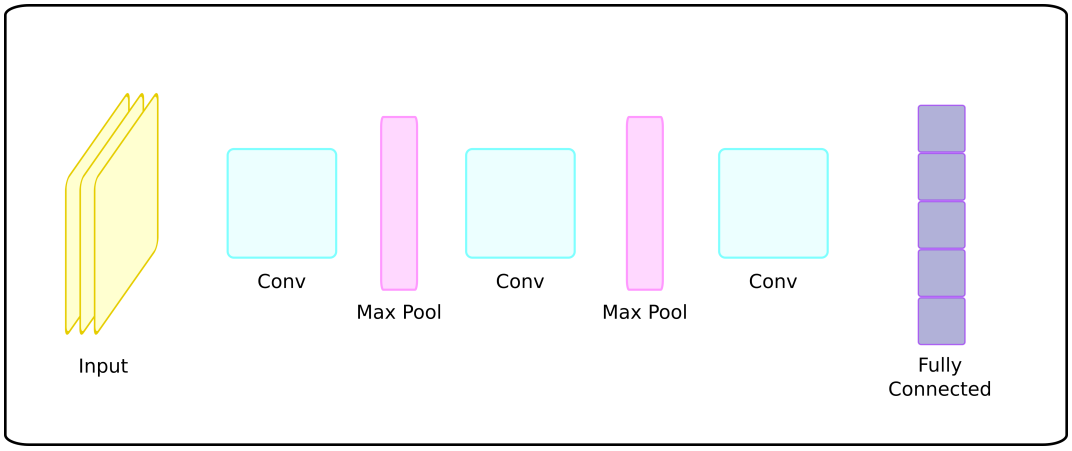

Figura 5 – Representação da Arquitetura Com *Pooling*

4.5.1.2 Etapa II

Avaliaram-se os efeitos proporcionados ao desempenho da rede pelo alongamento da camada de convolução, isto é, se para um maior número de máscaras de convolução obter-se-ia melhores valores tanto para a acurácia quanto para a predição. Neste sentido o parâmetro L determina o comprimento da camada de convolução, e por padrão todas as arquiteturas apresentavam profundidade L5, referente ao número de camadas convolucionais e com intervalo de variação entre L5 e L20, crescendo progressivamente de cinco em cinco unidades do valor inicial até o valor máximo.

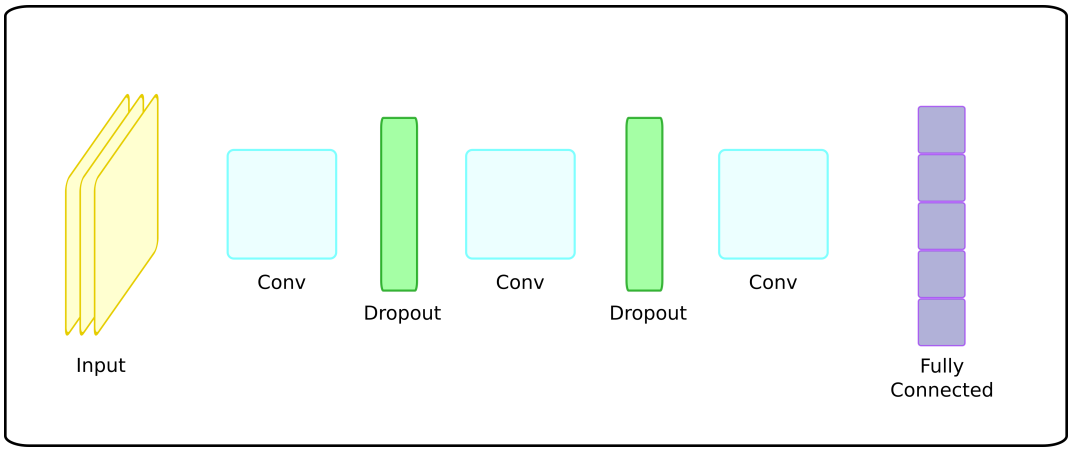

Figura 6 – Representação da Arquitetura Com *Dropout*

4.5.2 Fase II - Variação da Camada Totalmente Conectada (classificador)

Para os melhores modelos selecionados na fase anterior, realizou-se alterações quanto à camada totalmente conectada, variando-se o número de camadas e neurônios em cada uma destas. Esta fase se torna um tanto quanto mais complexa conforme o número de variações para o classificador demostram-se infinitas; partindo do pressuposto em que o número de neurônios seriam derivados de potências de dois, averiguou-se os efeitos obtidos com a inclusão de um número arbitrário de camadas, começando por apenas uma até o máximo de três camadas.

4.5.3 Fase III - Variação das Características da Imagem

Esta fase consistiu em verificar os efeitos das características da imagem sobre o desempenho da rede, para tanto selecionaram-se os melhores resultados das fases anteriores. Esta também se divide em duas etapas, sendo que na primeira, variou-se o tamanho das imagens de entrada sendo redimensionadas para tamanhos de 16, 32 e 64 píxeis; e na segunda converteram-se as imagens à escala de cinza.

4.5.4 Fase IV - Mudança de Datasets

Após as três fases anteriores, obteve-se o conjunto das potenciais arquiteturas, selecionando-se a melhor ou um conjunto das melhores para a construção de uma rede neural definitiva que seja submetida a uma avaliação quanto ao seu desempenho quando treinada e validada sobre outros *datasets*. Ao final, esperava-se determinar quão genéricas tais configurações seriam, uma vez que alterando-se o *dataset*, altera-se também a característica dos dados, e dessa forma o varia-se seu desempenho em relação aos mesmos.

4.5.5 Comentários Adicionais

Durante a Fase I geraram-se ao todo mais de 200 arquiteturas de redes neurais convolucionais, levando-se em consideração os quatro tipos de modelos e todas as possibilidade de variação entre seus parâmeros de configuração. Posteriormente, geraram-se mais de 40 arquiteturas de camadas totalmente conectadas durante a Fase II e nas Fases III e IV realizaram-se mais 25 testes sobre as dimenções da imagem de entrada, o modelo de cores e, por último, a validação dos melhores modelos sobre outros *datasets*. Como se percebe, a maior dificuldade envolvida neste trabalho esteve relacionada à exploração de várias arquiteturas, na tentativa de encontrar as mais apropriadas para o problema de classificação de sinais de trânsito. Na próxima seção serão tratados os resultados obtidos no decorrer deste experimento.

5 Resultados e Análise

Este capítulo traz a descrição dos resultados obtidos durante os experimentos. Na primeira seção são descritos os resultados parciais compostos pelas primeiras três fases do experimento. A segunda aborda os resultados finais obtidos durante a fase IV com a mudança de *dataset*, sendo utilizados os *datasets* Belga e Chinês.

5.1 Resultados Parciais

Ao longo do desenvolvimento do trabalho, obteram-se resultados parciais pelos quais identificaram-se configurações bem sucedidas; estes foram de suma importância para orientar a direção na qual as arquiteturas deveriam evoluir. É prudente ressaltar que nenhuma das conclusões expostas no decorrer deste texto devem ser tomadas como regras universais, uma vez que foram obtidas através de um estudo empírico aplicado especificamente sobre o *dataset* alemão. Tais conclusões careceriam de análise estatística, além de necessitarem uma validação mais ampla em relação a outros *datasets*. Nesta seção, apresentar-se-ão os resultados mais importantes obtidos em cada uma das fases de experimentação, a análise dos diversos parâmetros estudados e a forma como estes interferem no desempenho final de uma rede.

5.1.1 O Limiar de Acurácia

Através de análise experimental, percebe-se que cada arquitetura apresenta um limiar de acurácia, a partir do qual não importa quantas épocas se adicionem à fase de treinamento, o valor desta não apresentará alterações positivas. Este limiar está associado ao *underfitting* e, conforme detalhado nas seções anteriores, quanto mais simples uma arquitetura se apresenta, maiores serão as chances de que ela não consiga generalizar satisfatoriamente as características dos dados de entrada. Portanto, algumas arquiteturas apresentam ponto de convergência absurdamente baixos, em contrapartida outras apresentam limiares próximos da acurácia ideal. Este último tipo de arquitetura é o desejado para fins do presente trabalho: uma vez identificada uma coleção de arquiteturas com alto limiar de acurácia, selecionar-se-ia aquela cuja curva de aprendizado apresentasse maior acentuação (evolui mais rapidamente, apresentando valores mais altos após o transcorrer de um determinado número de épocas) e que, posteriormente, apresentasse o melhor desempenho durante a etapa de testes. Para se esquivar de uma solução fortemente dependente de uma entrada específica, o experimento conduzir-se-ia pela submissão de tais arquiteturas a diferentes *datasets*, a fim de validar sua supremacia em diferentes contextos.

5.1.2 Influência da Arquitetura da Camada de Convolução

5.1.2.1 Dimensão da Camada de Convolução

Analisando na Figura 7 o gráfico de evolução da curva de acurácia para diferentes arquiteturas do modelo A, percebe-se que a dimensão da máscara de convolução impacta diretamente sobre o desempenho da rede, alcançando-se uma acurácia próxima ao ideal através do uso de máscaras de convolução de tamanhos maiores do que a imagem de entrada (padronizadamente na primeira fase, o tamanho da imagem de entrada era 24x24) e quanto menores os valores de dimensão das mesmas, pior foi a acurácia obtida. O mesmo pode ser inferido a partir do desempenho, conforme demonstrado pelos gráficos na Figura 8, percebe-se que a evolução efetua-se cada vez mais rápido conforme o aumento da dimensão das máscaras, tendo a arquitetura A_L5Dx64 obtido a melhor pontuação e a arquitetura A_L5Dx8, a pior.

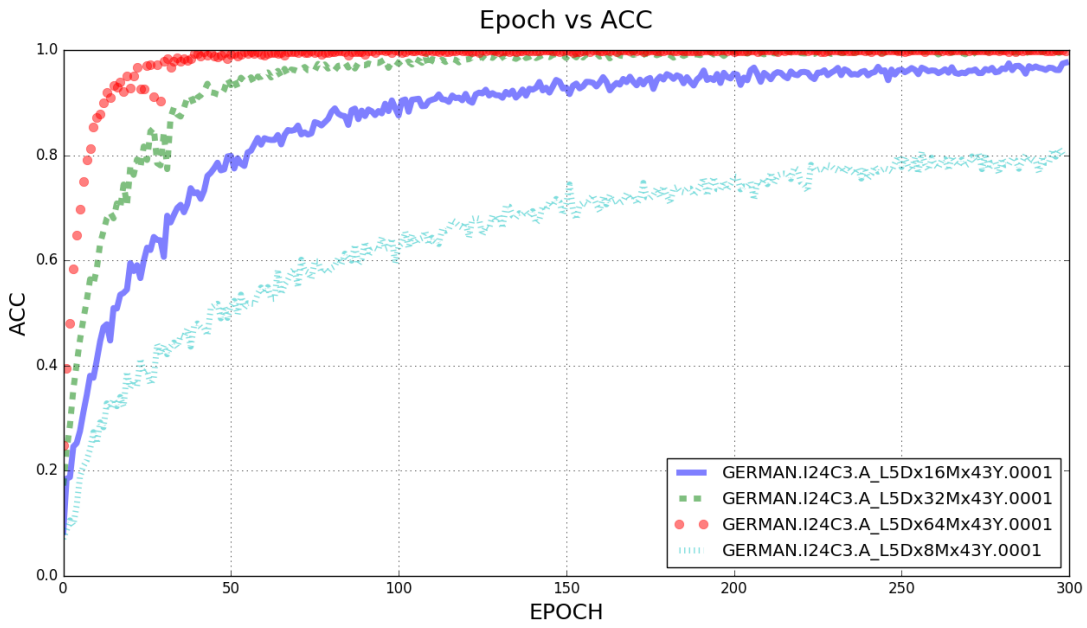

Figura 7 – Evolução da curva de Acurácia para Modelos A_L5Dx

De fato, a dimensão das máscaras de convolução interfere não apenas no limiar de acurácia mas principalmente na curva de aprendizado e, por consequência, arquiteturas construídas sobre máscaras de menor dimensão, além de consumirem um maior número de épocas para progredir, dificilmente atingirão maior valor de acurácia quando comparadas àquelas baseadas em máscaras de dimensões maiores. Além disso, os modelos B, FC e FD evidenciam que as máscaras maiores devem estar posicionadas mais próximas da camada de entrada, se possível a maior máscara de convolução, ou o conjunto destas, deve estar posicionado logo após a imagem de entrada, conforme sugerem os resultados representados

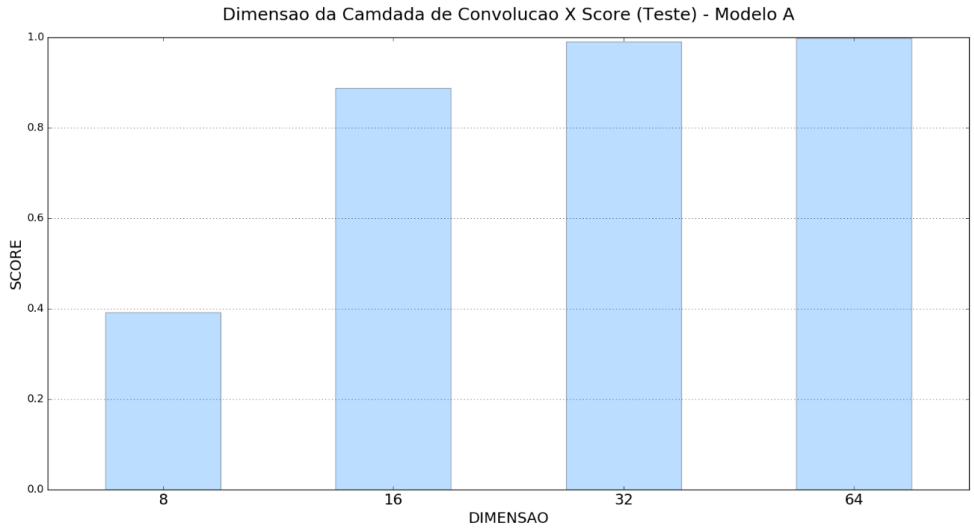

Figura 8 – Comparativo entre arquiteturas do Modelo A para a dimensão da camada de convolução e o desempenho alcançado durante a validação.

pelas Figuras 9 e 11. As Figuras 12 e 13 sugerem que esta diferença de desempenho tende a reduzir-se conforme as dimensões das máscaras de convolução aumentam.

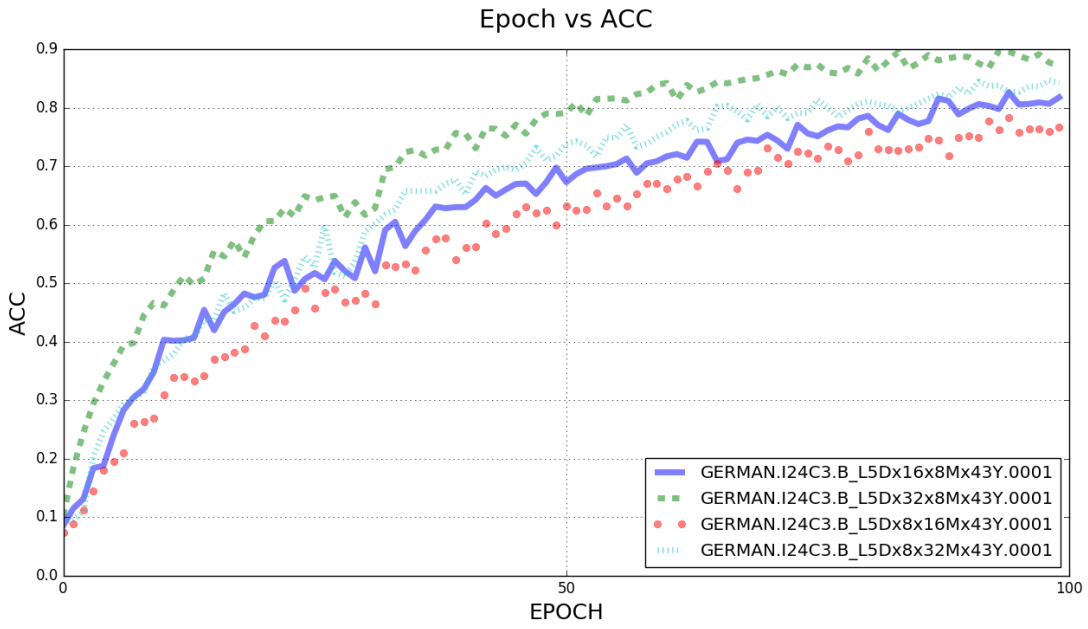

Figura 9 – Evolução da curva de Acurácia para Modelos B_L5 conforme o posicionamento relativo da maior máscara de convolução

Pode-se entender que o objetivo da máscara de convolução é extrair informações relativas às características da imagem através das operações sobre os *kernels* de convolução; dessa forma quanto maior a máscara, maiores serão as informações colhidas a partir

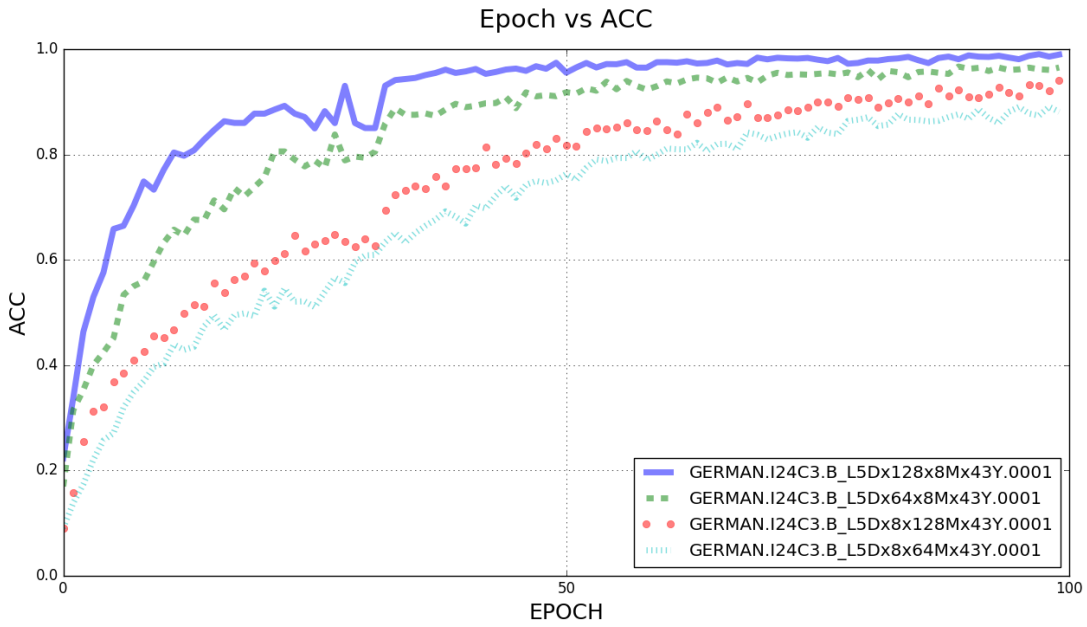

Figura 10 – Evolução da curva de Acurácia para Modelos B_L5 conforme o posicionamento relativo da maior máscara de convolução:

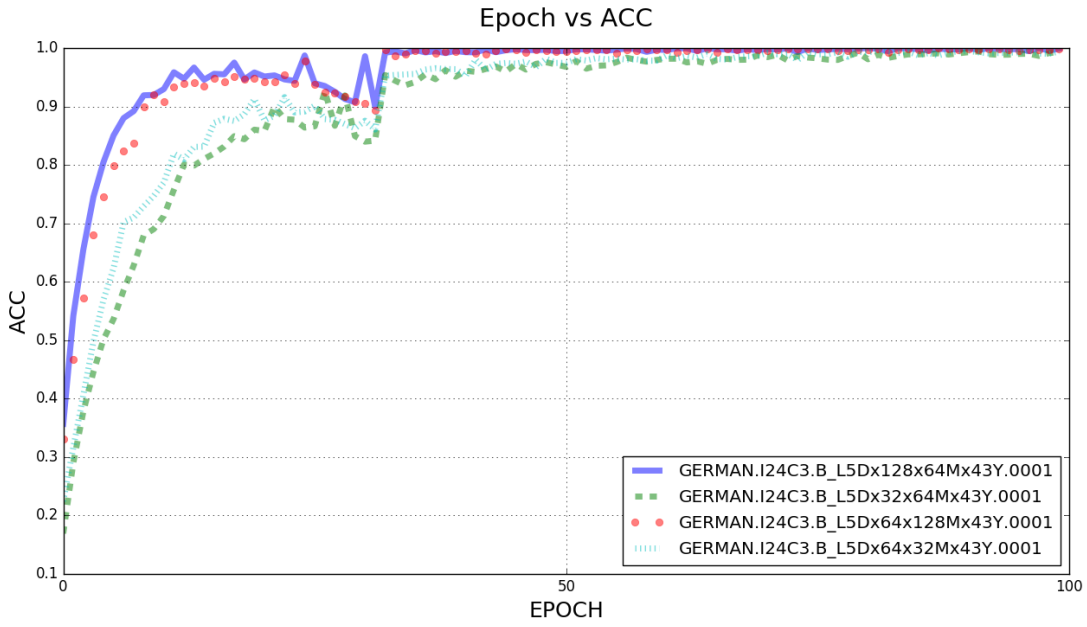

Figura 11 – Evolução da curva de Acurácia para Modelos B_L5 conforme o posicionamento relativo da maior máscara de convolução

da imagem original e repassada às camadas seguintes. Pressupõem-se ainda que utilizar máscaras de convolução maiores do que as dimensões da imagem acaba por extrapolar as informações contidas nesta e, dessa forma, produz um mapa de características com um

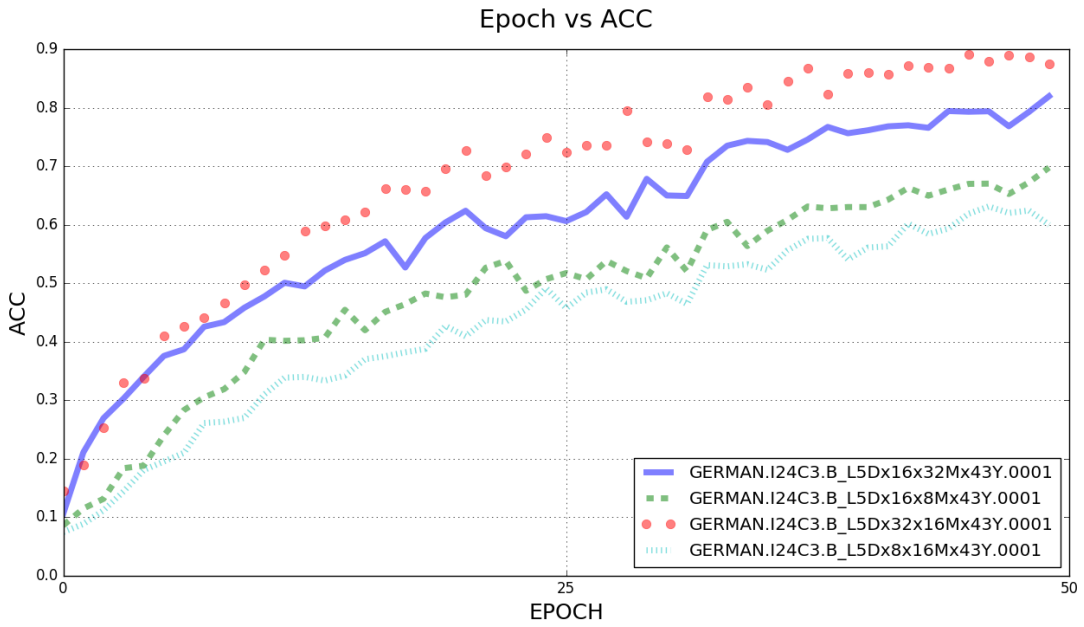

Figura 12 – Comparativo entre a evolução da curva de acurácia e proximidade da maior máscara de convolução para o Modelo B

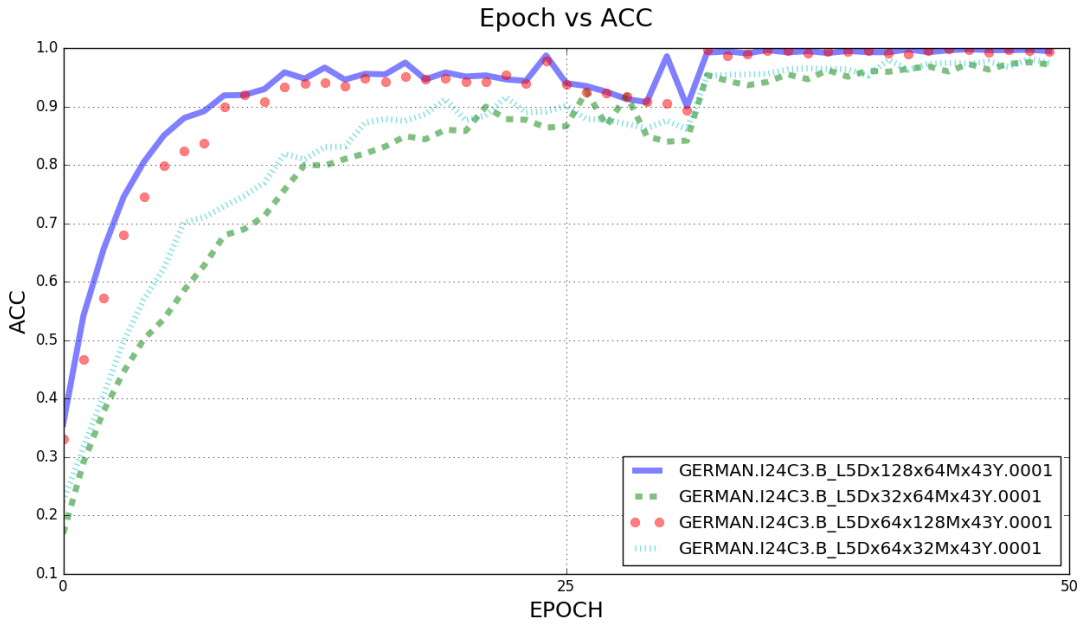

Figura 13 – Comparativo entre a evolução da curva de acurácia e proximidade da maior máscara de convolução para o Modelo B

maior volume de informações e talvez menor ruído, a isto estaria associado o desempenho superior obtido por tais arquiteturas, tal afirmação porém carece de provas matemáticas.

Infere-se ainda que o uso de camadas de *pooling* intercaladas com as camadas

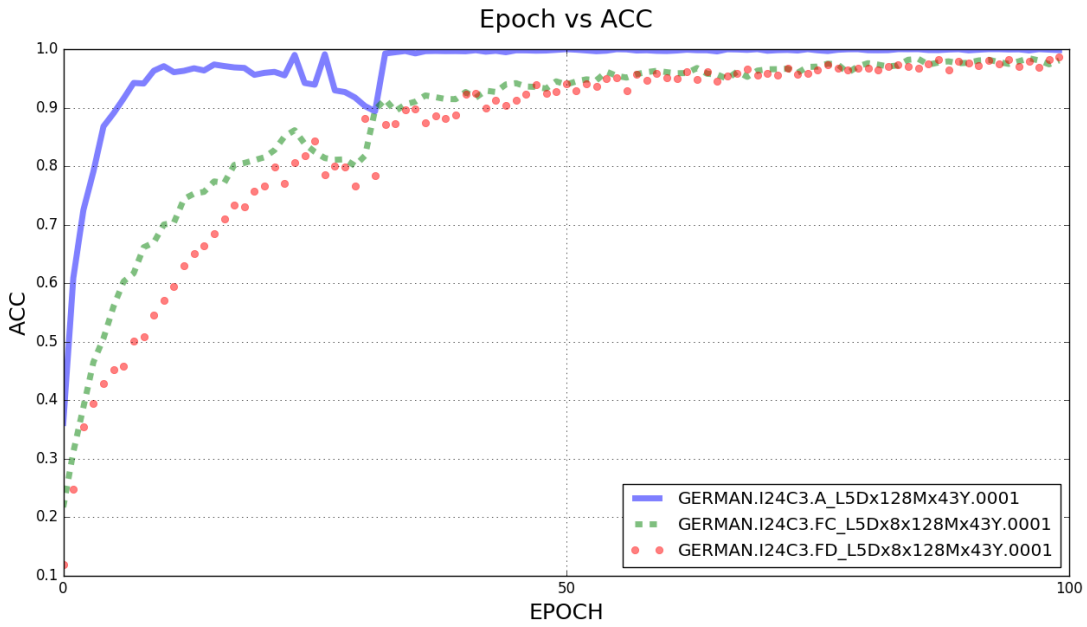

Figura 14 – Comparativo entre a evolução da curva de Acurácia para o melhor modelo A comparado aos modelos FC e FD

convolucionais seria responsável por reduzir ainda mais os níveis de ruído, conforme destaca o volume de informação útil ao simplificar imagem apenas às suas características mais básicas. No entanto, percebe-se que arquiteturas baseadas em máscaras convolucionais de dimensões reduzidas apresentam desempenho comprometido quando submetidas a esta configuração. Uma possível explicação para isto seria que estas produzem um volume de informação pequeno e parte considerável deste estaria sendo descartado em cada operação de *pooling*, dessa forma, após a sequência de operações de *pooling*, restaria apenas um conjunto muito simplificado de mapas de características.

Ainda, caso fosse necessário categorizar as arquiteturas entre boas e ruins e apontar a melhor e a pior dentre elas, baseando-se apenas nos resultados obtidos por esta análise, poder-se-ia facilmente concluir que que para o modelo A, a arquitetura cuja mascara de convolução de tamanho 64 obteve melhor desempenho, e a de tamanho 8 o pior; e para o modelo B, a arquitetura B_Dx128x64 o melhor e B_Dx8x16 o pior desempenho. Conforme esperado, quanto menores forem as máscaras e quanto mais próximas as menores dentre elas estiverem em relação à camada de entrada, piores serão os resultados alcançados.

As arquiteturas derivadas do modelo FC e FD apresentaram um comportamento semelhante às precedentes, conforme crescem as dimensões das máscaras de convolução, porém como pode-se perceber na figura 14, ambas as arquiteturas obtiveram um desempenho inferior ao alcançado pela arquitetura A_L5Dx128, o que significa que a melhor opção ainda é utilizar máscaras de dimensões constantes ou intercaladas.

Modelo/Arquitetura	$\mu Time(s)$	Time_100 (H)
Modelo A		
GERMAN.I24C3.A_L5Dx8Mx43Y.0001	14,77	0,410
GERMAN.I24C3.A_L5Dx16Mx43Y.0001	32,55	0,904
GERMAN.I24C3.A_L5Dx32Mx43Y.0001	54,04	1,501
GERMAN.I24C3.A_L5Dx64Mx43Y.0001	94,47	2,624
GERMAN.I24C3.A_L5Dx128Mx43Y.0001	85,85	2,384
Modelo B_D8x		
GERMAN.I24C3.B_L5Dx8x16Mx43Y.0001	9,984	0,277
GERMAN.I24C3.B_L5Dx8x32Mx43Y.0001	10,37	0,288
GERMAN.I24C3.B_L5Dx8x64Mx43Y.0001	11,30	0,313
GERMAN.I24C3.B_L5Dx8x128Mx43Y.0001	13,93	0,387
Modelo B_D12		
GERMAN.I24C3.B_L5Dx128x8Mx43Y.0001	41,17	1,143
GERMAN.I24C3.B_L5Dx128x16Mx43Y.0001	42,76	1,188
GERMAN.I24C3.B_L5Dx128x32Mx43Y.0001	46,89	1,302
GERMAN.I24C3.B_L5Dx128x64Mx43Y.0001	58,56	1,626
Modelo F		
GERMAN.I24C3.FC_L5Dx8x128Mx43Y.0001	15,34	0,426
GERMAN.I24C3.FD_L5Dx8x128Mx43Y.0001	65,42	1,817

Tabela 4 – Comparativo de tempo de processamento entre diferentes modelos e arquiteturas

5.1.2.2 Comprimento da Camada de Convolução

Os resultados obtidos através dos experimentos não foram suficientes para identificar uma relação direta na influência do comprimento da camada de convolução sobre o desempenho, mas tão somente o seu impacto sobre a curva de aprendizado. De forma geral, arquiteturas mais longas apresentam um crescimento mais lento do que arquitetura mais curtas, porém tal fato nem sempre é verdadeiro, e por esta razão não se pode afirmar a priori que uma arquitetura com menor número de máscaras terá uma curva melhor ou pior que outra com maior número de máscaras, de forma que o alongamento de uma arquitetura pode, ora resultar em melhora, ora em piora.

Para o modelo A, apenas se pode dizer que nenhuma das arquiteturas mais alongadas alcançaram algum grau de melhora sobre a arquitetura L5, como se percebe pelas Figuras 15 e 16, embora a Figura 17 sugere que esta diferença de desempenho tende a reduzir-se conforme o aumento da dimensão das máscaras de convolução. O Modelo FC apresenta curvas mais definidas quanto a proporção inversa entre comprimento da camada de convolução/desempenho, sendo que conforme ocorre o alongar da camada de convolução há uma queda progressiva no desempenho durante o treinamento 18.

Dessa forma assume-se que o alongar da camada de convolução não proporciona

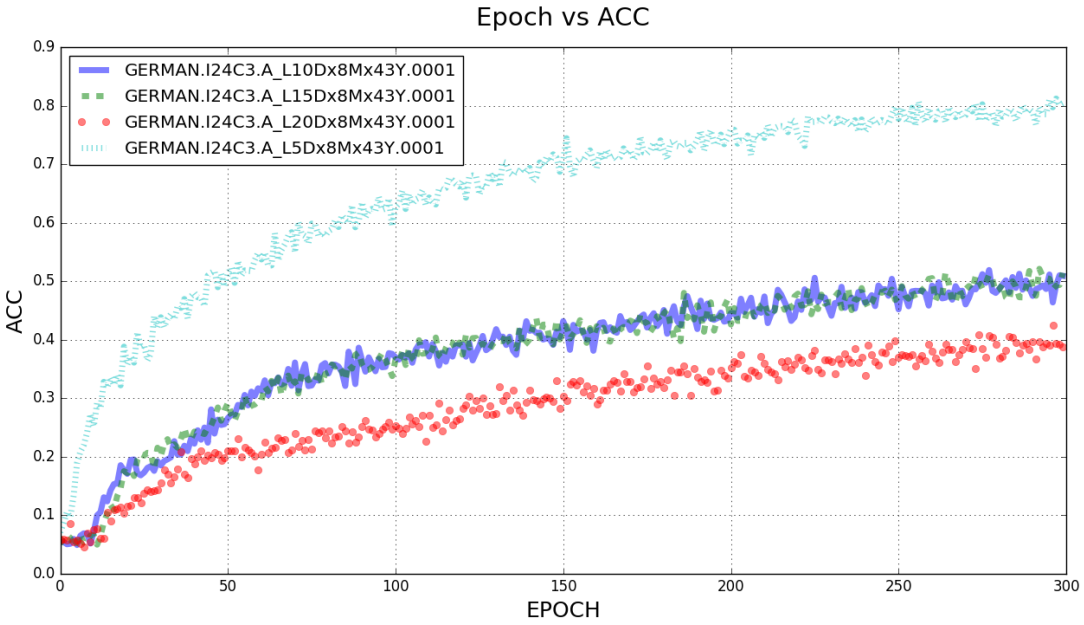

Figura 15 – Curva de Evolução da Acurácia para o modelo A_L?D8

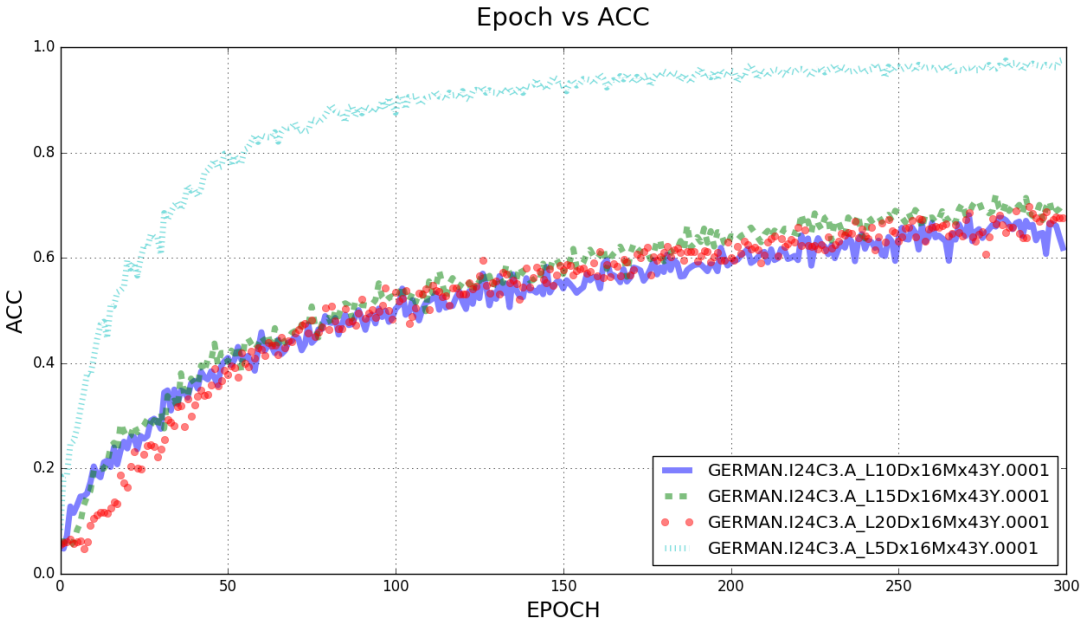

Figura 16 – Curva de Evolução da Acurácia para o modelo A_L?D16

melhorias diretas sobre a curva de aprendizado, seja este fenômeno ocasionado pelo alongamento das camadas de *pooling*, ou pelo aumento da complexidade do modelo; percebe-se ainda que conforme se aumenta a dimensão das máscaras de convolução, menor se torna o efeito negativo do alongamento da camada de convolução: enquanto para o modelo A_Dx8 cuja dimensão da máscara de convolução é 8x8 pixeis, o valor final de

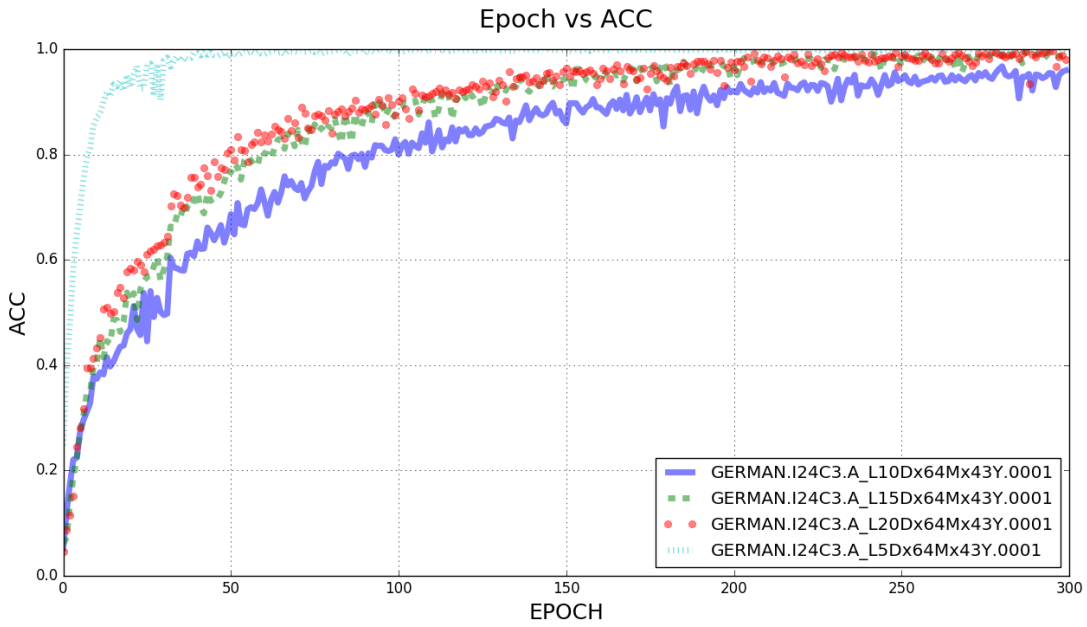

Figura 17 – Curva de Evolução da Acurácia para o modelo A_L?D64

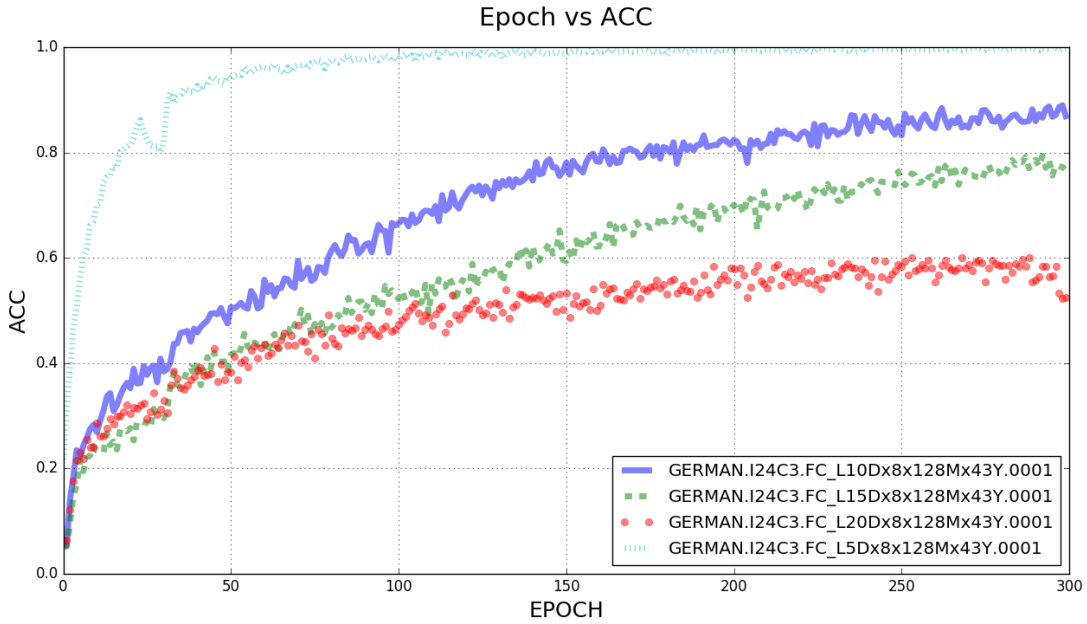

Figura 18 – Evolução da curva de Acurácia para o Modelo FC_L?D?.

acurácia apresenta uma queda de 40% entre a arquitetura L5, contendo cinco camadas convolucionais de comprimento, em relação à arquitetura L20; para o Modelo A_Dx64, esta queda reduz-se a uma diferença de apenas 5%.

Portanto, conclui-se que tendo como base apenas dois parâmetros, dimensão da máscara de convolução e comprimento, aparentemente um menor número de máscaras

de convolução, de dimensões maiores, é a opção mais interessante para a obtenção de bons resultados. O alongamento da camada de convolução torna mais longo o processo de treinamento, e por si só tende a não proporcionar melhoras significativas sobre o desempenho, sendo até mesmo prejudicial a este. Verá-se mais adiante que os modelos de maior desempenho são, em grande maioria, baseados em arquiteturas L5. Doravante pode-se afirmar que camadas de convolução muito extensas tendem a reduzir a eficácia da rede neural quando treinadas sobre um mesmo número de épocas que arquiteturas mais curtas; em outras palavras, arquiteturas mais simples obtém melhores resultados após um número relativamente pequeno de épocas.

5.1.2.3 Comparação entre os Modelos

Pela análise experimental identifica-se uma relação direta entre dimensão das máscaras de convolução e desempenho, o mesmo não pode ser dito acerca do comprimento da camada de convolução. Arquiteturas derivadas do Modelo A apresentam bom desempenho; arquiteturas derivadas do modelo B apresentam desempenho ligeiramente inferior às primeiras, porém, quase sempre consomem menor tempo de processamento (ver Tabela 4) e, portanto, dispor camadas grandes intercaladas com pequenas na forma NxMxN... demonstra-se uma estratégia vantajosa.

Adicionalmente, em alguns casos os modelos FD apresentaram desempenho ligeiramente superior ao modelo FC, porém ambos apresentam-se insatisfatórios quando comparados aos modelos A e B, considerando que estes últimos na maior parte das vezes alcançam desempenhos muito próximos ao desempenho ideal; de fato, como demonstra a tabela 7, todos os modelos foram capazes de produzir arquiteturas as quais alcançaram desempenho **SCR** superior a 99%, porém as cinco arquiteturas capazes de atingir valores acima de 99,9% são derivadas dos modelos A e B, sendo estes considerados, por consequência, modelos superiores.

5.1.2.4 Supressão da Camada de *Pooling*

Na seção anterior, observou-se a relação direta entre desempenho e dimensão da camada de convolução; embora houvesse dúvida quanto à influência da camada de *pooling* estar descartando informações essenciais para a classificação, e por esta razão estariam as camadas de dimensões reduzidas apresentando um desempenho inferior em relação às maiores, e a suposição de que esta regra deixaria de prevalecer com a supressão da camada de *pooling*, os resultados do experimento a reafirmam: confirmou-se a mesma relação através do uso de arquiteturas sem *pooling*, e a partir de tal, valida-se a ideia de que o desempenho é proporcional ao tamanho das máscaras de convolução, fenômeno equivalente ao observado durante a Etapa I. O desempenho é proporcional à dimensão da máscara de convolução e as máscaras de maior tamanho devem ser posicionadas mais

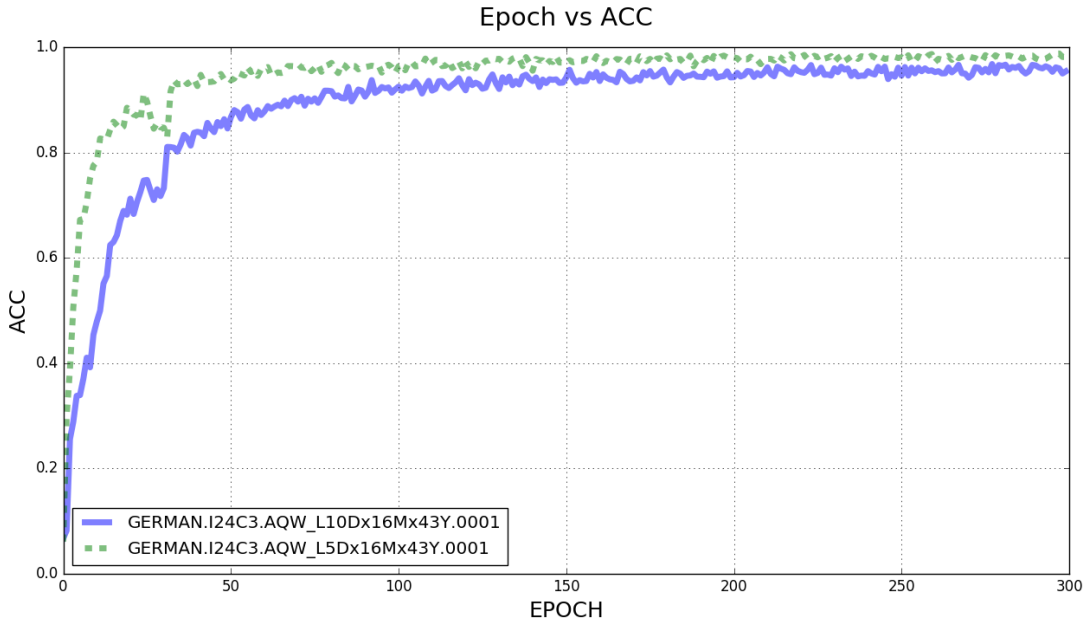

Figura 19 – Evolução da curva de acurácia para o modelo AQW_L?D16

próximas da camada de entrada.

Quanto à influência do comprimento da camada de convolução, a não existência de camada de *pooling* não interferiu sobre a tendência do desempenho apresentar-se inversamente proporcional ao comprimento da camada, porém o efeito deste alongamento revelou-se mais atenuado em relação às arquiteturas com *pooling*, como o observado na Figura 19. De qualquer forma, a supressão da camada de *polling* não foi suficiente para que o alongamento da camada de convolução resultasse arquiteturas superiores às arquiteturas L5.

Dessa forma, chega-se à conclusão de que a função da camada de *pooling* é otimizar o custo, uma vez que sua supressão torna o processamento durante a etapa de treinamento tarefa dispendiosa, consumindo muito tempo para computar uma única época. Entretanto, não se pode ignorar o fato de que os efeitos negativos do *pooling* expressam-se com maior evidência conforme reduz-se o tamanho das máscaras de convolução, desfavorecendo o desempenho de arquiteturas cujas dimensões sejam pequenas, uma vez que produzirão menor volume de informação, a qual simplificar-se-á ainda mais após as operações de *pooling*, e ao fim deste processo, o volume de características fornecidas ao classificador será diminuto.

Conclui-se portanto que utilizar camadas de *pooling* intercaladas a cada camada convolucional é uma forma de otimizar o tempo de computação, porém ao mesmo tempo é capaz de reduzir a capacidade de generalização da rede; de forma contrária, a dimensão da máscara convolucional, quanto menor, mais rápido conduzir-se-á o treinamento e menos

efetiva será a generalização; em relação ao alongamento da mesma, quanto maior, mais dispendioso tornar-se-á o treinamento, uma vez que é realizado um maior número de computações entre os *kernels* de convolução antes de que a informação chegue à camada totalmente conectada.

Ainda, seria injustificável a supressão da camada de *pooling* apenas sobre o argumento de que esta reduz o desempenho de arquiteturas baseadas em máscaras de convolução menores, uma vez que um desempenho superior poderia ser alcançado através de arquiteturas cujas as máscaras de convolução sejam maiores e estejam intercaladas por camadas de *pooling*, em um tempo consideravelmente inferior ao gasto pelas primeiras.

5.1.2.5 Conclusões sobre a Camada Convolucional

Efetuada a fase de experimentação da camada convolucional, obteve-se a relação das melhores arquiteturas, ordenadas pelo grau de acurácia e desempenho alcançados durante as etapas de treinamento e validação; neste tópico são levantadas as principais conclusões acerca das melhores configurações e o caminho para alcançar um bom nível de desempenho explorando a complexidade do extrator de características.

Pela análise da série de modelos A e B é possível inferir que a melhor opção seria utilizar modelos cuja camada de convolução seja mais curta, porém de dimensões maiores. A Tabela 7 demonstra os valores alcançados por tais modelos quando submetidos à validação, através da mesma verifica-se a não ocorrência de *overfitting* pela comparação entre o valor médio e máximo de acurácia e o coeficiente de acerto durante a validação, na última coluna, tem-se a relação entre a acurácia média e o coeficiente de validação, quanto maior este valor, melhor o desempenho da arquitetura em ambas as etapas.

Ainda através dos resultados obtidos a partir dos experimentos envolvendo as camadas convolucionais, é possível perceber que o não uso de max *pooling* acaba por comprometer o desempenho final da rede durante o teste ainda que mantenha rendimento compatível durante a fase de treinamento, isto sugere que a não utilização do *pooling* favorece a ocorrência de *overfitting*; supõem-se que a operação do *pooling* possa de certa forma filtrar o ruído dos dados de entrada, uma vez que simplifica a representação da imagem preservando apenas suas características relevantes.

Pela tabela 7 infere-se que os oito melhores resultados, os quais obtiveram desempenho superior a 0.999 são arquiteturas derivadas dos modelos A e B, sendo este último em especial, mais presente dentre as arquiteturas de sucesso, por esta razão supõem-se que o melhor modelo seria derivado da combinação de tais arquiteturas. Para a próxima etapa de experimentação, foram utilizados vários padrões de modelos dentre os quais foram chamados série de melhores modelos, dessa forma cada um destes após ser testado quanto ao seu desempenho, foi então submetido aos testes da próxima etapa.

Modelo	**mTime (s)**	**Time_100 (H)**
GERMAN.I24C3.AQ_L5Dx8Mx43Y.0001	14,77	1,23
GERMAN.I24C3.AQ_L5Dx8M128x43Y.001	27,87	2,32
GERMAN.I24C3.AQ_L5Dx8Mx8x43Y.001	28,88	2,40
GERMAN.I24C3.AQ_L5Dx8Mx8x128x43Y.001	29,64	2,47
GERMAN.I24C3.AQ_L5Dx8Mx8x16x43Y.001	31,63	2,63
GERMAN.I24C3.AQ_L5Dx8M64x43Y.001	31,67	2,63
GERMAN.I24C3.AQ_L5Dx8M32x43Y.001	32,23	2,68
GERMAN.I24C3.AQ_L5Dx8Mx16x43Y.001	32,51	2,70
GERMAN.I24C3.AQ_L5Dx8Mx8x32x43Y.001	33,79	2,81
GERMAN.I24C3.AQ_L5Dx8Mx8x64x43Y.001	37,10	3,09

Tabela 5 – Comparativo entre o tempo de treinamento e arquitetura do classificador

5.1.3 A Camada Totalmente Conectada

Durante a Fase I estudaram-se as principais configurações da camada de convolução; esta etapa objetiva-se a melhorar a capacidade de generalização do classificador. Dessa forma, ao término desta, identificada uma boa configuração para o classificador, deseja-se obter uma arquitetura mista a partir da melhor arquitetura de camada totalmente conectada acoplada à melhor arquitetura de camada de convolução desenvolvida na Fase I.

Em uma primeira abordagem, supõem-se que o classificador atue de forma independente da camada de convolução e dessa forma uma melhor arquitetura da camada totalmente conectada resultaria em uma melhora de desempenho por ser capaz de aproveitar com maior eficiência a informação extraída por essa.

Devido a este fato, e também a fim de otimizar o tempo de treinamento, realizou-se o experimento utilizando como modelo padrão, uma arquitetura de camada convolucional A_L1D8 e A_L5D8, sobre a qual foram desenvolvidos outras 25 arquiteturas pela variação da configuração do classificador. Além disso, alterou-se a taxa de aprendizado para o valor de 0,001.

Na Figura 21 observa-se um gráfico comparativo ente os melhores resultados para a arquitetura do classificador acoplado à arquitetura A_L5D8. Percebe-se que as arquiteturas dotadas de um classificador mais sofisticado foram capazes de gerar um ganho de desempenho em relação a arquiteturas baseadas em um classificador contendo apenas a camada de saída.

Para uma arquitetura cuja a camada de convolução é simples, a complexidade do classificador tende a melhorar consubstancialmente a sua acurácia durante o treino, conforme demonstra a Figura 20. Por outro lado, um bom extrator de característica seria suficiente em produzir um bom resultado, ainda que associado à um classificador mediano; de mesma forma, o inverso é equivalente, sendo possível encontrar um classificador suficientemente bom, capaz de alcançar boa acurácia, ainda que o extrator apresente

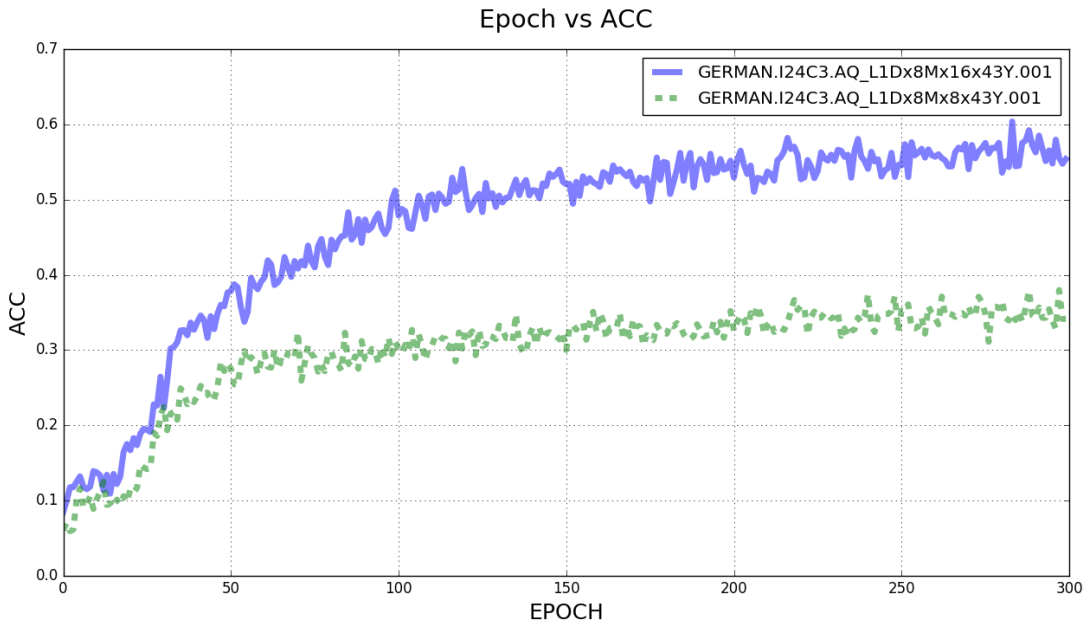

Figura 20 – Evolução da curva de acurácia para diferentes configurações do classificador

limiares baixos.

Dessa forma, o desempenho de uma rede pode evoluir sobre duas vias distintas: ou o modelo se aperfeiçoa através da complexidade do extrator, ou pela complexidade do classificador; haveria ainda um meio termo onde ambos seriam medíocres, porém complementares entre si.

Deve-se manter cuidado ao incrementar a complexidade do classificador, pois ocasionalmente, obtem-se resultados indesejados como o expresso na Figura 22, adicionalmente pela análise do tempo de treinamento para diferentes modelos de classificador, percebe-se que a medida que a complexidade deste aumenta o tempo de treinamento também se alonga.

Ainda, analisando-se o tempo de treinamento, outra desvantagem identificada é que aumentar a complexidade de ambos, extrator e classificador, ocasionaria um custo computacional elevado, visto que muitas computações fariam-se necessárias durante a etapa de convolução e no ajuste dos pesos da camada totalmente conectada (ver Tabelas 4 e 5), além do mais, atribuir a responsabilidade ao classificador aumenta as chances de *oveffintting* conforme visto na fundamentação teórica.

Sobre arquiteturas dotadas de um extrator sofisticado, a presença de um classificador mais eficiente não reflete em melhora de desempenho. Na Tabela 6 percebe-se que os modelos foram capazes de produzir desempenhos próximos aos produzidos na Fase I, porém foram incapazes de superá-los.

Arquitetura	**mACC**	**ACC,0**	**SCR,0**	**SCR***
GERMAN.I24C3.AQ_L5Dx16Mx32x43Y.0001	0,68	0,864	0,558	0,38
GERMAN.I24C3.AQ_L5Dx32Mx32x43Y.0001	0,85	0,978	0,894	0,76
GERMAN.I24C3.AQ_L5Dx64Mx32x43Y.0001	0,93	0,995	0,983	0,92
GERMAN.I24C3.A_L5Dx32Mx43Y.0001	0,94	0,993	0,991	0,93
GERMAN.I24C3.A_L5Dx64Mx43Y.0001	0,97	0,999	0,999	0,97

Tabela 6 – Comparativo entre Fase I e Fase II

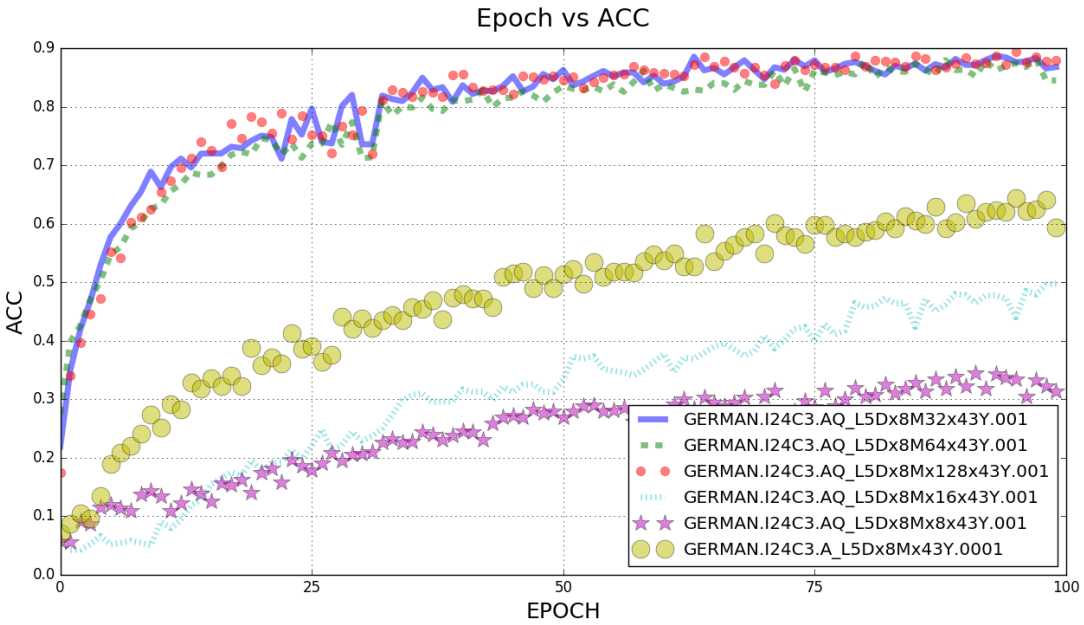

Figura 21 – Evolução da Curva de Acurácia para diferentes configurações do classificador

Dessa forma os resultados obtidos a partir do aumento de complexidade do classificador foram insatisfatórios, conforme foram incapazes de resultar em um ganho de desempenho quando associado a extratores eficazes (ver Tabela 6); dessa forma é preferível aumentar a complexidade do extrator de características e acoplá-lo a um classificador simples, sendo esta uma estratégia mais promissora através da qual foi possível obter um desempenho próximo ao ideal.

Chega-se à conclusão de que sobre o *dataset* GTSRB um extrator de características complexo é suficiente para a obtenção de resultados próximos do ideal. Nada se pode afirmar antecipadamente para outros *datasets*, sendo necessário validar tal desempenho empiricamente. Por esta razão, apontou-se as arquiteturas produzidas na Fase I como sendo melhores do que aquelas abordadas nesta seção.

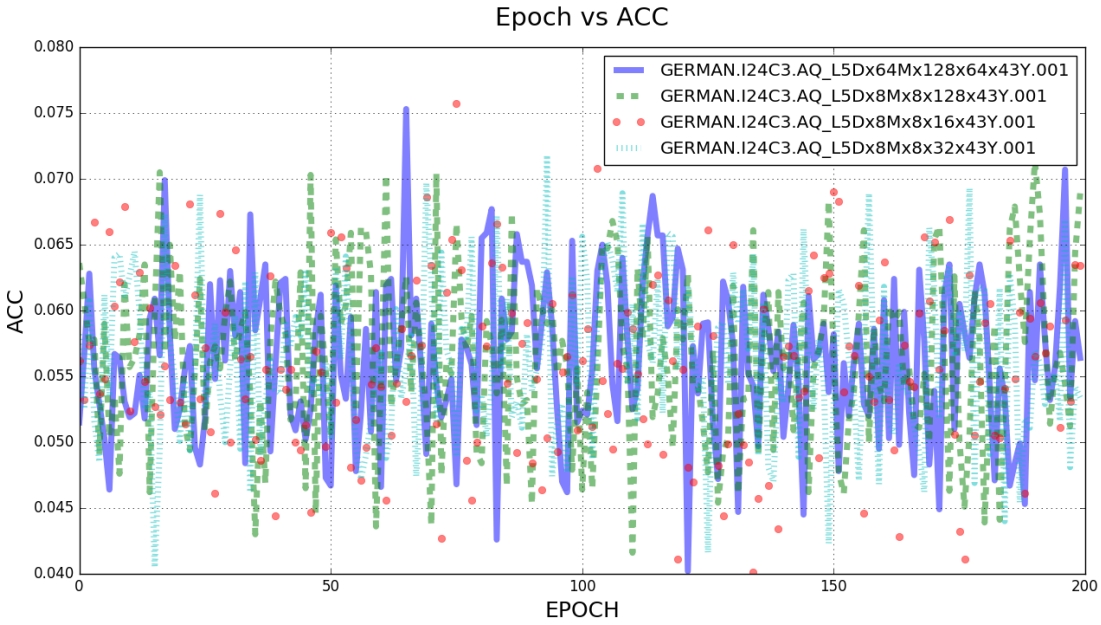

Figura 22 – Evolução da Curva de Aprendizado AQW_L?D16

5.1.4 Imagem de Entrada

5.1.4.1 A Dimensão da Imagem de Entrada

Variou-se o tamanho da imagem de entrada segundo o parâmetro I, entre os valores I8, I16, I32. Analisando a Figura 23 infere-se que o desempenho é proporcional ao tamanho da imagem de entrada; tal fato é facilmente justificado assumindo-se que a resolução da imagem é proporcional ao seu tamanho, visto que um maior número de píxeis seriam capazes de fornecer maior quantidade de informação. Adicionalmente, a própria operação de reescala (*resize*), na qual se ajusta uma imagem a um tamanho menor, reduz não apenas sua resolução, mas também descarta informações, conforme exclui linhas e colunas de píxeis, assim resultando em uma representação de dados com menor volume de características.

5.1.4.2 O Modelo de Cores e Representação dos Dados

Realizaram-se os testes concernentes à coloração, no qual cada imagem do *dataset* foi convertida em tons de cinza e reduzida a um único canal, atribuindo a cada pixel a tonalidade média dos três canais originais. Para agilizar o tempo de processamento, todas arquitetura testadas são variantes do modelo A_L5D8; ainda, devido à baixa curva de aprendizado apresentada pelos testes preliminares, decidiu-se aumentar a taxa de aprendizado de tais arquiteturas, sendo esta dez vezes maior do que aquela utilizada no aprendizado das arquiteturas multicanal.

Aparentemente a alteração da coloração da imagem provoca um efeito negativo

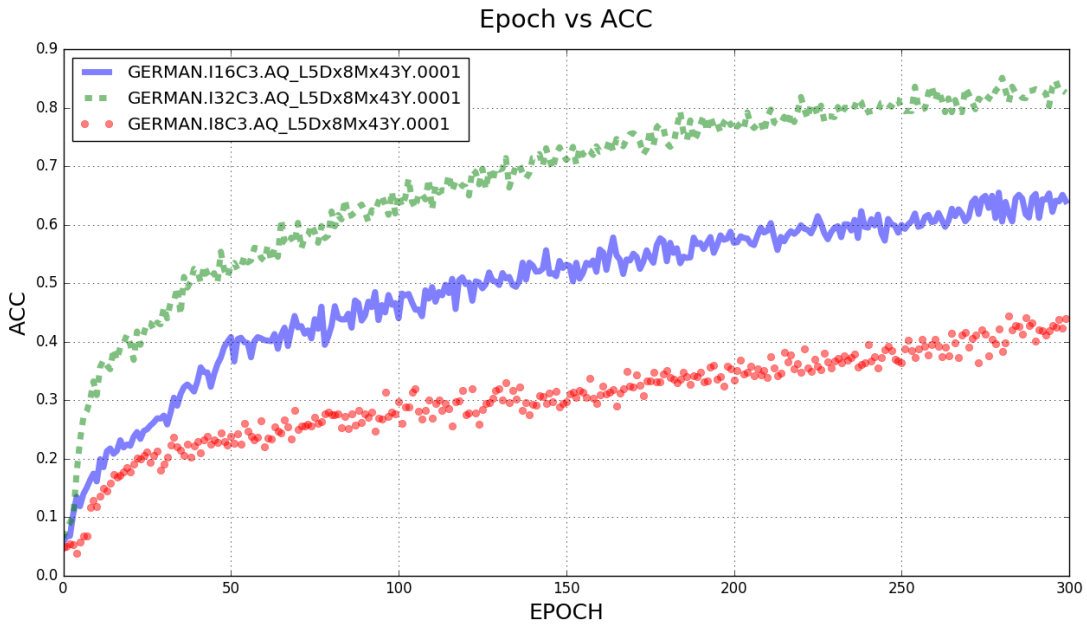

Figura 23 – Evolução da Curva de Acurácia para diferentes tamanhos de imagens de entrada

sobre o treinamento. Não se sabe ao certo a qual fator estaria este associado porém, considera-se que a presença de um único canal reduz o volume de características fornecido ao classificador; resta a dúvida quanto ao efeito da conversão a tons de cinza preservando-se os três canais. Conforme observa-se na Figura 24, todos os experimentos conduzidos na presença de um único canal obtiveram um desempenho inferior às arquiteturas multicanal.

5.2 A Arquitetura Definitiva e Submissão à outros *datasets*

Algumas das melhores arquiteturas desenvolvidas na Fase I e II foram submetidas aos *datasets* Belga e Chinês a fim de determinar a robustez de cada arquitetura quando submetida à diferentes *datasets*. Nesta seção, apresentar-se-ão os modelos para os quais obteve-se a maior taxa de sucesso, assim como os seus valores de acurácia durante o treino e a pontuação obtida durante a validação para os três *datasets*: alemão, belga e chinês. Deseja-se concluir as características das arquiteturas mais eficientes e validar sua robustez quando submetidas à *datasets* diferentes.

Observando a Tabela 7, onde se encontram os melhores resultados da Fase I, conclui-se que as arquiteturas desenvolvidas durante esta fase mostraram-se mais eficazes aos propósitos de classificação do que suas sucessoras na Fase II e III, tendo inclusive obtido resultados melhores do que aqueles auferidos pelas equipes vencedoras do campeonato alemão. Assim, a utilização de tais arquiteturas seriam suficientes para a obtenção de resultados próximos ao ideal sobre o *dataset* alemão de classificação (GTSRB). Neste

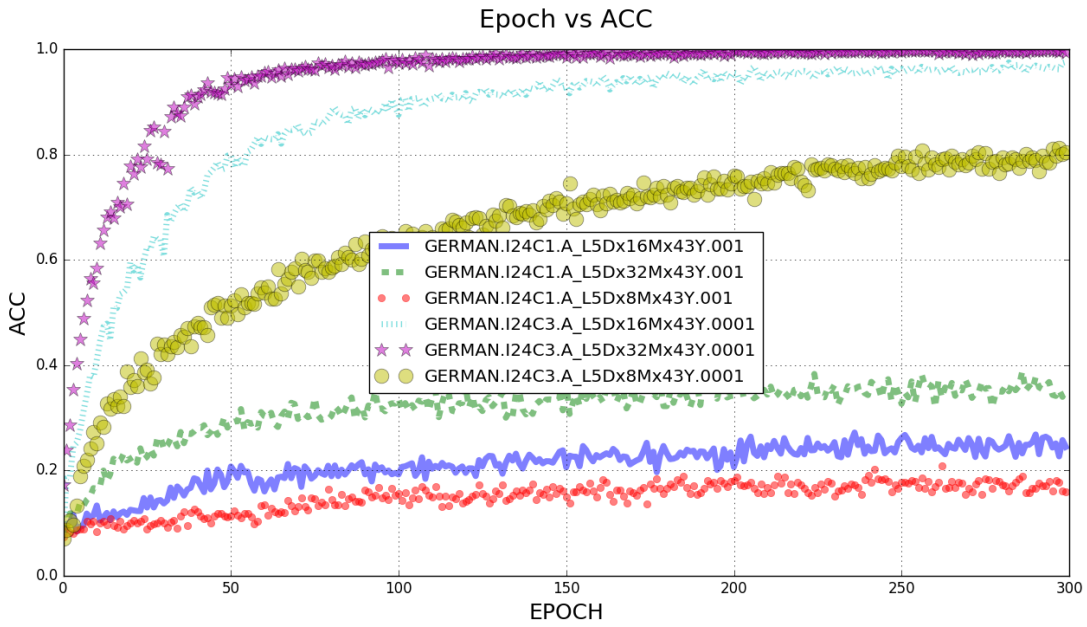

Figura 24 – Comparativo entre a evolução da curva de acurácia para modelos de cores RGB e monocanal

momento restaria apenas a dúvida quanto ao seu desempenho quando submetidas a outros *datasets*.

Abaixo apresentam-se os gráficos nas Figuras 25 e 26, onde se pode observar as curvas de evolução de acurácia das melhores arquiteturas da Fase I submetidas aos *datasets* Belga e Chinês. Conforme se percebe, tais curvas apresentam comportamento semelhante às curvas produzidas pelas mesmas arquiteturas durante a Fase I, sendo o maior valor de acurácia alcançado pela arquitetura A_L5Dx128, sobre todos os *datasets*. A Tabela 8 traz valores referentes ao processo como um todo, tendo os valores da acurácia média e máxima durante o treinamento e o desempenho durante teste (SCR), pela qual é possível detectar a presença de *overfitting* para ambos os *datasets*, Belga e Chinês, conforme os valores de SCR são relativamente baixos comparados aos valores de ACC e μACC.

Deste fato infere-se que infelizmente a arquitetura desenvolvida neste trabalho demonstrou-se extremamente bem adequada ao *dataset* alemão; porém teve um baixo rendimento quando submetida a outros *datasets*. Tal fato poderia estar atrelado à natureza destes *datasets*, devido à baixa amostragem de imagens, em relação ao *dataset* alemão. A fim de esclarecer tal dúvida, em um último teste validou-se o comportamento do mesmo conjunto de arquiteturas sobre o *datasets* CIFAR disponível na biblioteca do próprio TensorFlow.

Devido o *overfitting* apresentado na validação sobre outras bases de dados, decidiu-se averiguar qual seria o efeito de aumentar a complexidade do classificador e repetir os

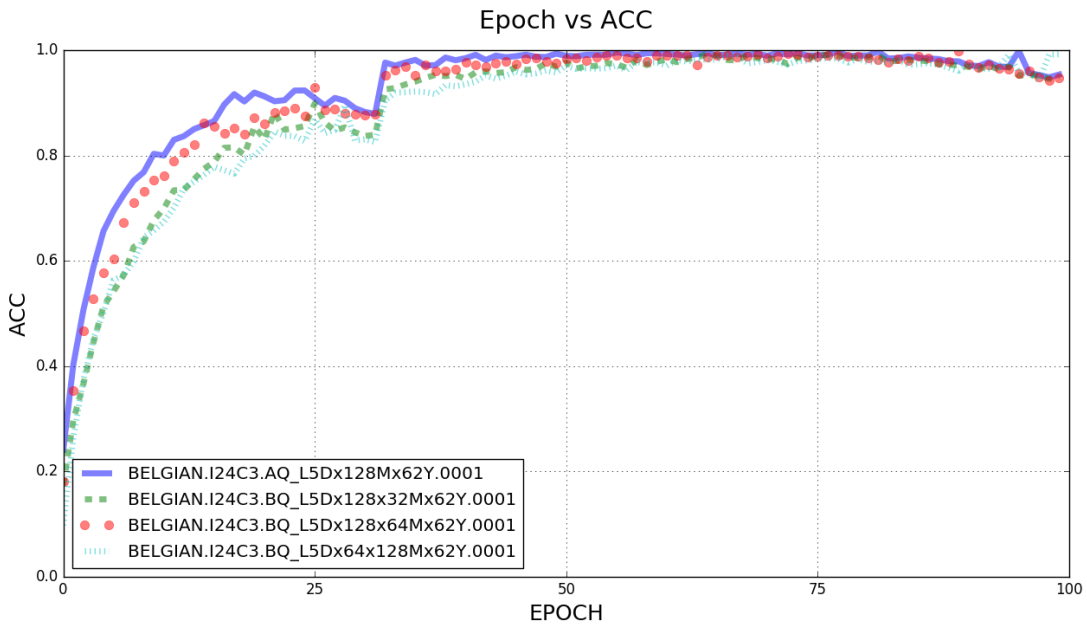

Figura 25 – Evolução da curva de acurácia para as melhores arquiteturas sobre o *Dataset* Belga

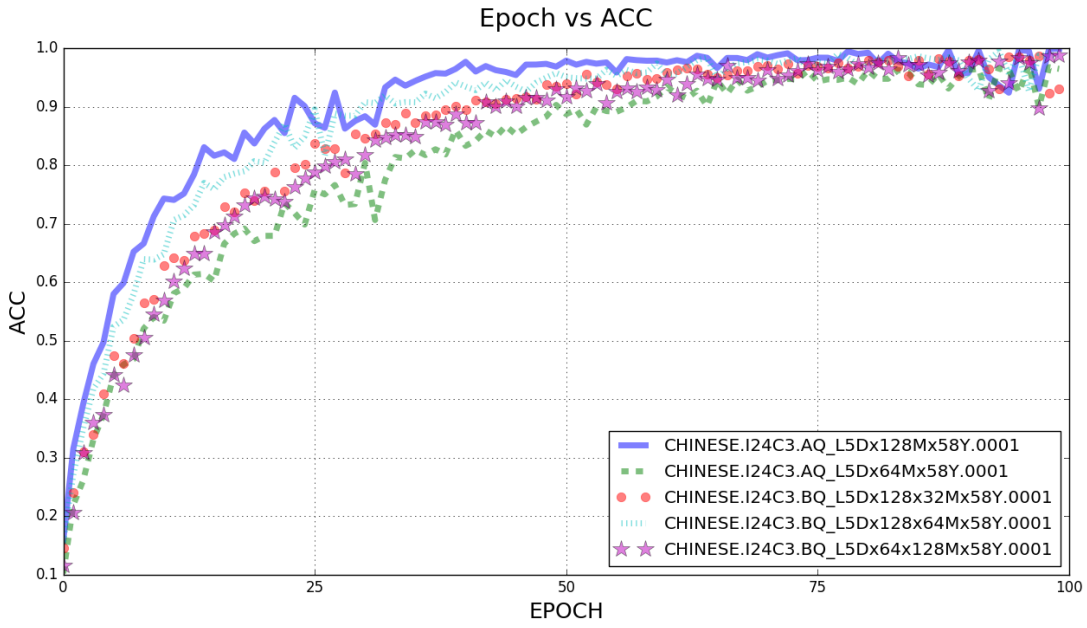

Figura 26 – Evolução da curva de acurácia para as melhores arquiteturas sobre o *Dataset* Chinês

Modelo	**L**	**D**	**μACC**	**ACC**	**SCR**	**SCR* μACC**
A*	**5**	**x128**	**0,988**	**1**	**0,999**	**0,988**
B*	**5**	**x128x64**	**0,986**	**1**	**0,999**	**0,985**
B	5	x64x128	0,984	1	1	0,984
B	5	x128x32	0,981	1	0,999	0,980
A	5	x64	0,979	0,999	0,999	0,978
B	5	x128x16	0,974	0,9978	0,999	0,974
B	5	x32x128	0,970	0,9992	0,999	0,969
B	5	x64x32	0,968	0,9968	0,999	0,968
B	5	x32x64	0,963	0,9999	0,997	0,961
B	5	x128x8	0,964	0,9962	0,995	0,960
B	5	x64x16	0,956	0,9984	0,994	0,951
FC	5	x8x128	0,952	0,9984	0,991	0,944
FD	5	x8x128	0,942	0,9976	0,998	0,941
*A***	*10*	*x128*	*0,921*	*0,9998*	*0,999*	*0,920*
*B***	*20*	*x128x64*	*0,916*	*0,9975*	*0,997*	*0,914*
*B***	*15*	*x128x64*	*0,897*	*0,9905*	*0,996*	*0,894*
AQW	5	x16	0,894	0,9649	0,995	0,890
B	10	x64x128	0,892	0,9969	0,997	0,890
*B***	*10*	*x128x64*	*0,879*	*0,9949*	*0,995*	*0,875*
B	10	x128x32	0,859	0,9929	0,995	0,854

Tabela 7 – Melhores modelos para a Fase I

Modelo	**μACC**	**ACC**	**SCR**	**SCR* μACC**
BELGIAN.AQ_L5Dx128	0,9658	0,992	0,7829	0,7561
BELGIAN.AQ_L5Dx64	0,9532	0,991	0,7843	0,7475
BELGIAN.BQ_L5Dx128x32	0,9526	0,991	0,7988	0,7609
BELGIAN.BQ_L5Dx128x64	0,9603	0,994	0,7869	0,7557
BELGIAN.BQ_L5Dx64x128	0,9490	0,992	0,8146	0,7731
CHINESE.AQ_L5Dx128	0,9579	0,999	0,3590	0,3439
*CHINESE.AQ_L5Dx64***	*0,9262*	*0,997*	*0,3761*	*0,3483*
CHINESE.BQ_L5Dx128x32	0,9424	0,962	0,4002	0,3771
CHINESE.BQ_L5Dx128x64	0,9537	0,903	0,4122	0,3931
CHINESE.BQ_L5Dx64x128	0,9379	0,995	0,3951	0,3706
GERMAN.AQ_L5Dx128	0,9892	0,999	0,9996	0,9888
GERMAN.AQ_L5Dx64	0,9665	0,999	0,9984	0,9651
GERMAN.BQ_L5Dx128x32	0,9823	0,998	0,9975	0,9799
GERMAN.BQ_L5Dx128x64	0,9856	1,000	0,9997	0,9854
GERMAN.BQ_L5Dx64x128*	**0,9822**	**0,999**	**0,9992**	**0,9815**
CIFAR.AQ_L5Dx64Mx43Y.001	0,89	0,925	0,780	0,69

Tabela 8 – Análise de desempenho das cinco melhores arquiteturas submetidas a outros *datasets*

Arquitetura	**μACC**	**ACC**	**SCR**	**SCR* μACC**
GERMAN.I24C3.AQ_L5Dx16Mx32x43Y.0001	0,68	0,864	0,558	0,38
GERMAN.I24C3.AQ_L5Dx32Mx32x43Y.0001	0,85	0,978	0,894	0,76
GERMAN.I24C3.AQ_L5Dx64Mx32x43Y.0001	0,93	0,995	0,983	0,92
GERMAN.I24C3.A_L5Dx32Mx43Y.0001	0,94	0,993	0,991	0,93
GERMAN.I24C3.A_L5Dx64Mx43Y.0001	0,97	0,999	0,999	0,97
CIFAR.I32C3.AQ_L5Dx64Mx43Y.001	0,89	0,925	0,780	0,69

Tabela 9 – Desempenho alcançado com uso de classificador complexo sobre outras bases de dados

procedimentos realizados anteriormente. Os resultados expressos na Tabela 9 demonstram que a substituição do classificador foi incapaz de proporcionar alguma alteração significativa sobre os resultados obtidos com uma única camada totalmente conectada. Dessa forma tais valores referem-se ao fato de que mesmo com o uso de um classificador mais sofisticado, os resultados obtidos não foram capazes de produzir desempenho semelhante ao alcançado por estas mesmas arquiteturas sobre o GTSRB.

6 Conclusão e Trabalhos Futuros

Ao término deste trabalho, tem-se um conjunto de arquiteturas bem ajustadas e que obtiveram relativo sucesso em classificar com precisão o *German Traffic Sign Recognition Benchmark*, tendo produzido dentre os melhores valores, 100% de acurácia e um desempenho de 99,97% na última fase do experimento, embora tenham apresentado resultados insatisfatório sobre outras bases de dados, este se sobresssaiu sobre o melhor resultado alcançado em *Man vs Computer* equivalente à 99,46% (STALLKAMP et al., 2012). Aparentemente, o *overfitting* experimentado sobre outros *datasets* ocorre devido à própria natureza dos mesmos.

Conclui-se que a tarefa de construir uma rede neural eficaz, realmente não possui regras exatas que sempre conduzirão à bons resultados, uma vez que tal tarefa depende da natureza do *dataset* em questão. Além do mais, geralmente existe mais de uma arquitetura para a qual se é capaz de obter bom desempenho.

Dessa forma seu desenvolvimento se estabelece através de um método empírico e incremental, necessitando a validação de uma centena de arquiteturas na tentativa de encontrar aquela que melhor se adapta à natureza do *dataset* em questão, ou talvez, deve-se alterar a este próprio, quando um volume muito pequeno de imagens ou uma grande diferença de amostras por classes se mostre um fator limitante para o desempenho da rede.

O uso de *kernels* de convolução com dimensões muito maiores que as da imagem de entrada, assim como sua eficácia em resultar ganho de desempenho, especialmente sobre o GTSRD, é uma questão para a qual não se encontrou documentação em nenhum outro trabalho, assim, deve-se investigar a fundamentação matemática por traz deste fenômeno.

Considerando-se trabalhos futuros, deseja-se desenvolver uma metodologia automática pelo emprego de um algoritmo evolutivo que gere uma rede suficientemente bem ajustada para alcançar bom desempenho dado um *dataset* arbitrário. Além disso, espera-se utilizar os conhecimentos adquiridos sobre classificação de imagens para implementar um algoritmo de detecção, e deste modo construir um reconhecedor, que uma vez desenvolvido, associar-se-ia a um buscador de imagens automático (baseado em técnicas de *crawling*) e, se possível, seria utilizado para desenvolver um *script* automático de busca de imagens de placas de trânsito. Dessa forma com o buscador, detector e classificador, almeja-se implementar um sistema baseado em redes neurais capaz de ampliar a própria base de dados pela qual foi treinado ou auxiliar na construção de novas bases de dados relacionadas ao domínio em questão. Se possível, tal sistema seria empregado na construção de bases de dados sobre sinalização brasileira.

Referências

AHIRE, J. B. *Perceptron and Backpropagation*. 2018. Disponível em: <https://medium.com/@jayeshbahire/perceptron-and-backpropagation-970d752f4e44>. Citado na página 25.

BECKER, D. *Rectified Linear Units (ReLU) in Deep Learning*. 2018. Disponível em: <https://www.kaggle.com/dansbecker/rectified-linear-units-relu-in-deep-learning>. Citado na página 30.

BERGER, C. *Perceptrons - The Most Basic Form of Neural Network*. 2016. Disponível em: <https://appliedgo.net/perceptron/>. Citado 2 vezes nas páginas 24 e 25.

BIMBRAW, K. Autonomous cars: Past, present and future - a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings*, v. 1, p. 191–198, 01 2015. Citado na página 20.

BMVA, T. B. M. V. A.; RECOGNITION, S. for P. *What is computer vision?* Nenhuma citação no texto.

BROWNLEE, J. *8 Inspirational Applications Deep Learning*. 2016. Disponível em: <https://machinelearningmastery.com/inspirational-applications-deep-learning/>. Citado na página 29.

BROWNLEE, J. *What is Deep Learning*. 2016. Disponível em: <https://machinelearningmastery.com/what-is-deep-learning/>. Citado 2 vezes nas páginas 17 e 29.

BUDUMA, N. *Data Science 101: Preventing Overfitting in Neural Networks*. 2019. Disponível em: <https://www.kdnuggets.com/2015/04/preventing-overfitting-neural-networks.html/2>. Citado na página 28.

CACCIA, L. C. M. *Réseaux de neurones artificiels*. 2018. Disponível em: <http://accromath.uqam.ca/2018/09/reseaux-de-neurones-artificiels/>. Citado na página 23.

CAFFE. Disponível em: <http://caffe.berkeleyvision.org/>. Citado na página 45.

CEA, C. L. A. e. a. A. *La Voiture Autonome*. 2017. Disponível em: <http://www.cea.fr/comprendre/Pages/nouvelles-technologies/essentiel-sur-voiture-autonome.aspx>. Citado na página 20.

CHANDRAYAN, P. *Brief history deep learning*. 2017. Disponível em: <https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663>. Citado na página 17.

CHANDRAYAN, P. *Deep Learning : Autoencoders Fundamentals and types*. 2017. Disponível em: <https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663>. Citado na página 29.

CIRESAN, D. et al. A committee of neural networks for traffic sign classification. *Proceedings of the International Joint Conference on Neural Networks*, p. 1918–1921, 07 2011. Citado na página 43.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, v. 2, 06 2005. Citado na página 33.

DIETERLE, F. *Overfitting, Underfitting and Model Complexity*. 2019. Disponível em: <http://www.frank-dieterle.de/phd/2_8_1.html>. Citado 2 vezes nas páginas 26 e 27.

DOUKKALI, F. *Batch normalization in Neural Networks*. 2017. Disponível em: <https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c>. Citado 2 vezes nas páginas 28 e 37.

EULOGIO, R. *Introduction to Random Forests*. 2017. Disponível em: <https://produto.mercadolivre.com.br/MLB-1184124730-kefir-de-leite-brinde-escolha-instrucoes-iogurte--_JM?quantity=1>. Citado na página 39.

FOOTE, K. D. *A Brief History of Deep Learning*. 2017. Disponível em: <https://www.dataversity.net/brief-history-deep-learning/>. Citado na página 17.

FUMO, D. *Classification Versus Regression—Intro To Machine Learning 5*. 2017. Disponível em: <https://medium.com/simple-ai/classification-versus-regression-intro-to-machine-learning-5-5566efd4cb83>. Nenhuma citação no texto.

GADAM, S. *Artificial Intelligence and Autonomous Vehicles*. 2018. Disponível em: <https://medium.com/datadriveninvestor/artificial-intelligence-and-autonomous-vehicles-ae877feb6cd2>. Citado na página 16.

GANDHI, R. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO—Object Detection Algorithms*. 2018. Disponível em: <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>. Citado 2 vezes nas páginas 35 e 36.

GANDHI, R. *Support Vector Machine—Introduction to Machine Learning Algorithms*. 2018. Disponível em: <https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>. Citado na página 40.

GEITGEY, A. *Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks*. 2016. Disponível em: <https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721>. Citado 2 vezes nas páginas 29 e 30.

GOOGLE. *TensorFlow — An open-source machine learning framework for everyone*. 2015. Disponível em: <https://www.tensorflow.org/>. Citado na página 45.

HAMMOND, K. *Pratical Artificial Intelligence for Dummies*. [S.l.: s.n.], 2015. 53 - 118 p. Citado na página 21.

INDURKYA, N.; DAMERAU, F. J. *Handbook of Natural language Processing*. [S.l.: s.n.], 2010. 23 - 26 p. Citado na página 23.

JIA, Y.; HUANG, C.; DARRELL, T. Beyond spatial pyramids: Receptive field learning for pooled image features. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012. Disponível em: <https://doi.org/10.1109/cvpr.2012.6248076>. Citado na página 35.

KURENKOV, A. *A Brief History of Game AI Up To AlphaGo*. 2016. Disponível em: <http://www.andreykurenkov.com/writing/ai/a-brief-history-of-game-ai/>. Citado na página 21.

LARSON, E. J. *A Brief History of Computer Chess*. Disponível em: <https://thebestschools.org/magazine/brief-history-of-computer-chess/>. Citado na página 21.

MARCOS, E.; RIBEIRO, E.; MERLO, E. AplicaÇÃo das redes neurais na concessÃo de crÉdito - um estudo de caso em uma empresa de consÓrcio. 01 2005. Citado na página 16.

MATHIAS, M. et al. Traffic sign recognition &#x2014 how far are we from the solution? In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013. Disponível em: <https://doi.org/10.1109/ijcnn.2013.6707049>. Citado 6 vezes nas páginas 17, 31, 37, 38, 41 e 47.

MAZUR, M. *A Step by Step Backpropagation Example*. 2015. Disponível em: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>. Citado na página 26.

NAGPAL, A. *L1 and L2 Regularization Methods*. 2017. Disponível em: <https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c>. Citado na página 28.

NAGPAL, A. *Over-fitting and Regularization*. 2017. Disponível em: <https://towardsdatascience.com/over-fitting-and-regularization-64d16100f45c>. Citado na página 27.

PAPAGEORGIOU, C.; OREN, M.; POGGIO, T. A general framework for object detection. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*. Narosa Publishing House, 2002. Disponível em: <https://doi.org/10.1109/iccv.1998.710772>. Citado na página 32.

PARTHASARATHY, D. *A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN*. 2017. Disponível em: <https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4>. Citado na página 34.

PATEL, S. *Chapter 2 : SVM (Support Vector Machine)*. 2017. Disponível em: <https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72>. Citado na página 40.

PURDY, C. G. F. K. W. *History of the automobile*. 2017. Disponível em: <https://www.britannica.com/technology/automobile/History-of-the-automobile>. Citado na página 16.

REDMON, J. et al. You only look once: Unified, real-time object detection. 06 2015. Citado na página 36.

REESE, H. *Autonomous driving levels 0 to 5: Understand the Differences*. 2016. Disponível em: <https://www.techrepublic.com/article/autonomous-driving-levels-0-to-5-understanding-the-differences/>. Citado na página 20.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence a Modern Approach*. [S.l.: s.n.], 1995. 53 - 118 p. Citado na página 22.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence a Modern Approach*. [S.l.: s.n.], 1995. 297 - 304 p. Citado na página 22.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence a Modern Approach*. [S.l.: s.n.], 1995. 263 p. Citado na página 23.

RUTA, A.; LI, Y.; LIU, X. Real-time traffic sign recognition from video by class-specific discriminative features. *Pattern Recognition*, Elsevier BV, v. 43, n. 1, p. 416–430, jan. 2010. Disponível em: <https://doi.org/10.1016/j.patcog.2009.05.018>. Citado 3 vezes nas páginas 16, 32 e 33.

SALTI, S. et al. Traffic sign detection via interest region extraction. *Pattern Recognition*, Elsevier BV, v. 48, n. 4, p. 1039–1049, abr. 2015. Disponível em: <https://doi.org/10.1016/j.patcog.2014.05.017>. Citado 3 vezes nas páginas 17, 31 e 32.

SCHAUL JUSTIN BAYER, D. W. S. Y. M. F. F. S. T. R. J. S. T. *Welcome to PyBrain*. 2010. Disponível em: <http://pybrain.org/>. Citado na página 44.

SERMANET, P.; LECUN, Y. Traffic sign recognition with multi-scale convolutional networks. In: . [S.l.: s.n.], 2011. p. 2809 – 2813. Citado na página 29.

SOLEM, j. E. *Programming Computer Vision with Python*. [S.l.: s.n.], 2012. 7 p. Citado na página 16.

STALLKAMP, J. et al. The german traffic sign recognition benchmark: A multi-class classification competition. In: *The 2011 International Joint Conference on Neural Networks*. IEEE, 2011. Disponível em: <https://doi.org/10.1109/ijcnn.2011.6033395>. Citado na página 38.

STALLKAMP, J. et al. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, Elsevier BV, v. 32, p. 323–332, ago. 2012. Disponível em: <https://doi.org/10.1016/j.neunet.2012.02.016>. Citado 4 vezes nas páginas 37, 38, 41 e 81.

TRICKS, C. *Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN,YOLO,SSD*. 2018. Disponível em: <https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/>. Citado na página 35.

TSANG, S.-H. *Review: SSD—Single Shot Detector (Object Detection)*. 2018. Disponível em: <https://towardsdatascience.com/review-ssd-single-shot-detector-object-detection-851a94607d11>. Citado na página 36.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* IEEE Comput. Soc. Disponível em: <https://doi.org/10.1109/cvpr.2001.990517>. Citado na página 33.

VIOLA, P.; JONES, M. J. Robust real-time face detection. *International Journal of Computer Vision*, Springer Nature, v. 57, n. 2, p. 137–154, maio 2004. Disponível em: <https://doi.org/10.1023/b:visi.0000013087.49260.fb>. Citado na página 32.

VOINIGESCU, E. *Les Neurones ont une Forme Propice à L'apprentissage Profond*. 2017. Disponível em: <https://www.cifar.ca/fr/nouvelles/2017/12/05/les-neurones-ont-une-forme-propice-%c3%a0-l-apprentissage-profond>. Citado na página 29.

WED, J. *Object Detection with Deep Learning: The Definitive Guide*. 2017. Disponível em: <https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/>. Citado na página 31.

ZAGORUYKO, S. et al. A multipath network for object detection. 04 2016. Citado na página 34.

ZAKKA, K. *A Complete Guide to K-Nearest-Neighbors with Applications in Python and R*. 2016. Disponível em: <https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/>. Citado na página 40.

ZAKLOUTA, F.; STANCIULESCU, B. Real-time traffic sign recognition in three stages. *Robotics and Autonomous Systems*, Elsevier BV, v. 62, n. 1, p. 16–24, jan. 2014. Disponível em: <https://doi.org/10.1016/j.robot.2012.07.019>. Citado na página 33.