UNIVERSIDADE FEDERAL DO MARANHÃO

Programa de Pós-Graduação em Ciência da Computação

Joaquim Martins Scavone

Otimizando CNNs Com Aprendizado Acumulativo Via Múltiplas Redes Neurais

São Luís
2020

Joaquim Martins Scavone

Otimizando CNNs Com Aprendizado Acumulativo Via Múltiplas Redes Neurais

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Ciência da Computação, ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal do Maranhão.

Programa de Pós-Graduação em Ciência da Computação

Universidade Federal do Maranhão

Orientador: Prof. Dr. Areolino de Almeida Neto

Coorientador: Prof. Dr. Geraldo Braz Junior

São Luís - MA

2020

Joaquim Martins Scavone

Otimizando CNNs Com Aprendizado Acumulativo Via Múltiplas Redes Neurais

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Ciência da Computação, ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal do Maranhão.

Trabalho aprovado em São Luís - MA, 03 de Novembro de 2020:

Prof. Dr. Areolino de Almeida Neto
Orientador
Universidade Federal do Maranhão

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

Prof. Dr. Alexandre César Muniz de Oliveira
Examinador Interno
Universidade Federal do Maranhão

Prof. Dr. Marcelo Lisboa Rocha
Examinador Externo
Universidade Federal do Tocantins

São Luís - MA
2020

A minha esposa Gerlany, minha coautora na vida, obrigado pelo seu apoio e dedicação.

Agradecimentos

Em primeiro lugar, gostaria de agradecer ao meu bom Deus, que vem sendo minha fortaleza em todos os momentos difíceis através de seu filho, Jesus Cristo.

A minha linda esposa Gerlany, o melhor e mais belo presente que já recebi, ao meu filho Samuel que renovou minhas esperanças em um mundo melhor com o seu sorriso.

Aos meus pais Miriam e Carlos, que se fizeram presentes junto a minha família pra suprir a minha ausência.

Ao meu irmão Anderson, que sempre me motivou.

A todos os amigos que se fizeram presente nessa etapa, especialmente ao Marcelo, Denis, Antônino e Ivan, que abriram a porta do seu lar para me abrigar durante esse período.

Ao Jonisson por compartilhar seu tempo e conhecimento comigo.

Ao Matheus, que esteve comigo em todas as etapas deste mestrado motivando e colaborando com suas ideias para meu projeto.

Ao meu amigo Moisés, que me estendeu a mão mesmo sem me conhecer.

A todos os meus professores, pelo conhecimento que me proporcionaram durante toda a vida acadêmica, principalmente ao meu orientador, professor Areolino, e ao meu coorientador, professor Geraldo, pelas orientações e paciência durante o mestrado. Aprendi muito mais do que conteúdo com os ensinamentos de vocês.

A todos os amigos do laboratório INOVTEC, pelas ideias, sugestões e brincadeiras. Obrigado.

Aos meus colegas de trabalho. professores Anderson, Bruno e Isabela, por estarem sempre ajudando, desdobrando-se para que eu pudesse concluir este trabalho.

A todos que compõem o PPGCC-UFMA: professores, alunos e funcionários.

Meus amigos, de todos os ganhos que tive nesse mestrado, com certeza vocês foram os mais importantes, pois não teria conseguido sem vocês, assim este titulo pertence a todos nós.

"Quando penso que cheguei ao meu limite descubro que tenho forças para ir além."

(Ayrton Senna.)

Resumo

O número de vítimas fatais em acidentes de trânsito é assustador. Muitos desses acidentes decorrem do desrespeito a sinalização, que, muitas vezes, acontece de maneira involuntária, por distrações, por exemplo. Esta problemática vem sendo tratada com muita atenção na comunidade científica. O que levou ao surgimento dos Sistemas Avançados de Auxílio ao Motorista (ADAS - *Advanced Driver Assistance Systems*), que são sistemas que podem interpretar a sinalização e fluxo na via e, a partir dessas informações, emitir alertas ao condutor ou até mesmo intervir na condução. As redes convolucionais já são amplamente utilizadas nos ADAS e estão promovendo verdadeiro avanço nesse área. Desta forma, este trabalho apresenta uma estratégia que utiliza redes neurais neste tipo de problema. A pesquisa desenvolvida realizou uma união das técnicas de múltiplas redes neurais autocoordenadas e redes neurais convolucionais, que demonstrou sua eficiência quando aplicada a redes já treinadas. A técnica proposta apresentou 95.33% de precisão, a possibilidade da diminuição do tempo de treinamento e uma nova estratégia de fuga de mínimos locais, o que abre um leque de novas pesquisas que podem ser realizadas.

Palavras-chave: reconhecimento e classificação de semáforos, aprendizagem profunda, rede neural convolucional, múltiplas redes neurais autocoordenadas.

Abstract

The number of fatalities in traffic accidents is staggering. Many of these accidents result from disrespect for signaling, which often happens involuntarily, due to distractions, for example. This issue has been treated with great attention in the scientific community. This led to the emergence of Advanced Driver Assistance Systems (ADAS), which are systems that can interpret signaling and flow on the road and, based on this information, issue alerts to the driver or even intervene in driving. Convolutional networks are already widely used in ADAS and are promoting real progress in this area. Thus, this work presents a strategy that uses neural networks in this type of problem. The developed research made a union of the techniques of multiple self-coordinated neural networks and convolutional neural networks, which demonstrated its efficiency when applied to already trained networks. The proposed technique achieved 95.33% accuracy, the possibility of reducing training time and a new strategy to escape local minimums, which opens up a range of new research that can be carried out.

Keywords: Traffic Lights Recognition, Deep Learning, Multiple Self-coordinated Neural Networks.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ADAS	*Advanced Driver Assistance Systems*
IA	Inteligência Artificial
RNA	Rede Neural Artificial
CNN	*Convolutional Neural Network*
MLP	*Multilayer Perceptron*
ROI	*Regions of Interest*
LDA	*Linear Discriminant Analysis*
kNN	*K-Nearest Neighbor Classifier*
SVM	*Support Vector Machines*
TLR	*Traffic Light Recognition*
RSL	Revisão Sistemática da Literatura
MRNA	Múltiplas Redes Neurais Artificiais
RBF	*Radial Basis Function*
ML	*Machine Learning*
API	*Application Programming Interface*
MLP	*Multi Layer Perceptron*
BP	*Backpropagation*
FA	Função de Ativação
ReLU	*Rectified Linear Units*
SELU	*Scaled Exponential Linear Units*
MC	Matriz de Confusão
SGD	*Stochastic Gradient Descent Optimizer*

Sumário

1 Introdução

No ano de 2017, mais de 182 mil pessoas envolveram-se em acidentes de trânsito apenas no Brasil. Destas, mais de 35 mil vieram a óbito (Ministério da Saúde, 2019). Nos Estados Unidos, em 2018, o número de vítimas passou de 139 mil (IIHS, 2020). E os dados mais recentes da OMS (2018) apontam que mais de 1,3 milhões de pessoas perderam sua vida por causa desse tipo de acidente no mesmo ano.

Urge citar que muitos dos acidentes ocorridos podem ser atribuídos ao desrespeito à sinalização, que muitas vezes acontece de maneira involuntária como por distrações, por exemplo. Assim, independente do motivo, o desrespeito ao sinal vermelho causou mais de 846 mortes nos Estados Unidos no ano de 2018, onde, pelo menos, metade eram pedestres, ciclistas ou outros veículos que foram atingidos por motoristas infratores (IIHS, 2020).

Com o intuito de combater o número gigantesco de acidentes de trânsito, estão sendo desenvolvidas pesquisas em tecnologias para auxiliar no monitoramento, na vigilância e até na condução de veículos. Portanto, os projetos focados na segurança da condução são denominados sistemas avançados de assistência ao motorista (ADAS - *Advanced Driver Assistance Systems*).

Um dos grandes desafios dos ADAS é a confiabilidade das predições do sistema, uma vez que as pessoas não estão prontas para entregar suas vidas a um sistema computadorizado que pode falhar, mesmo que sua precisão já supere humanos que executam o mesmo trabalho. Logo, sistemas como esses exigem uma alta acurácia e são muito sensíveis a erros. Por isso, há um esforço da comunidade científica para criar um sistema robusto que tenha seu erro próximo ou igual a zero.

Desse modo, existem inúmeras abordagens que tentam atender às expectativas de precisão dos ADAS. Alguns dos mais promissores focam em sensores dentro do contexto de cidades inteligentes (OCHOA; OLIVA, 2019). Em suma, trabalhos assim preveem a comunicação do veículo com a sinalização, outros veículos e sensores. Porém, projetos que requerem a automatização de todos os processos vinculados ao trânsito ainda estão muito distantes da realidade de países em desenvolvimento, como o Brasil.

Em teoria, a automatização do trânsito em países subdesenvolvidos será gradual e individual, ou seja, o veículo usado será automatizado pelo seu proprietário ao mesmo tempo que os novos virão de fábrica já com os recursos necessários. Por certo, a janela de tempo de interação entre veículos automatizados, semi-automatizados e sem automatização, durante esse processo de migração, será considerável. E, se a migração da infraestrutura seguir padrões habituais, provavelmente, a automatização deve surgir nas capitais e depois se estender ao interior das cidades, o que pode levar anos.

Tendo em vista o tempo da migração de veículos e infraestrutura, os ADAS devem ser capazes de reconhecer não apenas sensores, mas também objetos, veículos, pessoas e sinalizações tradicionais. Aliás, esse imenso conjunto de dados deve ser processado de maneira instantânea e por um equipamento adequado, permitindo que o sistema torne-se útil e acessível. Logo, o sistema deve ser capaz de tomar uma decisão imediata, como por exemplo, acionar os freios do veículo. Contudo, um processo com tantas variáveis dificilmente será alcançado de maneira algorítmica.

Uma alternativa que vem ganhando cada vez mais relevância para problemas com alta complexidade é a inteligência artificial (IA). Entre as várias linhas de pesquisa sobre IA, as redes neurais artificiais (RNAs) vêm destacando-se (DOMINGOS, 2017). Existem várias arquiteturas de RNAs, entre elas as redes convolucionais (CNNs - *Convolutional Neural Networks*). As CNNs, geralmente, são usadas para extrair características de imagens e reconhecimento de objetos. Nessa área elas vêm obtendo resultados impressionantes (SERMANET *et al.*, 2014).

Outrossim, em soluções por via algorítmica, o programador deve projetar todas as alternativas possíveis de interação do sistema, pois em casos omissos, ele comporta-se de maneira inesperada. Por conseguinte, o conceito de IA prevê uma espécie de autoprogramação em que o próprio sistema cria seu algorítimo a partir da exposição ao comportamento esperado (DOMINGOS, 2017). Esse processo é chamado de treinamento.

O treinamento de redes convolucionais, que são um dos tipos de redes profundas, é um processo que despende tempo e que, frequentemente, possui ajustes manuais no processo. Usualmente, a dinâmica do treinamento passa pela escolha do modelo da rede, hiper-parâmetros da rede e do treinamento. E ainda, em alguns casos, são aplicados pré e/ou pós processamento nos dados de treinamento. Após a escolha de todos os hiper-parâmetros, o treinamento é iniciado.

Depois da fase de treinamento, se a CNN não atingir um resultado esperado, a escolha de sua estrutura ou hiper-parâmetros deve ser alterada. Cada vez que esse processo ocorre, toda a rede deve ser retreinada. Isso praticamente descarta todo esforço empreendido, assim como o conhecimento adquirido pela rede.

As CNNs vêm tornando-se referência em classificação, reconhecimento e detecção de objetos. Vários modelos de CNNs possuem um classificador formado por uma rede Perceptron multicamadas (MLP - *Multilayer Perceptron*), geralmente presente nas últimas camadas da rede. Apesar disso, as pesquisas sobre redes convolucionais aprofundam-se cada vez mais nas melhorias das convoluções (XIE *et al.*, 2017; SIMONYAN; ZISSERMAN, 2015). Em alguns casos, o classificador tornou-se apenas uma saída para a rede, possuindo somente uma camada (HUANG *et al.*, 2017).

Desse modo, uma rede MLP com as dimensões e hiper-parâmetros adequados,

poderia solucionar problemas de classificação de imagens (AGHDAM; HERAVI, 2017). Apesar disso, as pesquisas vêm deixando as MLPs de lado devido aos vastos recursos computacionais necessários para o treinamento de uma rede profunda desse tipo. Porém, o potencial do classificador ainda é um problema em aberto e pode ser a chave para um novo salto nas pesquisas relacionadas a reconhecimento de imagens.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma técnica híbrida entre redes convolucionais e múltiplas redes neurais artificiais autocoordenadas com o propósito de contribuir com as pesquisas sobre reconhecimento automático de semáforos.

1.1.2 Objetivos específicos

Para atingir o objetivo geral, os seguintes objetivos específicos são propostos:

- Propor um modelo matemático de adaptação da camada completamente conectada das redes convolucionais para adição de novas redes;
- Avaliar a eficácia do modelo proposto em uma rede convolucional presente na literatura com imagens de uma base pública de sinais de trânsito;
- Comparar a precisão da rede com outros modelos conhecidos e;
- Avaliar a eficácia do uso de redes auxiliares como estratégia de fuga de mínimos locais durante os treinamentos.

1.2 Justificativa

As CNNs estão cada vez mais precisas em suas classificações, entretanto ainda há muito espaço para melhora, sobretudo em problemas críticos como classificação e reconhecimento de semáforos. É importante ressaltar que existe uma convergência das pesquisas com CNNs que usualmente estão focadas nas mudanças das camadas convolucionais, mas o avanço da ciência depende da divergência e da exploração de várias possibilidades.

Assim, existe um tênue equilíbrio entre número de parâmetros de treinamento e recursos disponíveis, o que inviabiliza a utilização de redes MLPs muito profundas. Este trabalho apresta uma técnica que permite o crescimento das camadas completamente

conectadas por meio de um novo modelo de treinamento sem a necessidade de aumento nos recursos computacionais.

O trabalho baseia-se no estudo de Almeida Neto, Góes e Nascimento (2010), o qual apresentou uma técnica de treinamento com aprendizado acumulativo, sendo intitulada como Múltiplas Redes Neurais Artificias (MRNA) Autocoordenadas. Essa estratégia apresentou uma abordagem em que os treinamentos anteriores não são descartados, mas aprimorados. Portanto, o modelo consiste no uso de várias redes para atacar um problema sem dividi-lo. Além disso, a técnica apresenta outros exemplos de sucesso na literatura (OLIVEIRA; ALMEIDA NETO, 2013; TEIXEIRA; ALMEIDA NETO, 2016).

A revisão da literatura não encontrou nenhuma aplicação de MRNAs de Almeida Neto, Góes e Nascimento (2010) em redes CNN. Logo, este trabalho apresenta uma abordagem inédita que consiste na adaptação do modelo de MRNAs para a criação de um novo conceito de dimensão no classificador das redes CNNs. A técnica consiste na união de mais de um classificador à mesma estrutura convolucional, de maneira que os classificadores cooperem entre si para resolver o problema através da estratégia de treinamento acumulativo.

Assim sendo, o presente trabalho justifica-se por contribuir diretamente com as pequisas sobre redes convolucionais, trazendo uma abordagem inédita de estrutura e treinamentos para as redes. Somado à isso, o modelo foi validado em base pública de semáforos de trânsito o que, por si só, demonstra sua relevância para a comunidade como possível ferramenta ao enfrentamento dos inúmeros acidentes de trânsito.

1.3 Motivação

Existem inúmero projetos que visam ao auxílio e à automação dos processos relacionados ao trânsito. Todavia, em sua grande maioria, esses processos requerem um alto investimento em infraestrutura, o que está longe de ser uma realidade do Brasil. A principal motivação desse trabalho é contribuir com uma área de pesquisa que vem gerando aplicação de baixo custo, permitindo que países subdesenvolvidos possam também reduzir o número de vítimas de acidentes.

1.4 Trabalhos relacionados

Esta seção dedica-se a apresentar os trabalhos relacionados ao estudo em questão, o qual envolve três pontos predominantes. Dessa forma, esta seção também foca nesses pontos. O primeiro é o problema, o reconhecimento e classificação de semáforo (TLR - *Traffic Light Recognition*). Como os problemas de TLR são comuns na literatura, este

trabalho filtrou os estudos com interseções em relação à técnica utilizada e à base de imagens escolhida. O segundo ponto nesta seção diz respeito aos trabalhos sobre redes convolucionais que ilustram a evolução das soluções dessa área com foco em trabalhos, os quais contribuíram com ideias para esta investigação. Por fim, os trabalhos relacionados à intervenção proposta por este estudo: treinamento acumulativo utilizando múltiplas redes neurais.

Os estudos relacionados à TLR começaram pela combinação de várias técnicas de processamento de imagens (LINDNER; KRESSEL; KAELBERER, 2004; SIOGKAS; SKODRAS; DERMATAS, 2012; FAIRFIELD; URMSON, 2011). Haltakov *et al.* (2015) utilizam um método de segmentação semântica, que é empregado para reconhecimento das regiões de interesse (ROI - *Regions of Interest*). Em seguida, são confirmados e categorizados por um classificador geométrico e de cores combinado com um sistema de detecção temporal.

Em Siogkas, Skodras e Dermatas (2012), o sistema incorpora um módulo de pré-processamento de cores para aprimorar as regiões vermelha e verde na imagem. Ademais, uma rápida transformação de simetria radial é utilizada para a detecção de candidatos a semáforos e os resultados falsos positivos são minimizados usando a verificação de persistência espaço-temporal.

Todavia, nos últimos anos, as abordagens de aprendizagem profundas superaram os algoritmos de última geração em uma ampla variedade de problemas (FERNANDEZ *et al.*, 2018). E, consequentemente, não tardou o aparecimento de técnicas que usam redes neurais para reconhecimento e classificação de semáforos na literatura. Algumas técnicas mantiveram a estrutura inicial de processamento de imagens, porém utilizaram redes neurais com poucas camadas para classificar as características retiradas do pré-processamento.

No trabalho de Soares *et al.* (2018), foram adotadas técnicas de processamento de imagem para identificação de ROIs e extração de suas características. Em seguida, os dados são enviados à uma MLP para classificá-los. Em Ouyang *et al.* (2020), utilizou-se um módulo heurístico de seleção de região candidata para identificar todos os semáforos possíveis, os quais são passados a um classificador formado por uma pequena CNN para classificá-los.

Bao *et al.* (2019) apresentam um sistema de reconhecimento de imagem combinado com uma CNN. Isso é possível porque além da classificação do semáforo, o relógio de contagem regressiva e a distância do veículo do semáforo também são identificados, podendo assim propor uma estratégia de decisão para o condutor.

Em Li *et al.* (2018), foi utilizado um conjunto de técnicas para detecção de semáforos como *Fuzzy, Bulb* e detecção multi tamanho. Já Michael e Schlipsing (2015) combinaram

a análise de discriminantes linear (LDA - *Linear Discriminant Analysis*), k-vizinhos mais próximos (kNN - *K-Nearest Neighbor Classifier*) e máquina de vetores de suporte (SVM - *Support Vector Machines*). Ambos os trabalhos mostraram que combinar técnicas pode ser o caminho para a solução do problema de reconhecimento e classificação de semáforos.

Apesar de a combinação de várias técnicas mostrar-se eficaz, muitas pesquisas preferiram utilizar apenas uma CNN para realizar essa tarefa, como Fernandez *et al.* (2018) que apresentam uma CNN capaz de reconhecer semáforos com um ótimo desempenho. Contudo, a pesquisa não aborda a classificação do estado do semáforo. Jensen, Nasrollahi e Moeslund (2017) demonstram o desempenho da rede *You Only Look Once* (YOLO) na classificação de semáforos. A rede YOLO vem sendo utilizada com ótimos resultados para detecção de objetos para vários contextos.

Weber, Wolf e Zöllner (2016) criaram a DeepTLR, uma rede CNN profunda que foi desenvolvida especificamente para detecção e classificação de semáforos. DeepTLR foi baseada na Alexnet. Em Pon *et al.* (2018), foi apresentado, segundo os autores, a primeira rede capaz de detectar semáforos e placas de trânsito, simultaneamente, mostrando o potencial de redes neurais para solução de problemas desse tipo.

O uso de mais de uma rede para resolver um problema é comum na literatura e já foi até utilizado no problema de TLR (BEHRENDT; NOVAK; BOTROS, 2017). O artigo de Behrendt, Novak e Botros (2017) apresentou um sistema de detecção que usa duas redes neurais trabalhando em conjunto. Uma detectando áreas com possíveis semáforos e outra para classificá-los entre: verde, vermelho, amarelo, desligado ou sem semáforo. Esse trabalho obteve resultados expressivos, o qual demonstra que esse tipo de abordagem ainda é uma forte alternativa para os problemas de classificação.

Em virtude do grande número de estudos de TLR, alguns trabalhos dedicaram-se a fazer uma revisão sistemática da literatura (RSL) sobre o assunto. Entre as RSLs sobre o problema, o trabalho de Jensen *et al.* (2016) apresenta o estado da arte do reconhecimento e classificação de semáforos, tabulando dados de 25 trabalhos publicados entre 2009 e 2015. Jensen *et al.* (2016) sugerem que sejam adotadas métricas padrões para avaliação de desempenho de trabalhos dessa espécie, pois a adoção das métricas padrões facilitaria a comparação entre estudos, além de uma melhor compressão do estado da arte.

O estudo de Deng *et al.* (2010) apresenta um conjunto de imagens intitulado Imagenet. Segundo os autores, A Imagenet é o maior e mais completo conjunto de imagens indexado existente, a partir do qual foi criado um desafio para avaliar o desempenho de redes classificadoras, como redes convolucionais. Com a análise desse desempenho Imagenet, é possível elaborar uma hipótese sobre as correntes de evolução das CNNS.

A CNN é um tipo de técnica que visa a diminuir a quantidade de parâmetros a serem treinados, reduzindo assim o tempo de treinamento e o investimento em *hardwares*

(AGHDAM; HERAVI, 2017). Conforme supracitado, o uso de várias redes é um ponto frequente na literatura. Entretanto, os trabalhos que aprofundam essa técnica, geralmente, dividem os problemas em sub-problemas, atribuindo a cada rede um sub-problema para resolver, desenvolvendo uma estrutura de coordenação do treinamento e tratamento das saídas das redes, como no trabalho de Behrendt, Novak e Botros (2017).

Ainda é pertinente citar que outras abordagens que usam mais de uma rede no mesmo problema podem ser encontradas na literatura, como por exemplo, as redes adversárias (GOODFELLOW *et al.*, 2014). Essa técnica, basicamente, utiliza duas redes, a geradora e a classificadora, para solucionar um problema. A rede classificadora tem o objetivo de aprender os padrões das imagens que devem ser reconhecidos. Ao passo que a rede geradora vai aumentar o número de amostras, criar ruídos e tentar enganar a classificadora. No final do treinamento, espera-se uma rede capaz de criar amostras de entradas quase idênticas às originais e outra capaz de classificá-las com alto poder de precisão.

Apesar de um dos objetivos das CNNs ser diminuir o número de parâmetros de treinamentos, já existem estudos que demonstram que o tamanho da rede está co-relacionado com a sua capacidade de classificação. Simonyan e Zisserman (2015) demonstram como a profundidade pode melhorar o desempenho de redes CNNs. O estudo apresenta as redes VGG16 e VGG19 que possuem respectivamente 16 e 19 camadas antes do classificador. O estudo enfatiza que a profundidade da rede é predominante para sua capacidade de generalização.

No entanto, o trabalho de Xie *et al.* (2017) traz uma nova bordagem a ser considerada. O artigo apresenta uma terceira dimensão para as redes convolucionais, denominado cardinalidade. Segundo esse estudo, a substituição de uma grande camada convolucional por várias camadas pequenas e em paralelo melhorou o desempenho da rede sem grandes mudanças de consumo de recursos computacionais.

Redes crescem em largura através do número de seus neurônios, em profundidade pelo seu número de camadas e em cardinalidade por meio das convoluções em paralelo na mesma camada. Contudo, o conceito de cardinalidade ainda não foi implementado nas camadas do classificador. Logo, este trabalho propõe realizar tal feito e para isso, é utilizada a técnica de Almeida Neto, Góes e Nascimento (2010).

A proposta das MRNA autocoordenadas de Almeida Neto, Góes e Nascimento (2010) foi unir várias MLPs objetivando um trabalho em conjunto para controlar uma haste flexível. Um dos motivadores do trabalho foi a sensação de perda do esforço realizado quando o treinamento não era bem sucedido. Por isso, além de propor uma combinação de várias redes, o estudo descreve uma estratégia de aprendizado acumulativo.

A técnica de treinamento aplicada às MRNA autocoordenadas prevê que um

treinamento incompleto não será descartado, mas será utilizado como base inicial para o treinamento da próxima rede. De forma simplificada, o objetivo da rede seguinte é aprender conhecimentos desconsiderados pelas redes anteriores (ALMEIDA NETO; GÓES; NASCIMENTO, 2010).

As MRNA foram utilizadas em outros problemas com sucesso. Em Oliveira e Almeida Neto (2013), por exemplo, elas foram utilizadas para gerenciar um controlador de semáforos, diminuindo o tempo de espera e de congestionamento nas vias. Em Teixeira e Almeida Neto (2016), o trabalho apresentou os desempenhos das MRNAs em duas redes de estruturas diferentes, a MLP e a rede da função da base radial (RBF - *Radial Basis Function*). Ademais, também foi apresentado um modelo de inserção pacífica das redes, possibilitando que as redes posteriores sejam inseridas sem causar oscilações na curva do erro. Por fim, as MRNAs apresentam características de treinamentos gradativos, acumulativos e pacíficos, o que lhes tornam ideais para intervenções em redes em produção.

1.5 Organização do Trabalho

Este trabalho está organizado da seguinte forma. Além do atual capítulo, o Capítulo 2 trata da fundamentação teórica do estudo. Em sequência, o Capítulo 3 apresenta a metodologia adotada neste trabalho. Logo após, o Capítulo 4 apresenta e discute os resultados obtidos. O trabalho finaliza-se no Capítulo 5 com as conclusões e trabalhos futuros.

2 Fundamentação teórica

A IA vem popularizando-se ultimamente e, apesar de parecer um conceito novo, diversos estudos sobre a técnica existem há décadas. Um exemplo disso é o atendimento via *chat* realizado por robôs computacionais, bem como o uso de visão computacional para reconhecimento de objetos. Portanto, os estudos nessa área tendem a evoluir cada vez mais, pois o objetivo da IA é ensinar o computador a aprender (DOMINGOS, 2017). Nesse contexto, é comum o equívoco entre o conceito de aprendizado de máquina (ML - *Machine Learning*) e o de IA, sendo aquele apenas uma das áreas da IA.

O aprendizado de máquina pode ser definido como "a capacidade de melhorar o desempenho na realização de alguma tarefa por meio da experiência"(MITCHELL, 1997). Antes do surgimento das mais diferentes formas da IA, os *softwares* eram exclusivamente algorítmicos. Assim, "um algoritmo é uma sequência de instruções que informa ao computador o que ele deve fazer"(DOMINGOS, 2017).

Na programação algorítmica, provavelmente, um projetista estuda o problema e escreve instruções sobre o seu comportamento. Segundo Domingos (2017), o aprendizado de máquina seria a capacidade de um computador escrever os seus próprios algoritmos. Então, após apresentadas uma determinada entrada e sua respectiva saída, o computador deverá ser capaz de encontrar a correlação entre estas e armazená-las para respostas futuras sobre o mesmo problema.

Existem cinco principais linhas de pesquisa sobre aprendizado de máquina: Simbolista, Evolucionária, Bayesiana, Analogista e Conexionista (DOMINGOS, 2017). Domingos (2017) define a linha conexionista como a que busca fazer uma engenharia reversa do cérebro humano. Essa analogia está correta, uma vez que os estudos relacionados à área têm sua origem no trabalho de McCulloch e Pitts (1943), o qual descreve um modelo matemático que representaria um neurônio humano. Esse último tornou-se a base para a criação de um neurônio artificial.

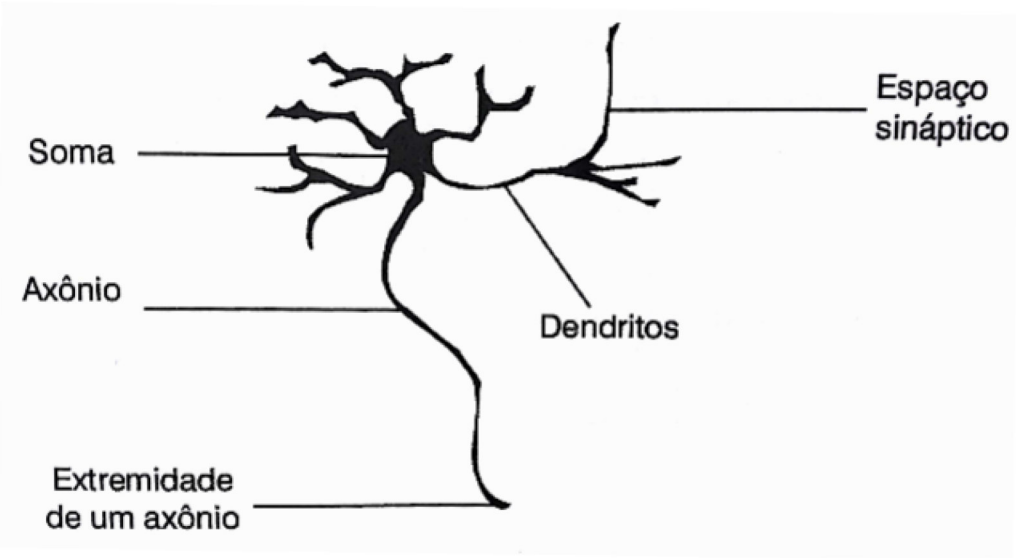
2.1 Neurônio Artificial

Definitivamente, o cérebro humano está longe de ser superado, por isso é compreensível que as bases de criação de uma IA partam do estudo e tentativa de replicação de seu funcionamento. Esse órgão é formado por bilhões de células específicas denominadas neurônios e são os de tipologia biológica que têm a função de receber uma descarga elétrica, processá-lá e decidir se deve ser propagada.

A Figura 1, apresenta um modelo de neurônio biológico, os quais são interligados

como uma rede através de um líquido sináptico que propaga os disparos elétricos por eles produzidos. Ao receber uma descarga através de seus dendritos, o soma calcula a intensidade da carga recebida. Urge citar que o soma de um neurônio tem uma certa tolerância a descargas recebidas e quando ultrapassadas, redimensiona e propaga a corrente recebida por meio do seu axônio (BRAGA; CARVALHO; LUDERMIR, 2007). Esse processo denomina-se ativação.

Figura 1 – Neurônio biológico.



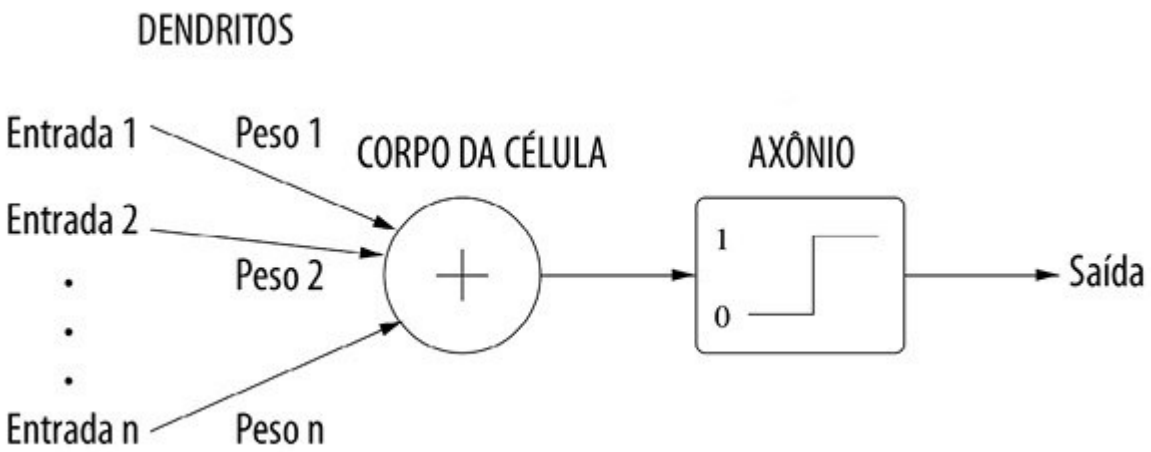
Fonte: Braga, Carvalho e Ludermir (2007)

Os disparos dos neurônios acontecem de forma simultânea e aproximadamente mil vezes por segundo (DOMINGOS, 2017). Essa simples célula, quando colocada em rede, é responsável por atos incríveis como o pensamento lógico, movimentos, emoções, lembranças, entre muitos outros.

O modelo matemático de McCulloch e Pitts (1943) apresentou uma proposta de um neurônio artificial ilustrado na Figura 2. Assim, o neurônio artificial tem um conjunto de entradas que faz analogia aos pulsos elétricos vindos de outros neurônios e os dendritos são representados pelos pesos. São estes que vão determinar a importância da entrada correspondente à ativação do neurônio atual, ou seja, quanto maior o peso, maior a influência da entrada na ativação do neurônio atual.

Por fim, a saída, geralmente, é o resultado da adição de todos os produtos das entradas pelos seus pesos correspondentes. Ao passo que o axônio do neurônio biológico, no artificial, é representado por uma função denominada função de ativação. O resultado dela corresponde à saída do neurônio.

Figura 2 – Modelo do neurônio artificial.



Fonte: Domingos (2017)

2.2 Rede Perceptron

O trabalho de McCulloch e Pitts (1943) inspirou vários outros sobre redes neurais, o que desencadeou a criação de diversas arquiteturas, entre elas a rede Perceptron. Criada por Rosenblatt (1957), a rede citada possui três camadas: entrada, processamento e saída.

A Figura 3 apresenta um esboço da rede Perceptron. A camada de entrada, assim como em vários outros modelos de RNAs, é formada por um vetor representado pela variável X. Esse vetor de entrada simboliza os elementos que influenciam no problema, pois cada um dos elementos de X tem sua devida importância na ativação dos neurônios.

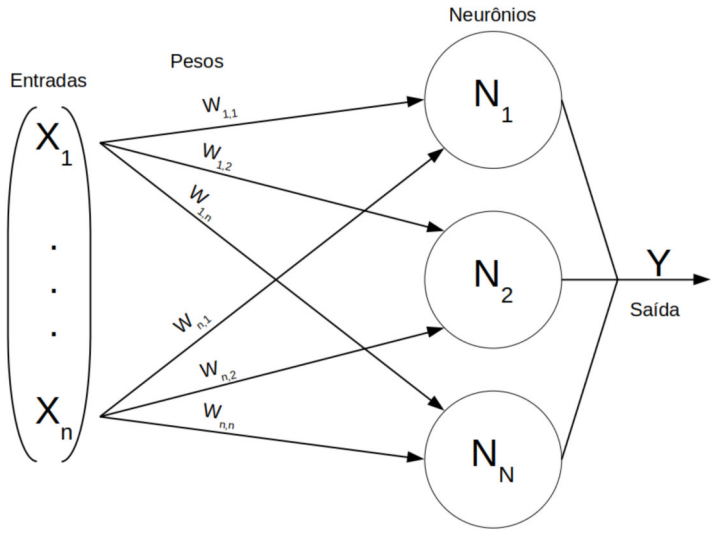
A correlação entre o valor de entrada e sua importância é denominada peso, a qual pode ser visualizada na Figura 3 representada pela variável W. O conjunto de pesos de uma rede pode ser considerado o conhecimento da mesma. Dessa forma, o processo de treinamento consiste em suas atualizações, a partir de valores normalmente aleatórios.

A camada de processamento é formada pelos neurônios internos, a qual geralmente recebe o produto de todos padrões de entrada pelos seus pesos correspondentes. Em seguida, a entrada desses neurônios passa pelas funções de ativação. Em uma visão ilustrativa, apresentam-se os neurônios dispostos em alinhamento vertical, o que dá suporte a expressão, corriqueiramente usada, largura da rede.

Após a ativação, os valores são repassados para a camada de saída. Esta é responsável pelo processamento final e, geralmente, tem seu valor representado pela variável Y. O resultado da última camada deve conter a resposta esperada para a entrada correspondente.

Além da estrutura, Rosenblatt (1957) também apresentou um modelo de treinamento que deveria ser adotado. Esse processo pretendia que a rede fosse capaz de atualizar seus pesos, buscando uma solução aproximada da desejada. Resumidamente, o treinamento

Figura 3 – Rede Perceptron.



Fonte: Autor.

consistia na comparação do resultado almejado d pelo resultado obtido pela rede Y. Então, esse tipo de treinamento denomina-se treinamento supervisionado.

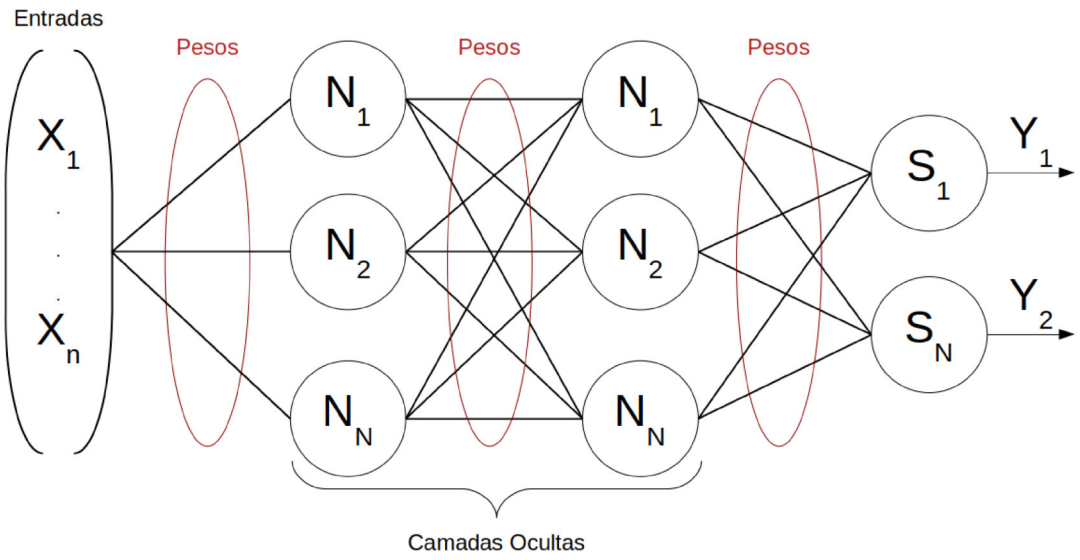
Apesar de inovadora para a época, a Perceptron, com a largura correta, só conseguia resolver problemas lineares, o que não representa a grande maioria dos problemas. Como na década não existiam outras alternativas, as redes neurais ficaram esquecidas até que Rumelhart, Hinton e Williams (1986) apresentaram sua rede Perceptron Multicamadas.

2.3 Rede Perceptron Multicamadas

Após anos sem grandes descobertas em RNAs, Rumelhart, Hinton e Williams (1986) lançaram um modelo de sua rede Perceptron multicamadas (MLP - *Multi Layer Perceptron*) usando os mesmos conceitos da Perceptron convencional, todavia com capacidade de ter mais do que apenas uma camada (BRAGA; CARVALHO; LUDERMIR, 2007). A Figura 4 apresenta uma ilustração da MLP e evidencia que a grande revolução dessa rede foi o algoritmo de retropropagação (BP - *Backpropagation*) do erro apresentado por ele. A partir dessa função é possível estimar o erro de uma camada anterior através do gradiente da função e realizar a atualização dos pesos de modo a diminuir o erro.

Na MLP, os neurônios são organizados em camadas e cada uma delas liga-se à subsequente até a camada de saída da rede. A entrada de um neurônio é formada pela soma do produto da saída de todos os neurônios da camada anterior pelos seus respectivos pesos, como apresentado na Equação 2.1 (HAYKIN, 1999). Nessa equação, net_i representa a entrada do neurônio i, out_j é a saída do neurônio j e w_{ij} é o peso da conexão entre o

Figura 4 – Rede MLP.



Fonte: Autor.

neurônio j e o neurônio i.

$$net_i = \sum_{j=1}^{n} out_j \cdot w_{ij} \tag{2.1}$$

A saída dos neurônios é dada por meio do cálculo da função de ativação apresentado na Equação 2.2, sendo realizado a partir do somatório de todas as suas entradas (HAYKIN, 1999). Ademais, a MLP é uma rede de treinamento supervisionado, ou seja, precisa ser conhecida a saída desejada na fase de treinamento para que, através dessa informação, possa atualizar os pesos.

$$out_i = f_i(net_i) \tag{2.2}$$

O erro da camada de saída da rede é calculado pela Equação 2.3, onde e_i é o erro do neurônio i, d_i é o resultado esperado para aquele neurônio e y_i é saída atual do neurônio i. A partir do erro da camada de saída, é possível calcular o erro das camadas anteriores por meio da retropropagação.

$$e_i = d_i - y_i \tag{2.3}$$

A Equação 2.4 mostra como o erro pode ser retropropagado para camadas anteriores, em que e_j representa o erro de um neurônio em uma camada oculta, e_i é o erro do neurônio

i da camada posterior, f'_i é a derivada da função de ativação do neurônio i da camada posterior e w_{ij} representa o peso que liga o neurônio j ao neurônio i da camada subsequente.

$$e_j = \sum_{i=1}^{n} e_i \cdot f'_i \cdot w_{ij} \tag{2.4}$$

Em teoria, com a apresentação do algoritmo de retropropagação, seria possível a implementação de redes tão profundas que poderiam resolver qualquer problema. Contudo, é comum que redes com essa característica não consigam atualizar os pesos das camadas iniciais, uma vez que o valor do erro vai ficando menor a cada camada. Esse fenômeno é intitulado como perda do gradiente.

Outro caso comum na literatura é o *overfit*. Esse termo é usado para descrever um estado em que a rede possui uma alta precisão nas amostras de treino, entretanto não consegue generalizar esse conhecimento para amostras desconhecidas.

Por conseguinte, o maior entrave para as redes muito profundas é o consumo de *hardware*. Redes com muitas camadas levam muito tempo para concluir um treinamento e se este não for satisfatório, deverá ser reiniciado.

O processo de reinicialização de um treinamento consiste na mudança da estrutura da rede, nos hiper-parâmetros de treinamento e reinicialização do pesos. Quando isso ocorre, todo o conhecimento adquirido no treinamento anterior é perdido e todo processo recomeça.

2.4 Múltiplas Redes Neurais Artificiais Autocoordenadas

O conceito de múltiplas redes neurais artificiais (MRNA) auto-coordenadas consiste na utilização de mais de uma rede para resolver um determinado problema (ALMEIDA NETO; GÓES; NASCIMENTO, 2010). Métodos parecidos estão presentes na literatura, porém a metodologia de Almeida Neto, Góes e Nascimento (2010) propõe um diferencial, o qual é a inserção das redes de maneira sequencial evitando conflitos entre as redes e dispensando o uso de um coordenador.

Isto posto, o método citado consiste em um modelo de treinamento em que as redes são inseridas em sequência. Assim que a primeira rede estabiliza da curva do erro, outra rede é inserida com a finalidade de adquirir o conhecimento ignorado pela primeira. O resultado do modelo é o somatório das saídas de todas as redes, conforme mostra a Equação 2.5. O processo de inserção de redes é repetido até que a solução apresentada

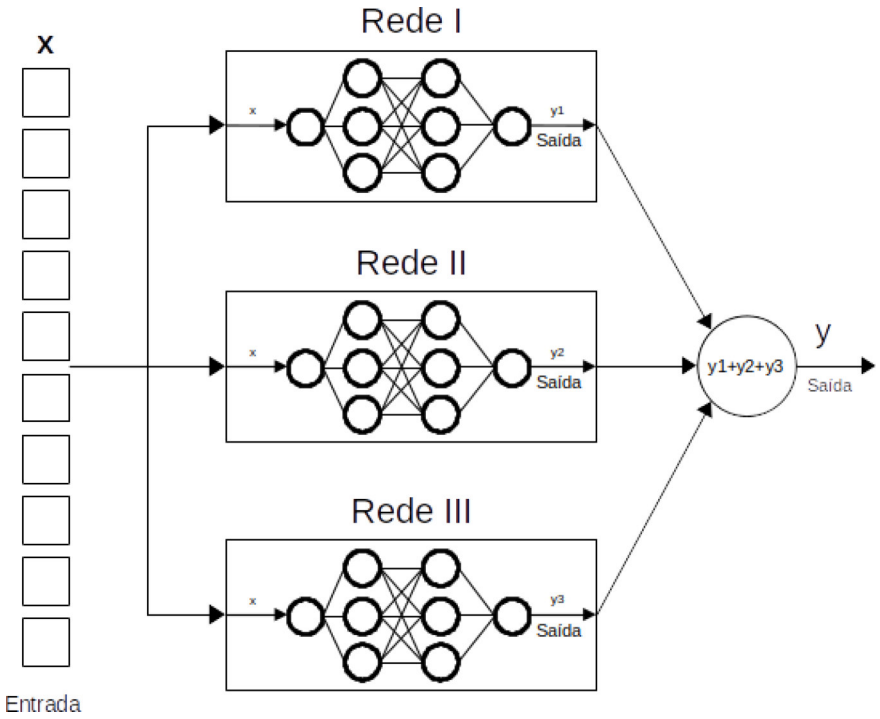
pelo conjunto das n redes seja considerada satisfatória.

$$y = \sum_{i=1}^{n} y_i \tag{2.5}$$

A Figura 5, ilustra o funcionamento do modelo. Nota-se que todas as redes compartilham o mesmo conjunto de dados de entrada. E apesar do treinamento sequencial, as saídas de todas a redes já inseridas são levadas em consideração no treinamento atual. Então, para que isso ocorra, a rede em treinamento é ajustada de acordo com a Equação 2.6. Nela, o erro do conjunto das redes e é obtido através da subtração da saída almejada d pelo somatório das saídas de todas a redes.

$$e = d - \sum_{i=1}^{n} y_i \tag{2.6}$$

Figura 5 – Dinâmica do funcionamento MRNA autocoordenadas.



Fonte: Autor.

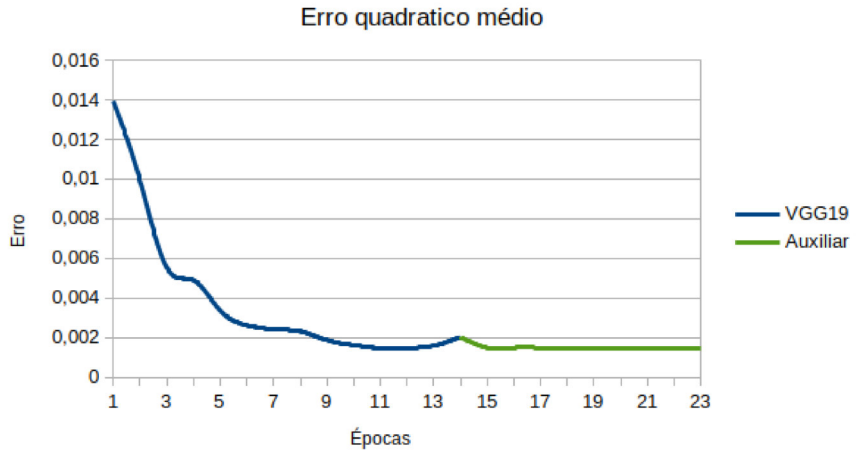
Com a inserção de uma nova rede com pesos aleatórios, o modelo é levado para outro ponto no espaço de busca, possibilitando fugas de mínimos locais. Basicamente existem duas técnicas de inserção das redes auxiliares de Almeida Neto, Góes e Nascimento (2010), que neste trabalho são intituladas de pacífica e aleatória. Na inserção pacífica, os pesos que ligam a última camada oculta à camada de saída são zerados (TEIXEIRA; ALMEIDA NETO, 2016). Ademais, como a saída da rede é um produto dos pesos pela entrada da camada, a saída da rede auxiliar será zero até que os pesos iniciais sejam

reajustados. Esta simples e eficaz técnica permite que a rede inserida não interfira na curva do erro do modelo, como observado na Figura 6a.

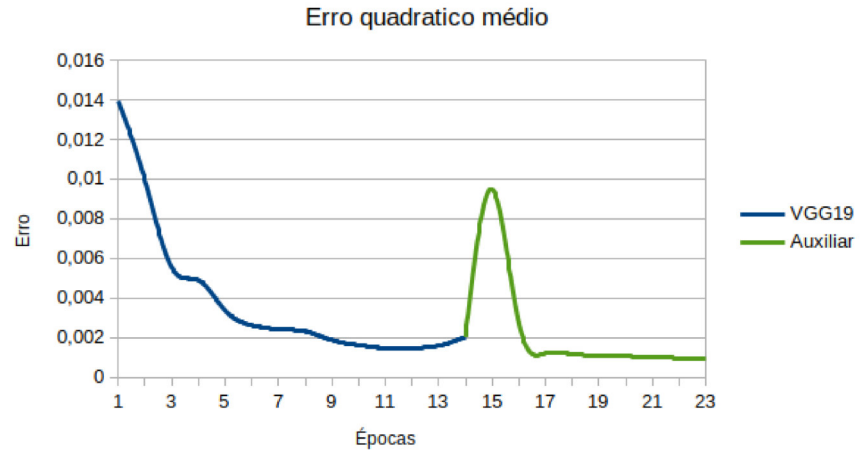
Enquanto a técnica de inserção pacífica quase sempre mantenha a tendência de queda na curva do erro, a inserção totalmente aleatória geralmente causa uma instabilidade que é refletida em um pico no gráfico de erro, como ilustrado na Figura 6b. Apesar disso, em alguns casos, a inserção totalmente aleatória pode ser mais eficaz, conforme demonstrado neste estudo.

Figura 6 – Exemplo dos tipos de inserção das redes auxiliares.

a) Inserção pacífica.



b) Inserção completamente aleatória.



Fonte: Autor.

Urge citar que habitualmente o termo profundidade é usado para ilustrar o número de camadas de uma RNA e largura, a quantidade de neurônios nas camadas. Logo, pode-se imaginar a adição de uma nova rede utilizando a estratégia de MRNA autocoordenadas, como a criação de uma nova dimensão à rede principal. Isso leva a questão do potencial do

método proposto, o qual combina a profundidade e largura das CNNs com a profundidade e largura de suas redes auxiliares.

2.5 Redes Neurais Convolucionais

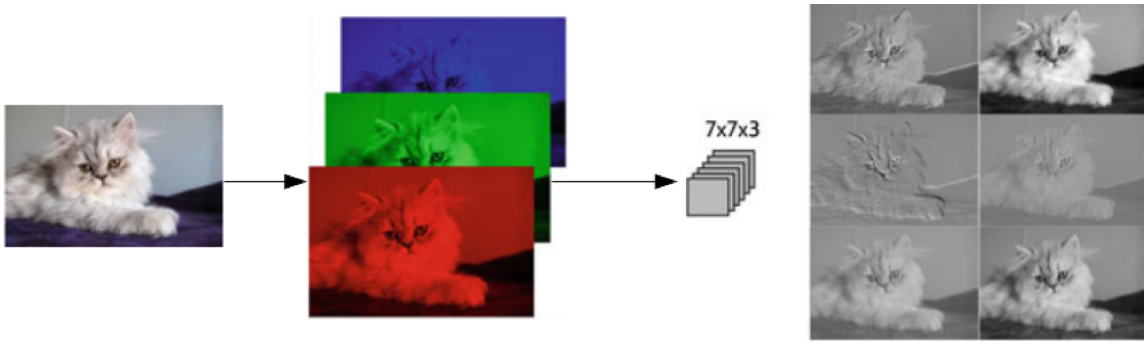
Uma rede convolucional é uma rede neural profunda que, geralmente, é utilizada para classificação de imagens ou sons. A classificação é um agrupamento de objetos com características semelhantes. E, embora seja uma técnica que data de meados dos anos 80, as redes convolucionais ficaram esquecidas por décadas, pois a rede neural com camadas suficientes para realizar esse tipo de classificação não existia (KIM, 2017).

Por conseguinte, desde 2012, as CNNs voltaram conquistando vários adeptos devido a evolução com o aprendizado profundo (KIM, 2017), pois já existem inúmeras variantes desse tipo. Contudo, a grande maioria das CNNs é formada basicamente por três tipos de camadas: convolução, *pooling* e completamente conectada.

2.5.1 Convolução

A camada convolucional tem por objetivo extrair características das imagens, conforme ilustrado na Figura 7. E a imagem passa por diversos filtros que têm a extração de características da imagem original como resultante do processo: bordas, formas e texturas, por exemplo. Portanto, o processo de convolução pode ser visualizado como a renderização da imagem anterior com suas características ressaltadas.

Figura 7 – Exemplo de Convolução.



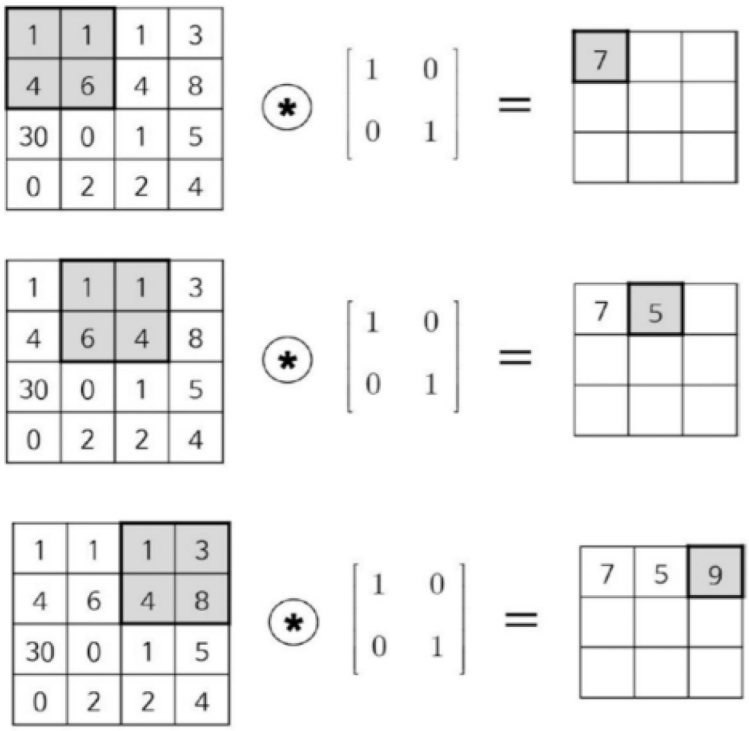
Fonte: adaptada de Aghdam e Heravi (2017).

A Figura 8, apresenta o funcionamento do filtro de convolução. A partir dela é possível observar que a imagem é percorrida por filtros de 2×2 *pixels*. De acordo com o exemplo, o passo, do inglês *strides*, é o deslocamento do filtro sobre a matriz contendo os dados. Por conseguinte, é comum a definição de que os filtros sejam pesos compartilhados entre os neurônios (KANG; WANG, 2014).

Ainda segundo o exemplo da Figura 8, verifica-se que o resultado da convolução é a soma dos elementos da matriz do produto entre a matriz principal e o filtro na posição atual. Durante o treinamento, os filtros são atualizados e ao final, cada um deles deve ser capaz de reconhecer uma característica da imagem.

Geralmente, as redes CNNs apresentam um grande número de convoluções, o que gera um demasiado aumento das amostras de treinamento. Então, para reduzir o volume de dados, é comum que, após uma sequência de convoluções, seja realizada uma subamostragem das imagens derivadas. Esse processo é conhecido como *pooling*.

Figura 8 – Filtro de convolução.



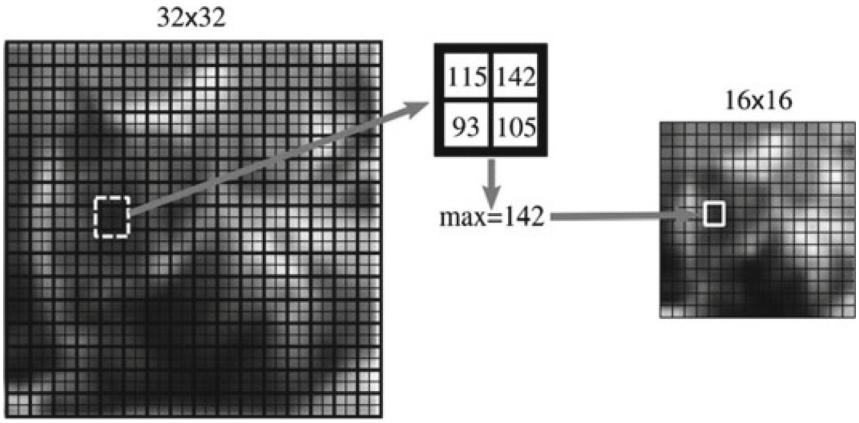
Fonte: Adaptada de Kim (2017).

2.5.2 *Pooling*

A camada de *pooling* é responsável pela diminuição da imagem usando uma solução não-linear. A Figura 9, exemplifica a aplicação do *pooling* máximo no resultante de uma convolução, na qual é possível notar que a matriz de 32×32 é reduzida para a resolução 16×16. E, da mesma forma da convolução, o *pooling* desloca-se na imagem usando um parâmetro chamado passo. O *pooling* máximo não é o único tipo de subamostragem, pois existem outras técnicas, como o *pooling* médio e mínimo, por exemplo.

A camada completamente conectada é responsável pela classificação dessas imagens. Usualmente, as amostras chegam na entrada do classificador em formato matricial. Por esse motivo, é comum encontrar uma camada, habitualmente chamada de *flatten*, dedicada

Figura 9 – Max *pooling*.



Fonte: Adaptada de Aghdam e Heravi (2017).

à conversão da matriz de entrada em um vetor. A partir daí, a dinâmica de treinamento segue as características que já foram abordadas na Seção 2.3.

2.5.3 Funções de Ativação

Como visto anteriormente, o núcleo de um neurônio artificial é uma das inúmeras funções de ativação existentes. Apesar de não ser uma regra, cada camada utiliza a mesma função de ativação, inclusive as camadas convolucionais. Nesta seção, serão apresentadas as funções de ativação utilizadas neste trabalho.

A função linear consiste no produto do valor de entrada por uma constante, como representado pela Equação 2.7, onde a constante a é um valor real diferente de zero.

$$f(x) = ax \tag{2.7}$$

A *Rectified Linear Units* (ReLU) é uma função não-linear que converge até seis vezes mais rápida que a função tangente hiperbólica(KRIZHEVSKY; SUTSKEVER; HINTON, 2012). Essa última é apresentada na Equação 2.8, enquanto a ReLU consiste em zerar valores negativos e manter os valores positivos.

$$ReLU(x) = max(0, x) \tag{2.8}$$

A *Scaled Exponential Linear Units* (SELU) é uma variante da ReLU que permite saídas negativas e mantém uma maior instabilidade na curva do erro (KLAMBAUER *et al.*, 2017). A função de SELU está representada pela Equação 2.9 e recomenda-se o valor

de $\lambda > 1$ para seu melhor desempenho.

$$SELU(x) = \lambda \begin{cases} x, & \text{se } x > 0 \\ \alpha e^x, & \text{se } x \leq 0 \end{cases} \tag{2.9}$$

A *SoftSign*, apresentada na Equação 2.10, foi desenvolvida por Bergstra *et al.* (2009). A função retorna valores entre -1 e 1. E, segundo os autores, a *SoftSign* superou, em vários testes, funções como sigmoide e tangente hiperbólica.

$$SoftSign(x) = \frac{x}{(1 + |x|)} \tag{2.10}$$

A função *SoftMax* é usada habitualmente nas CNNs, sendo também a função de ativação da camada de saída da maioria das redes citadas neste estudo. A função, apresentada na Equação 2.11, consiste no cálculo da probabilidade de distribuição dos valores de um vetor. Logo, durante a classificação, o valor um é dividido entre as classes. Esse processo ocorre de acordo com o valor da classe em relação ao resultante da soma de todas as outras saídas da camada (NWANKPA *et al.*, 2018).

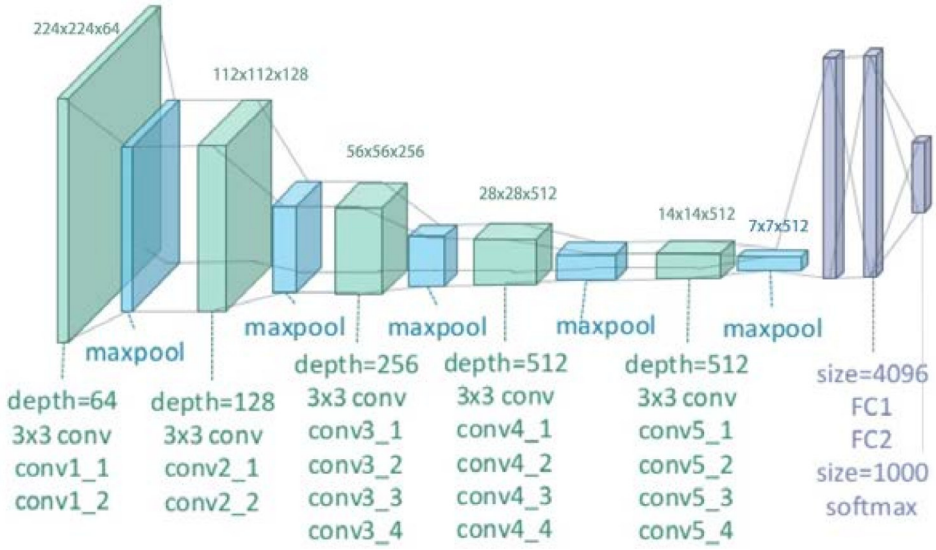
$$SoftMax(x_i) = \frac{e^{x_i}}{\sum_j e^j} \tag{2.11}$$

2.6 VGG19

A rede VGG19 foi apresentada no estudo de Simonyan e Zisserman (2015) com o objetivo de apresentar a correlação entre a profundidade da rede e sua capacidade de classificação. Este trabalho selecionou essa rede pois, apesar de profunda, ela possui um menor número de parâmetros a serem treinados em relação às redes com desempenho semelhantes. A escolha permitiu que a rede fosse treinada com *hardwares* disponíveis e em um tempo acessível, possibilitando um maior número de simulações.

A rede VGG19 possui o total de 19 camadas organizadas em seis blocos intercalados com *pooling* máximo e sua arquitetura pode ser verificada na Figura 10. Assim, os cinco primeiros blocos são formados por camadas de convolução com filtros 3×3. O primeiro deles é formado por duas camadas com 64 neurônios. Ao passo que o segundo é formado por mais duas camadas de convolução de 128 neurônios. O terceiro é formado por quatro camadas de 256 neurônios. Já o quarto e o quinto blocos são formados por quatro camadas de 512 neurônios cada uma. E, por fim, o sexto bloco é formado por uma MLP de três camadas, em que as duas primeiras são de 4096 neurônios e a última com 1000.

Figura 10 – Organização das camadas da VGG19.



Fonte: Zheng, Yang e Merkulov (2018).

A arquitetura apresentada possui um classificador com essas proporções porque foi avaliada no desafio Imagenet[1]. A Imagenet é a maior base de imagens marcadas do planeta (DENG *et al.*, 2010) e está catalogada em mil classes. Logo, a última camada da VGG19 possui um neurônio para cada objeto rotulado na Imagenet.

A VGG19 utiliza o aprendizado supervisionado em seu processo de treinamento. Por consequência, é essencial que a base utilizada possua imagens previamente rotuladas com as classes que se deseja ensinar a rede. Antes do início do processamento, os pesos da rede devem ser definidos. Eles podem ser iniciados aleatoriamente, utilizando estimativas do projetista ou através de técnicas de transferência de aprendizagem.

O treinamento de uma rede convolucional exige uma quantidade expressiva de amostras para que seja eficaz (ZHENG; YANG; MERKULOV, 2018). Logo, a transferência de aprendizado possibilitou a utilização do conhecimento de redes treinadas com um grande número de dados para problemas que possuem bases com menor número de amostras (ZHENG; YANG; MERKULOV, 2018).

Na prática, a técnica supracitada consiste no carregamento dos pesos resultantes de um treinamento satisfatório para outro problema. E em seguida, a adequação das dimensões da rede para o problema que se pretende aprender.

Além da função de atualização do pesos apresentada na Seção 2.3, existem outras formas de realizar esse processo. Esse parâmetro, geralmente, é conhecido como otimizador da rede. Além dos parâmetros específicos que cada função de otimização exige, todos os otimizadores têm em comum um parâmetro denominado taxa de aprendizado. A escolha

[1] http://www.image-net.org/challenges/LSVRC/

do otimizador e da sua taxa de aprendizado são fatores de suma importância para o treinamento do modelo.

Com a estrutura da rede, o tipo de inicialização dos pesos e o otimizador definidos, o treinamento é inicializado. Além dos hiper-parâmetros já citados, existem outros que podem ser adotados para auxiliar no gerenciamento do treinamento, como: o número máximo de épocas e o limite de épocas sem atualizações consideráveis na acurácia das predições, por exemplo.

Visando a diminuir o consumo do *hardware*, cada época de treinamento de uma rede CNN pode ser dividida em lotes. A cada lote, a rede realiza o processo de cálculo e a retropropagação do erro, similar ao processo visto na Seção 2.3. Porém, as camadas de convolução são formadas por pesos comunitários, logo o processo de convolução tem que ser inversamente realizado para que se possa estimar o erro e, assim, aplicar a atualização dos pesos.

3 Materiais e Métodos

Este trabalho propõe um metodologia de treinamento complementar, gradual e acumulativo baseado em MRNA autocoordenadas. As MRNAs já demonstraram sua eficácia como alternativa ao modelo tradicional de treinamento (ALMEIDA NETO; GÓES; NASCIMENTO, 2010; OLIVEIRA; ALMEIDA NETO, 2013; TEIXEIRA; ALMEIDA NETO, 2016). Diante desse contexto, há a intenção de determinar se a estratégia pode ser aplicada com êxito nas camadas completamente conectadas das redes convolucionais. Portanto, objetiva-se que as redes em produção, as quais possuam problemas em aberto, consigam avançar sem mudança na estrutura já treinada.

Os modelos de CNNs cogitados para o experimento foram: VGG19 (SIMONYAN; ZISSERMAN, 2015), ResNeXt (XIE *et al.*, 2017) e a DenseNet (HUANG *et al.*, 2017). A VGG19 foi escolhida por apresentar melhor eficiência em relação ao tempo de treinamento comparada à base escolhida. Este parâmetro foi decisivo, pois permitiu um maior número de treinamentos e, consequentemente, um resultado mais robusto.

Para comprovar a eficiência do método, a proposta de treinamento foi dividida em duas fases, segundo ilustrações da Figura 11. A primeira delas consiste na adaptação da VGG19 utilizada no desafio Imagenet[1] à base escolhida. Então, a adaptação foi realizada por técnicas de pré-processamento e alterações na camada completamente conectada da rede, as quais serão descritas nas seções posteriores. Na segunda fase, as redes auxiliares foram adicionadas e a eficiência do modelo, mensurada.

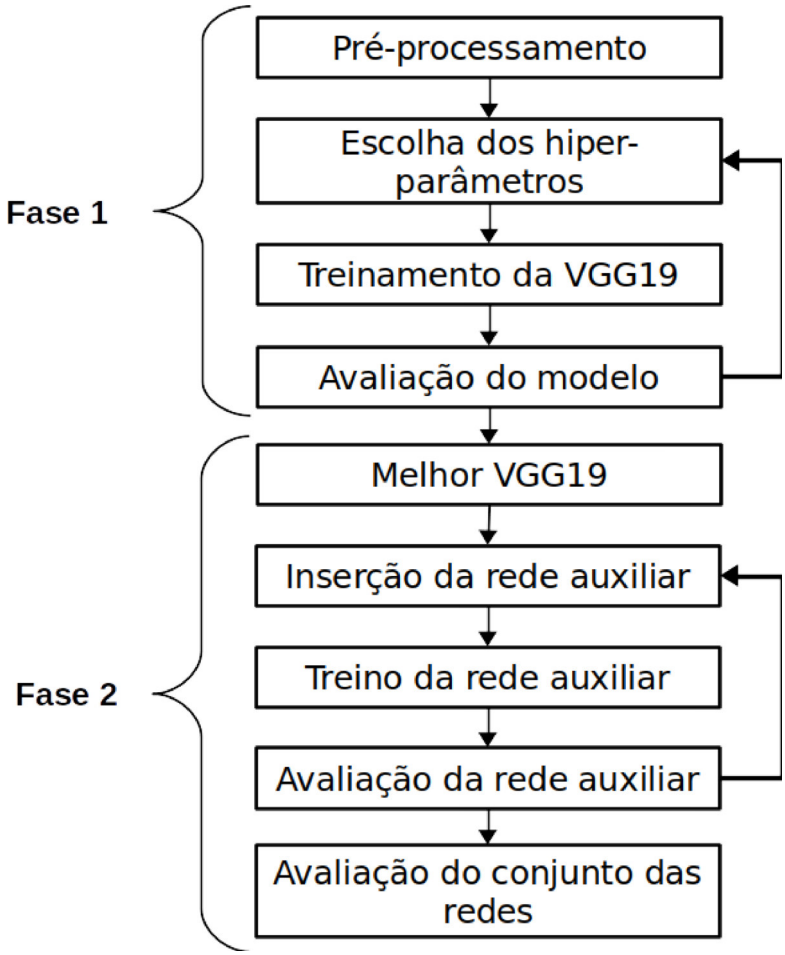
Com relação à programação, Python foi a linguagem escolhida para a codificação. Essa decisão visa à fácil replicação do estudo em outros trabalhos, uma vez que está presente em vários estudos da literatura. Além disso, essa linguagem possui uma biblioteca para implementação de CNNs, de maneira descomplicada, denominada TensorFlow[2]. Para a criação dos modelos no TensorFlow, utilizou-se a interface de programação de aplicação (API - *Application Programming Interface*) Keras[3].

3.1 Base de Imagens

Várias bases foram cogitadas para realização do experimento. Por conseguinte, a escolha levou em consideração o tamanho das imagens, a quantidade de amostras da base e as marcações presentes na mesma que deveriam possuir no mínimo as marcações: sem semáforo, semáforo em vermelho e semáforo em verde. Aliás, a presença das bases em

[1] http://www.image-net.org/challenges/LSVRC/
[2] https://www.tensorflow.org/
[3] https://keras.io/

Figura 11 – Diagrama da metodologia do treinamento.



Fonte: Autor.

outros trabalhos foi um fator predominante, tendo em vista a possibilidade de confirmação dos resultados mediante comparações com outros da literatura.

Entre as bases disponíveis, a LaRA (CHARETTE; NASHASHIBI, 2009), que é um vídeo das ruas da cidade de Paris, foi utilizada. Nessa base, há 11.179 imagens com baixa resolução (640×480) permitindo o armazenamento de um maior número de amostras na memória simultaneamente o que, por sua vez, acelera o processo de treinamento.

Outrossim, a base é rotulada como: sem semáforo, vermelho, amarelo, verde e ambíguo (*background, red, orange, green e ambiguos*). A etiqueta ambígua teve suas amostras descartadas por não estar no contexto previsto para classificação de semáforos.

3.2 Primeira fase - treinamento da VGG19

A primeira fase do treinamento concentrou-se na busca de uma VGG19 com robustez em reconhecimento e classificação de semáforos, pois se a rede principal não estivesse com uma acurácia próxima aos modelos presentes na literatura, a relevância do trabalho poderia ser questionada. Antes do treinamento, a base passou por alguns pré-processamentos visando à adaptação ao modelo e a redução no tempo de treinamento.

3.2.1 Pré-processamento

As amostras da base foram divididas entre treino, validação e testes, seguindo, respectivamente, as proporções: 70%, 10% e 20%. A base é um vídeo que possui vinte e cinco quadros por segundo criando uma relação de sequência temporal. Esse fator temporal é usado como confirmação de predição em alguns trabalhos na literatura, como citado anteriormente.

O fato da base ser um vídeo, poderia levantar dúvidas sobre a robustez do experimento. Em uma separação aleatória das imagens, algumas de uma mesma cena, contudo em quadros diferentes, poderiam ser postas nos grupos de treino, validação e testes. Isso, por sua vez, inflaria positivamente a predição da rede. Esse fator foi levado em consideração na divisão das amostras, impedindo que quadros do vídeo de um mesmo período temporal aparecessem em grupos diferentes.

Como exemplo, imagens de semáforo em amarelo (etiqueta amarelo) são apenas 58 amostras, as quais representam o mesmo semáforo no mesmo período de tempo. Se pelo menos uma das amostras dessa etiqueta estivesse presente na fase de treinamento, isso resultaria em 100% de acertos na fase de testes, mesmo não garantindo que a rede realmente tivesse aprendido a classificar os semáforos em amarelo. Outrossim, caso todas essas amostras ficassem apenas no teste e/ou na validação, a rede não saberia como classificá-las, devido à ausência de amostras desse tipo na fase de treinamento. Sendo assim, seguindo outros trabalhos, as imagens com essa marcação foram descartadas.

Com os devidos cuidados na distribuição das amostras e tentando respeitar ao máximo a disposição apresentada, a organização das imagens utilizadas no experimento resultou nos quantitativos apresentados na Tabela 1.

A entrada padrão da VGG19 é uma matriz de 224×224 com três canais (vermelho, verde e azul). Esse padrão diverge do tamanho da imagem da base que é de 640×480. Neste caso, pode-se alterar a entrada padrão da rede ou reduzir o tamanho da imagem. Mas como o intuito era manter a estrutura da VGG19 o mais fidedigna possível, optou-se pela conservação do padrão de entradas e a redução das imagens.

A Figura 12 apresenta as duas técnicas utilizadas de redução da imagem. A

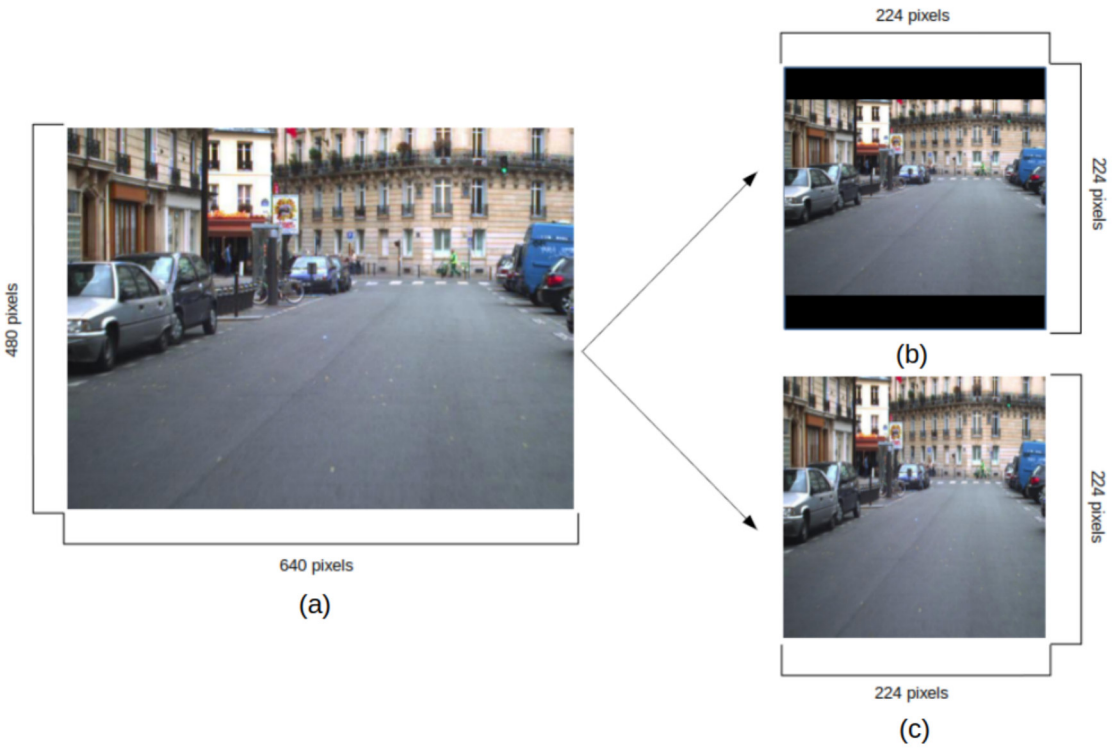
Tabela 1 – Distribuição das amostras da base LaRA.

Etiqueta	**Treino**	**Validação**	**Testes**	**Total**
Sem semáforos	3565	396	990	4951
Vermelho	2280	253	633	3166
Verde	1340	148	371	1859
Total	7185	797	1994	9976

Fonte: Autor.

Figura 12a, apresenta uma imagem da base em tamanho original e a Figura 12b, a imagem redimensionada mantendo suas proporcionalidades e, por isso, os *pixels* faltantes foram preenchidos pela cor preta. Na imagem 12c, é ilustrado o redimensionamento sem o respeito à proporcionalidade, em que é possível observar uma deformação.

Figura 12 – Redimensionamento das amostras.



Fonte: Autor.

Algumas técnicas, como a geração de novas imagens a partir da rotação das amostras, foram testadas. Entretanto, nenhuma dessas técnicas apresentou ganho real na fase de testes e dessa forma, não foram utilizadas.

3.2.2 A escolha dos hiper-parâmetros

Como já foi apresentada, a VGG19 disponível no Keras[4] foi utilizada como rede principal. O modelo possui uma rede MLP responsável pela classificação das amostras na parcela final de sua estrutura. Contudo, a rede foi treinada para a base Imagenet (DENG *et al.*, 2010) que é formada por mil classes. E esta diverge da base escolhida, pois possui apenas três classes: sem semáforos, vermelho e verde.

Por conter uma MLP muito grande, a alteração apenas da saída da rede de mil para três classes mostrou-se ineficiente. Isso decorreu, conforme teorizado, devido ao fato de que uma rede com muitos neurônios tendeu ao *overfit* e por isso, julgou-se necessário a alteração da saída da VGG19.

Isto posto, a dinâmica de escolha dos hiper-parâmetros da primeira fase do treinamento consistiu no tipo de compressão das imagens, no número de camadas da MLP, taxa de aprendizado, número de neurônio e funções de ativação de cada camada. Além disso, o hyperopt ficou responsável pela seleção do otimizador, que neste contexto é a função matemática utilizada para atualizar os pesos. Após a seleção de um conjunto de hiper-parâmetros, deu-se início a fase de treinamento.

3.2.3 Treinamento da VGG19

Na etapa de treinamento, algumas configurações foram realizadas para otimizar o processo. Em redes pequenas, as quais possuem poucos neurônios de entrada ou poucas amostras de treinamento, é comum que todas as amostras sejam passadas de uma vez só para a rede. Isso permite que a rede atualize os pesos apenas no final da época, o que, em teoria, aperfeiçoaria o ajuste dos pesos.

O número de amostras utilizadas no treinamento gerou uma carga bem maior do que o suportado pelo *hardware* disponível. Dessa forma, foi impossível carregar todas as imagens para a rede de uma única vez. Logo, as amostras foram divididas em lotes, conhecidos como *batch size*. Para o treinamento da rede, foram utilizados lotes de oito imagens, gerando 899 lotes por época e a cada um deles, a VGG19 atualizava seus pesos.

O treinamento em lotes permite maior rapidez, uma vez que com a memória disponível, foi possível o armazenamento de todas as imagens pré-processadas e treinamento dos lotes dispensando o acesso ao disco. O ganho de desempenho deu-se pela dispensa do acesso citado, durante o treinamento, o que tornaria o processo mais moroso.

Os treinamentos foram fixados em no máximo duzentas épocas e a cada vez que a rede obtivesse um resultado melhor, em comparação aos já realizados, os pesos eram salvos. Com o intuito de evitar treinamentos indesejáveis, foi estipulado um limite de três épocas de tolerância sem melhora na validação. Desse modo, caso o critério de tolerância

[4] https://keras.io/api/applications/vgg/#vgg19-function

fosse atendido, o treinamento seria abortado e a rede retornaria os pesos com a melhor acurácia.

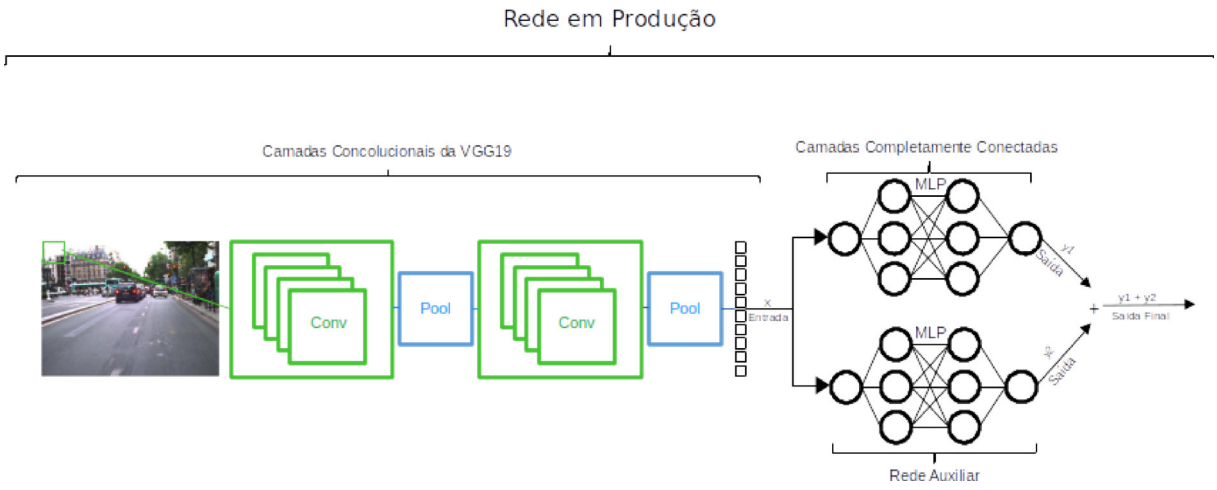
3.2.4 Avaliação da VGG19

A rede VGG19 foi avaliada com as imagens separadas para o teste. Para aferir sua precisão, utilizou-se a matriz de confusão (MC). Outrossim, com a finalidade de demonstrar que as redes auxiliares podem ser aplicadas às redes em produção, buscou-se encontrar uma rede com valores aproximados aos do estado da arte e então, aprimorá-la com as redes auxiliares. Logo, enquanto não se obtiveram resultados próximos ao estado da arte, os hiper-parâmetros foram alterados e o treinamento reiniciado.

3.3 Segunda fase - redes auxiliares

A contribuição do trabalho começa na proposta de inserção das redes auxiliares. Após o treinamento da rede VGG19, caso uma solução ótima não seja encontrada, o modelo deve ser retreinado. Nesta abordagem, o próximo passo é tentar melhorar a classificação da rede partindo da adição de MLPs auxiliares. A Figura 13 apresenta como ficaria o modelo com os dois classificadores: a rede nativa VGG19 e a rede auxiliar inserida na segunda fase do processo. É válido ressaltar que nas seções posteriores será apresentada uma alternativa a esse processo.

Figura 13 – Rede VGG19 com uma rede auxiliar.



Fonte: Autor.

3.3.1 Inserção da rede auxiliar

Apesar da Figura 13 apresentar uma rede com dois classificadores, essa abordagem não está limitada a esse número. Como ilustrado, as redes auxiliares foram inseridas exatamente na mesma camada que o classificador da rede principal, compartilhando assim os mesmos padrões de entrada.

Exauridas as tentativas de melhorar a rede principal, inicia-se a inserção de redes auxiliares. Esse processo objetiva, assim como em Almeida Neto, Góes e Nascimento (2010), aprender o que a rede principal desconsiderou. Logo, o modelo pode ser visualizado com uma rede CNN com múltiplos classificadores.

De acordo com o abordado no capítulo anterior e ilustrado na Figura 6, existem dois tipos possíveis de inserção para as redes auxiliares: a inserção pacífica e a completamente aleatória. Esses modelos de inserção foram utilizados nos treinamentos.

3.3.2 Treinamento da rede auxiliar

Certamente, a maior diferença deste trabalho para os que existem na literatura é o seu modelo de treinamento, o qual tentou ser o mais fiel possível ao modelo de Almeida Neto, Góes e Nascimento (2010), contudo adaptações foram necessárias.

O treinamento utilizando o modelo tradicional mostrou-se lento e, por diversas vezes, excedeu a memória disponível. Assim, o problema foi resolvido diminuindo o consumo de *hardwares* durante a fase de treinamento. Para isso, o conjunto das redes já treinadas só foi usado na primeira época de treinamento, uma vez que os seus pesos não serão atualizados.

Durante a primeira época do treinamento da primeira rede auxiliar, ou seja, a partir do momento que a rede principal não será mais treinada, a saída da camada *flatten* foi armazenada. Esse processo dispensa a necessidade das imagens passarem pelos blocos de convolução novamente, pois as informações armazenadas passam a ser o padrão de entrada para todas a redes auxiliares posteriores.

Como o padrão de entrada possuía um vetor bem menor do que a matriz de entrada original, permitiu o aumento do lote de treinamento, que inicialmente era formado por oito imagens. E, após o processamento, suportou até 32 amostras, diminuindo consideravelmente o número de lotes.

Para melhorar ainda mais o desempenho, a saída do modelo durante o treinamento e os dados de validação também foram armazenados. Portanto, isso permitiu que a rede auxiliar tenha seu treinamento independente do restante do modelo. Logo, todas as redes auxiliares foram treinadas com a saída da camada *flatten* chamada de X'. Já a saída desejada d foi modificada para que o treinamento da rede auxiliar tornasse-se independente

da execução das redes já treinadas.

A saída desejada da rede auxiliar k em treinamento segue a Equação 3.1. A saída desejada d_k é o valor desejado total menos a saída de todas as redes já treinadas y_i menos a saída da rede k em treinamento.

$$d_k = d - \sum_{i=1}^{n} y_i - y_k \tag{3.1}$$

Onde:
n = *total de redes*
k = *rede em treinamento*

Imaginando que cada rede auxiliar adiciona uma nova dimensão à principal, o modelo apresentado permite a paralelização dos treinamentos das redes auxiliares para escolher a que melhor atende a dimensão atual. As redes auxiliares passaram pelo mesmo grifo de treinamentos da rede principal, pois, se o modelo não evoluiu na validação por três épocas consecutivas, o treinamento foi abortado, tendo como limite máximo de duzentas épocas de treinamento.

3.3.3 Avaliação da rede auxiliar

Após o treinamento da rede auxiliar, o modelo completo das redes foi validado através do conjunto de teste, em que a saída do modelo é a somatória da saída de todas as redes auxiliares e a rede principal, conforme mostra a Equação 3.2.

$$y = \sum_{i=1}^{n} y_i \tag{3.2}$$

Se a rede contribuir significativamente com o resultado na fase de teste, é incluída ao modelo permanentemente. Logo após, o calculo da saída desejada, d_k, é refeito e armazenado levando a nova rede em consideração. Em seguida, o modelo passa para o treinamento em uma nova dimensão. De modo contrário, a rede é descartada. Em caso de inserção e descarte de três redes auxiliares, o treinamento era encerrado.

3.3.4 Avaliação do conjunto das redes

Com o intuito de avaliar o modelo através da comparação aos trabalhos presentes na literatura, utilizaram-se as seguintes métricas de avaliação: a precisão e o *recall*. Para tal, as equações utilizadas seguem as seguintes siglas: Verdadeiro Positivo (VP), Falso Positivo (FP) e Falso Negativo (FN).

A precisão mostra a capacidade de classificar as amostras de maneira correta. Quando a rede indica a presença de objetos que não pertencem a uma classe como sendo pertencentes, o valor da precisão diminui (JENSEN *et al.*, 2016). O modelo matemático usado para quantificá-la é apresentado na Equação 3.3.

$$precisão = \frac{VP}{VP + FP} \tag{3.3}$$

Já o *recall* diminui quando a rede deixa de classificar um objeto pertencente a uma classe (JENSEN *et al.*, 2016). A fórmula matemática para se obter o *recall* está exposta na Equação 3.4.

$$recall = \frac{VP}{VP + FN} \tag{3.4}$$

O ideal é que a precisão e o *recall* sempre tenham valores próximos, elevando o grau de confiabilidade do modelo (JENSEN *et al.*, 2016). Como o modelo de predição possui mais de uma classe, o resultado final é a média ponderada do resultado obtido por cada classe.

4 Resultados e Discussão

Este capítulo dedica-se a explanação dos resultados obtidos nesse estudo. Os dados são apresentados, assim como na metodologia, em duas fases: treinamento da rede principal e a inclusão das redes auxiliares. A fim de proporcionar um melhor entendimento sobre o funcionamento e a eficiência do método, várias simulações foram realizadas ao decorrer deste estudo. Portanto, como seria inviável descrever todos os testes realizados, foram selecionados três experimentos classificados, respectivamente, como: pior caso, caso mediano e o melhor caso.

4.1 Rede Principal

A rede VGG19 possui um classificador que se resume à uma MLP. A estrutura desse classificador, usado no desafio Imagenet[1], possui uma saída com mil classes. Durante a primeira fase do treinamento, alterou-se a estrutura do classificador para se adaptar a base utilizada. Desse modo, a Tabela 2 apresenta alguns exemplos das estruturas das MLPs usadas. Os dados estão organizados em: nome da camada, seus respectivos números de neurônios e a função de ativação utilizada na camada [2]. A estrutura original do classificador da VGG19 pode ser observada na Tabela 2a.

A primeira abordagem para a escolha da nova estrutura da rede partiu da substituição apenas da camada de saída por uma com três neurônios, como apresentado na Tabela 2b. Apesar de infrutíferas tentativas de treinamento alterando os outros hiper-parâmetros, esta estrutura resultou no pior caso. Acredita-se que o péssimo desempenho deu-se pelo mau dimensionamento do classificador ao problema, levando ao *overfit* do modelo.

O caso mediano, exposto na Tabela 2c, apresenta uma estrutura com menos neurônios, entretanto com uma camada a mais que a rede original. Mesmo o caso mediano apresentando resultados relevantes, o destaque central fica ao melhor caso. A melhor rede principal veio da combinação da estrutura apresentada na Tabela 2d com os seguintes hiper-parâmetros de treinamento: para o redimensionamento das imagens, utilizou-se a compressão com distorção, conforme a Figura 12c. O otimizador escolhido foi a descida do gradiente estocástico (SGD - *Stochastic Gradient Descent Optimizer*)[3] com taxa de aprendizado de 0,01.

[1] http://www.image-net.org/challenges/LSVRC/
[2] https://keras.io/activations/
[3] https://keras.io/optimizers/

Tabela 2 – Exemplos de estruturas do classificador da VGG19.

a) Original.

	N	**F.A.**
Entrada	4096	Relu
Escondida	4096	Relu
Saída	1000	SoftMax

b) Pior caso.

	N	**F.A.**
Entrada	4096	Relu
Escondida	4096	Relu
Saída	3	SoftMax

c) Caso mediano.

	N	**F.A.**
Entrada	800	Relu
Escondida	100	Relu
Escondida	100	Relu
Saída	3	SoftMax

d) Melhor caso.

	N	**F.A.**
Entrada	4096	Relu
Escondida	300	Relu
Saída	3	SoftMax

Onde:
N = Número de neurônios
F.A. = Função de Ativação

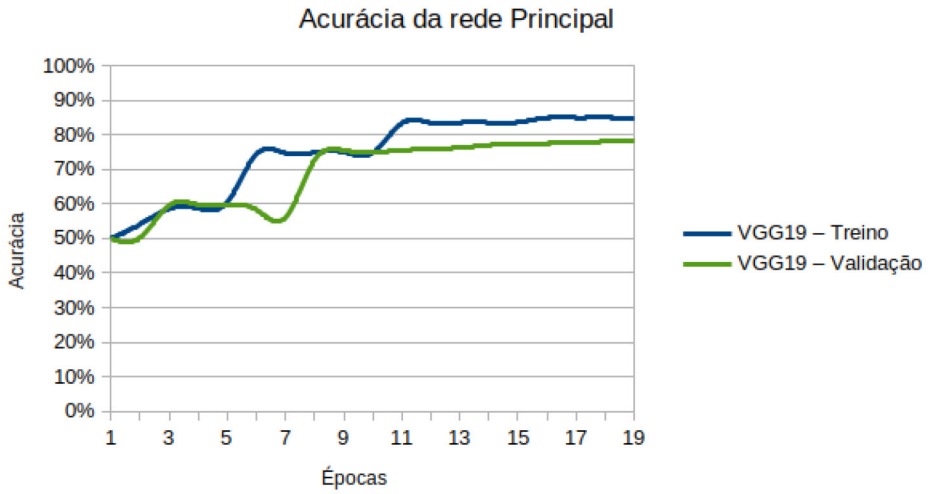
Fonte: Autor.

Os resultados dos treinamentos abordados estão ilustrados na Figura 14. Os gráficos seguem o mesmo padrão, utilizando a cor azul para as amostras de treinamento e a verde para as de validação. Nota-se que, já na fase de treinamento, a rede com melhor caso levou vantagem sobre as outras, tendo sua acurácia de classificação próxima aos 100%, como ilustrado na Figura 16a. Contudo, o que deve ser levado em consideração para a escolha do melhor modelo é o desempenho com as amostras de testes. Assim, esse desempenho foi avaliado com auxilio de matrizes de confusão como apresentado na Figura 15.

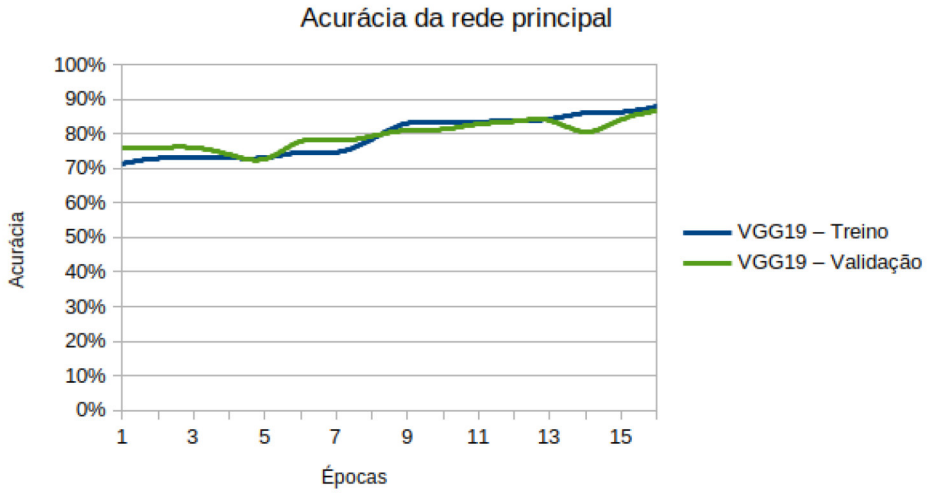
Por fim, a fase de testes apresentou como a pior precisão o valor de 50,07% (Figura 15a), o resultado do caso mediano foi de 79,92% (Figura 15b) e o melhor caso de 84,2% (Figura 15c). Como o objetivo do trabalho é contribuir com as pesquisas de redes convolucionais mostrando a eficácia do método para melhorar redes já treinadas, o melhor caso de treinamento serviu como referência para escolher uma estrutura para a rede auxiliar.

Figura 14 – Exemplos de treinamentos da VGG19.

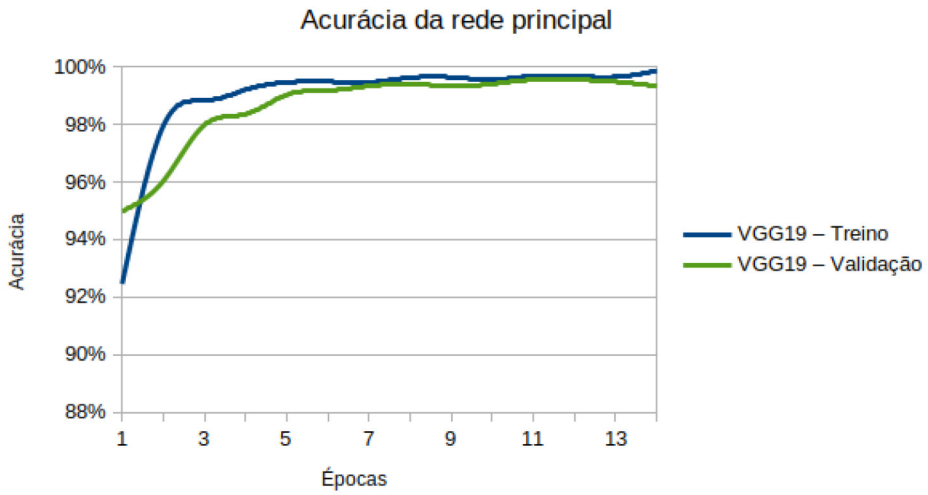
a) Pior caso.



b) Caso mediano.



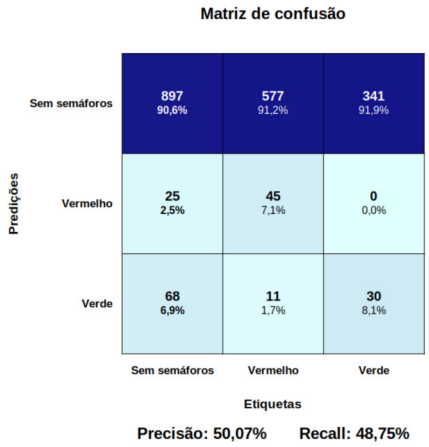
c) Melhor caso.



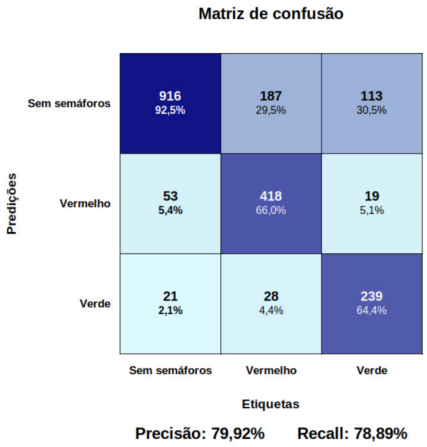
Fonte: Autor.

Figura 15 – Testes da VGG19.

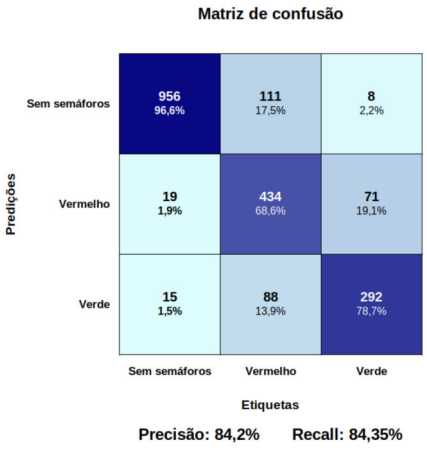
a) Pior caso.



b) Caso mediano.



c) Melhor caso.



Fonte: Autor.

4.2 Redes Auxiliares

Após a identificação de uma rede VGG19 que corresponderia à uma rede em produção, a próxima etapa foi a inserção de redes auxiliares. Visando a abranger o maior espaço de busca e evitar vício na escolha da estrutura e dos hiper-parâmetros da rede, os treinamentos das MLPs auxiliares foram automatizados utilizando a biblioteca hyperopt[4].

A meta-eurística utilizada pela biblioteca teve como espaço de busca as funções de

[4] https://github.com/hyperopt/hyperopt

ativação para cada camada da rede. Além disso, o hyperopt foi responsável pela escolha do otimizador utilizado, tendo como referência todos os implementados no keras.

Outro hiper-parâmetro automatizado foi a taxa de aprendizado, a qual variou entre 0,001 a 0,05. Ademais, o processo automatizou a escolha do número de neurônios de cada camada disponível, variando entre quatro neurônios a cinco mil neurônios. Outra decisão atribuída ao hyperopt foi a seleção do tipo de inserção de rede, que poderia variar entre pacífica ou aleatória, como ilustrado na Figura 6.

Tabela 3 – Estrutura da melhor rede auxiliar.

	Neurons	**Activation Function**
Entrada	2169	Relu
Escondida	1354	SoftSign
Saída	3	Selu

Fonte: Autor.

Os testes foram realizados em redes de duas a seis camadas, que unidas às simulações manuais, totalizaram 425 experimentos. A estrutura de rede auxiliar unitária que obteve o melhor resultado possui três camadas, conforme apresentada na Tabela 3. Além disso, essa estrutura utilizou em seu treinamento o otimizador descida do gradiente estocástico. Além disso, a taxa de aprendizado escolhida foi 0,005. Por fim, o método de inserção da rede que obteve o melhor desempenho foi com pesos aleatórios, ilustrada na Figura 6b, com seus pesos inicias variando entre um negativo e um positivo.

Vale pontuar que, apesar de os testes indicarem que a inserção pacífica apresenta uma convergência mais rápida e uma melhor estabilidade na curva do erro, as inserções aleatórias mostraram-se mais eficientes em relação ao conjunto de testes.

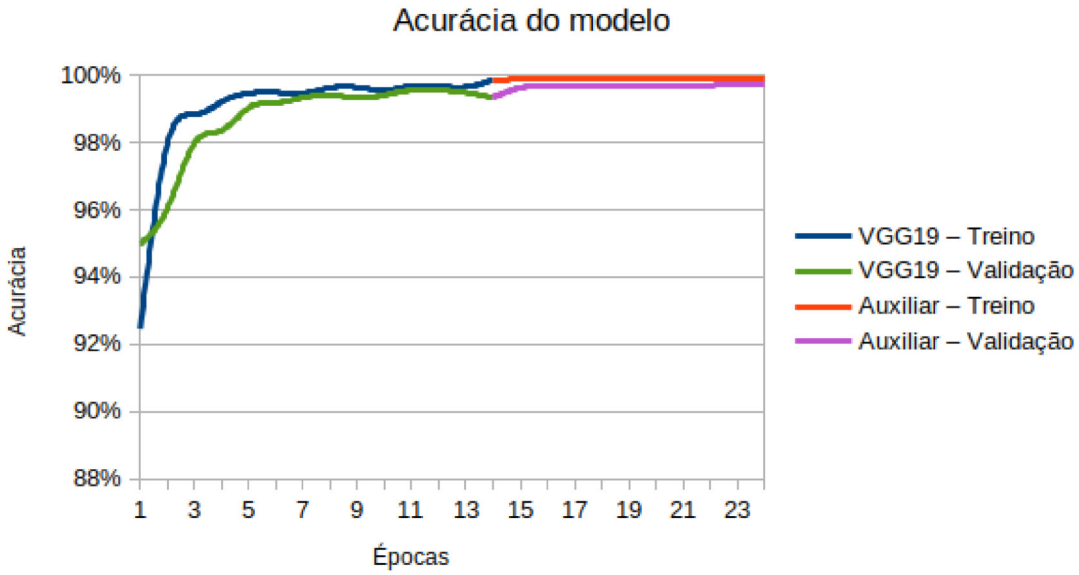
O gráfico que apresenta o ponto de conexão no treinamento das redes pode ser observado na Figura 16a. Logo após a inserção da rede, o treinamento da MLP auxiliar mostrou menor oscilação nas curvas de treino e de validação. A inserção de uma rede auxiliar à VGG19 aumentou sua precisão na fase de generalização, ou seja, a fase em que a rede é testada com imagens inéditas.

O ganho com o uso da rede auxiliar pode ser observado pela comparação dos testes realizados apenas pela VGG19, ilustrado na Figura 15c, com os testes utilizando a rede adicional, conforme a Figura 16a. Ao realizar a comparação, percebe-se que a rede auxiliar aumentou em 11,33% a precisão da rede principal. Entretanto, a inserção de mais redes não melhorou o modelo.

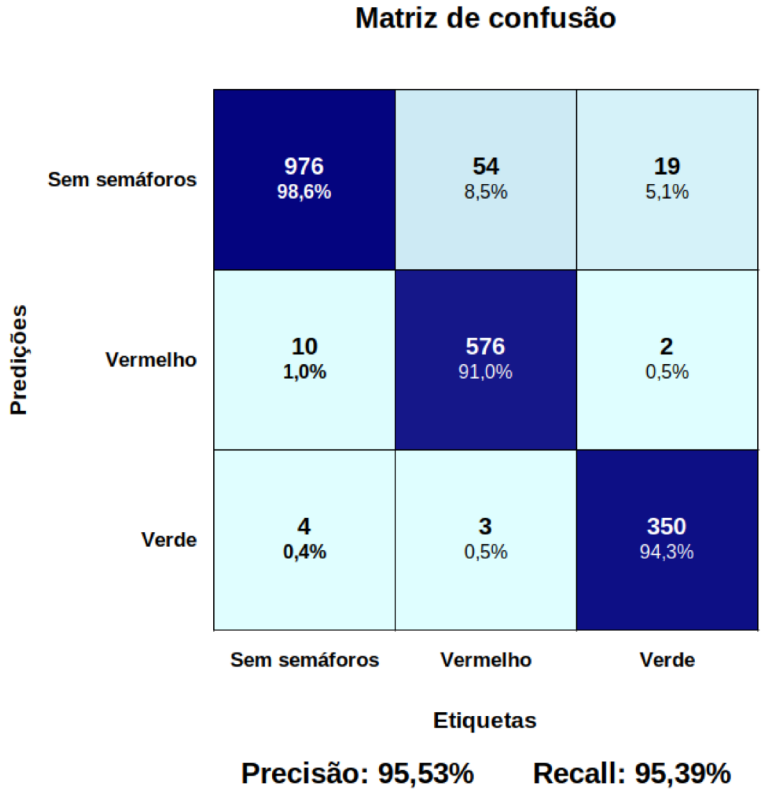
Para entender melhor o comportamento da técnica, foi utilizada a mesma estrutura da rede auxiliar encontrada para o melhor caso, no caso mediano e no pior caso. O cenário

Figura 16 – Rede auxiliar adicionada ao melhor caso.

a) Gráfico do treino do melhor caso.



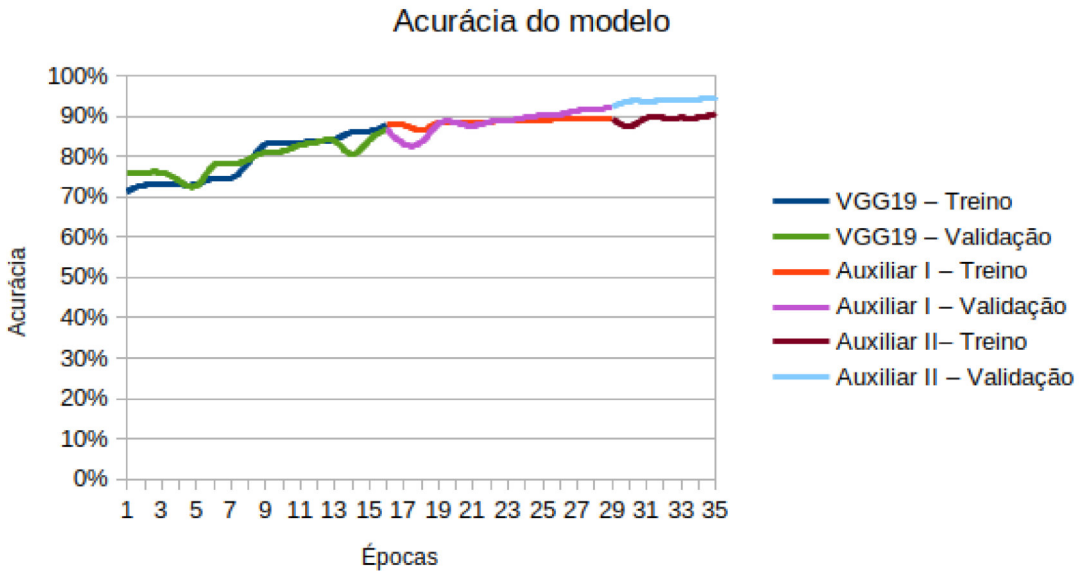
b) Matriz de confusão do melhor caso.



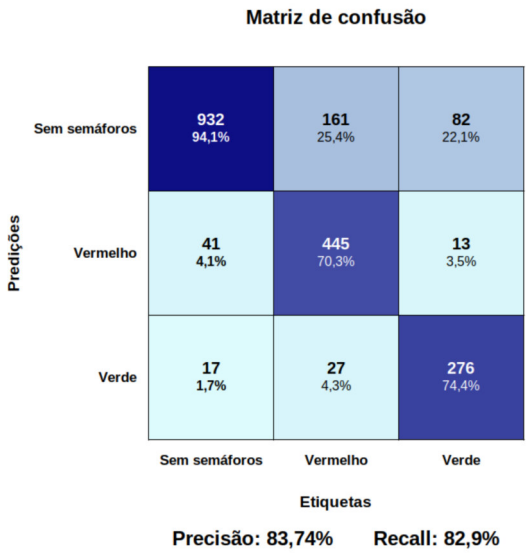
Fonte: Autor.

Figura 17 – Redes auxiliares adicionadas ao caso mediano.

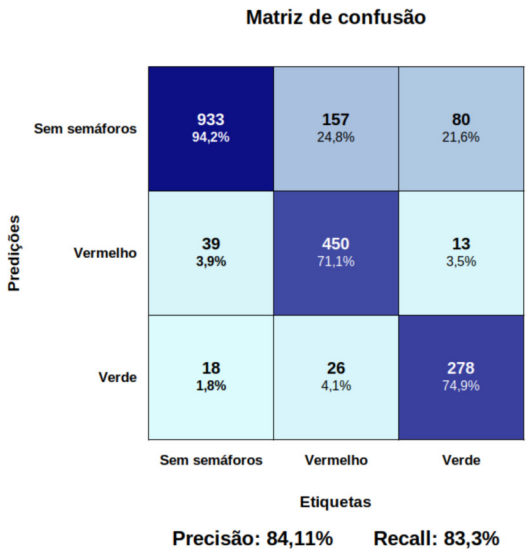
a) Gráfico do treino.



b) Matriz de confusão com uma rede auxiliar.



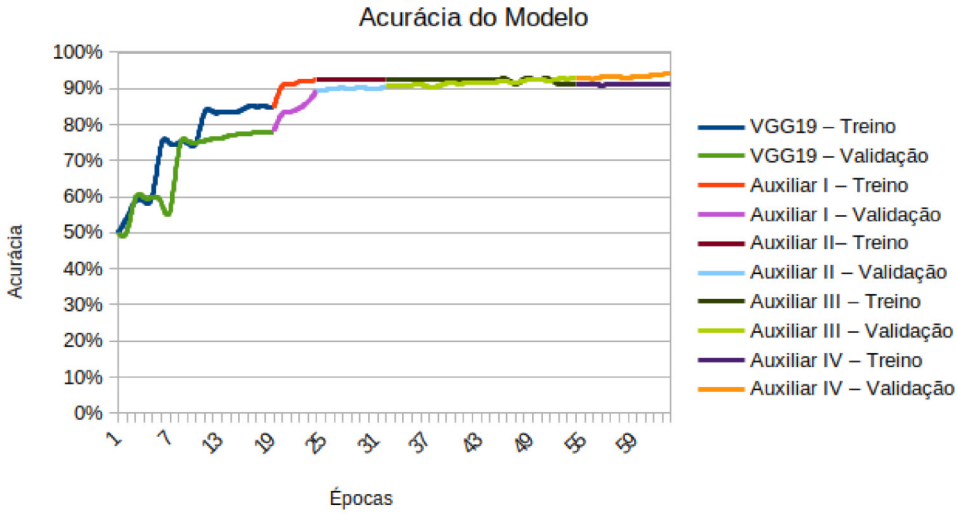
c) Matriz de confusão com duas redes auxiliares.



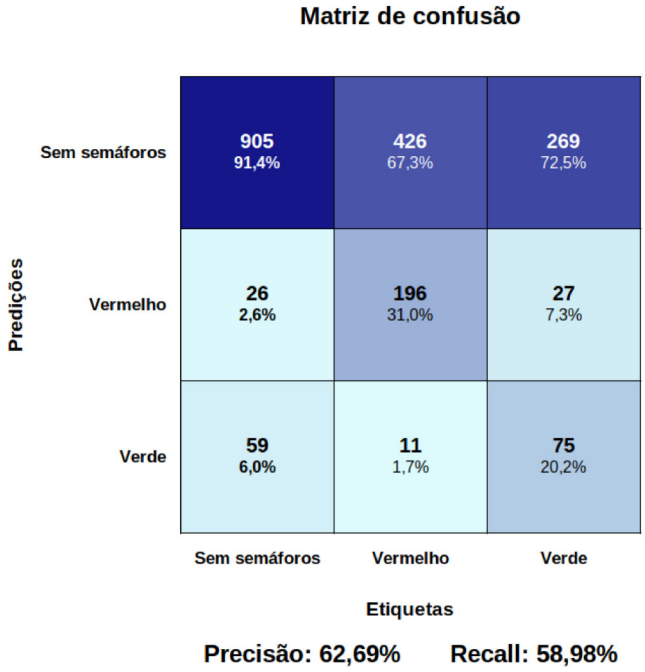
Fonte: Autor.

Figura 18 – Redes auxiliares adicionadas ao pior caso.

a) Gráfico do treino.



b) Matriz de confusão com do modelo final.



c) Resultado das redes auxiliares.

Rede auxiliar	**Precisão**	***Recall***
I	60,05%	56,67%
II	61,73%	58,27%
III	62,3%	58,73%
IV	62,69%	58,98%

Fonte: Autor.

mediano obteve melhora com adição de duas redes auxiliares, conforme pode ser observado na Figura 17. Já no pior caso, foram adicionadas quatro redes auxiliares, conforme a Figura 18, contudo, sem um resultado expressivo no modelo final, quando comparado ao melhor caso. Conclui-se que, nestes exemplos, as redes não conseguiram resultados satisfatórios pela falta de exatidão na extração das características por parte das camadas convolucionais, o que poderia ser solucionado com um retreinamento da CNN utilizada.

O resultado obtido pela utilização das redes auxiliares, em teoria, poderia ser obtido apenas pela VGG19 após ajustes em sua arquitetura e nos hiper-parâmetros. Entretanto, o uso da técnica permitiu um aprendizado acumulativo, o que resultou em um treinamento mais rápido, como pode ser observado na Tabela 4. Isto foi possível, pois somente a rede auxiliar foi treinada, a qual possui uma estrutura bem menor que a rede principal. Esse processo facilitou a descoberta de uma solução considerada a melhor.

Tabela 4 – Tempo médio de treinamento por época.

Fase	**Segundos**
VGG19	$\cong$ 555
Primeira época das redes auxiliares	$\cong$ 134
Demais épocas das redes auxiliares	$\cong$ 29

Fonte: Autor.

4.3 Estratégia de fuga do mínimo local

Além de complementar o treinamento de uma rede profunda, a rede auxiliar também mostrou-se eficaz quando utilizada na fuga de um mínimo local. Diante disso, a Figura 19 ilustra um teste realizado para esse tipo de situação. No teste específico, uma rede VGG19 utilizando como classificador a sequência de camadas apresentada na Tabela 5 foi submetida a um treinamento, o qual é ilustrado como primeiro treino na Figura 19. Após sua finalização, com pesos iniciados aleatoriamente, a rede errava aproximadamente 50% das amostras de teste.

Tabela 5 – Estrutura da MLP utilizada na VGG19.

	Neurônios	**Função de ativação**
Entrada	4096	Relu
Escondida	380	Relu
Saída	3	Linear

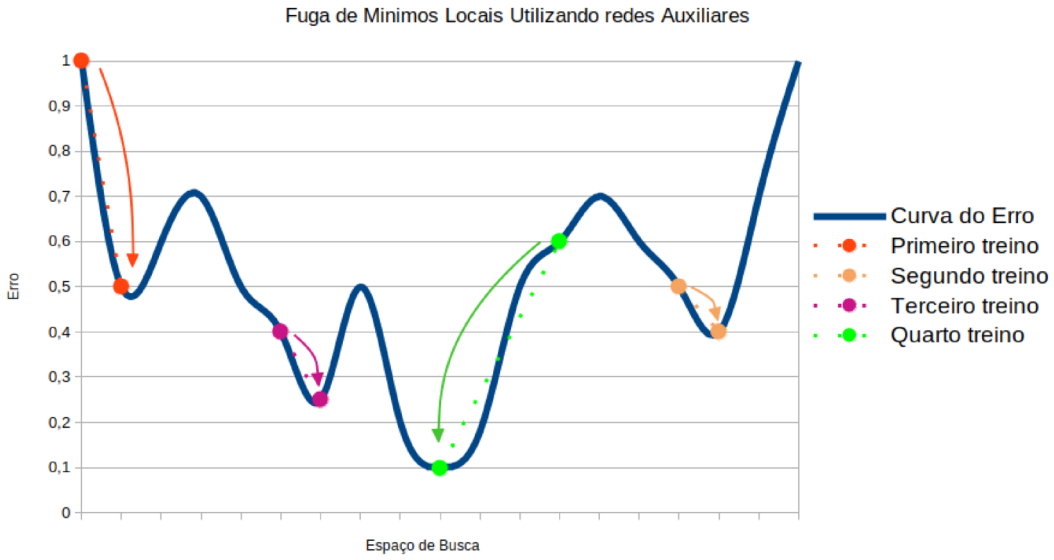
Fonte: Autor.

Em um treinamento tradicional, alterações na estrutura da rede ou em seus hiper-

parâmetros deveriam ser realizadas seguidas de um novo treinamento, dobrando o tempo gasto e desperdiçando o esforço decorrido. Para evitar isso, foi inserida uma nova rede para auxiliar a VGG19.

A rede auxiliar utilizada possuía a mesma estrutura apresentada na Tabela 3, com exceção da função de ativação da camada de saída, que foi alterada para linear. Após a realização do treino da MLP auxiliar, percebeu-se que o modelo foi colocado em outro ponto do problema, porém com o conhecimento adquirido no primeiro treinamento intacto. A rede auxiliar resultou na queda do erro para aproximadamente 40%. O segundo treino está representado pela marcação em amarelo na Figura 19.

Figura 19 – Redes auxiliares utilizadas como estratégia de fuga de mínimos locais.



Fonte: Autor.

Em sequência, a rede principal voltou a ser treinada e a rede auxiliar permaneceu inalterada. Essa fase pode ser observada como terceiro treino na Figura 19. Dessa forma, novamente a rede foi colocada em outro ponto do problema, entretanto sem perder o conhecimento obtido, podendo ajustar suas camadas convolucionais novamente.

Por fim, a rede auxiliar foi retirada elevando o erro acima do patamar inicial, o que ilustra como as redes ficaram interdependentes. Todavia, ao final do quarto treino, que utilizou apenas a rede VGG19, foi possível superar os treinamentos anteriores, indicando que o uso de uma rede auxiliar pode facilitar o ajuste dos pesos e permitir a fuga de um mínimo local.

Por conseguinte, após o quarto treino, algumas redes auxiliares foram inseridas com o intuito de melhorar o desempenho da rede principal, porém sem ganhos expressivos. Apesar desse tipo de simulação ser promissora, esta não foi aprofundada e ainda está em fase inicial de pesquisa.

4.4 Comparação com a literatura

A Tabela 6 apresenta uma comparação dos resultados obtidos neste trabalho em relação aos trabalhos em destaque que utilizaram a mesma base de imagens. Este estudo superou artigos pertinentes como Weber, Wolf e Zöllner (2016) e Haltakov *et al.* (2015). Apesar de não superar o trabalho de Bao *et al.* (2019), os resultados são aproximados e relevantes. Vale ressaltar que Bao *et al.* (2019) utilizaram o histórico das imagens em sua classificação, podendo não obter o mesmo resultado com imagens aleatórias.

Tabela 6 – Comparação do modelo com estudos relacionados em predição na base LaRA.

Método	**Precisão**	***Recall***
Weber, Wolf e Zöllner (2016)	85,6%	90,7%
Haltakov *et al.* (2015)	89,2%	85,13%
Bao *et al.* (2019)	97,90%	95,52%
VGG19	84,2%	84,35%
Método proposto	95,46%	95,44%

Fonte: Autor.

Esta técnica apresenta uma solução que facilmente pode ser generalizada para todos as CNNs dedicadas à classificação, até mesmo para o modelo de Bao *et al.* (2019). Todavia, não apresenta a dependência do histórico de imagens para eliminação de falsos positivos e negativos.

Contudo, este trabalho não tinha como objetivo superar o estado da arte do problema de classificação e reconhecimento de semáforos, mas sim apresentar uma pesquisa que pode contribuir com a evolução dos treinamentos e arquiteturas das CNNs, podendo ser aplicadas em qualquer problema de classificação de imagens.

5 Conclusão

Os acidentes de trânsito vitimam milhares de pessoas todos os anos. Muitos desses acidentes são causados por desrespeito a sinalização. Por se tratar de uma cena que está em constante transformação, a classificação e o reconhecimento de semáforos tornam-se uma difícil tarefa no processo de automação do trânsito. Por isso, este trabalho apresentou uma possível solução para o problema de TLR. Para isso, foi utilizado um novo modelo de treinamento para CNNs baseado em MRNA autocoordenadas.

Como foi apresentado, as CNNs vêm destacando-se em desafios relacionados a reconhecimento de sons e imagens. Logo, muito do seu avanço decorre da incessante busca de melhorias dessas redes. Essas melhorias, geralmente, têm relação com o crescimento da profundidade ou largura da rede. Porém, redes muito profundas exigem *hardwares* especializados, o que acaba elevando o custo dessas aplicações.

A fim de deixar uma contribuição para as pesquisas da área de aprendizado de máquina, este trabalho apresentou um modelo de treinamento acumulativo, que diminuiu o tempo do treinamento, o que por sua vez, acelerou o processo de descoberta de uma solução ótima. Além disso, com a metodologia apresentada, a rede pode ser treinada em *hardwares* de baixo custo.

Para validação do modelo, utilizou-se uma rede VGG19 que, depois de treinada, foi concatenada a uma rede auxiliar. Em síntese, a nova rede aprendeu os conhecimentos ignorados pela primeira rede durante o treinamento. Esta técnica permite que qualquer CNN apossa ter mais de um classificador, que é treinado de maneira autocoordenada e sequencial. Para validar o método, o modelo foi treinado na base de imagens de trânsito LaRA, atingindo $95,46\%$ de precisão no reconhecimento e classificação de semáforos, o que reforçou a eficiência da técnica apresentada.

5.1 Contribuições deste trabalho

Apesar de o foco principal do trabalho não ser exatamente sobre reconhecimento e classificação de semáforos, este contribuiu diretamente com as pesquisas da área. Nesta investigação, mesmo utilizando menos recursos de pré-processamento e técnicas de validação temporal, ainda assim, os resultados obtidos foram próximos do estado da arte. O que mostrou que a técnica tem aplicabilidade.

Os resultados só puderam ser alcançados por conta da nova estrutura de CNNs, com dois ou mais classificadores. Outro ponto muito importante foi a exposição de um novo método de treinamento que, além de diminuir o tempo de busca por uma solução

ótima, utiliza menos recursos de *hardwares*.

Além disso, o modelo de treinamento apresentou uma proposta de armazenamento das predições das redes já treinadas, o que diminuiu o tempo de treinamento por época em aproximadamente 78%, quando comparado ao modelo treinado de forma convencional. Ademais, foi apresentado um estudo em fase inicial sobre o uso da técnica como estratégia de fuga de mínimos locais, onde se obtiveram resultados promissores.

A técnica proposta neste estudo possibilitou um aprendizado acumulativo que, por sua vez, diminuiu o número de treinamentos necessários e de parâmetros calculados, o que resultou em um ganho considerável no tempo de cada treinamento. Vale ressaltar que a técnica não se limita a VGG19, logo podendo facilmente ser replicada a qualquer outra arquitetura de CNN, o que abre uma nova fronteira de estudos.

5.2 Trabalhos futuros

A partir desta pesquisa, é possível visualizar diversas outras aplicações para a técnica desenvolvida. No primeiro momento, pretende-se avaliar a técnica em outros contextos de conhecimento como agricultura e pesquisas médicas. Somado a isso, a técnica será avaliada em competições da área de visão computacional.

Entre as inúmeras evoluções possíveis, destaca-se a utilização da técnica nas camadas convolucionais, almejando o treinamento dessas camadas de maneira sequencial e acumulativa ou, até mesmo, a junção de duas redes CNNs completas.

Outro desdobramento interessante para o futuro desta pesquisa seria a utilização de um algoritmo inteligente para identificar o momento ideal para a inclusão de uma nova rede, como um algoritmo genético, por exemplo.

Como apresentado, a técnica utilizou a biblioteca hyperopt para automatizar a escolha da estrutura das redes auxiliares. Por sua vez, a biblioteca já permite a execução de simulações em paralelo utilizando um *cluster* de computadores. Futuramente, pretende-se explorar essa funcionalidade para paralelizar os treinamentos o que, em teoria, deixaria o modelo ainda mais rápido.

Enfim, este trabalho apresentou uma técnica promissora, que abre um novo e vasto campo de pesquisas e poderá vir a ser uma importante técnica de aprendizado de máquinas.

Referências

AGHDAM, H. H.; HERAVI, E. J. Guide to convolutional neural networks : a practical application to traffic-sign detection and classification. Cham, Switzerland: Springer, 2017. ISBN 978-3-319-57549-0. Citado 4 vezes nas páginas 16, 20, 30 e 32.

ALMEIDA NETO, A. de; GÓES, L. C. S.; NASCIMENTO, C. L. Accumulative learning using multiple ANN for flexible link control. IEEE Transactions on Aerospace and Electronic Systems, v. 46, n. 2, p. 508–524, 2010. ISSN 00189251. Citado 7 vezes nas páginas 17, 20, 21, 27, 28, 36 e 42.

BAO, C.; CHEN, C.; KUI, H.; WANG, X. Safe driving at traffic lights: An image recognition based approach. Proceedings - IEEE International Conference on Mobile Data Management, v. 2019-June, n. Mdm, p. 112–117, 2019. ISSN 15516245. Citado 2 vezes nas páginas 18 e 55.

BEHRENDT, K.; NOVAK, L.; BOTROS, R. A deep learning approach to traffic lights: Detection, tracking, and classification. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). [S.l.: s.n.], 2017. p. 1370–1377. Citado 2 vezes nas páginas 19 e 20.

BERGSTRA, J.; DESJARDINS, G.; LAMBLIN, P.; BENGIO, Y. 03 - Quadratic Polynomials Learn Better Image Features. Montreal, p. 1–11, 2009. Citado na página 33.

BRAGA, A. de P.; CARVALHO, A. C. P. de Leon Ferreira de; LUDERMIR, T. B. Redes Neurais Artificiais Teoria e Aplicações. 2. ed. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editara, 2007. Citado 2 vezes nas páginas 23 e 25.

CHARETTE, R. de; NASHASHIBI, F. Real time visual traffic lights recognition based on Spot Light Detection and adaptive traffic lights templates. In: 2009 IEEE Intelligent Vehicles Symposium. IEEE, 2009. v. 1, n. March, p. 358–363. ISBN 978-1-4244-3503-6. Disponível em: <http://ieeexplore.ieee.org/document/5164304/>. Citado na página 37.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; Kai Li; Li Fei-Fei. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, p. 248–255, 2010. Citado 3 vezes nas páginas 19, 34 e 40.

DOMINGOS, P. O Algoritmo Mestre. São Paulo: Novatec Editora Ltda, 2017. Citado 4 vezes nas páginas 15, 22, 23 e 24.

FAIRFIELD, N.; URMSON, C. Traffic light mapping and detection. Proceedings - IEEE International Conference on Robotics and Automation, IEEE, p. 5421–5426, 2011. ISSN 10504729. Citado na página 18.

FERNANDEZ, C.; GUINDEL, C.; SALSCHEIDER, N. O.; STILLER, C. A Deep Analysis of the Existing Datasets for Traffic Light State Recognition. IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC, IEEE, v. 2018-November, p. 248–254, 2018. Citado 2 vezes nas páginas 18 e 19.

GOODFELLOW, I. J.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; BENGIO, Y. Generative adversarial nets. Advances in Neural Information Processing Systems, v. 3, n. January, p. 2672–2680, 2014. ISSN 10495258. Citado na página 20.

HALTAKOV, V.; MAYR, J.; UNGER, C.; ILIC, S. Semantic Segmentation Based Traffic Light Detection at Day and at Night. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 9358, p. v–vi, 2015. ISSN 16113349. Citado 2 vezes nas páginas 18 e 55.

HAYKIN, S. Neural networks: a comprehensive foundation. 2 ed. ed. New York: Prentice Hall, 1999. 938 p. Citado 2 vezes nas páginas 25 e 26.

HUANG, G.; LIU, Z.; Van Der Maaten, L.; WEINBERGER, K. Q. Densely connected convolutional networks. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, v. 2017-January, p. 2261–2269, 2017. Citado 2 vezes nas páginas 15 e 36.

IIHS, I. I. for H. S. Red light running. 2020. <https://www.iihs.org/iihs/topics/t/red-light-running/qanda>. Acessado em: 21 mai. 2020. Citado na página 14.

JENSEN, M. B.; NASROLLAHI, K.; MOESLUND, T. B. Evaluating State-of-the-Art Object Detector on Challenging Traffic Light Data. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). [S.l.: s.n.], 2017. p. 882–888. ISSN 2160-7516. Citado na página 19.

JENSEN, M. B.; PHILIPSEN, M. P.; MØGELMOSE, A.; MOESLUND, T. B.; TRIVEDI, M. M. Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives. IEEE Transactions on Intelligent Transportation Systems, v. 17, n. 7, p. 1800–1815, 2016. ISSN 1524-9050. Citado 2 vezes nas páginas 19 e 44.

KANG, K.; WANG, X. Fully Convolutional Neural Networks for Crowd Segmentation. 2014. Disponível em: <http://arxiv.org/abs/1411.4464>. Citado na página 30.

KIM, P. MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence. New York, NY: Apress, 2017. Citado 2 vezes nas páginas 30 e 31.

KLAMBAUER, G.; UNTERTHINER, T.; MAYR, A.; HOCHREITER, S. Self-normalizing neural networks. Advances in Neural Information Processing Systems, v. 2017-December, p. 972–981, 2017. ISSN 10495258. Citado na página 32.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. [S.l.: s.n.], 2012. p. 1097–1105. Citado na página 32.

LI, X.; MA, H.; WANG, X.; ZHANG, X. Traffic Light Recognition for Complex Scene With Fusion Detections. IEEE Transactions on Intelligent Transportation Systems, v. 19, n. 1, p. 199–208, jan 2018. ISSN 1524-9050. Citado na página 18.

LINDNER, F.; KRESSEL, U.; KAELBERER, S. Robust recognition of traffic signals. IEEE Intelligent Vehicles Symposium, Proceedings, p. 49–53, 2004. Citado na página 18.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, v. 5, n. 4, p. 115–133, dec 1943. Disponível em: <http://arxiv.org/abs/1805.08936http://link.springer.com/10.1007/BF02478259>. Citado 3 vezes nas páginas 22, 23 e 24.

MICHAEL, M.; SCHLIPSING, M. Extending traffic light recognition: Efficient classification of phase and pictogram. In: 2015 International Joint Conference on Neural Networks (IJCNN). [S.l.: s.n.], 2015. p. 1–8. ISSN 2161-4407. Citado na página 18.

Ministério da Saúde. Homens são os que mais morrem de acidentes no trânsito. 2019. <https://www.saude.gov.br/noticias/agencia-saude/45466-homens-sao-maiores-vitimas-de-acidentes-no-transito>. Acessado em: 10 fev. 2020. Citado na página 14.

MITCHELL, T. M. Machine Learning. 1. ed. [S.l.]: McGraw-Hill, 1997. (McGraw-Hill series in computer science). ISBN 9780070428072,0070428077. Citado na página 22.

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. Activation Functions: Comparison of trends in Practice and Research for Deep Learning. p. 1–20, 2018. Disponível em: <http://arxiv.org/abs/1811.03378>. Citado na página 33.

OCHOA, A.; OLIVA, D. Smart traffic management to support people with color blindness in a Smart City. 2018 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2018, IEEE, 2019. Citado na página 14.

OLIVEIRA, M. B. W. d.; ALMEIDA NETO, A. d. Optimization of traffic lights timing based on multiple neural networks. Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI, IEEE, p. 825–832, 2013. ISSN 10823409. Citado 3 vezes nas páginas 17, 21 e 36.

OMS, O. Death on the road. 2018. <https://extranet.who.int/roadsafety/death-on-the-roads/#ticker//all>. Acessado em: 21 mai. 2020. Citado na página 14.

OUYANG, Z.; NIU, J.; LIU, Y.; GUIZANI, M. Deep CNN-Based real-time traffic light detector for self-driving vehicles. IEEE Transactions on Mobile Computing, IEEE, v. 19, n. 2, p. 300–313, 2020. ISSN 15580660. Citado na página 18.

PON, A. D.; ADRIENKO, O.; HARAKEH, A.; WASLANDER, S. L. A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection. Proceedings - 2018 15th Conference on Computer and Robot Vision, CRV 2018, p. 102–109, 2018. Citado na página 19.

ROSENBLATT, F. The Perceptron - A Perceiving and Recognizing Automaton. 1957. 460–1 p. Citado na página 24.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. Nature, v. 323, n. 6088, p. 533–536, oct 1986. ISSN 0028-0836. Disponível em: <http://www.nature.com/articles/323533a0>. Citado na página 25.

SERMANET, P.; EIGEN, D.; ZHANG, X.; MATHIEU, M.; FERGUS, R.; LECUN, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings, 2014. Citado na página 15.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, p. 1–14, 2015. Citado 4 vezes nas páginas 15, 20, 33 e 36.

SIOGKAS, G.; SKODRAS, E.; DERMATAS, E. Traffic lights detection in adverse conditions using color, symmetry and spatiotemporal information. VISAPP 2012 - Proceedings of the International Conference on Computer Vision Theory and Applications, v. 1, p. 620–627, 2012. Citado na página 18.

SOARES, J. C. da S.; BORCHARTT, T. B.; PAIVA, A. C. de; ALMEIDA NETO, A. de. Methodology Based on Texture, Color and Shape Features for Traffic Light Detection and Recognition. Proceedings of the International Joint Conference on Neural Networks, v. 2018-July, 2018. Citado na página 18.

TEIXEIRA, A. P.; ALMEIDA NETO, A. de. Multiple Neural Networks Using MLP and RBF Networks 2 . Artificial Neural Networks. v. 11, p. 1–9, 2016. Citado 4 vezes nas páginas 17, 21, 28 e 36.

WEBER, M.; WOLF, P.; ZÖLLNER, J. M. DeepTLR: A single deep convolutional network for detection and classification of traffic lights. In: 2016 IEEE Intelligent Vehicles Symposium (IV). [S.l.: s.n.], 2016. p. 342–348. Citado 2 vezes nas páginas 19 e 55.

XIE, S.; GIRSHICK, R.; DOLLÁR, P.; TU, Z.; HE, K. Aggregated residual transformations for deep neural networks. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, v. 2017-January, p. 5987–5995, 2017. Citado 3 vezes nas páginas 15, 20 e 36.

ZHENG, Y.; YANG, C.; MERKULOV, A. Breast cancer screening using convolutional neural network and follow-up digital mammography. In: MAHALANOBIS, A.; ASHOK, A.; TIAN, L.; PETRUCCELLI, J. C. (Ed.). Computational Imaging III. SPIE, 2018. v. 10669, p. 1–13. Disponível em: <https://doi.org/10.1117/12.2304564>. Citado na página 34.

Apêndices

Optimizing CNN's using Cumulative Learning with Auxiliary Networks for Traffic Light Recognition

Joaquim Martins Scavone[1,2], Jonnison Lima Ferreira[1], Emerson Rogério Alves Barea[2], Geraldo Braz Júnior[1], Areolino de Almedia Neto[1]
[1]Federal University of Maranhão, Maranhão, Brazil
[2]Federal Institute of Tocantins, Tocantins, Brazil
joaquim.scavone@ifto.edu.br

***Abstract*—The number of fatal victims due to traffic accidents is frightening. Neural networks are already used to try to minimize this problem, however there is still an open task training those networks with maximum effectiveness. This work proposes an extension of the concept of Multiple Self-coordinated Neural Networks applied to convolutional neural networks in order to recognize traffic lights. The idea is to present a method that performs self-improvements in the network and avoid local minimums in new training. The proposed method can contribute by reducing the time spent on training in choosing hyperparameters. The preliminary results were promising, reaching 95.33% of accuracy and opening the range of new research that can be carried out in this area.**

***Keywords*—Traffic Lights Recognition, Deep Learning, Multiple Self-coordinated Neural Networks**

I. Introduction

In 2017, more than 182,838 people got involved in traffic accidents in Brazil, and of this total, more than 35 thousand died [1]. Likewise, in 2016, disrespect for the red light killed more than 810 people in the United States [2]. Many traffic accidents are caused by driving errors, which is one of the reasons that management systems gained attention from the scientific community in the last decade.

Motivated by the high number of accidents, researchers are developing works that assist in monitoring, surveillance, and even in the automated driving of vehicles. Projects focused on driving safety are called Advanced Driver Assistance Systems (ADAS) and are increasingly popular among researchers, and their results are being used in new car models.

ADAS require high accuracy and is very sensitive to errors. An error can cost a life, so there is an effort by the scientific community to create robust systems that have an error close to or equal to zero.

Over time, new technologies have contributed to ADAS evolution, as more recently, the techniques related to Artificial Intelligence (AI).

Currently, Artificial Neural Networks (ANNs) stands out on AI researches. There are several ANN architectures, including Deep Convolutional Neural Networks (CNN), which have convolutive layers frequently used to extract characteristics from images and objects to recognition [3].

Despite its efficiency, CNNs are deep networks with multiple layers. This condition makes the training phase a lengthy process that is performed manually. An example of deep CCN is VGG19, which has 19 layers. The choice of CNNs training hyperparameters considers the number of layers, their sequence, activation functions, learning rate, among other things. After defining all these parameters, the network is then trained, and the whole process must start over again if an acceptable solution was not found.

978-1-7281-7539-3/20/$31.00 ©2020 IEEE

The works on deep learning have accelerated the research on autonomous vehicles in the last decade [4]–[6]. The changes in the convolutions layers [8], [9], and the increase in the depth of networks [10] have contributed to the CCNs accuracy improvement, but still do not support the necessary effectiveness in ADAS.

An important characteristic of the CCNs is the need to change the entire structure, or the hyperparameter used if the network does not obtain the expected results after training. Each time this process occurs, the entire network must be retrained, practically discarding any effort made in previous processing. Seeking to minimize this behavior [11] presented a technique called Multiple Artificial Neural Networks (MANN), which allows a problem to be attacked as a top by several networks, through a peaceful collaboration, dispensing with the use of a coordinator, and can be applied in already trained networks. This technique presents some successful examples [11]–[13].

This work presents an evolution to the MANN technique applied to a VGG19 CNN, adding auxiliary networks to improve the recognition and classification of traffic lights in an already-trained network, introducing the concept of cumulative training to convolutional networks. Among the highlights and most relevant aspects, this work *(i)* evaluate the effectiveness of the proposed model using a convolutional network with images taken from a traffic lights public base; *(ii)* compares the network accuracy with other well-known models; and *(iii)* evaluate the effectiveness of using auxiliary networks as a strategy to escape local minimums during training.

The rest of this work is organized as follows: the section II presents the related works to the theme of our work; the III section details the proposed methodology; section IV carried out the experiments and show and discusses the results; while the section V presents the conclusions and points to proposals for future work.

II. Related Work

There is a vastly number of works that deal with traffic lights recognition and classification. Some of them only combines the image processing techniques [14] [15] [16]. Others uses semantic segmentation to ROI recognition, just after they are confirmed and classified by geometric and color classifiers, combined with a time detection system [25]. In [15] the work is extended and incorporates a color preprocessing module to enhance the red and green regions in the image, and a rapid transformation of radial symmetry is used for the detection of traffic light candidates reaching false positive results reduction using the spatio-temporal persistence check. In [19] a set of techniques was used to detect traffic lights, such as Fuzzy, Bulb and Multi Size Detection. Already [20] combines the analysis of linear discriminant (LDA), k-nearest neighbors (kNN) and support vector machine (SVM).

In recent years, deep learning approaches have surpassed state-of-the-art algorithms in a wide variety of problems [6], consequently the appearance of techniques that use neural networks for the recognition and classification of semaphores in the literature did not take long. Some techniques maintained the initial structure of image processing, but used neural networks with few layers to classify the characteristics taken from preprocessing. In [17], image processing techniques were used to identify ROI's and extract their characteristics, right after the data was sent to a Multilayer Perceptron (MLP) to classify them. A heuristic module for selecting the candidate region [18] was used to identify all possible traffic lights, and right after a classifier formed by a small Convolutional Neural Network (CNN) to classify ROIs. Also, [26] presents an image recognition system combined with a CNN, which besides the traffic light classification can also identify the countdown timer, and the distance from the traffic light vehicle, thus being able to propose a decision strategy for the driver.

Although the use of several techniques has proven to be effective, many researches have preferred to use only one CNN to accomplish this task, such as [6] which presents a CNN capable of recognizing traffic lights with an excellent performance, however the research does not address the state classification traffic light. In [21] was demonstrated the performance of the YOLOv2 network in traffic light classification. Also, [22] proposes DeepTLR, a deep network specific for traffic light detection and classification based on Alexnet. And, in [5] was proposed a model capable of detecting traffic lights and traffic signs simultaneously, showing the potential of neural networks to solve such problems was presented.

CNN's efficiency in classifying and recognizing traffic lights has been demonstrated in each new study, but there is space for improvements. For this reason, we propose a hybrid concept of extend [11] MANN applied to a convolutional network.

The MANN concept provides for the use of more than one network to tackle the same problem. In [23], it is also proposed to use more than one network, the generator and the classifier, to solve a problem. However, it differs with only one network doing classification task. MANN proposes networks to work in harmony applying the principles of cooperation. The network called "Multi-task" can be compared to MANN, however its distinction is in the training method, since Multi-task trains the entire network at once, just dividing the classification into different micro-networks. MANN trains one network at a time, keeping all the others frozen in the meantime. Another relevant difference between the architectures is the fact that MANN does not fragment the problem to solve it, all networks attack the problem as a whole.

III. Methodology

In this section, we present the proposed methodology. We propose enhance a base pre-trained network, VGG19 using a coordinated process of network addition to the original, with the purpose of continuous increment of performance, named Auxiliary Networks.

For a better understanding of our proposal, we divided this methodology into stages, each with well-defined functions. Figure 1 presents the entire process, called here as $Simulation$, that corresponds to the execution of the whole process, consisting of the $Preprocessing$, $Training$, and $Evaluation$ stage, assisted by the use of $Auxiliary\ Networks$.

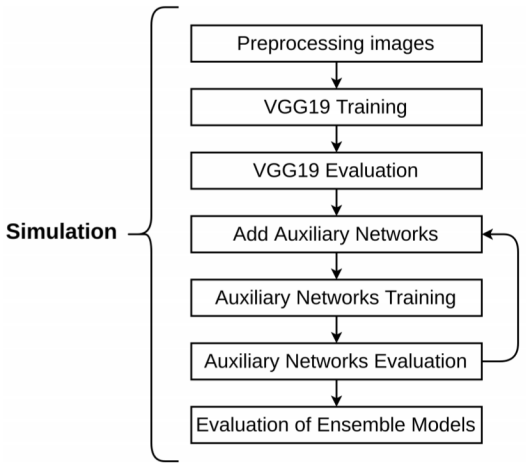


Fig. 1. Network training workflow.

To validate this model, we use images taken from LaRA base [27], and carry out some simulation experiments. This database was chosen because it is in the public domain, has several references in the literature, and contains low-resolution images (640 x 480 pixels), enabling the execution of training with a moderate computational cost.

A. Preprocessing

First step is preprocessing where we reduced the resolution of the images taken from LaRA base to 224 x 224 pixels, with 3 channels (RGB), allowing the use of pre-trained weights taken from the ImageNet challenge in some simulations. In addition to image transformations, CNN received blocks of samples, optimizing the use of available hardware. As the basis of the images is a video broken into frames, the blocks sought to prevent images corresponding to the same period from

participating in the training, validation, or tests simultaneously, avoiding the overlapping of images in the same scenario at different stages of the simulation.

For image selection, we discarded images with tagging outside the context of the desired classification (tags: background, red, yellow, green, off), regardless of the base used. In the LaRA database, the only unwanted class was ambiguous, which corresponds to images that are not related to traffic lights; and yellow, because they contain a smaller number of samples than other classes, with little variation in the context of the image, making it difficult to separate between training, validation, and testing. Besides, the class yellow is not presented in other works, making it impossible to validate the results.

We did not use other preprocessing techniques. In general, preprocessing techniques depend heavily on the dataset used, considering that one of the objectives of the MANN is to minimize the number of training sessions and the time spent in choosing training parameters [11].

B. Training and Evaluation

To carry out the training and evaluation stages, the VGG19 network was changed to support the traffic light classification requirements using the LaRA base. For this, the VGG19 MLP layer, which initially produced 1000 classes in its output, was modified to support only 3 classes (background, red and green).

After that, the first phase of training was responsible for choosing a sequence of layers for fully connected that would obtain a relevant result, and then on making improvements in the already-trained network.

To allow composition of fully connected network attach in the CNN vector layer, we created a connection structure that allows both fully connected and auxiliary share the same input. The sum of the output of the leading network and all its auxiliaries produces the output of the set of networks (ensemble models), as illustrated in Figure 2 and represented by the 1.

$$y = \sum_{i=1}^{n} y_i \tag{1}$$

C. Enhancing using Auxiliary Networks

Auxiliary networks are MLP networks, connected in the vectorization layer of the convolutional outputs leading network. The idea is that the auxiliary network learns characteristics neglected by CNN's native classifier, improving its performance. Figure 2 illustrates how the model works together, seeking this improvement.

The main principle is to add new train unit to the previous network with the intent of loss reduction. To add and train a new auxiliary network, an adaptation of the expected output d was used for each specific network, as demonstrated in 2, where d_k is the difference between the output from the original markup of the d images, for the output of all networks, except the network in training. In that case, the new network would have to learn what the previous networks were not capable of learning before. The final output is then composed by the sum of all other outputs. During the testing phase, the label is defined by the neuron with the highest activation value.

$$d_k = d - \sum_{i=1}^{n} y_i - y_k \tag{2}$$

where:
n = all networks
k = training network

Also, in order to help identification of hyperparameters, we used the Hyperopt library to test various auxiliary network structures, evaluating which structure would be most beneficial to the overall ensemble model, taking into account the accuracy of the testing phase. After defining the structure of the networks, the training was carried out using the LaRA base, and finally, the architecture was evaluated with images that were not present during the training and validation phases.

IV. Experimental Results and Discussion

Preprocessing the images from the LaRA dataset resulted in 9976 images used in this experiment. The images were grouped into training, validation, and tests. The Table I presents the grouping of images concerning the simulation phase and their classifications.

TABLE I
Composition of the LaRA base and amount of images used in each stage

Label	Training	Validation	Testing	Total
Background	3565	396	990	4951
Red	2280	253	633	3166
Green	1340	148	371	1859
Total	7185	797	1994	9976

The CNN VGG19 model available on Keras has an MLP composed of 1000 classes on the ImageNet dataset; however, the number of classes desired in this experiment is only 3. In the LaRA dataset, the native model of this CNN converges to errors close to 0 quickly, but it has a poor generalization for new images, indicating an overfit.

To attempt an approximation between training and testing, we applied several changes to the structure of the fully connected layers (MLP) and their hyperparameters. During the process of choosing the best network structure, taking into account the accuracy in the testing phase, the optimizers, learning rate were changed, and dropout was used between the MLP layers in some tests. Finally, a reduction in the number of neurons in the second layer of the MLP showed a good result and was considered satisfactory for the first stage of the experiment.

Using the Stochastic Gradient Descent (SGD)[1] optimizer, with a learning rate of 0.01, maintaining its original convolutional layers, and pre-trained weights taken from the ImageNet challenge, VGG19 had its MLP changed to the model presented in Table II, which shows a three-layer MLP: input, hidden and

[1]https://keras.io/optimizers/

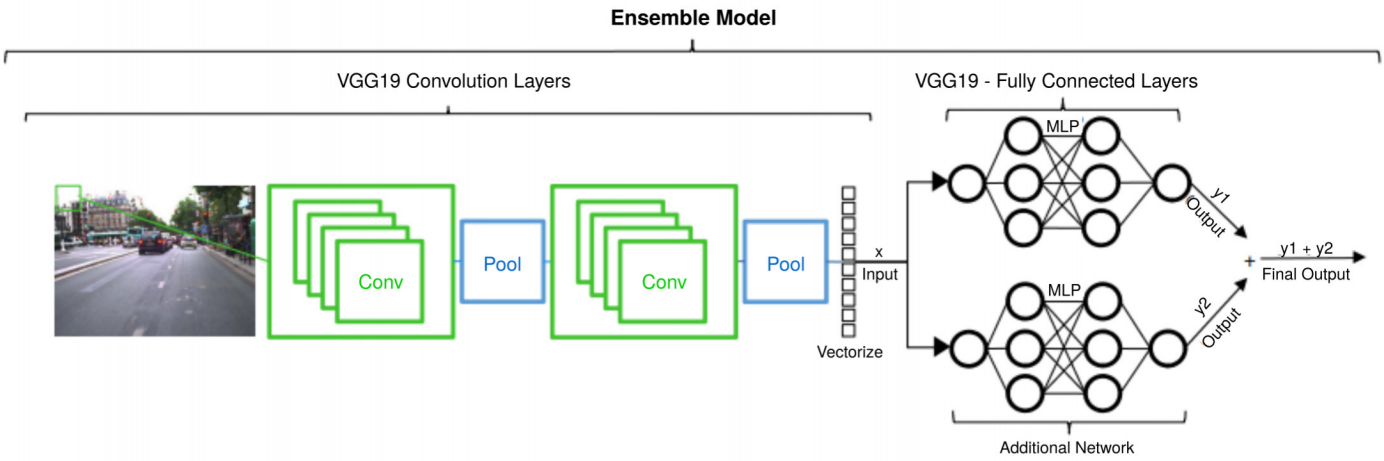


Fig. 2. Example of a CNN with auxiliary network.

output; the respective numbers of neurons and the activation functions[2] used. This structure achieved the best result, corresponding to 84.2% accuracy in the testing phase.

TABLE II
VGG19 FULLY CONNECTED LAYER STRUCTURE.

Layer	Neurons	Activation Function
Input	4096	Relu
Hidden	300	Relu
Output	3	SoftMax

As VGG19 representing a production network already trained, the next step was the insertion of auxiliary networks. As several hyperparameters can be modified during the training of a neural network, the Hyperopt library was used to automate the search for the auxiliary network structure that best contributed to the leading network. The search space took into account the activation and optimizer functions implemented in Keras; the learning rate, which ranged from 0.001 to 0.05; and the number of neurons in each layer, which varied between four and five thousand. This procedure was applied to networks of 2 to 6 layers.

After 425 simulations of unitary auxiliary network structures, we observed that the network with three layers obtained the best result, Table III. This network used the SGD optimizer and a learning rate of 0.005. In this case, the auxiliary network increased the accuracy of the leading network by 11.33% in the testing phase.

TABLE III
AUXILIARY NETWORK STRUCTURE.

Layer	Neurons	Activation Function
Input	2169	Relu
Hidden	1354	SoftSign
Output	3	Selu

Figure 3 shows the connection point in network training. After connecting the networks, the accuracy of the ensemble model improves considerably. The auxiliary network training demonstrates less fluctuation in accuracy and validation, while the insertion of an auxiliary network to VGG19 increased its accuracy in the generalization phase, that is the test phase with new images.

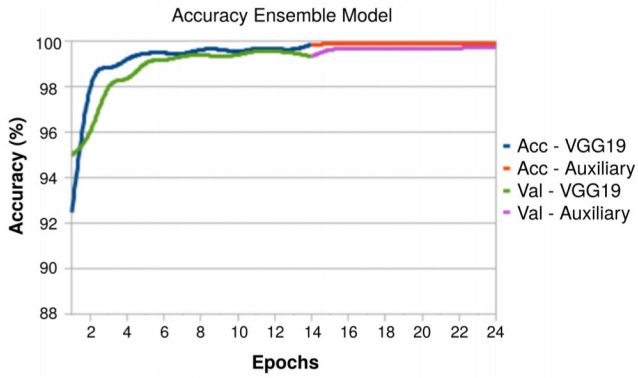


Fig. 3. Accuracy of training using auxiliary networks.

After the phase of discovering the best structure, other networks were added to the leading network. However, the insertion of these new networks did not improve the generalization. This fact occurs by the high accuracy in the training phase, which leaves little opportunity for adjustments.

On other bases, or with a more significant number of samples, the use of more than one auxiliary network may be necessary. The result obtained by concatenating the networks, in theory, could be obtained only by VGG19 after adjustments in its architecture and hyperparameters. However, the use of auxiliary networks allows for cumulative learning and faster training, since only the auxiliary network that has a much smaller structure, is trained. This method can facilitate the discovery of an optimal solution.

Besides complementing the training of a deep network, the auxiliary network also proved useful when used to escape from a local minimum, the image 4 illustrates a test performed for

[2]https://keras.io/activations/

this situation. In this test, a VGG19 structure similar to the one previously, however, with linear output, was trained with randomly started weights (stage 1), resulting in an approximately of 50% sample test error.

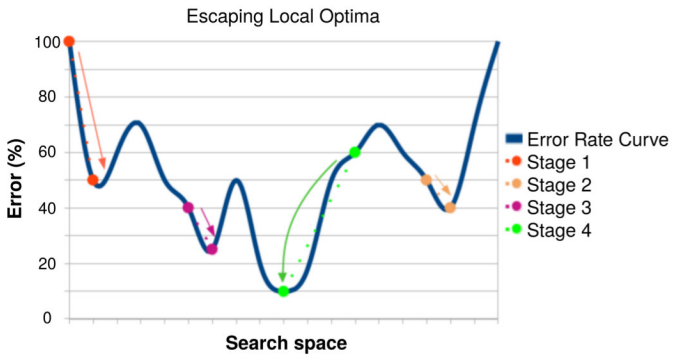


Fig. 4. Search of minimum global using auxiliary networks.

In a traditional training condition, changes should be made to the network structure or its hyperparameters, followed by new training, doubling the time spent, and wasting the effort of previous processing. Using a new network to assist VGG19, we have additional training with a much lower computational cost, considering that the auxiliary network has a more straightforward structure and requires fewer calculations for its training.

After adding an auxiliary network (stage 2), the network goes to at another point of the problem; however, as the knowledge acquired in stage 1 was intact, the error rate dropped to approximately 40% after training the auxiliary network. In sequence, the leading network (VGG19) was retrained, and the auxiliary network remained unmodified (stage 3). Over again, the network goes to another point of the problem, without losing the knowledge obtained, helping its convolutive layers.

Finally, the auxiliary network was removed, raising the error above the initial level, demonstrating how the networks became interdependent. However, at the end of the last training session (stage 4), VGG19 exceeded previous training steps, demonstrating that the use of an auxiliary network can assist in weight adjustments, allowing the escape of a local minimum.

A. *Discussion*

Table IV presents a comparison of the results of the proposed methodology with other relevant works that use the same dataset. The results obtained in this work overcome [22] and [25]. Despite not surpassing the [26], the results are approximate and relevant, as it is worth mentioning that it uses the history of the images and their classification, and may not obtain the same result with random images.

Our technique presents a solution that can easily be generalized to all CNNs dedicated to classification, such as in the CCN used by [26] technique itself. However, it is not limited to works that depend on image history.

However, this work did not aim to overcome the state of the art of the traffic light classification and recognition problem; however, it intends to contribute to the evolution of CNNs training, can be applied to any image classification problem.

TABLE IV
RESULTS OF TRAFFIC LIGHT DETECTION

Method	**Dataset**	**Accuracy(%)**	**Recall(%)**
[22]	LaRA	85.60	90.70
[25]	LaRA	89.20	85.13
[26]	LaRA	97.90	95.52
[10]	LaRA	84.20	87.16
Proposed Method	LaRA	95.53	95.49

V. CONCLUSIONS AND FUTURE WORKS

This work focused on the problem of recognition and classification of traffic lights, proposing a methodology that presents an adaptation of the MAAN technique for CNNs, proposing new architectures and training techniques that can speed up the process of discovering an optimal solution.

Although the accuracy of the model is not the focus of this work, our solution reached a result close to state of the art, being able to integrate an ADSS system for traffic lights recognition.

As future works, we propose a study that evaluate techniques for better use of memory resources, in addition to the application of the methodology proposed herein other databases, validating its applicability in other contexts.

REFERENCES

[1] Ministério da Saúde, "Homens são os que mais morrem de acidentes no trânsito", 2019, ¡https://www.saude.gov.br/noticias/agencia-saude/45466-homens-sao-maiores-vitimas-de-acidentes-no-transito¿, accessed in 2020-02-10.
[2] The Insurance Institute for Highway Safety (IIHS). Red light running, 2017.
[3] P. Sermanet, D. Eigen, X. Zhang, M, Mathieu, R. Fergus, Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," arXiv e-prints, 2013.
[4] Z. Ouyang, J. Niu, Y. Liu and M. Guizani, "Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles," in IEEE Transactions on Mobile Computing, vol. 19, no. 2, pp. 300–313, 1 Feb. 2020.
[5] A. Pon, O. Adrienko, A. Harakeh and S. L. Waslander, "A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection," 2018 15th Conference on Computer and Robot Vision (CRV), Toronto, ON, 2018, pp. 102–109.
[6] C. Fernández, C. Guindel, N. Salscheider and C. Stiller, "A deep analysis of the existing datasets for traffic Light state recognition," 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 248–254.
[7] J. Deng, W. Dong, R. Socher, L. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp.248–255.
[8] G. Huang, Z. Liu, L. v. d. Maaten and K. Q. Weinberger, "Densely connected convolutional networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 2261-2269.
[9] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, "Aggregated residual transformations for deep neural networks," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 5987-5995.
[10] K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.
[11] A. D. Almeida Neto, L. C. S. Goes and C. L. Nascimento, "Accumulative learning using multiple ANN for flexible link control," in IEEE Transactions on Aerospace and Electronic Systems, vol. 46, no. 2, pp. 508-524, April 2010.

[12] M. B. W. D. Oliveira and A. D. Almeida Neto, "Optimization of Traffic Lights Timing Based on Multiple Neural Networks," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, 2013, pp. 825-832.

[13] A. P. Teixeira and A. D. Almeida Neto, "Multiple neural networks using MLP and RBF networks", Journal of Convergence Information Technology(JCIT), 2016.

[14] F. Lindner, U. Kressel and S. Kaelberer, "Robust recognition of traffic signals," IEEE Intelligent Vehicles Symposium, 2004, Parma, Italy, 2004, pp. 49–53.

[15] G. Siogkas, E. Skodras, and E. Dermatas, "Traffic lights detection in ad-verse conditions using color, symmetry and spatiotemporal information,"inVISAPP, 2012.

[16] N. Fairfield and C. Urmson, "Traffic light mapping and detection," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 5421–5426.

[17] J. C. da Silva Soares, T. B. Borchartt, A. C. de Paiva and A. de Almeida Neto, "Methodology based on texture, color and shape features For traffic light detection and recognition.," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, 2018, pp. 1–7.

[18] Z. Ouyang, J. Niu, Y. Liu and M. Guizani, "Deep CNN-based real-time traffic light detector for self-driving vehicles," in IEEE Transactions on Mobile Computing, vol. 19, no. 2, pp. 300–313, 1 Feb. 2020.

[19] X. Li, H. Ma, X. Wang and X. Zhang, "Traffic light recognition for complex scene with fusion detections," in IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 1, pp. 199–208, Jan. 2018.

[20] M. Michael and M. Schlipsing, "Extending traffic light recognition: Efficient classification of phase and pictogram," 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, 2015, pp. 1–8.

[21] M. B. Jensen, K. Nasrollahi and T. B. Moeslund, "Evaluating state-of-the-art object detector on challenging traffic light data," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, 2017, pp. 882–888.

[22] M. Weber, P. Wolf and J. M. Zöllner, "DeepTLR: A single deep convolutional network for detection and classification of traffic lights," 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, 2016, pp. 342–348.

[23] I. J. Goodfellow, J. Pouget-Abadie,M . Mirza, B. Xu, D. Warde-Farley,S. Ozair, A. Courville and Y. Bengio,"Generative adversarial nets," Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[24] Y. Huang, W. Wang, L. Wang and T. Tan, "Multi-task deep neural network for multi-label learning," 2013 IEEE International Conference on Image Processing, Melbourne, VIC, 2013, pp. 2897-2900.

[25] V. Haltakov, J. Mayr, C. Unger, and S. Ilic, "Semantic segmentationbased traffic light detection at day and at night," inPattern Recognition,J. Gall, P. Gehler, and B. Leibe, Eds., 2015, pp. 446–457.

[26] C. Bao, C. Chen, H. Kui and X. Wang, "Safe driving at traffic lights: An image recognition based approach," 2019 20th IEEE International Conference on Mobile Data Management (MDM), Hong Kong, Hong Kong, 2019, pp. 112-117.

[27] R. de Charette and F. Nashashibi, "Real time visual traffic lights recognition based on Spot Light Detection and adaptive traffic lights templates," 2009 IEEE Intelligent Vehicles Symposium, Xian: IEEE, 2009, pp. 358-363.

[28] J. Bergstra, D. Yamins, D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," Proc. Of the 12th Python in science conf. (SCIPY 2013),2013.