UNIVERSIDADE DO ESTADO DO AMAZONAS – UEA
ESCOLA SUPERIOR DE TECNOLOGIA – EST
CURSO DE ENGENHARIA ELÉTRICA

JAMES FRANKLIN PEREIRA MONTEIRO

SISTEMA DE DETECÇÃO E IDENTIFICAÇÃO AUTOMÁTICA DE PLACAS DE TRÂNSITO BRASILEIRAS APLICADA AO AUXÍLIO À DIREÇÃO VEICULAR UTILIZANDO TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

MANAUS
2021

JAMES FRANKLIN PEREIRA MONTEIRO

SISTEMA DE DETECÇÃO E IDENTIFICAÇÃO AUTOMÁTICA DE PLACAS DE TRÂNSITO BRASILEIRAS APLICADA AO AUXÍLIO À DIREÇÃO VEICULAR UTILIZANDO TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

Projeto de Pesquisa desenvolvido durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Bacharel em Engenharia Elétrica

Orientador: Prof. Dr. Jozias Parente de Oliveira

MANAUS

2021

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
Cleinaldo de Almeida Costa
Vice-Reitor:
Cleto Cavalcante de Souza Leal
Diretor da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo
Coordenador do Curso de Engenharia Elétrica:
Israel Gondres Torné

Banca Avaliadora composta por: *Data da defesa: 30/12/2021.*
Prof. Jozias Parente de Oliveira, Dr. *(Orientador)*
Prof. Bruno da Gama Monteiro, Me.
Prof. Fábio de Sousa Cardoso, Dr.

CIP – Catalogação na Publicação

Monteiro, James Franklin Pereira Monteiro

Sistema de detecção e identificação automática de placas de trânsito brasileiras aplicada ao auxílio à direção veicular utilizando técnicas de processamento digita de imagens / James Franklin Pereira Monteiro/; [orientado por] Jozias Parente de Oliveira. – Manaus: 2021.

p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2021.

1.Detecção e identificação de placas de trânsito. 2. RCNN. 3. Processamento Digital de Imagens. De Oliveirta, Jozias Parente.

JAMES FRANKLIN PEREIRA MONTEIRO

SISTEMA DE DETECÇÃO E IDENTIFICAÇÃO AUTOMÁTICA DE PLACAS DE TRÂNSITO BRASILEIRAS APLICADA AO AUXÍLIO À DIREÇÃO VEICULAR UTILIZANDO TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Nota obtida: 9,8 (nove vírgula oito)

Aprovada em 30 / 12 / 2021

Área de concentração: Processamento Digital de Imagens

BANCA EXAMINADORA

Orientador: Jozias Parente de Oliveira, Dr.

Avaliador: Bruno da Gama Monteiro, Me.

Avaliador: Fábio de Sousa Cardoso, Dr.

Manaus 2021

Dedicatória

Aos meus pais, por terem dedicado suas vidas para me tornar a pessoa que sou hoje. Agradeço imensamente por todo o amor, suporte e auxílio que me foram dados para chegar a este momento tão importante. Dedico-lhes essa conquista, como forma de gratidão.

Agradecimentos

Agradeço aos meus pais, por todo o apoio e incentivo.

Agradeço ao meu orientador Jozias, por ter me aceito e orientado durante o desenvolvimento deste trabalho.

Agradeço aos meus amigos Matheus Nery e Ewerton Cassiano, além de todos os colegas de classe por todos os ensinamentos ao longo dos anos.

RESUMO

Os Sistemas Avançados de Assistência ao Motorista (ADAS) são sistemas que buscam auxiliar o motorista a exercer a atividade da direção veicular, de forma passiva ou ativa. O ADAS possui diversas funcionalidades, no entanto, destaca-se a capacidade que o automóvel efetue um mapeamento do ambiente ao redor, identificando situações de risco para o condutor, passageiros ou pedestres, devendo atuar de forma efetiva ou alertar sobre tais condições. Uma das aplicações deste mapeamento de ambiente, refere-se a sua usabilidade em detecção e identificação de placas de sinalização de trânsito. Para que esse processo seja feito com um índice adequado de precisão, são necessários utilização de técnicas de visão computacional, processamento digital de imagens e, mais recentemente, Redes Neurais. Com isso, este trabalho propõe-se a desenvolver um sistema de detecção e identificação automática de placas brasileiras de sinalização de trânsito, aplicadas ao auxílio à direção veicular, de forma a utilizar técnicas de processamento digital de imagens. O modelo é composto por um sistema de RCNN – (*Region Based Convolutional Neural Networks*), constituído com duas etapas: detecção e classificação. Na primeira, utilizando técnicas de processamento de imagem e modelo de SVM para efetuar a extração da zona de interesse, obtendo uma precisão da taxa de 67,2 % em ambiente de teste, com bases de *benchmark*. A segunda etapa consiste em utilizar o resultado do processo anterior para efetuar a classificação dos sinais de trânsito, seguindo um modelo de Rede Neural Convolucional clássico, chegando a uma taxa de precisão de 99,26 %, em teste com bases de *benchmark*, e para base de dados de placas brasileiras, 85,29 %, sendo considerado satisfatório para o conjunto de dados.

Palavras-chave: Detecção e identificação de placas de trânsito, RCNN, Processamento Digital de Imagens.

ABSTRACT

The Advanced Driver Assistance Systems (ADAS) are systems that seek to help the driver to drive the vehicle safely, either passively or actively. The ADAS has several characteristics, however, the vehicle's ability to map the surrounding environment stands out, identifying hypotheses of risk for the driver, passengers, or pedestrians, and must act effectively or alert about such conditions. One of the applications of this environment mapping refers to its usability in detecting and identifying traffic signs. For this process to be carried out with an adequate level of precision, the use of computer vision techniques, digital image processing and, more recently, Neural Networks are used. Thus, this work proposes to develop a system for the detection and automatic identification of Brazilian traffic sign plates, applied to aid in vehicular driving, to use digital image processing techniques. The model consists of a RCNN - (Region Based Convolutional Neural Networks) system, consisting of two stages: detection and classification. In the first, using image processing techniques and SVM model to extract the zone of interest, obtaining a rate accuracy of 67.2% in a test environment, testing with benchmark dataset. The second step consists of using the result of the previous process to carry out the classification of traffic signals, following a classic Convolutional Neural Network model, reaching an accuracy rate of 99.26%, in a test with benchmark dataset, and for database of Brazilian plates, 85.29%, being considered satisfactory for the dataset.

Keywords: Traffic Sign Detection and Classification, RCNN, Digital Image Processing

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADAS – *Advanced Driver Assistance Systems* (em português, *Sistema avançado de assistência ao motorista*)
BelgiumTSD – *Belgium Traffic Sign Dataset*
BRTSD – *Brazilien Traffic Sign Dataset*
CNN – *Convolutional Neural Network*
CONTRAN – *Conselho Nacional de Trânsito*
DENATRAN – *Departamento Nacional de Trânsito*
GTSRB – *German Traffic Sign Recognition Benchmark*
GTSDB - *German Traffic Sign Detection Benchmark*
HOG – *Histogram of Oriented Gradients*
HSV – *Hue, Saturation, Value*
IDE – *Integrated Development Environment*
MLP – *Multilayer Perceptron*
MSE – *Mean squared error*
OpenCV – *Open Source Computer Vision Library*
PDI – *Processamento Digital de Imagem*
RBG – *Red, blue and Green*
RCNN – *Region-based Convolutional Neural Network*
ReLU – *Rectifield Linear Unit*
RNA – *Redes Neurais Artificiais*
RNC – *Rede Neurais Convolucionais*
ROI – *Region of Interest*
SGDM – *Stochastic Gradient Descent with Momentum*
SSE – *Sum of Squared Residuals*
SVM – *Support Vector Machine*
UFRGS – *Universidade Federal do Rio Grande do Sul*
UFSC – *Universidade Federal de Santa Catarina*

SUMÁRIO

INTRODUÇÃO

O Sistema Avançado de Assistência ao Motorista (ADAS) possui o intuito de auxiliar o motorista, tornando o ato de dirigir mais seguro, desta forma este sistema visa reduzir a alarmante estatística de 1,2 milhões de mortes no trânsito globalmente, conforme (WHO, 2011), buscando minimizar os erros humanos na condução veicular que, por sua vez, corresponde a 94% da causa de incidentes ocorridos, segundo Singh (2015).

Para alcançar tal objetivo de aprimorar a segurança veicular, este sistema utiliza de ferramentas que monitoram e analisam o ambiente ao redor do veículo através da aquisição e processamento digital de imagens, podendo desta forma realizar tarefas como: alerta de colisão, detecção de pedestres, alerta de condições da via, detecção de objetivos, entre outros. Além disso, o sistema visa minimizar os problemas causados pela excessiva e constante carga de informações ao motorista, especificamente em condições não favoráveis, o que torna o ato da direção veicular mais perigoso.

Desde sua concepção o ADAS vem evoluindo juntamente com a sua aplicação e ênfase em veículo semiautônomos e autônomos, desta forma abrangendo uma maior gama de objetos a serem devidamente detectados e identificados, tal como a precisão e velocidade do reconhecimento dos objetos próximos ao veículo.

Tendo em vista este contexto, propõe-se a criação de um sistema de detecção e identificação de forma automática de placas de trânsito brasileiras aplicado ao sistema de auxílio à direção veicular. Desta forma, partindo da aquisição de imagens e tratamento; que por sua vez é constituído em digitalização, conversão para escala de cinza, limiarização e segmentação a imagem; seguido do treinamento, teste e avaliação do modelo desenvolvido, sendo assim, o modelo apresentado seria examinado em toda a sua extensão.

Buscando uma apresentação de forma clara e objetiva, a pesquisa está ordenada em 4 capítulos, além das referências.

Capítulo 1 – Referencial Teórico: Expõe os principais conceitos sobre o ADAS, modelos e distinções sobre as placas de trânsito brasileiras, a problemática sobre a legibilidade das indicações de sinalização, assim somo sobre os conceitos de imagem, o processamento digital da mesma e tomada de decisão. Aplicação de visão computacional para a resolução de problemas envolvendo detecção e classificação de objetos.

Capítulo 2 – Metodologia: Consiste no capítulo que exemplifica os processos de desenvolvimento adotados para este trabalho, assim como, as etapas de testes a serem aplicadas. Exemplificando e descrevendo as ferramentas, plataformas e conjuntos de dados utilizados.

Capítulo 3 – Implementação: Descreve como os procedimentos expostos na metodologia foram aplicados.

Capítulo 4 – Resultados: Expõe os resultados obtidos a partir da implantação da metodologia proposta, servindo como parâmetro para comparação da eficiência do modelo proposto no trabalho em relação aos benchmarks tradicionais, desta forma, podendo-se aferir uma conclusão, que será apresentada no fim do trabalho.

1 REFERENCIAL TEÓRICO

Com o problema e a sua respectiva proposta de solução sido apresentadas, faz-se necessário explanar sobre as ferramentas e métodos primordiais para a viabilidade da proposição deste trabalho. Tendo em vista que a hipótese se refere a elaboração de um sistema capaz de efetuar a detecção e correto reconhecimento de placas brasileiras de sinalização de trânsito de forma precisa e eficaz, através de aquisição e processamento de imagens reais, faz-se necessário discorrer sobre os conceitos de ADAS, modelos e distinções sobre as placas de trânsito brasileiras, a problemática sobre a legibilidade dos indicações de sinalização, assim somo sobre os conceitos de imagem, o processamento digital da mesma e tomada de decisão.

1.1 SISTEMAS AVANÇADOS DE ASSISTÊNCIA AO MOTORISTA – ADAS

Sistemas avançados de assistência ao motorista (*Advanced Driver Assistance system* - ADAS) representa um conjunto de sistemas e subsistemas, que podem ser inteligentes e capazes de auxiliar o condutor, de forma passiva ou ativa. Segundo Brookhuis (2001), o conceito do ADAS inclui dentre diversas funcionalidades a detecção de “pontos cegos”, controle de cruzeiro adaptativo, controle de cruzeiro inteligente automatizado, condução de pelotão, entre outros. Além disso, segundo Reif (2014), o sistema ADAS tem principal objetivo fazer com que o automóvel efetue um mapeamento do ambiente ao redor, identificando situações de risco para o condutor, passageiros ou pedestres, devendo atuar de forma efetiva ou alertar sobre tais condições.

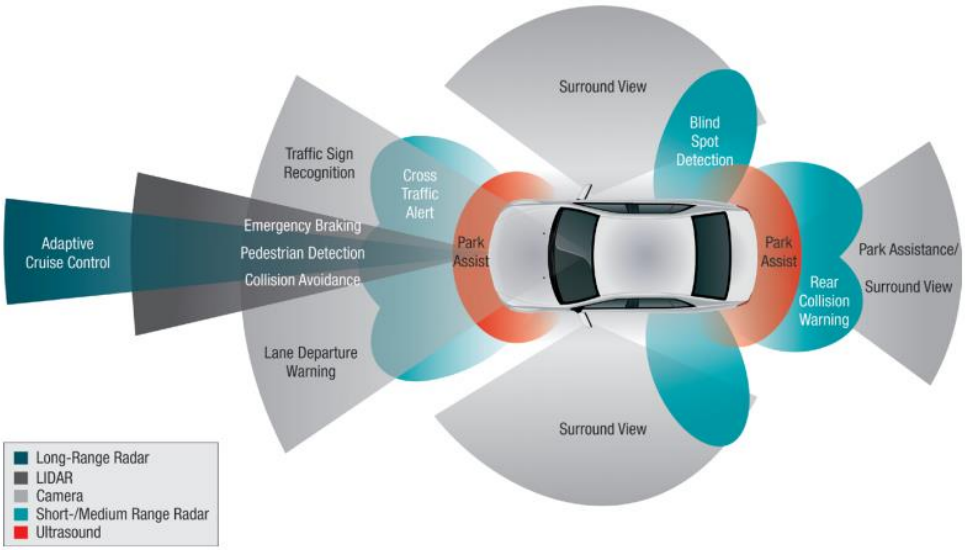
Segundo Winner (2015), o sistema ADAS pode ser classificado em cinco níveis distintos de aplicação:

a) **nível 1:** caracterizado pela ausência total de automação, o condutor é único e inteiramente responsável pelo controle do veículo;
b) **nível 2:** o condutor mantém as condições do nível anterior, no entanto, o sistema passa a apresentar um ou outro processo automatizado;
c) **nível 3:** passa a existir um certo grau de automação pelo controle do veicular. o condutor deve-se manter atento com este ativo;
d) **nível 4**: passa a existir um elevado grau de automação, no qual o condutor não necessita ficar atento na operação do veículo, podendo optar por ter o controle total do automóvel ou não;

e) **nível 5:** passa a haver a automação de todos os processos envolvidos, excluindo de forma total o controle manual, ou seja, não há a necessidade de um condutor.

A Figura 1, explana sobre a atuação do ADAS no controle longitudinal e lateral do veículo.

Figura 1 Exemplo de ADAS e suas funções de monitoramento e controle



Fonte: (CHITNIS, 2014)

Sobre o ADAS, pode-se ainda distinguir seus modos de operação, segundo Winner (2015), existem os seguintes três modelos de operação:

a) **modo a:** refere-se ao modo mais básico do adas, onde o mesmo provê informações e funções de aviso sobre situações externas, não havendo interferência direta no controle do automóvel. ex.: detecção e reconhecimento de placas de sinalização de trânsito, aviso sonoro de proximidade;
b) **modo b:** neste o ADAS, encarrega-se de forma imediata do controle parcial do carro. ex.: velocidade de cruzeiro, correção do posicionamento do veículo em rodovias;
c) **modo c:** neste o adas é capaz de assumir o total controle do veículo, em situações extremas, visando evitar falhas humanas. ex.: frenagem de emergência.

Com os itens apresentados anteriormente, notam-se os benefícios da concepção do ADAS, visando o reduzir ou até mesmo eliminar os erros do condutor. Segundo Brookhuis (2001), os benefícios da implementação do ADAS são de grande significância pois o mesmo é capaz de permitir que os condutores que utilizam o ADAS possam apresentar uma condução mais eficiente e segura; minimiza-se os problemas oriundos do clima, visão e ambiente.

1.2 PLACAS BRASILEIRAS DE SINALIZAÇÃO DE TRÂNSITO

Segundo o Código de Trânsito Brasileiro, as sinalizações de trânsito são definidas como:

> Sinais de trânsito são elementos de sinalização viária que se utilizam de placas, marca viárias, equipamentos de controle luminosos, dispositivos auxiliares, apitos e gestos, destinados exclusivamente a ordenar ou dirigir o trânsito dos veículos e pedestres (DENATRAN, 2008)

O Conselho Nacional de Trânsito (CONTRAN) é o órgão responsável por regularizar, fiscalizar e definir padrões de sinalizações, sendo estas segmentadas nas seguintes categorias:

- Sinalização vertical de regulamentação;
- Sinalização vertical de advertência;
- Sinalização vertical de indicação;
- Sinalização vertical horizontal;
- Sinalização semafórica;
- Sinalização de obras e dispositivos auxiliares.

De acordo com o tema proposto por este trabalho e sua respectiva delimitação de desenvolvimento, torna-se enfático a necessidade de expor os conceitos referentes às duas primeiras categorias de sinalizações verticais.

Sobre as sinalizações verticais, conforme regulamentação do CONTRAN, elas devem possuir formas padronizadas, associadas as mensagens que se quer transmitir.

Sobre as sinalizações verticais de regulamentação, o Manual de Brasileiro de Sinalização de Trânsito vol 1, afirma que a sinalização vertical de regulamentação tem por finalidade transmitir aos usuários as condições, proibições, obrigações ou restrições no uso das vias urbanas e rurais, (CONTRAN, 2005). Este grupo possui apenas as cores vermelho, branco e preto dispostas nas placas, como exemplificado na Figura 2.

Figura 2 Exemplos de sinalizações verticais brasileiras de regulamentação

Parada obrigatória	R-1	
Dê a preferência	R-2	
Velocidade máxima permitida	R-19	

Fonte: (CONTRAN, 2005)

Sobre as sinalizações verticais de advertência, o Manual de Brasileiro de Sinalização de Trânsito vol 2, afirma que a sinalização vertical de advertência tem por finalidade alertar aos usuários as condições potencialmente perigosas, obstáculos ou restrições na via ou adjacentes, (CONTRAN, 2005). Este grupo possui apenas as cores amarelo ou laranja e preto dispostas nas placas, como exemplificado na Figura 3.

Figura 3 Exemplos de sinalizações verticais brasileiras de advertência

	A-1a	Curva acentuada à esquerda
	A-1b	Curva acentuada à direita
	A-2a	Curva à esquerda

Fonte: (CONTRAN, 2007)

1.3 CONCEITO DE IMAGEM

As imagens podem ser descritas como uma função *f(x,y)*, onde as variáveis *x* e *y* representam a localização de um ponto qualquer na imagem, e f representa a intensidade luminosa no determinado ponto, ou seja, pode-se afirmar que uma imagem é a representação de uma matriz de intensidade luminosa, capaz de ser equacionada numa função *f(x,y)*. Segundo Guimarães (2008), uma imagem digital *f[m,n]* descrita em uma espaço discreto 2D é derivada de uma imagem analógica *f(x,y)* de um espaço contínuo

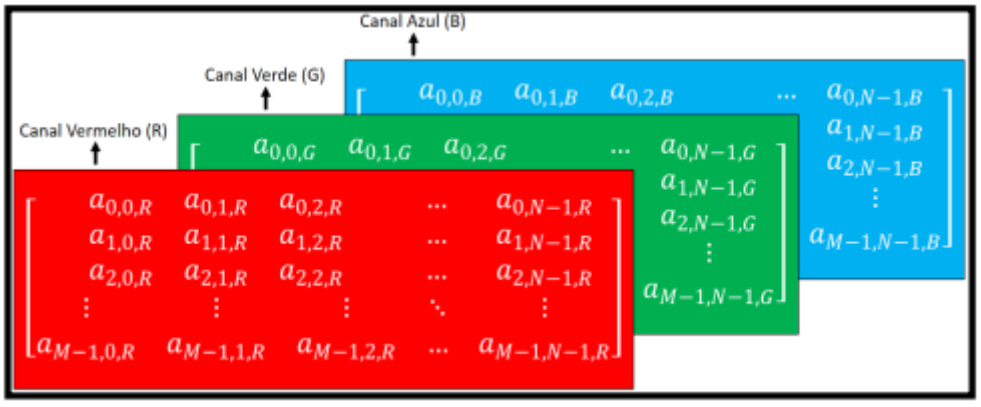
2D através de um processo de amostragem que é frequentemente referido como digitalização. Portanto, nota-se que imagens digitais são nada mais que matrizes onde cada unidade é chamada de pixel, onde a área MxN representa o número de pixels total da imagem, como pode ser visualizado na Figura 4.

Figura 4 Matriz de uma imagem em níveis de cinza

Fonte: (FILHO e NETO, 1999)

Sobre os pontos de intensidade, podemos afirmar que existem apenas três tipos de imagens: binária, monocromática e a de cores. A primeira, possui apenas preto (nível 0) e branco (nível 1). A segunda, varia os tons de cinza em decimais entre 0 e 1, de forma discreta, ou seja, há um limite de níveis de cinza, geralmente imposto por limitações de memórias virtuais para o armazenamento das imagens, este limite de geralmente segue as potências de 1 (64, 128, 256...). A última, refere-se a fotos coloridas no formato RGB, um composto de três matrizes de intensidade de vermelho, verde e azul, respectivamente, as composições das matrizes quanto ao formato de RGB podem ser visualizadas na Figura 5.

Figura 5 Matriz de uma imagem RGB

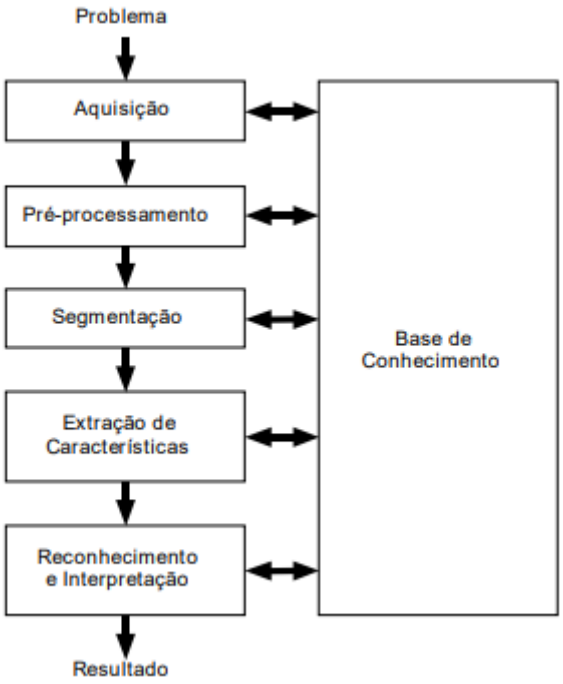


Fonte: (FILHO e NETO, 1999)

1.4 TÉCNICAS DE PROCESSAMENTO DIGITAL DE IMAGENS

Sobre as etapas e técnicas que envolvem o sistema de aquisição e processamento de imagem, conforme Filho e Neto (1999), elas envolvem os procedimentos e sequência que estão representados e dispostos na Figura 6.

Figura 6 Sistema de processamento de imagens



Fonte (FILHO e NETO, 1999)

1.4.1 Problema

Refere-se ao domínio do problema, que no projeto em questão seriam as placas de sinalização brasileiras de trânsito.

1.4.2 Aquisição

Refere-se ao primeiro passo no processo propriamente dito. Para que ocorra a aquisição de imagens é necessário a utilização de um sensor e um digitalizador. Segundo Filho e Neto (1999), o sensor converterá a informação óptica em sinal elétrico e o digitalizador transformará a imagem analógica em uma imagem digital. Em sequência a

este processo a imagem é escalonada para um tamanho mais adequado ao sistema, preparando-a para os próximos processos.

1.4.3 Pré-processamento

De forma geral, esta etapa do sistema visa aperfeiçoar a qualidade da imagem para as etapas seguintes, tendo em vista que o objeto de interesse disposto pode apresentar pixels ruidosos, contrastes e brilho inadequado, entre outras imperfeições. Para que as irregularidades sejam sanadas, esta fase é subdivida nos seguintes tópicos:

a) **aprimoramento:** refere-se a etapa onde são utilizadas diversas técnicas de intensidade, como filtros especiais, ou transformadas de Fourier com a finalidade de exibir partes ou detalhes de uma figura antes obscura por problemas de aquisição ou do próprio ambiente externo. Além disso, é possível efetuar outros tipos de aprimoramento, como o contraste e saturação da imagem;
b) **restauração:** é uma etapa que é executada caso a imagem digitalizada esteja degradada. Sendo assim, envolve modelos probabilísticos ou matemáticos para estabelecer características que haviam sido perdidas por diferentes fatores que variam desde o efeito do tempo na degradação da imagem até a más condições do ambiente externo;
c) **processamento de imagem em cores:** refere-se a etapa onde a imagem é colocada num formato mais apropriado para os tópicos seguintes, desta forma, neste ponto a mesma é transformado para um formato em tons de cinza, ou em formato binário;
d) **compressão:** refere-se a etapa que lida com a relação de capacidade de armazenamento e processamento inerente a uma imagem, ou a largura de banda para transmitir a mesma.

1.4.4 Segmentação

Corresponde a etapa onde uma imagem é dividida em diversas unidades, de forma significativa, ou seja, nos objetos de interesse que a compõem. Refere-se a das etapas de maior dificuldade de implementação do processamento digital de imagens.

1.4.5 Extração de características

Refere-se à etapa que buscar extrair características das imagens resultantes através de descritores que permitam caracterizar com precisão e com bom poder se discriminação entre os objetos destacados na imagem. É importante salientar que nesta etapa a entrada continua sendo uma imagem, no entanto, a saída resultante é um conjunto de dados correspondentes àquela imagem.

1.4.6 Reconhecimento e interpretação

Segundo Filho e Neto (1999), refere-se a última etapa do sistema, onde é atribuído um rótulo a um objeto baseado em suas características, traduzidas por seus descritores, o ato de rotular consiste no reconhecimento. A tarefa de interpretação, consiste em atribuir um significado a um conjunto de objetos reconhecidos. Para a resolução do problema proposto neste trabalho, nota-se que nesta etapa o reconhecimento refere-se ao processo de detecção de que há uma placa de sinalização de trânsito na imagem de origem, e a etapa de interpretação representa a identificação de qual sinalização aquele objeto faz referência.

Por fim, nota-se que todas as etapas estão relacionadas a base de conhecimento, que consiste no local onde são armazenados todo o conhecimento sobre o problema a ser resolvido, já existente, ou minimamente as informações necessárias para a resolução da problemática.

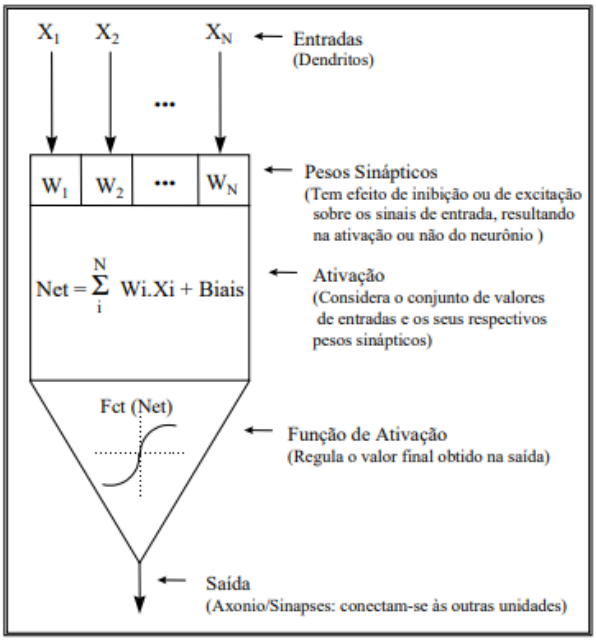
1.5 REDES NEURAIS

Segundo Russel e Novig (2003), redes neurais artificiais (RNA), são baseadas no conceito simplificado do funcionamento biológico do cérebro, onde um neurônio possui a função de processar e disseminar sinais elétricos.

De forma simplificada, pode-se afirmar que o conhecimento de uma RNA está conectado a sua estrutura de rede, onde tem-se em evidências as conexões, ou sinapses, entre as unidades, ou neurônios, que a compõem. É de importante valia, ressaltar que é atribuído um peso sináptico, valor numérico, a cada conexão, o que por sua vez caracteriza a força dela. O processo de aprendizagem de uma RNA consiste na adaptação

dos seus pesos sinápticos, o que leva a criação de um caminho com uma resposta ao problema proposto embasada no peso das conexões estabelecida. A Figura 7 exemplifica a relação de pesos sinápticos atribuídos a estrutura de rede de um neurônio artificial.

Figura 7 Exemplo de Neurônio Artificial

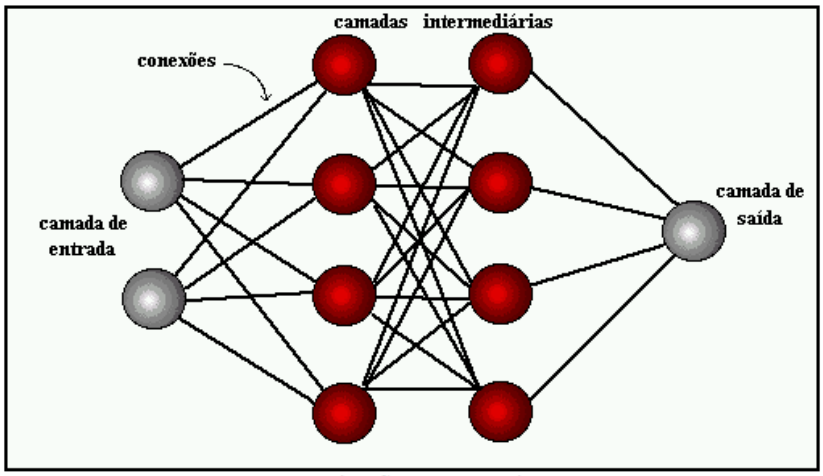


Fonte: Osório e Bittencourt (2000)

Sobre as camadas de conexão, usualmente estas são classificadas conforme o exposto na Figura 8, e possuem as seguintes atribuições, segundo Carvalho (2009):

- **Camada de entrada:** onde os padrões são apresentados à rede;
- **Camadas intermediarias ou escondidas**: onde é feito grande parte do processamento, através das conexões ponderadas;
- **Camada de saída:** onde o resultado final é concluído e apresentado.

Figura 8 RNA - Organização em camadas



Fonte: Carvalho (2009)

Sobre o processo de aprendizagem inerente a uma RNA, conforme Osório e Bittencourt (2000), o aprendizado ocorre de forma gradual e iterada, onde os pesos são modificados por diversas vezes, conforme os aprendizados já obtidos. O estágio de aquisições de conhecimento, utiliza apenas de um conjunto de dados de aprendizado, não sendo este equivalente ao conjunto de dados geral a ser trabalhado. Segundo Osório e Bittencourt (2000), os métodos de aprendizado neural podem ser divididos em três grandes classes:

- **Aprendizado supervisionado**: neste, o usuário dispõe de um comportamento de referência preciso que ele deseja ensinar a rede, desta forma, o sistema deve medir a diferença entre seu comportamento atual e o desejado, e então corrigir o sistema e pesos adotados.
- **Aprendizado semi-supervisionado:** neste, o usuário dispõe apenas das indicações imprecisas sobre o comportamento desejado. Este método também é chamado de aprendizado por esforço, pois dispões apenas de medidas de sucesso e insucesso.
- **Aprendizado não supervisionado:** neste, os pesos da rede são modificados em função de critérios internos, tais como, por exemplo, a repetição de padrões de ativação em paralelo de vários neurônios.

Sobre as aplicações de RNAs, as mesmas podem ser utilizadas em diferentes tipos de tarefas, tais como: o reconhecimento de padrões, classificações, transformações de

dados, predições controle de processos e aproximações de funções. No caso do trabalho aqui proposto, utiliza-se da primeira aplicação supracitada. De forma mais geral, segundo Osório e Bittencourt (2000), todas as tarefas anteriormente citadas, podem ser agrupadas em dois grandes grupos:

- **Redes para aproximação de funções:** redes que apresentam saídas com valores contínuos e usualmente empregados para a aproximações de funções.
- **Redes para a classificação de padrões:** redes que devem atribuir para cada item que lhe é fornecido uma classe ao qual este item pertence, ou seja, esta rede efetua uma classificação do item enquadrando o em um determinado grupo de acordo com suas características.

1.5.1 SVM – *Support Vector Machine*

Uma das técnicas de aprendizagem de máquina supervisionado mais difundidas é conhecida como SVM – *Support Vector Machine*. Segundo Akande (2014), este método tornou-se bastante popular pois alinha uma grande capacidade de generalização para soluções dos problemas propostos com uma ótima eficiência.

Segundo Hsu, Chang e Lin (2003), o objetivo central do SVM é realizar a criação de um modelo que possa realizar a predição de a qual classe de um determinado dado a partir dos seus atributos. Sendo assim, para efetuar a correta classificação entre dos dados em análise, é necessário haver um conjunto de dados de treinamento e testes, informando as classes de cada dado presente no conjunto.

Seguem as Equações 1 e 2, usadas para rotulo e otimização de um SVM.

$$(x_i, y_i), i = 1, \dots l \tag{1}$$

Onde $x_i \in R^n$ e $y_i \in 1, -1^l$

$$\min \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{2}$$

Sujeito à $y_i(w^T \varphi(x_i) + b) \geq 1 - \xi_i$ e $\xi_i \geq 0$

Segundo Hsu, Chang e Lin (2003), as funções kernel são descritas para mapear os dados em um espaço dimensional superior. A utilização das funções kernel é comumente a escolha para a solução do problema proposto neste trabalho.

Utilizou-se a função radial *basis function* que é descrita na Equação 3.

$$K(x1, x2) = \exp\left(-\gamma \left(\left|\left|x_i - x_j\right|\right|\right)^2, \gamma > 0\right) \quad (3)$$

1.5.2 Redes Neurais Artificiais – Perceptron Multicamadas

Como exposto anteriormente, as Redes Neurais Artificiais possuem o seu funcionamento básico espelhado nas redes neurais humanas, utilizando da atribuição de pesos e cálculos matemáticos para distribuir pesos e criar conexões entre os neurônios em questões para que sejam realizadas tomadas de decisões baseadas no caminho que possui a maior correlação com as características apresentadas na entrada do sistema. Dito isso, segundo Haykin (2009), estes modelos podem ser utilizados para a criação de algoritmos que visão realizar o reconhecimento de imagens, sinais, vozes, entre outros. De forma geral, estes algoritmos são fundamentados em criar relacionamento, ou conexões, entre os neurônios, que por sua vez são unidades de processamento dispostas em camadas sequenciais e são ligados entre si através de sinapses, que possuem pesos distintos.

O processo anterior pode ser exemplificado através da Figura 7, onde as entradas do neurônio recebem os sinais X1, X2... Xn; os pesos sinápticos são representados por W1, W2,...,Wn, que possuem efeito na inibição ou excitação dos sinais que estão entrando no neurônio, resultando na ativação ou não do mesmo; a junção aditiva ou ativação do neurônio é composta pelo é composta pelo somatório dos sinais de entrada e seus respectivos pesos com a adição do *Biais*, unidade utilizada na rede neural para ajudar a ajudar os pesos. Em seguida o resultado da junção aditiva é submetido a função de ativação, segundo Osório e Bittencourt (2000), uma função de ativação tem o propósito de regular o valor final obtido pelo neurônio, desta forma, esta função acaba por restringir o resultado obtido pelo neurônio a um valor finito, comumente, os intervalores de 0 e 1 ou -1 e 1 são aplicadas para limitar as opções de resposta. As funções de ativação mais difundidas e utilizadas são:

a) Função *Reactified Linear Unit* (ReLU): é a função que mais se adapta para o desenvolvimento de Redes Neurais, pois o cálculo do gradiente de um neurônio é computacionalmente imperceptível, o que auxilia na prevenção do crescimento exponencial dos requerimentos computacionais para realizar as operações de Redes Neurais, de acordo com Goodfellow, Bengio e Courville (2016)
b) Função Limiar: é a função que efetua a normalização do sinal de saída em duas possibilidades de valores, 0 ou 1;
c) Função Sigmoid: é a função que por definição atribui valores crescente, é comumente aplicada em algoritmos de classificação de imagens, pois é possível inferir a precisão da classificação escolhida pelo modelo.

Segundo Haykin (2009), a primeira etapa para o desenvolvimento de uma RNA é a definição da estrutura que será adotada pela mesma, determinando como será a atribuição dos neurônios utilizados, assim como a quantidade de camadas e conexões. Dentre as possibilidades de estruturas que podem ser adotadas, será utilizada a abordagem da Perceptron Multicamadas.

A Perceptron Multicamadas (*MultiLayer Perceptron* – MLP) apresenta uma estrutura de conjuntos virtuais de unidade sensoriais na camada de entrada, apresentando uma ou mais camadas de neurônios ocultas e uma camada de saída. Sendo estas redes inteiramente conectadas, de tal forma que cada um dos neurônios que a compõe uma camada conecta-se com todos da camada anterior, como pode ser visto na Figura 8.

Segundo Haykin (2009), os dados que são inseridos na estrutura de MLP compreendem informações características do problema em estudo, podendo ser relacionados a forma geométrica ou cor do objeto, sendo essas informações extraídas de prévias nas etapas de pré-processamento.

Quanto ao funcionamento do MLP, segundo Haykin (2009), os modelos que utilizam da estrutura MLP necessitam de treinamentos sob supervisão, desta forma, o algoritmo pode aprender com um conjunto de dados prévio onde os resultados já são conhecidos, se forma que seja possível criar padrões para entrada com características semelhantes às de treino.

Dos algoritmos que utilizam MLP, o back-propagation tem destaque, pois possui uma boa eficiência, realizando o treinando da RNA e sendo aplicável em diversos problemas complexos. Este modelo baseia-se no aprendizado por tentativa e erro, usando um conjunto de dados de entrada e saída já identificados. Desta forma, é possível efetuar

a correção de erro, com isso o modelo passa a aprender baseado em erros anteriores, uma vez que os pesos de cada conexão da RNA são iniciados de forma aleatória, o que acarreta num volume de erro muito grande, no entanto, a cada repetição é calculada os valores de SSE (*Sum Of Squared*) - Soma dos Quadrados dos Erros e MSE (*Mean Squared Error*) – Erro Quadrático Médio, o que permite ajustar a atribuição de pesos de cada conexão, desde a camada de saída até a entrada, fazendo com que na próximo iteração haja um resultado mais próximo do esperado para o treinamento.

Sobre as peculiaridades de uma MLP, uma das principais é quanto a quantidade de interações realizadas no período de treino de um modelo, de tal forma que essa quantidade seja suficientemente boa para que o modelo apresente taxas de aproveitamento e erro aceitáveis, evitando o *overfitting*, quando há excesso de treinamento, o que ocasiona em que as informações provenientes na entrada sejam apenas memorizadas e não se fato aprendidas, criando um impacto nas conexões feitas para chegar à conclusão, tornando-a ineficaz para conjuntos de dados diferentes. Outra possibilidade é o *underfitting*, quando o modelo para a apresentar uma alta taxa de erro durante o treino com valor próximo ao erro na validação, geralmente ocasionados por dados faltantes na base de treino ou poucas interações realizadas, uma vez que a taxa de erro da rede é elevada no início do treino, mas diminui de acordo com as novas iterações e consecutivo aprendizados do modelo.

1.6 DEEP LEARNING

Segundo Grace et al. (2018), uma das áreas de pesquisa mais desenvolvidas atualmente é a que abrange os conceitos de Deep Learning (DL), ou Aprendizado Profundo, uma vez que estes modelos possuem uma vasta área de aplicação como reconhecimento de fala, reconhecimento de objetos, sinais, visão computacional e outros.

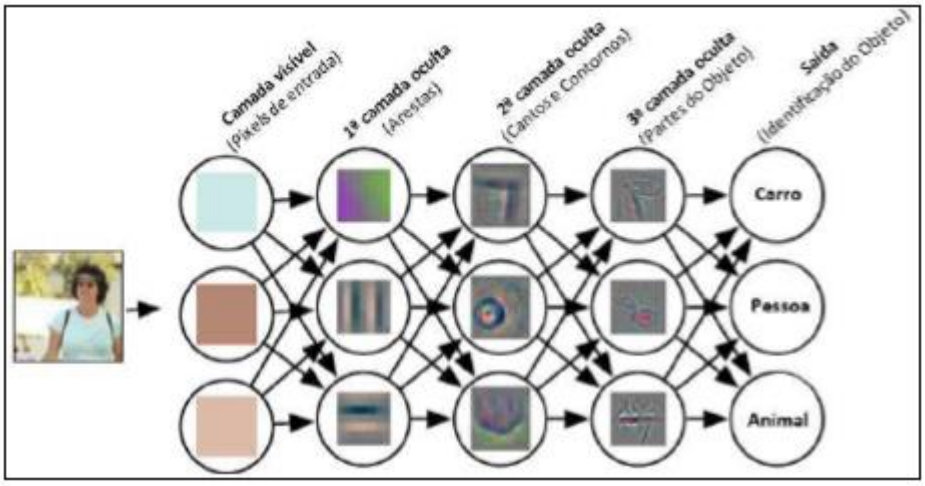
De acordo com Copeland (2016), o Aprendizado Profundo apresenta técnicas de significativa importância para análise de dados não categorizados, utilizando de RN em processamento de imagens, reconhecimento de voz, entre outros.

1.6.1 Reconhecimento de imagens utilizando Deep Learning

Segundo Goodfellow, Bengio, Courville (2016), para utilização de algoritmos de Deep Learning em processos de reconhecimento de imagens, é necessário efetuar o treinamento do modelo inserindo com imagens que representem as características das classes a serem classificadas, apresentando condições diferentes entre si, como: ângulo, luminosidade, situações distintas. Desde forma o modelo é capas de aprender a efetuar o reconhecimento de imagens com base nas características das imagens previamente utilizadas para treino.

Observando a Figura 9, fica exemplificado o funcionamento de um modelo de Deep Learning aplicado a reconhecimento, onde ele coleta as características da imagem de entrada, como cor, formato. Onde o modelo efetua o processamento dessas características em uma estrada de neurônios distribuídas em camadas de forma segmentada e hierarquizada, avisando assim uma interconexão entre as camadas utilizadas

Figura 9 Reconhecimento de Imagens usando Deep Learning - conceito



Fonte: Oliveira (2018)

De forma geral, os processos de *deep learning* são semelhantes as redes neurais artificiais, porém efetuam uma verificação mais profunda acerca dos dados, pois utilizam um número mais significativo de camadas de neurônios, em comparação com modelos de RNA padrão, o que permite um aprendizado mais elaborado do modelo e a utilização de mais características da imagem em estudo para problemas de classificação. Segundo Oliveira (2018), a melhor técnica de *deep learning* buscando performance e custo computacional reduzido para reconhecimento de imagens é utilização de *Neural Network*

Convolucions (CNN), ou Rede Neural Convolucional (RNC). Este modelo é caracterizado por efetuar o processamento de sinais na sua forma mais simplória, principalmente tratando-se sinais digitais, imagens e áudios, as características destes sinais são obtidas através da utilização de um processo de convoluções, realizadas sequencialmente.

A extração das principais características da imagem que serão utilizadas pelo modelo de aprendizagem de máquina ocorre através da camada de CNN, como exposto anteriormente. Essa camada é posicionada anteriormente a etapa do modelo, pois possui a função de obter os principais e mais significativos componentes da imagem em estudo e fornecer para o modelo, que sequencialmente efetua os testes e predições quanto as características recebidas.

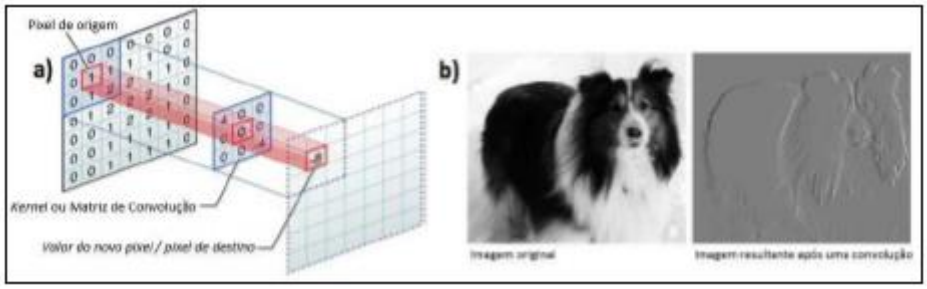
1.7 REDES NEURAIS CONVOLUCIONAIS

As Redes Neurais Convolucionais utilizam um processo de convoluções sequenciais em um conjunto de dados, valendo-se da aplicação de filtros e métodos previamente aprendidos para obter as características desejáveis. Segundo Pacheco (2017), a grande vantagem desse método em relação a RNA tradicional quando aplicado em imagens, é que este possui o poder de efetuar a codificação das propriedades, fazendo com que o treinamento diminua sem a afetar a precisão do modelo.

1.7.1 Convolução

A convolução consiste num processo que possui a finalidade de criar um sinal resultante de duas funções ou sinais, através do produto deles. Segundo Goodfellow, Bengio e Courville (2016), a convolução é uma operação que efetua a transformação linear de uma matriz primária, podendo estar ser um conjunto de 3 matrizes caso seja a representação de uma imagem RGB, de outra forma, para imagens em tons de cinza ou preta e branca, através da matriz primária é formada o kernel ou matriz de convolução. Em seguida, efetua-se a convolução propriamente dita, a multiplicação da matriz primária pela matriz de convolução, resultando numa terceira matriz que carrega as informações características da imagem ou matriz original, procedimento pode ser vista na Figura 10.

Figura 10 Convolução aplicada em imagem

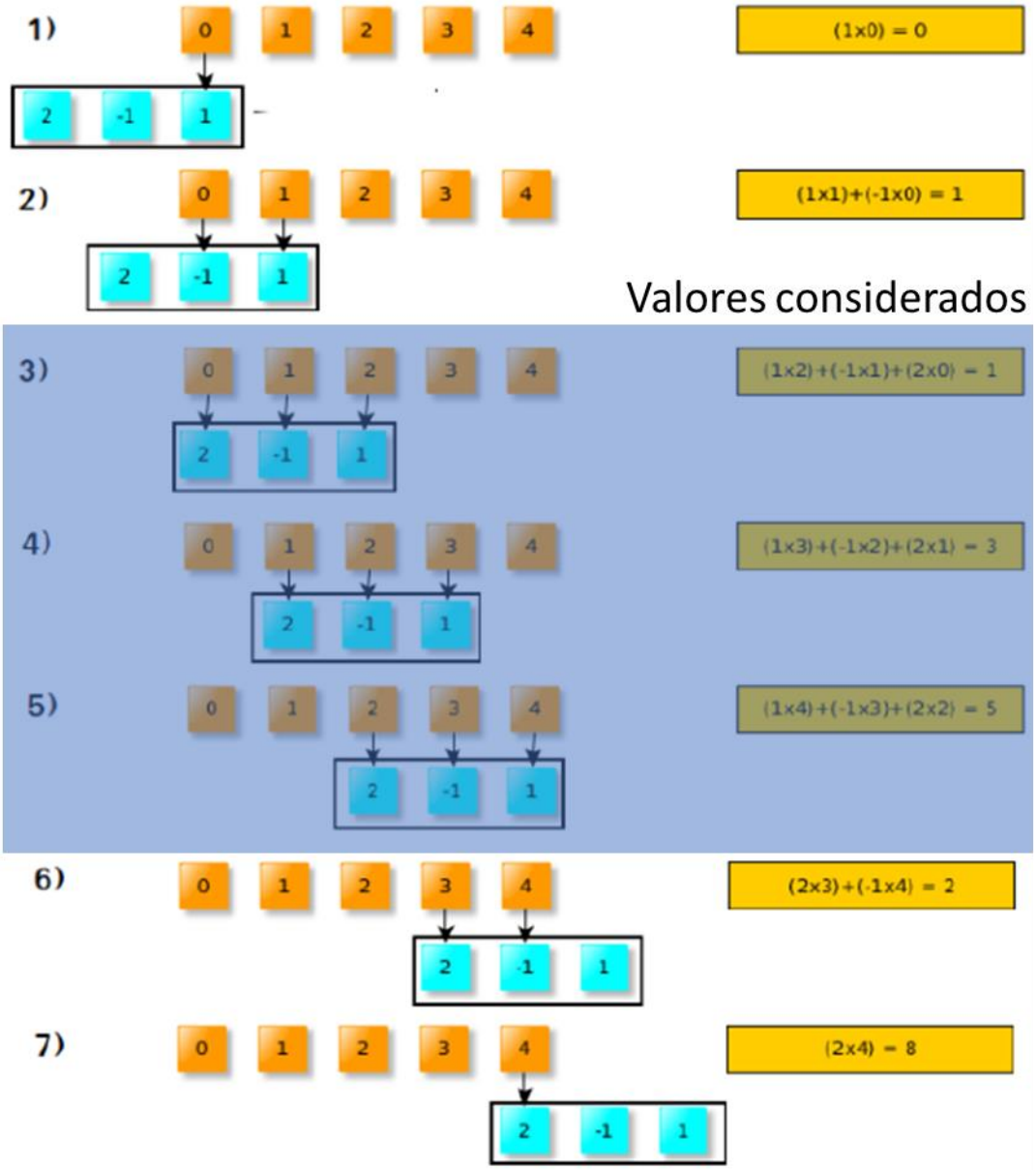


Fonte: Oliveira (2018)

1.7.2 Processo de convolução

A Figura 11 exemplifica as etapas de um processo convolucional, onde X = [0 1 2 3 4] e W = [1 -1 2], correspondem a dois sinais distintos. Primeiramente, é gerado a matriz invertida de W, chamada de W’ que corresponde a [2 -1 1]. Em seguida, é iniciado o processo de justaposição de matriz do sinal W’ sobre X, de tal forma que W’ é deslocada da esquerda para direita, se forma que acha a multiplicação dos valores que estão justapostos e em sequência a soma deles. O resultado de cada uma das interações é apresentado no fim, como a saída do processo convolucional.

Figura 11 Cálculo de uma convolução



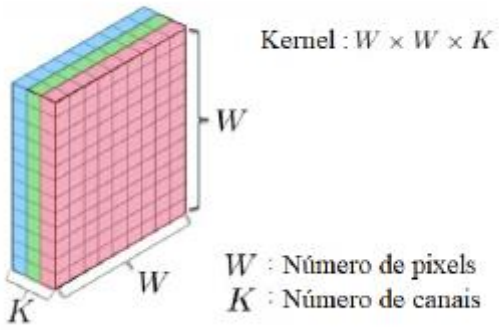
Fonte: Adaptado de Preto (2018)

Quando aplicado em processos de RNC, os valores de X correspondem as informações de pixels de uma imagem, por usa vez, o W' corresponde aos pesos, filtros ou *kernel*. São apenas considerados resultados válidos, aqueles onde o *kernel* é totalmente multiplicado com os pixels de entrada.

1.7.3 Camada Convolucional

Esta camada utiliza os conceitos de convolução apresentados anteriormente, no entanto, o kernel passa a ser uma matriz bidimensional, já que uma imagem possui duas dimensões, conforme ilustrado na figura 12. Há distinções para o tratamento de imagens em tom de cinza para imagens RGB, no primeiro caso, a matriz possuirá apenas um canal de entrada, já no segundo, apresentará 3 canais, um para cada composição de cor, fazendo com que a matriz possua 3 dimensões.

Figura 12 Kernel de uma imagem RGB



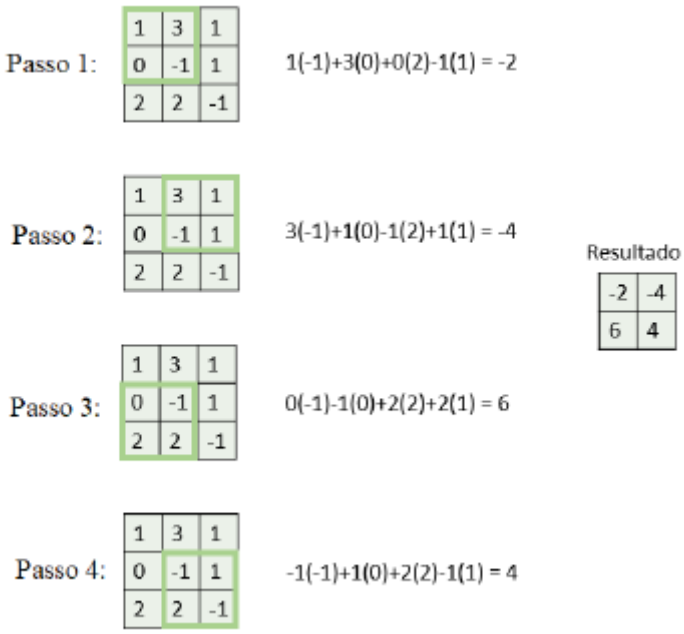
Fonte: Matsui, Cuturi, Vert e Birkenes (2007)

Na Figura 13 é ilustrado um exemplo de como a camada kernel percorre na matriz de entrada, nota-se que esse processo se assemelha ao exemplificado no tópico anterior num sinal vetorial. Ainda sobre o processo de "varredura", há alguns conceitos importantes quanto a aplicação em imagens. O primeiro, é o *Stride*, passo, que basicamente informa quantos pixels laterais a amada de kernel deve percorrer lateralmente em cada interação. O segundo, refere-se ao conceito de dilatação, que consiste em um vetor que abrange dois números inteiros, sendo utilizado para realizar o aumento da área de entrada vista pela camada, sem exigir requerimentos computacionais mais sofisticados. O terceiro, é o *Zero Padding*, utilizado para preencher com zero os pixels que abrangem toda a borda da matriz original, esse procedimento ocorre para que haja um controle do tamanho da matriz resultante, gerando um padrão. O tamanho da matriz final pode ser obtido através da Equação 4 abaixo:

$$\frac{t - \big((f-1)d + 1\big) + 2p}{s} + 1 \qquad (4)$$

Onde t representa o tamanho da entrada, f o tamanho do filtro, d o fator de dilatação, o *padding* e s o *stride*.

Figura 13 Processo de convolução em matriz



Fonte: Adaptado de Preto (2018)

1.7.4 Camada de Pooling

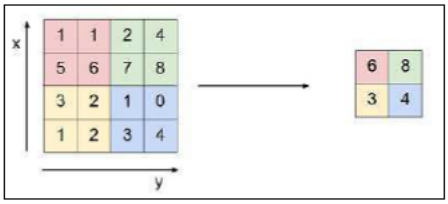
A Camada de Pooling é responsável por reduzir as dimensões de uma imagem, porém, mantendo as informações mis relevantes da mesma. Desta forma, o Pooling gera uma saída resumida da imagem anteriormente filtrada, minimizando os custos computacionais.

De forma geral, existem dois tipos de processos de Pooling, o Max Pooling e o Average Pooling. O primeiro, consiste em dividir a matriz de entrada em regiões e manter o maior valor apresentado por região. O segundo, consiste em dividir a matriz em regiões e calcular o valor médio de cada região.

Segundo Oliveira (2018), a função de Max Pooling é a mais utilizada e indicada para análise de imagens uma vez que pelo fato de considerar os valores máximo de cada região proporciona uma melhor visibilidade das áreas de alto contraste, que geralmente

podem ser associadas a objetos e corres distintas em um ambiente. A Figura 14 representa como ocorre a aplicação da função de Max-Pooling

Figura 14 Representação da função Max-Pooling



Fonte: Adaptado de Daniela de Oliveira Preto (2018)

1.7.5 Camada de SoftMax

A Camada de SoftMax, é a responsável por obter a probabilidade de a imagem de entrada ser de determinada classe, sendo assim, ocorre posteriormente a camada totalmente conectada.

O valor da probabilidade que é resultante dessa camada é obtido através da Equação 5:

$$P(cr\ |x,0\ ) = \frac{P(\grave{}x,0|cr)P(cr)}{\sum_{j=1}^{k} P(\grave{}x,0|cj)P(cj)} \quad (5)$$

Onde $0 \leq\ P(\grave{}cr|x,0) \leq 1, \sum_{j=1}^{k} = 1,\ P(\grave{}x,0|cr)$, corresponde a probabilidade condicional de uma determinada classe r, e P(cr) corresponde a probabilidade prévia de pertencer a esta classe r.

1.7.6 Camada de classificação

Corresponde a camada que efetua o cálculo de perdas de entropia de cada classe de saída, uma vez que esta é mutuamente exclusiva com o valor atribuído a classificação. Sendo assim, essa camada é posiciona de tal forma que receba os valores oriundas da camada de SoftMax, efetuando o cálculo de perda correspondente a cada classe de saída. É utilizada a Equação 6 para o cálculo de perda.

$$loss = \sum_{i=1}^{N} \sum_{j=1}^{K} t_{ij} \ln y_{ij}$$

(6)

Onde N representa o número de amostras, K o de classes, Tij corresponde a indicação de que a i-ésima amostra pertence a j-ésima classe, e Yij corresponde ao valor recebido da camada anterior.

1.7.7 Camada completamente conectada

Esta camada é aquela em que cada neurônio está conectado a todas as saídas da camada anterior, desta forma, todos os neurônios estão interconectados. Neste ponto todas as características adquiridas em outras camadas são utilizadas para encontrar padrões sobre a informação recebida na entrada, de forma tal que haja os ajustes de pesos para cada conexão. A fórmula que representa os ajustas de peso e tomada de decisão nessa etapa é representada na Equação 7.

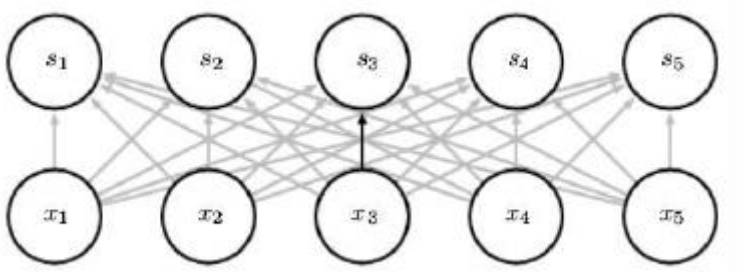
$$S = X * W + b$$

(7)

Onde, X é a informação recebida, W é a matriz que contém os pesos de cada conexão e b, a tendência.

A Figura 15 exemplifica as conexões efetuadas por todas as células presentes nas camadas completamente conectadas.

Figura 15 Camada totalmente conectada



Fonte: Goodfellow, Bengio, Courville (2016)

1.8 DETECÇÃO DE REGIÕES DE INTERESSE

Para problemas que envolvem situações de detecção e classificação em conjunto, foi desenvolvido o conceito de Rede Neurais Convolucionais baseadas em regiões, uma vez que uma RNC normalmente não consegue apenas efetuar os dois processos, ou seja, não conseguem identificar qual a região deve ser classificada, fazendo apensar a segunda etapa do processo.

1.8.1 Redes Neurais Convolucionais baseadas em regiões

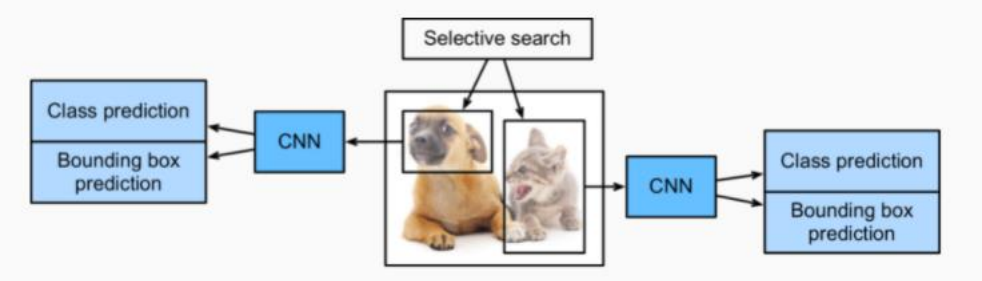
Sobre as Redes Neurais, há um seguimento de aplicação para detecção e identificação de regiões, estas são as chamadas Redes Neurais Convolucionais Baseadas em Regiões - R-CNN (*Region-based Convolucional Neural Network*). Segundo Girshick (2014), esta visa segmentar uma imagem em sub-regiões, posteriormente cada região possui a dimensão alterada para ser aplicada a estrutura de CNN já exposta, por fim, a rede neural efetua a classificação da região.

Segundo Girshick (2014), as etapas de um algoritmo de R-CNN consistem em:

1. Realizar uma pesquisa seletiva para extrair várias propostas de regiões de alta qualidade da imagem de entrada, sendo estas selecionadas em escalas, formas e tamanhos. Cada região é rotulada com uma classe e delimitada por uma *boundboxing*;
2. É realizado a escolha de um CNN, pré-treinado, sendo os resultados truncados antes da camada de saída. Cada proposta de região é redimensionada para ser aceitável aos padrões da CNN, de forma que produza os recursos extraídos para a proposta de região;
3. As características extraídas e classe rotulada de cada região, são utilizadas no treinamento de máquinas de vetores de suporte – SVM, onde cada módulo deste determina de forma individual se um exemplo contém uma classe específica;
4. Por fim, cada característica e rótulos atribuídos as *boundboxing*, são aplicadas para o treinamento de um modelo de regressão linear, de forma a prever a classe da região selecionada.

A Figura 16 exemplifica os processos de R-CNN anteriormente descritos.

Figura 16 Processo de uma RCNN



Fonte: Girschick (2014)

1.8.2 HOG – *Histogram of Oriented Gradients*

O HOG (*Histogram of Oriented Gradients*) ou Histograma da Orientação dos Gradientes, consiste num descritor de imagens utilizado principalmente para projetos de detecção de objetos, pois o resultado proveniente dos filtros de HOG apresenta as características da imagem em análise, podendo o resultado ser extraído e adicionado em um *pipeline* de aprendizado de máquina para a detecção da classe à qual as características ali encontradas pertencem, sendo amplamente utilizada pois possui um larga capacidade de representar a imagem ou objeto em questão por um simples vetor.

De forma geral, segundo Mallick (2016), o funcionamento do HOG consiste em coletar informações de importância significativa sobre a direção de gradiente de uma imagem dividida em diversas células. Sendo assim, o algoritmo calcula o módulo e a direção dos gradientes, divide a imagem em múltiplas células, calcula o histograma dos gradientes e os concatena em vetor.

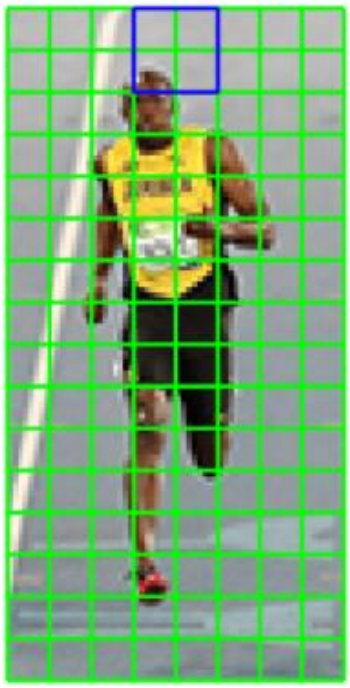
Sendo assim, a primeira etapa do algoritmo é responsável por calcular os gradientes horizontais e verticais da imagem, e a magnitude dos gradientes, que segundo Dalal e Triggs (2005), em caso de imagem em cores, cada componente terá o valor de gradiente calculado pelo *script*, porém apenas a componente com o maior valor de gradiente continuará a ser utilizada no decorrer do processo. Nota-se que o cálculo dos gradientes tira da imagem muitas informações desnecessárias para processos de detecção, permitindo que fique apenas as regiões de alta frequência, mais comumente encontradas em contornos de objetos. A Figura 17 representa os processos de gradientes sendo aplicados a uma imagem da base de dados a ser utilizada neste trabalho.

Figura 17 Gradientes

Fonte: Autoria própria

A segunda etapa, segundo Mallick (2016), consiste em dividir a imagem em múltiplas células, geralmente o tamanho de célula utilizado é de 8x8 pixels, porém este tamanho pode ser adequado para cada aplicação. Vale ressaltar que o tamanho das células influência no tamanho do vetor final, que carrega as informações características da imagem em análise, ou seja, dependendo do valor atribuído a essa variável pode haver um excesso ou escassez de informações. A Figura 18 exemplifica as divisões da imagem original em células 8x8.

Figura 18 Divisão da imagem em células 8x8



Fonte: Mallick (2016)

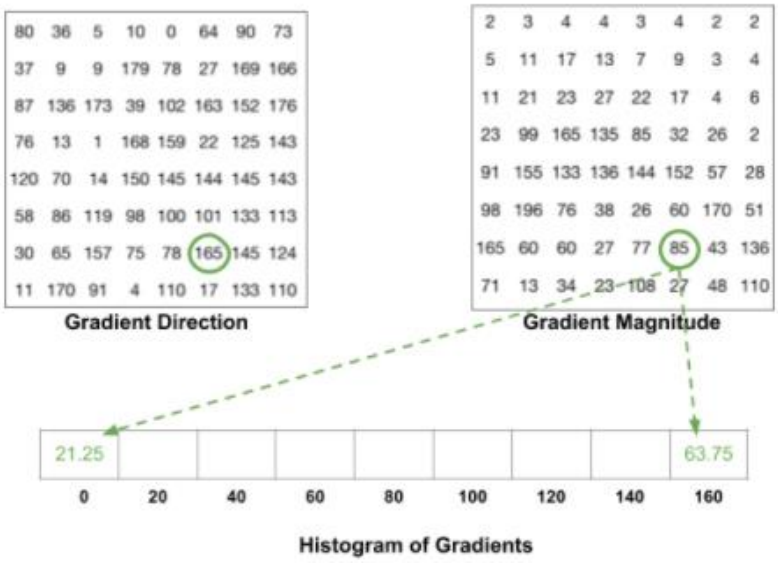
A terceira etapa, consiste em calcular o histograma do gradiente de cada célula. Primeiramente, é calculado o gradiente, desta forma, o algoritmo calcula a diferença entre o pixel da esquerda e da direita, processo também efetuado para os pixels verticais. Com esse processo finalizado, são encontrados dois valores, o primeiro representa a mudança no eixo x e o segundo no eixo y do pixel em análise. Com a obtenção desses valores, é possível utilizar as Equações 8 e 9 a seguir, para encontra-se o módulo e a orientação do gradiente.

$$m(x,y) = \sqrt{(gx(x,y)^2 + gy(x,y)^2)} \quad (8)$$

$$\theta(x,y) = \tan^{-1}\frac{gy(x,y)}{gx(x,y)} \quad (9)$$

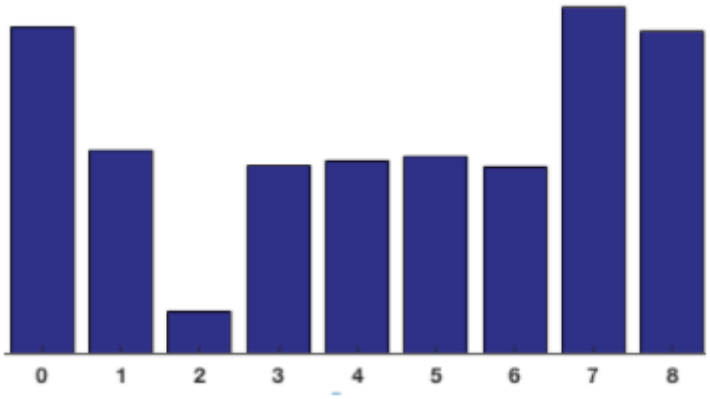
Posteriormente as etapas de cálculo referente ao módulo e orientação do gradiente, é construído o histograma, utilizando 9 – *bin*, ou seja, 9 divisões, sendo que cada divisão é essencialmente separada por ângulos de 20º, correspondente a direção do gradiente. Vale ressaltar que não há representações negativas, sendo assim, os ângulos variam entre 0º a 180º. A Figura 19 exemplifica o processo de construção de um histograma de gradientes e a Figura 20 exemplifica um histograma de 9-*bin.*

Figura 19 Construção do histograma de gradientes - HOG



Fonte: Mallick (2016)

Figura 20 9-bin histograma

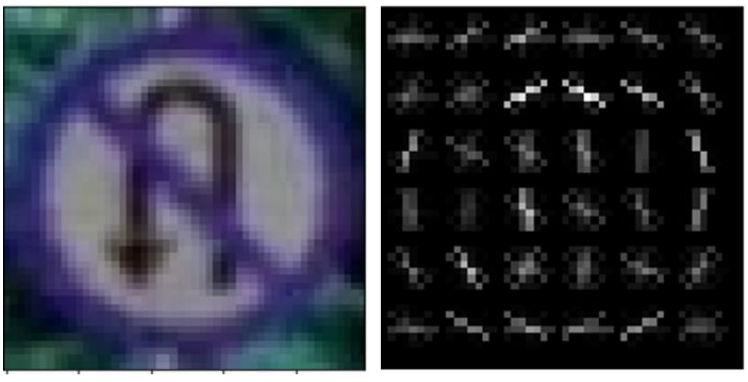


Fonte: Mallick (2016)

A última etapa consiste no processo de normalização de todas as células presentes na imagem, de forma tal que seja possível minimizar qualquer assimetria significante, uma vez que o valor resultante nos gradientes é influenciado pelas condições de luminosidade, podendo afetar a seleção dos objetos de interesse. Segundo Dalal e Triggs (2005), é mais vantajoso que o processo de normalização seja feito utilizando-se um grupo de células que possuam sobreposição entre elas.

Após estas quatro etapas, anteriormente descritas, o vetor HOG está finalizado e disponível para ser utilizado em um *pipeline* de aprendizado de máquina, para problemas de detecção de objetos. Na Figura 21 pode-se observar a comparação entre a imagem de entrada no processo de HOG e o resultado obtido.

Figura 21 Resultado do HOG em uma imagem



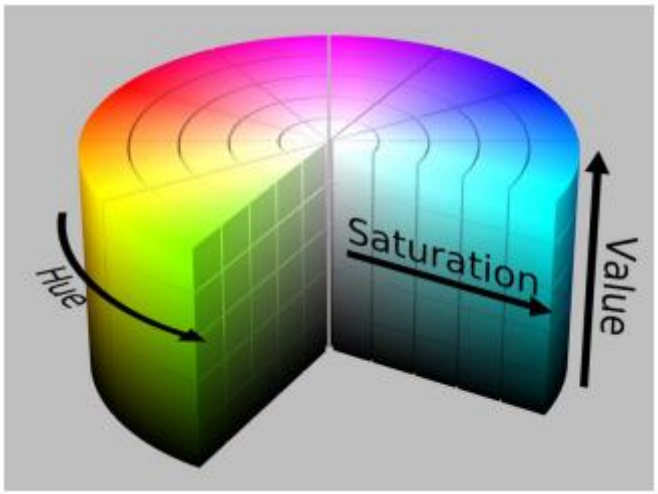
a) Original b) Resultado com HOG de células 8x8

Fonte: Autoria própria

1.8.3 HSV – *Hue, Saturation e Value*

Para efetuar a segmentação da imagem, levando em conspiração as cores padrões das placas de trânsito, é comumente utilizado o formatado HSV, pois esta formatação decompõe o espaço de cores em três: tonalidade, saturação e brilho. A tonalidade e a dimensão podem definir a cor em questão como vermelho, amarelo e azul. A saturação, por sua vez, define a pureza da cor, ou seja, quanto mais pura, mais tons de cinza ela apresenta. O brilho é utilizado para definir a proximidade da cor em questão para o branco, ou seja, quanto maior o brilho mais próximo do branco a mesma se encontra, no contrário, mais próximo do preto. Segue uma representação do espaço de cor HSV na Figura 22.

Figura 22 Descrição do espaço de cores HSV



Fonte: Duraes, Maciel e Barros (2018)

A aplicação do HSV é muito comum em processo de segmentação que se baseiam na cor dos objetos, pois este modelo de processo é menos custoso a nível computacional, em comparada a segmentações baseada na forma do objeto.

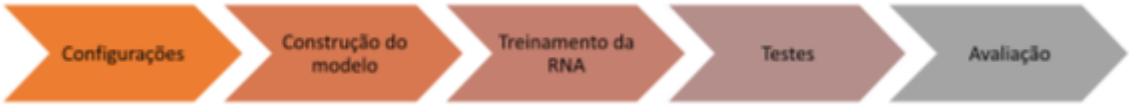
2 METODOLOGIA

O trabalho em questão, cujo objetivo busca realizar uma pesquisa exploratória sobre o material bibliográfico e de laboratório. Serão utilizados os procedimentos técnicos de pesquisa bibliográfica e experimental. Serão utilizados o método de abordagem hipotético-dedutivo e o método de procedimento em sua elaboração. Para coleta de dados será utilizada documentação indireta e observação direta, sobre a análise e interpretação de seus dados, qualitativos, ocorrerá de forma global.

Incialmente, serão realizadas pesquisas biográficas na área de processamento digital de imagens, redes neurais, sistemas de tomada de decisão aplicados a imagens, reconhecimento de objetos, tecnologias de hardware reprogramáveis e sistemas microprocessados. Estas pesquisas servirão para etapas de coletas de dados sobre as técnicas mais recentes e o estado da arte, de forma a viabilizar o desenvolvimento do projeto proposto.

A metodologia base proposta é representada na Figura 23:

Figura 23 Metodologia a ser adotada



Fonte: Autoria própria

A primeira etapa refere-se à definição, aquisição e configurações necessárias das ferramentas e bases a serem utilizadas para a composição do projeto, além da composição de um banco de dados que possuem itens suficientes para os testes e validações referentes ao projeto. Esta etapa destina-se também a criação do projeto referente ao sistema de detecção e identificação de imagens e sua integração.

A segunda etapa, dá-se pela construção e desenvolvimento direto do modelo de redes neurais, utilizando dos conhecimentos anteriormente adquiridos através das referências já expostas e ferramentas anteriormente listadas.

A terceira etapa, refere-se as fases de treinamento do modelo de RNA construído anteriormente, em que será utilizada um segmento da base geral para efetuar diversos

treinos no modelo e desta forma, para aperfeiçoamento por meio do correto ajuste de pesos inerente a cada sinapse.

A quarta etapa, refere-se a submissão de múltiplos testes ao modelo de RNA. Para este tópico adotar-se-á a metodologia de segmentar os testes pelos tipos e formatos de placas de sinalização de trânsito, de forma tal que possibilite uma melhor percepção sobre como as conexões estão sendo efetuadas e facilitando uma análise de melhoria.

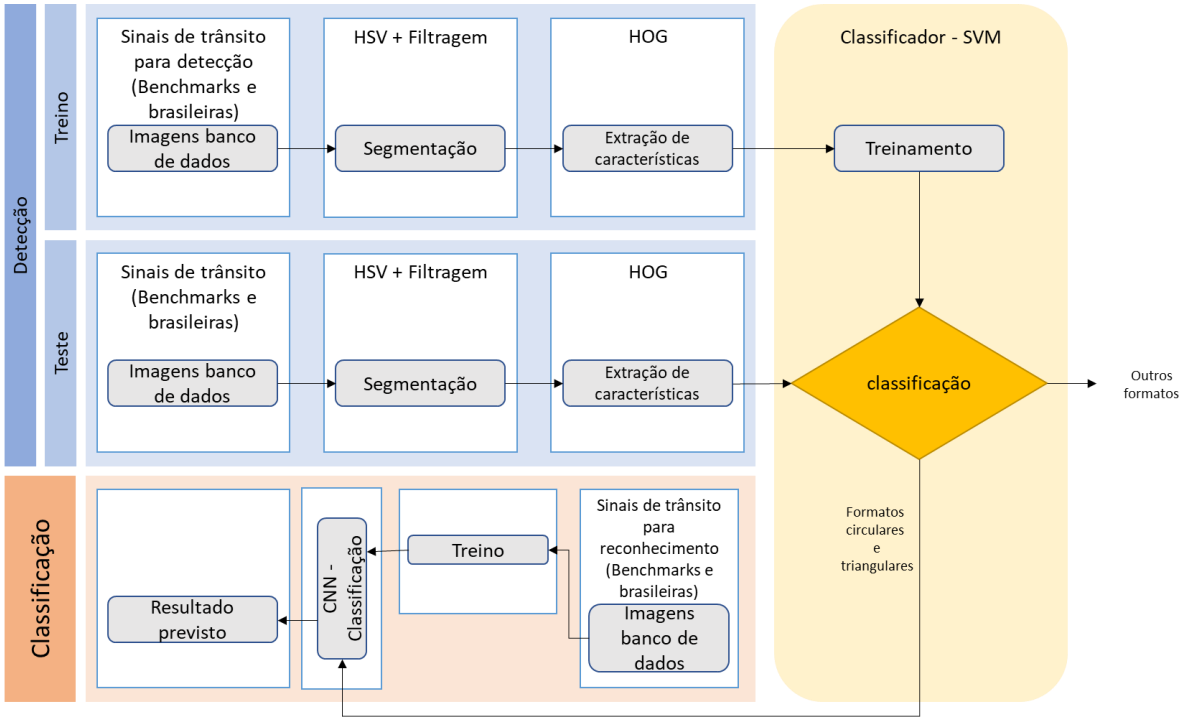
A última etapa, refere-se à avaliação dos resultados obtidos através do procedimento adotado, validando as hipóteses estipuladas para o desenvolvimento do processo em questão. Desta forma, os resultados serão discutidos, analisados entre si e comparados com métricas estipuladas em referências, para que seja possível encontrar meios mais favoráveis e pontos de melhorias.

2.1 FLUXO GERAL

Seguindo os conceitos expostos na fundamentação teórica e os métodos já conhecidos de detecção e classificação de objetos, dividiu-se o processo anteriormente proposto em cinco etapas de implementação, de forma mais específica: Banco de dados; Plataforma – Google Colab, Python e Dependências; Detecção; Configurações e Parâmetros; Classificação – Arquitetura de CNN.

Este fluxo geral, segmentado em etapas de detecção e classificação dos dados é representado na Figura 24 abaixo:

Figura 24 Fluxo Geral



Fonte: Autoria própria

2.2 BANCOS DE DADOS

Para o desenvolvimento desde trabalho, foram utilizadas bases de dados construídas com o propósito para situações de implementação destas em algoritmos de detecção e classificação de placas de sinalização de trânsito. De forma geral, os bancos de dados feitos para análises mais robustas em detecção, já possuem divisões entre os dados referentes a treino e teste, assim como arquivos com características de interesse da imagem para o problema a ser solucionado, no caso, as coordenadas de *boundboxing* com a região de interesse (ROI), tamanho da mesma e categoria pertencente. Já as bases que com foco no processo de classificação de imagem, também apresentam divisões entre os dados referentes a treino e teste, no entanto, as imagens que as compõe, em geral, estão recortadas, excluindo o ambiente em que a imagem foi obtida e apresentando apenas o objeto de interesse, no caso, as placas de sinalização de trânsito, assim como arquivos com características de interesse das imagens, no caso, tamanho da mesma e categoria pertencente.

Para as questões que envolvem detecção dos objetos de interesse, utilizaram as seguintes bases no estudo: *German Traffic Sign Detection Benchmark* - GTSDB, *Belgian Traffic Sign Dataset* - BelgiumTSD e *Brazilian Traffic Sign Detection* - BRTSD.

O GTSDB é um conjunto de dados que representa um dos *benchmarks* mais reconhecidos na literatura para implementação de algoritmos de detecção de sinais de trânsito, este possui cerca de 900 imagens com cenários de trânsito reais na Alemanha (sendo 600 de conjunto de treinamento e 300 para testes) e foi desenvolvido pela universidade de *Ruhr-Universität Bochum*. Segundo Houben et al. (2013), esta base possui sinais de trânsito que estão dispostos em sinais proibitórios, de perigo e mandatório – permitindo uma abordagem de acordo com a forma e cores.

O BelgiumTSD representa outro conjunto de dados bastante difundido como benchmark em situação de detecção, este possui cerca de 9006 imagens, com presença de 4565 placas de trânsito, sendo desenvolvido pelo Instituto Federal de Tecnologia de Zurique.

O BRTSD é conjunto de dados contendo 2112 cenários de trânsito brasileiro distintos, desenvolvida pela Universidade Federal do Rio Grande do Sul - UFRGS, a base apresenta mais de 3000 placas de sinalização de regulamentação e mais de 500 de sinalização de advertência. No entanto, esta base não apresenta anotações oficiais, portanto foi utilizada com a finalidade de exemplificação e não de gerar uma análise quantitativa, diferentemente das demais.

Para as questões que envolvem detecção dos objetos de interesse, utilizaram as seguintes bases no estudo: *German Traffic Sign Recognition Benchmark* - GTSRB, BelgiumTSD, para classificação, e Banco de Dados I – desenvolvido por Daniel Bitencourt em 2018.

O GTSRB é considerado um conjunto de *benchmark* de algoritmos de reconhecimento de sinais de trânsito, este possui cerca de 51840 imagens de placas de sinalização alemãs, em diferentes circunstâncias, sendo possível a identificação de 43 classes distintas. O GTSDRB foi desenvolvido pela universidade de *Ruhr-Universität Bochum*.

O BelgiumTSD, para classificação, corresponde ao conjunto de dados utilizando como *benchmark* para classificação de placas belgas de trânsito, foi criado em 2011, possuindo cerca de 9006 imagens, com presença de 4565 placas de trânsito, sendo desenvolvido pelo Instituto Federal de Tecnologia de Zurique.

Banco de Dados I – desenvolvido por Pereira (2018), na Universidade Federal de Santa Catarina - UFSC, é conjunto de dados que contém 475 imagens de placas de sinalização brasileira, sendo destas: 266 placas circulares, 184 em formato de losango e 25 placas triangulares. Cada categoria é dividida em grupos de treino e teste, que representam 80% e 20%, respectivamente, de cada uma das classes presentes, sendo: 11 classes de placas circulares, 6 de placas com forma de losango e 2 de placas triangulares.

Devido a inexistência de um banco de dados utilizado como benchmark para detecção e classificação de placas de sinalização brasileiras, assim como a inexistência de um banco de dados oficial e público, optou-se por utilizar para fins de análises quantitativas das técnicas propostas neste trabalho a utilização das bases de GTSRB, GTSDB e BelgiumTSD. Já que ambas as bases de detecção permitem a avaliação da performance e precisão do modelo de extração de ROIs e as bases de classificação proporcional treinamento e validação adequados para o modelo. Sendo assim, os bancos de dados com placas de sinalização brasileiras são utilização para exemplificar a possibilidade da aplicação do modelo de detecção e reconhecimento aqui proposto.

2.3 PLATAFORMA – GOOGLE COLAB, PYTHON E DEPENDÊNCIAS

A plataforma IDE utilizada para o desenvolvimento dos algoritmos de detecção e classificação das placas de trânsito foi o *Google Colab*. Segundo Gunawan (2018), o *Google Colab* corresponde a um ambiente semelhante ao *Jupyter Notebook*, de forma gratuita, sem a requisição de configurações prévias e inteiramente executável em serviços de nuvem, podendo ser eventualmente escalável, em versões pagas, para suportas análises mais robustas. Sendo todas estas possibilidades usufruídas através de um navegador web com conexão a internet, ainda sendo possível o acesso a dados diretores de repositórios no *Google Drive*, *GitHub* e *Kaggle*, por exemplo. Além de possibilitar a utilização das versões de *Python* e bibliotecas de aprendizados de máquina como *Keras* e *Tensorflow*.

A linguagem de programação adotada para o desenvolvimento dos algoritmos propostos foi Python. Segundo Raschka (2015), Python é uma das linguagens de programação mais populares para aplicações em ciência de dados, possuindo uma larga gama de bibliotecas já desenvolvidas para este tipo de aplicação. De acordo com Raschka (2015), apesar da linguagem em questão ter um desempenho inferior à linguagens de baixo nível, para execução de tarefas de computação intensiva, o Python possui bibliotecas como *OpenCV*, *Numpy* e *SciPy*, que foram desenvolvidas com base em

implementações em C, para operações rápidas e vetorizadas em matrizes multidimensionais. Para aplicações envolvendo aprendizado de máquina, de modo geral é aplicado o uso da biblioteca de *Scikit-Learn* e *TensorFlow*.

OpenCV (Open Source Computer Vision Library) representa uma biblioteca que possui funcionalidade em múltiplas plataformas e aplicações, onde sua principal funcionalidade é contribuir para o desenvolvimento de softwares. Para aplicações de processamento de imagens, essa biblioteca possibilita o processamento de filtros em imagens, detecção de objetos e aplicação no ciclo de aprendizagem de uma RNA.

Segundo Pedregosa et al. (2011), o Scikit-Learn aproveita no ambiente rico em desenvolvimento do python para fornecer implementações de última geração de muitos algoritmos de aprendizado de máquina, como SVM e outros modelos, de forma a manter uma interface fácil, prática e integrada ao Python.

TensorFlow e *Keras*. Segundo Vasilev et al. (2019), o *TensorFlow* é uma das mais populares bibliotecas de aprendizagem profunda. Desenvolvida pelo google, possui destaque por não fazer requerimento de uma GPU em específico para executar as atividades, esta faz uso de forma automática dos requerimentos disponíveis. Já o *Keras* é uma biblioteca que alto nível para redes neurais, usando *TensorFlow* como *backend*, pode performar rápidas e simples experimentações. Detecta automaticamente GPU disponível para uso.

2.4 DETECÇÃO

Quanto ao processo de detecção, de forma geral representa o processo e etapas de tratamento e pré-processamento das imagens que serão recebidas pelo modelo de detecção.

Primeiramente uma imagem de entrada em tons de zina apresenta a resolução de umas imagens de entrada de cinza como para uma resolução de 32 x 32.

Em segundo, representa a composição restrita a montagem do filtro inicial, ou kernel, que efetua a extração das primeiras características da imagem e reduz a resolução anterior da imagem.

Em terceiro, a função de *pooling*, que tem por objetivo criar uma matriz de menor resolução, portando as principais comuns das imagens de entrada. Esta também é responsável por criar varrições entre as imagens que são direcionadas para o modelo, de forma que havendo alguma imagem com a placa de trânsito deteriorada, com ruído na

imagem ou outra variação, ainda seja possível que o modelo a reconheça de forma aceitável.

Em quarto, a função de *flattening*, está é responsável por receber as características anteriores e dividi-las em grupos ainda menores, focando em aspectos e formas mais específicos, sendo estes transformados em vetores no processo seguinte.

Em quinto, as matrizes anteriormente obtidas são convertidas em vetores (SVM), informando características específicas de uma imagem e os direcionando para o processo de classificação.

2.4.1 Extratores de características e ROIS

Como exposto no item de Referencial Teórico, optou-se em utilizar o método de HOG, pois segundo Mallick (2016), este método extrator de característica possui a melhor taxa de acerto em relação aos demais métodos de extração cuja comparações foram efetuadas.

2.4.1.1 HOGs

Nas configurações do HOG é possível efetuar configurações que afetam a performance e desempenho em testes,

Como:

- *Orientations* = Definido como 8, representando o número de demarcações que se pretende criar.
- *Pixels_per_cell* = Definido como 4x4, representando o tamanho da célula da qual foi criada o histograma.

2.5 CONFIGURAÇÕES E PARÂMETROS DA CNN

Seguem alguns parâmetros e funcionalidades que são ajustados e necessários para a correta implementação dos modelos aqui propostos.

a) SGDM – *Stochastic Gradient Descent with Momentum*

O gradiente estocástico representa uma função que é responsável por minimizar a *loss function*, ou função de perda, para isso utiliza os pesos e vieses através da seguinte fórmula geral:

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) \quad (10)$$

Onde l é o número de iterações, α > 0 representa a taxa de aprendizado, $E(\theta)$ é a função de perda e $\nabla E(\theta)$ o gradiente da função de perda.

Segundo Zeiler (2012), a fórmula do SGD que leva em consideração o Momentum vem sendo utilizada com sucesso em aplicação de redes neurais, pois ao contrário da forma clássica, esta faz com que os valores da função de perda tendam a ir a uma única direção, evitando oscilações que depreciariam o modelo. Desta forma, segue a função modificada com a inclusão do Momentum.

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}) \quad (11)$$

b) InitialLearnRate

Este corresponde ao parâmetro que representa o quanto o peso da rede pode ser ajustado em cada iteração.

c) LearnRateSchedule

Corresponde ao parâmetro que efetua a atualização da taxa e aprendizado em determinado período. A fórmula adotada foi:

$$r * \left(0{,}1^{\frac{epoch}{10}}\right) \quad (12)$$

Onde lr, é a taxa de *Learning_Rate* inicial, em 0,01 e *epoch* é a iteração atual.

d) LearnRateDropFactor

Representa o número pré-determinado que multiplica o fator de aprendizado num período escolhido

e) LearnRateDropPeriod

Representa o período em que a taxa de aprendizado é atualizada

f) L2Regularization

É a função responsável por efetuar a regulação dos pesos da *loss function*, ou função de perda, de forma que haja a redução do *overfitting*. Sendo assim, há uma modificação na função de perda para incluir a regulação:

$$r(\theta) = E(\theta) + \gamma\left(\frac{1}{2}\right)w^T w \quad (13)$$

Onde w represento peso atribuído e o y o coeficiente de regulação

g) Epochs

Representa o período em que uma rede que possui *backpropagation* repassa todos os dados recebidos na entrada ao fim, e efetua o processo inverso, levando-os do fim as camadas de entrada.

h) Interactions

Representa o número de vezes em que as imagens de treinamento percorrem a rede em ambos os sentidos. Sendo cada vez que é percorrido em ambos os sentidos considerado uma iteração.

2.6 CLASSIFICAÇÃO – ARQUITETURA CNN

Seguindo os parâmetros e fluxo de redes neurais convolucionais apresentadas anteriormente, optou-se se efetuar teste com as seguintes propostas de rede presentes na Tabela 1.

Tabela 1 Modelos de CNN propostas

A	B	C	D
Camada de entrada			
Conv - 32	Conv – 32	Conv – 32	Conv - 64
ReLU			
MaxPool			
Conv - 32	Conv – 32	Conv – 32	Conv - 128
ReLU			
MaxPool			
Conv - 64	Conv – 32	Conv – 32	
ReLU			
MaxPool			
Conv - 64	Conv – 32		
ReLU			
MaxPool			
Conv - 128			
ReLU			
MaxPool			
Conv - 128			
ReLU			
MaxPool			
Completamento conectada			
ReLU			
Completamente conectada			
SoftMax			
Camada de Classificação			

Fonte: Autoria própria

Os resultados quanto a performance e serão discutidos no item de Resultados, todos os modelos de Redes Neurais propostas foram testados utilizando a base de dados GTSRB, para fins de um *benchmark* confiável, no entanto processo foi também aplicada a base de dados com placas brasileiras.

3 IMPLEMENTAÇÃO

Nesta seção serão demonstrados os processos anteriormente expostos na metodologia sendo implementados em prática com as bases já citadas.

3.1 PRÉ-PROCESSAMENTO, DETECÇÃO DE IMAGEM E EXTRAÇÃO DOS PONTOS DE INTERESSE

A primeira etapa consiste em efetuar um pré-processamento nas imagens do banco de dados, de forma tal que seja possível distinguir as placas de sinalização de trânsito do ambiente ao redor e posteriormente selecionar apenas a região de interesse onde ela se encontra. Levando quem consideração que a imagem de entrada é uma informação RGB, sendo bastante sensível a questões de luminosidade, a imagem de entrada em RGB é convertida em HSV, pois para este formato é irrelevante as diferentes condições de luminosidade presentes na captura, de forma a minimizar os problemas do modelo de conhecimento quanto a luminosidade. O formato HSV possui três componentes de cores: a tonalidade (H), informa a cor; a saturação (S), descreve a pureza da cor e o valor (V), que representa a brilho da cor.

Seguem os cálculos aplicados para obtenção das grandezas dos componentes HSV:

$$\max = \sup(R,G,B)\,, \min = \inf(R,G,B)$$

$$V = max$$

$$S = \begin{cases} (max - \min)/\max\,, & se\ \max \neq 0 \\ 0.\ caso\ contrário \end{cases}$$

$$H = \begin{cases} 0º, se\ \max = min \\ 60º * \dfrac{G-B}{\max - min} + 0º\,, se\ R = max \\ 60º * \dfrac{B-R}{\max - min} + 120º\,, se\ G = max \\ 60º * \dfrac{R-G}{\max - min} + 240º\,, se\ B = max \end{cases}$$

$$H = H + 360\ º, se\ H < 0$$

(14)

A segunda etapa consiste na aplicação de filtros na imagem, onde são utilizadas as técnicas de limiar para detectar as cores das placas e sinalização de trânsito, sendo o

procedimento mais importante referente a determinação do valor atribuído ao limiar. O principal objetivo dessa etapa é eliminar os falsos candidatos que foram erroneamente detectados como placas de trânsito pela limiarização, para isso é efetuado o processo de segmentação e filtragem, aplicando operações de erosão e dilatação a imagem segmentada. Para melhorar os resultados da segmentação, é definido um tamanho aceitável para delimitação de objetos de interesse, caso a região detectada gere maior área considerada erro, e portanto, eliminada

A terceira etapa consiste na extração de características das imagens através das regiões de interesse, neste trabalho foi utilizado a técnica de HOG, anteriormente explanada. Para calcular os recursos do HOG, o ambiente presente na imagem é subdividido em pequenas regiões, células, seguindo do agrupamento destas células em blocos e normalização dos blocos, evitando problemas de variância das iluminações. Posteriormente, as regiões detectadas são normalizadas em tamanho de 36x 36 pixels, divididas em blocos 8x8. Em seguidas essas características extraídas são direcionadas para o treinamento e subsequente teste, de uma camada de SVM.

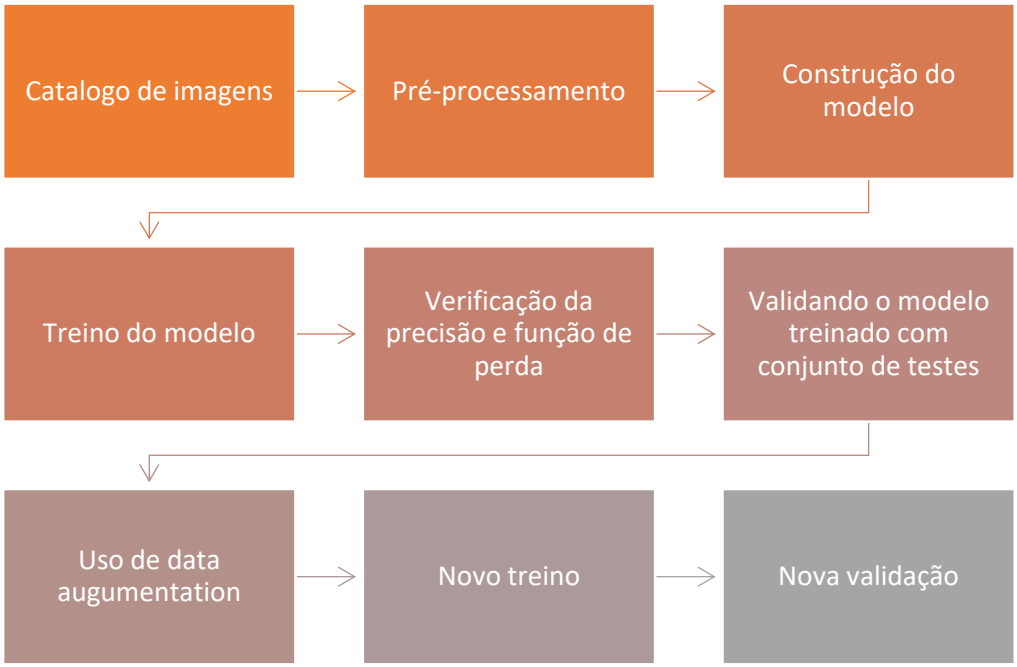
A quarta etapa e última do processo de detecção refere-se a utilização da técnica de SVM, onde é focado o reconhecimento da forma geométrica obtidos no processo anterior de segmentação. Desta forma, uma vez que os vetores de características são calculados e informados através da HOGs, estes são usados no método de SVM. Este método é de interessante aplicação pois possui uma robustez a diversos fatores como translações, rotações e escala. O método SVM buscas encontrar um hiperplano ótimo para separar de forma mais distinta possível as classes em questão, assumindo que são linearmente separáveis. O SVM tem a função de classificador binário, mas pode ser utilizado para resolver problema multiclasse, através da combinação de diversos SVMs binários. Quando a aplicação é referente a problemas de multiclasses, há quatro tipos de SVMs: um contra todos; em pares; com código de saída de correção de erro e todos de uma vez. De forma geral, o que apresenta melhor performance quando aplicado em situações de multiclasse é o “um contra um”. Nesta forma, utilizou-se um classificador SVM linear de forma que este diferencie o formato da placa de trânsito recebida entre: circular, triangular, outros.

Ressaltando que os códigos para desenvolvimento de tais etapas estão presentes no Apêndice A.

3.2 CLASSIFICAÇÃO – CNN

Após a etapa de identificação do formato da placa de sinalização, é utilizado um modelo de Redes Neurais Convolucionais para efetuar a corretar detecção dos sinais. Como exposta na metodologia, o modelo proposto é o correspondente na Figura 25 abaixo, ressaltando que os códigos para desenvolvimento de tais etapas estão presentes no Apêndice B:

Figura 25 Fluxo de implementação da CNN



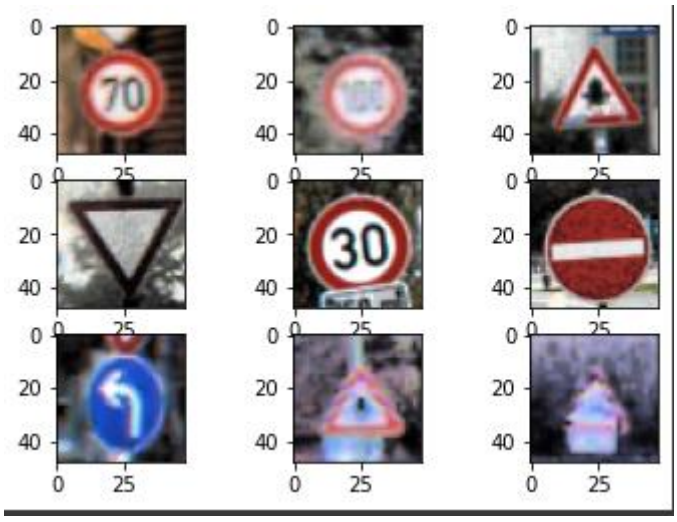
Fonte: Autoria própria

A primeira etapa consiste em realizar a separação das imagens presentes no banco de dados em classes e em pastas distintas de treino e teste para o modelo. Desta forma o mesmo passa a ter material para efetuar processos de treino e validação de um modelo de CNN. Como exposto anteriormente, é necessário que haja esse tipo de separar dos dados a serem utilizados de tal forma que seja possível evitar problema de *overfitting* e *underfitting*.

A segunda etapa consiste em realizar os pré-processamentos necessário nas imagens que serão utilizadas. Primeiramente, ocorre a conversão do formato RGB para HSV, seguindo da equalização do histograma dela. Em segundo lugar, a imagem possui seu tamanho reajustado, inicialmente para 48x48, que é o tamanho adotado para a camada

de entrada do modelo. Em terceiro lugar, são obtidas as informações das classes correspondentes a cada imagem e vetorizadas para utilização posterior do modelo. A Figura 26 expõe algumas imagens após etapa de pré-processamento.

Figura 26 Algumas imagens da base GTSRB após pré-processamento



Fonte: Autoria própria

A terceira etapa consiste na construção do modelo propriamente dito, para isso, usou das bibliotecas *keras* e *tensorflow*. Sendo assim, utilizam-se das funções destas bibliotecas para a criação da estrutura da CNN, seguindo os modelos expostos na metodologia, de forma geral, seguem os seguintes estágios: Camada de entrada, Camadas de convolução, Ativação ReLU, Camadas de Max Pooling, Camada totalmente conectada, Ativação Relu, Camada totalmente conectada, Camada de SoftMax.

Alguns dos parâmetros adotados na construção do modelo: *InitialLearnRate* = 0,01; A função de SGD foi adotado para minimizar a função de perda, sob uma taxa de decaimento de 10^{-6}, momentum = 0,9. Além disso o parâmetro de *LerningRateScheduler* foi utilizado como exposto no item anterior. Aqui ainda foram iniciadas para observação as métricas de *accuracy*, ou precisão, do modelo e *loss function*, ou função de perda, sendo aqui utilizada a *categorical crossentropy*.

A quarta etapa consiste em efetuar o treino do modelo anteriormente construído, neste estágio, o conjunto de dados referente a treino é adicionado ao modelo, de forma tal que este passe a conhecer os dados que serão trabalhados, num certo nível de semelhança com os de teste, sendo assim, o modelo efetua os ajustes de pesos e melhoras as conexões e predições a cada iteração com os dados de treino.

A quinta etapa consiste em verificar a precisão do modelo e valores da função de perda para cada iteração. É esse estágio que se tem uma avaliação mais visual do progresso de aprendizagem do modelo de acordo com cada iteração feita pelo mesmo. Os modelos propostos para teste foram avaliados nessa etapa, assim como o tempo médio de execução.

A sexta etapa consiste em avaliar o modelo treinado com base no conjunto de dados referente a teste. Neste estágio, o modelo já treinado é utilizado para predizer as classes do conjunto de testes e posteriormente ter a taxa de acerto avaliada.

A sétima etapa consiste na tentativa de melhorar o cenário de treino do modelo, de tal forma, que este possua um treinamento maior para ocasiões de aquisição diversas. Para tal, foi utilizada a função de *DataAugmentation* nos dados de treino, ela representa uma técnica usada para aumentar a quantidade de dados através da criação de cópias ligeiramente modificadas dos dados já presentes no conjunto de dados, esta função tem que a atribuição de agir como um regularizador, ajudando a reduzir o *overfitting* num treino do modelo. Pelo fato de também apresentar mais dados, é necessário que haja um maior número de iterações para melhores ajustes.

A oitava etapa consiste em efetuar um novo treino do modelo, com a base de dados já tendo passado pelo processo de *Augmentation* na etapa anterior.

A nona etapa consiste em reavaliar o modelo com base na curva de precisão do modelo e valores da função de perda para cada iteração. Desde forma, verificando se houve melhora com a aplicação da técnica de *Augmentation*.

4 RESULTADOS

Neste tópico são expostos os resultados seguindo a metodologia e implantação descritos anteriormente.

4.1 RESULTADOS DE DETECÇÃO

O modelo de detecção utilizando SVM proposto na metodologia e exposto no Apêndice A, foi testado utilizando a base de dados BelgiumTSD para nível de benchmarks, devido ao grande número de imagens e placas presentes na mesma, sendo que é possível e exemplificado a universalização e aplicação do mesmo a placas brasileiras contanto que haja base de treinamento adequado, uma vez que as formas e cores entre o modelo brasileiro e o belga, apresentam similaridades.

Seguem os resultados para ambientes de treino de teste no modelo SVM estão expostos na Tabela 2, a seguir.

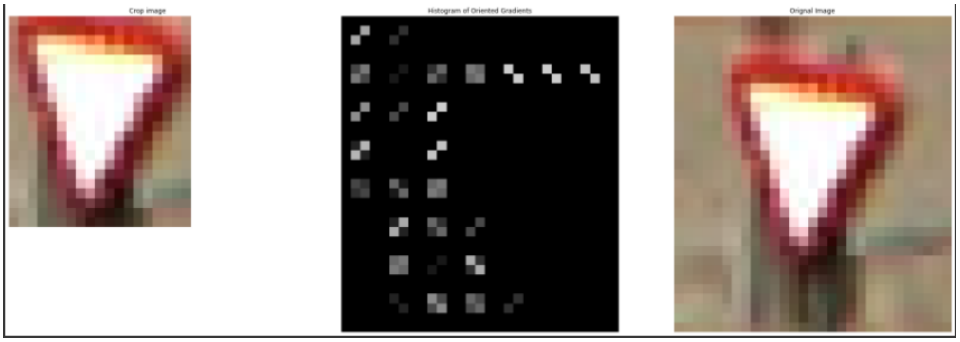
Tabela 2 Resultados modelo de SVM

Modelo	**Precisão**
Treino	0,799
Teste	0,672

Fonte: Autoria própria

Segue um exemplo de saída do modelo de SVM, na figura 27, com a imagem real apresentando plano de fundo e características de ambiente, é utilizando de etapas de segmentação, usando técnicas de HOG, para obtenção da região de interesse, resultando na imagem recortada apenas com a placa de trânsito na saída do modelo.

Figura 27 Resultado de saída do modelo de SVM - exemplo



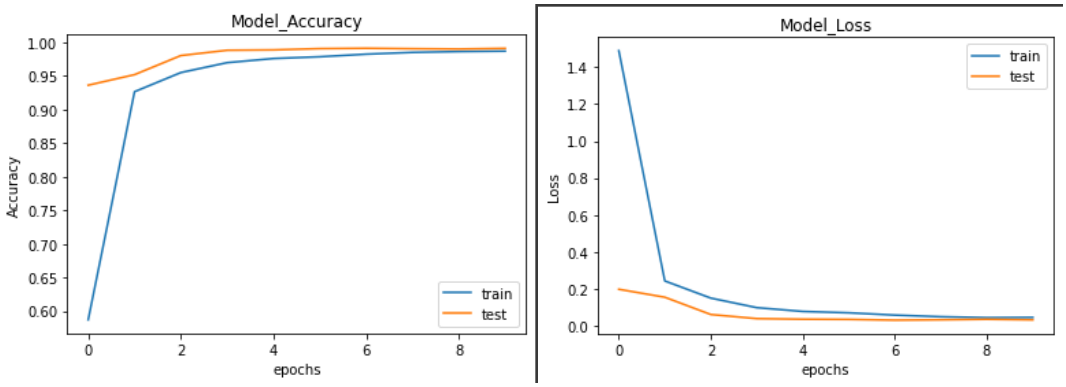
Fonte: Autoria própria

4.2 RESULTADOS DE CLASSIFICAÇÃO

4.2.1 Redes Neurais propostas

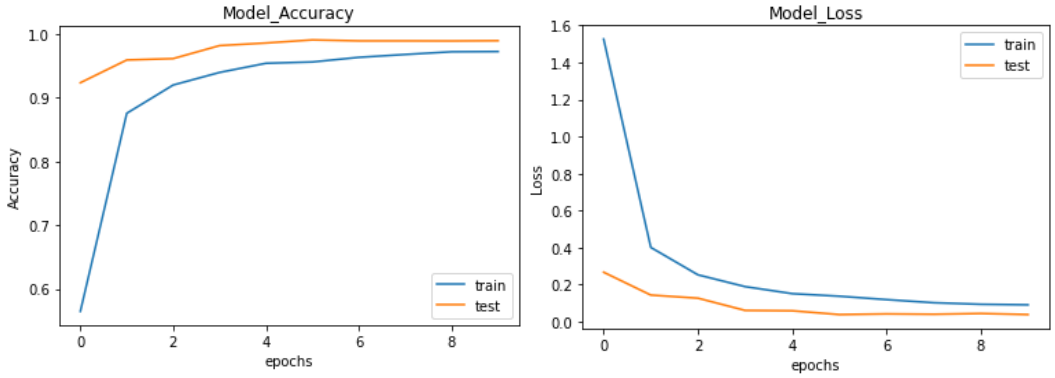
Seguem nas Figuras 28, 29, 30 e 31 os resultados das curvas de precisão e função de perda para cada modelo proposto na metodologia.

Figura 28 Treinamento RNC A



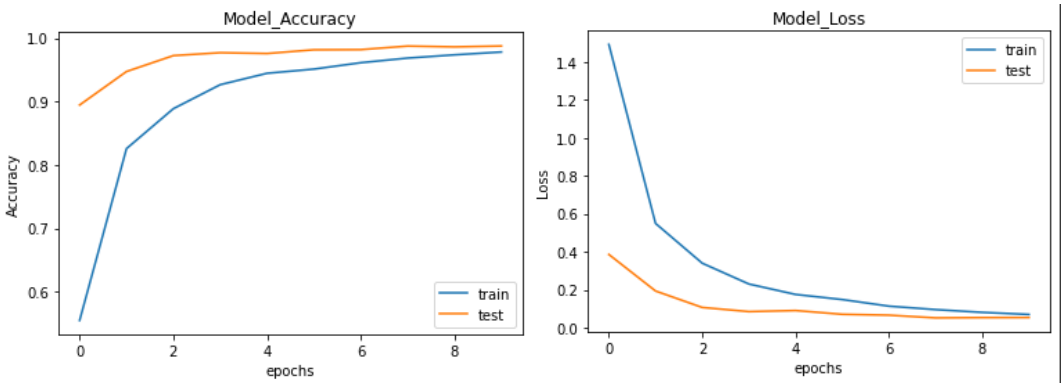
Fonte: Autoria própria

Figura 29 Treinamento RNC B



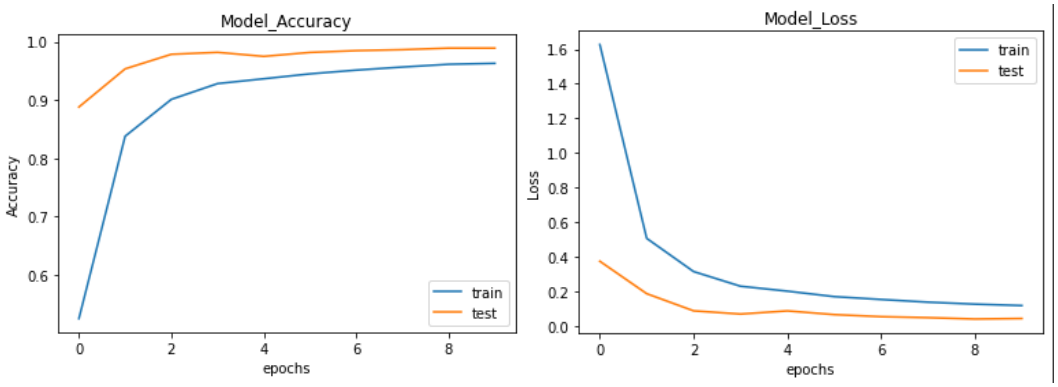
Fonte: Autoria própria

Figura 30 Treinamento RNC C



Fonte: Autoria própria

Figura 31 Treinamento RNC C



Fonte: Autoria própria

Com os resultados obtidos é possível montar uma tabela com as principais características apresentadas pelos diferentes modelos em ambiente de treinamento, dados são expostos na Tabela 3.

Tabela 3 Resultados do treinamento dos modelos propostos

Rede	Accuracy	Loss	Val_accuracy	Val_loss	Tempo médio por epoch (s)
A	0,9870	0,0453	0,9911	0,0322	400
B	0,9724	0,0905	0,9894	0,0380	255
C	0,9780	0,0692	0,9875	0,0529	700
D	0,9631	0,1165	0,9892	0,0416	200

Fonte: Autoria própria

Observa-se que a Rede A, que possui mais camadas e aplicação de filtros, apresentou um valor de precisão próximo de 99%, em detrimento de um tempo de treinamento total de 1 hora e 6 minutos. Já a Rede B, com duas camadas convolução a menos que A, apresentou uma precisão de aproximadamente 97% num tempo total de 42 minutos. Por sua vez, a Rede C, com apenas duas camadas de convolução e aumento significativo na quantidade de filtros, apresentou uma precisão maior que C, no entanto, possuiu o maior tempo de treinamento, totalizando cerca de 1 hora e 56 minutos. Por fim, a Rede D, apresentou o pior índice de precisão dos modelos propostos, no entanto, foi o mais rápido no período de treino, cerca de 33 minutos.

Levando em conta os valores e discussões apresentados anteriormente, optou-se por seguir com o modelo proposto na Rede A, considerando o como melhor custo-benefício entre tempo de treino, precisão, aplicação de filtros e função de perda.

4.2.2 Aplicação da Rede Neural escolhida

Ainda utilizando-se o conjunto de dados GTSRB, obtiveram-se os seguintes resultados de classificação da CNN, mostrada na matriz de confusão presente na Tabela 4. Performance geral de teste do modelo:

Tabela 4 Resultado Rede A aplicada a conjunto de teste

Rede	Accuracy	Loss	Tempo médio (s)
A	0,9926	0,032	64

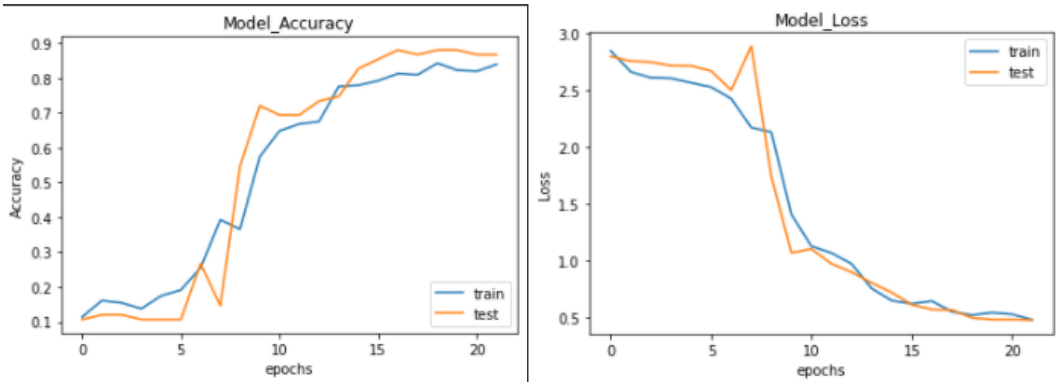
Fonte: Autoria própria

Para fim de demonstração de aplicação num conjunto de dados com placas brasileiras de trânsito, como exposto na Figura 32, utilizando-se da base de dados desenvolvida por Pereira (2018), na UFSC. A Rede A proposta foi treinada com um valor de 22 iterações, apresentando a seguinte curva de precisão e de função de perda na Figura 33.

Figura 32 Algumas amostras de imagens do conjunto de dados BR

Fonte: Autoria própria

Figura 33 Treinamento RNC A - conjunto BR



Fonte: Autoria própria

Quando aplicada aos dados com placas de trânsito brasileira, a Rede A apresentou a seguinte performance de treino, exposta da Tabela 5.

Tabela 5 Resultado do treinamento em conjunto de dados BR

Rede	Accuracy	Loss	Val_accuracy	Val_loss	Tempo médio (s)
A	0,8389	0,4771	0,8667	0,4753	4

Fonte: Autoria própria

Utilizando-se o conjunto de dados referente a testes com placas brasileiras de trânsito, obtiveram-se os seguintes resultados de classificação da CNN. Performance geral de teste do modelo é apresentada na Tabela 6:

Tabela 6 Resultados do modelo em conjunto de dados BR - teste

Rede	**Accuracy**	**Loss**	**Tempo médio por epoch (s)**
A	0,8529	0,4839	1

Fonte: Autoria própria

CONCLUSÃO

O trabalho em questão teve o intuito de desenvolver um sistema de detecção e identificação automática de placas brasileiras de sinalização de trânsito, aplicadas ao auxílio à direção veicular, de forma a utilizar técnicas de processamento digital de imagens.

Foi apresentado um breve referencial teórico sobre os principais conceitos utilizados para o desenvolvimento da pesquisa e implantação do modelo proposto como: o ADAS e suas necessidades para melhoria da segurança dos motoristas; das placas de trânsito brasileiras segundo as normas regulamentadoras nacionais; assim como também foram apresentados os conceitos de imagem e as principais técnicas de processamento digital de imagem; os conceitos de redes neurais artificias e SVM, assim como suas aplicações; conceito de *deep learning* e como as técnicas do mesmo podem ser utilizadas para reconhecimento de imagens e por fim, de técnicas de detecção e extração de objetos de interesse.

O modelo de RCNN proposto, seguiu do princípio da utilização de bases já estabelecidas para níveis de comparação com benchmarks, chegando a uma taxa de precisão de 99,26 % para base de GSTRB e para base de dados de placas brasileiras o valor de 85,29 %, sendo considerado satisfatório para o conjunto de dados e condições já exploradas no decorrer do trabalho. O modelo optado foi um RCNN, que por sua fez é desenvolvido em duas etapas, uma etapa focada em detecção de placas de trânsito, utilizando técnicas de processamento de imagem e modelo de SVM para efetuar a extração da zona de interesse, obtendo uma precisão da taxa de 67,2 % em ambiente de teste, para o conjunto de dados utilizado (BelgiumTSD), a segunda etapa consiste em utilizar o resultado proveniente do modelo de SVM para efetuar a classificação dos sinais de trânsito, seguindo um modelo de Rede Neural Convolucional clássico cuja precisão já foi abordada.

Avaliando os resultados obtidos e comparando os com os objetivos definidos no início do projeto, pode-se considerá-los satisfatório, uma que foi possível:

- Desenvolver um modelo de detecção e classificação conjunta;
- Obter um resultado comparável com os benchmarks tradicionais, validando a eficácia do modelo proposto;
- Utilizar dados de placas brasileiras de trânsito para treino de teste do modelo;

- Obter resultados satisfatórios de predição para placas de sinalização de trânsito brasileiras, uma vez que não há uma base de benchmark padrão para este tipo de conjunto de dados.

Por fim, sugere-se a continuidade deste trabalho, com a finalidade de inicialmente gerar uma base de dados obtidas em necessários de trânsito real, de forma que possa ser utilizada para futuros benchmarks, e efetuar a classificação dos dados presentes na mesma, a fim de que seja possível ter uma maior confiabilidade na aplicação do modelo cenários de placas de trânsito brasileiras. Outra possibilidade, seria a adaptação do código e técnicas aqui propostas para a criação de um sistema embarcado, que efetuaria a detecção de forma automática e em tempo real, obtendo a informação de uma câmera e exibindo o resultado em uma tela.

REFERÊNCIAS

AKANDE, A. **Performance comparison of svm and ann inpredicting compressive strength of concrete**. v. 16, p. 88-94, 2014.

BROOKHUIS, Karel A.; DE WAARD, Dick; JANSSEN, Wiel H. Behavioural impacts of Advanced Driver Assistance Systems–an overview. **European Journal of Transport and Infrastructure Research**, [S.l.], v. 1, n. 3, junho 2001. ISSN 1567-7141.

CARVALHO, André Ponde de Leon. **Redes Neurais Artificiais**. 2009. Disponível em: https://sites.icmc.usp.br/andre/research/neural/#:~:text=Redes%20Neurais%20Artificiais%20s%C3%A3o%20t%C3%A9cnicas,adquirem%20conhecimento%20atrav%C3%A9s%20da%20experi%C3%AAncia. Acesso em:20/09/2020.

CHITNIS, Kedar; STASZEWSKI, Roman; AGARWAL, Gaurav. **TI Vision SDK Optimized Vision Libraries for ADAS Systems**. Texas Instruments, 2014.

COPELAND, B. R. **Is Free Trade Good for the Environment?**. The American Economic Review, 2016

CONTRAN. **Manual Brasileiro de Sinalização de Trânsito – Volume 1 – Sinalização Vertical de Regulamentação**. 2005. Disponível em: http://vias-seguras.com/documentos/documentos_temas_o_a_z/doc_sinalizacao_e_seguranca_do_transito/manual_brasileiro_de_sinalizacao_de_transito_volume_i. Acesso em: 19/09/2020.

CONTRAN. **Manual Brasileiro de Sinalização de Trânsito – Volume 2 – Sinalização Vertical de Advertência**. 2007. Disponível em: https://www.mobilize.org.br/midias/pesquisas/manual-brasileiro-de-sinalizacao-vol-ii.pdf. Acesso em: 19/09/2020.

DALAL, N.; TRIGGS, B**. Histograms of oriented gradients for human detection**. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. v. 1, p. 886-893 vol. 1. ISSN 1063-6919.

DENATRAN. **Conselho Nacional De Trânsito - Código de Trânsito Brasileiro**. 2008. Disponível em: https://goo.gl/M2KkMC. Acessado em 19/09/2020.

DURAIS, Thiago de Jesus Oliveira.; MACIEL, Lucas Soares; BARROS, Wagner Ferreira de. **Processamento Digital de Imagens Aplicado a Identeificação Automática de Placas de Trânsito**. Revista CEREUS, 2018. ISBN 21757275.

FILHO, Ogê Marques; NETO, Hugo Vieita; **Processamento Digital de Imagens**. Rio de Janeiro, Brasport, 1999. ISBN 8574520098.

GIRSHICK, R. et al. **Rich feature hierarchies for accurate object detection and semantic segmentation**. United States, 2014.

GOODFELLOW, I; BENGIO, Y; COURVILLE, A. **Deep Learning**: Adaptive Computation and Machine Learning series. Cambridge, MA: MIT Press: 2016.

GRACE, K., SALVATIER, J., DAFOE, A., ZHANG, B., EVANS, O. **When Will AI Exceed Human Performance?**. Evidence from AI Experts., 2018

GUIMARÃES, Marly. Amostragem e Quantização. **Digital Signal Processing,** Manaus, jun. 2008.

GUNAWAN, Teddy S; ASHRAF, Arselan; RIZA, Bob Subhan; HARYANTO, Edy Victor; ROSNELLY, Rika; MARTIWI, Mira; JANIN, Zuriati. **Development of vídeo-based emotion recognition using deep learning with google colab**. TELKOMNIKA Telecommunication, Computing, Electronics and Control. Vol 18, No 5, Outubro de 2020, pp 2463-2471. ISSN: 16936930

HAYKIN, S. **Neural networks and learning machines**, 3rd ed. Ontario, Canada: Pearson, 2009.

HOUBEN, Sebastian; STALLKAMP, Johannes; SALMEN, Jan; SCHLIPSING, Marc; IGEL, Christian. **Detection of traffic signs in real-world images: the german traffic sign detection benchmark**. INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'13), 2013, Dallas. Proceedings. Piscataway: IEEE, 2013. p.1–8

HSU, C.-W.; CHANG, C.-C.; LIN, C.-J. **A practical guide to support vector classification**. v. 101, p. 1396-1400, 2003.

MALLICK, S. **Histogram of Oriented Gradients**. 2016. [Online; Acessado em 15 de dezembro de 2021]. Disponível em: hhttp://www.learnopencv.com/histogram-of-oriented-gradients

MATSUI, Takashi; CUTURI, Marco; VERT, J-P; BIRKENES, Oystein.. **A kernel for time series based on global alignments**. In Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on, volume 2, pages II–413. IEEE, 2007

OLIVEIRA, Wellington. **Software para reconhecimento de espécies florestais a partir de imagens digitais de madeiras utilizando deep learning.** Universidade Tecnologica Federal do Parana, 2018.

OSÓRIO, Fernando; BITTENCOURT, João Ricardo; **Sistemas Inteligentes baseado em Redes Neurais Artificiais aplicados ao Processamento de Imagens**. 2000. Disponível em:https://www.researchgate.net/profile/Fernando_Osorio2/publication/228588719_Sistemas_Inteligentes_baseados_em_redes_neurais_artificiais_aplicados_ao_processamento_de_imagens/links/0912f51001cc71ad2b000000/Sistemas-Inteligentes-baseados-em-redes-neurais-artificiais-aplicados-ao-processamento-de-imagens.pdf. Acesso em: 15/09/2020.

PACHECO, A. **Redes neurais convolutivas**. Brasil, 2017.

PEDREGOSA, Fabian; VAROGUAUX, Gaël; GRAMFORT, Alexandre; MICHEL, Vincent; THIRION, Bertrand. **Scikit-lern: Machine Learning in Python**. Journal of Machine Learning Research 12. 2011

PEREIRA, Daniel Bitencourt. **Análise e Implementação de Sistema Para Detecção de Placas Brasileiras de Sinalização de Trânsito**. Brasil, 2018

PRETO, Daniela de Oliveira. **Reconhecimento De Placas De Trânsito Por Meio De Deep Learning**. Brasil, 2018

RASCHKA, Sebastian. **Python Machine Learning: Unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics**. PACKT Publishing. 2015. 13 p. ISBN 9781783555130.

REIF, K. **Brakes, Brake Control and Driver Assistance Systems**. ViewegTeubner Verlag, 2014. ISBN 3658039779.

REZAEI, M.; KLETTE, R. **Computer Vision for Driver Assistance**. Springer-Verlag GmbH, 2017. ISBN 3319505491.

RUSSEL, S.; NOVIG, P. **Artificial Inteligence: A Modern Approach**. 2a. edição. Prentice Hall, 2003.

SINGH, S. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. **National Highway Traffic Safety Administration**, n. February, p. 12, 2015.

VASILEV, Ivan; SLATER, Daneil; SPACAGNA, Gianmario; ROELANTS, Peter; ZOXXa, Valentino. **Pyhton Deep Learning**. Packt Publishing, Ed 2 , 2019. ISBN 9781789349702

WHO. Global status report on road safety 2015. **WHO Library Cataloguing-in-Publication Data Global**, p. 340, 2015.

WINNER, H. et al. **Handbook of Driver Assistance Systems**. Springer-Verlag GmbH, 2015. 1602 p. ISBN 3319123513.

ZIELER, Matthee D. **ADADELTA: NA ADAPTATIVE LEARNING RATE METHOD**. USA. 2012

APÊNDICE A – ALGORITMO DE DETECÇÃO

```
import glob
import pandas as pd
import cv2
import os
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from skimage.feature import hog
from sklearn.decomposition import PCA
from sklearn.svm import SVC
import joblib
import tensorflow as tf

Test_Images_Directory ='Testing'
Training_Images_Directory ='Training'

#Le cada arquivo csv com características das imagens e os concatena
num único arquivo
csv_files_training=glob.glob(Training_Images_Directory+'/**/*.csv',
recursive=True)
main_Training=pd.read_csv(texto,sep=';',header=None)
for i in range(1,len(csv_files_training)):
    new_doc=pd.read_csv(csv_files_training[i],sep=';')
    main_Training=main_Training.append(new_doc, ignore_index=True)
main_Training

#Aplicação de HOG - seleciona a ROI de interesse, salva as
coorenadas
def images_to_hog(main,Images_Directory):
    Features=[]
    Labels=[]
    for i in range(0,len(main)):
        img_path=Images_Directory+'/00000'[:-
len(str(main['ClassId'][i]))]+str(main['ClassId'][i])+'/'+main['Fil
ename'][i]
        img = cv2.imread(img_path)
        img=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img = cv2.medianBlur(img,3)
        crop_image=img[main['Roi.Y1'][i]:main['Roi.X2'][i],main['Ro
i.X1'][i]:main['Roi.Y2'][i]]
        crop_image=cv2.resize(crop_image, (64, 64)) #modifica o
tamanho para 64*64.
        # Aplica a biblioteca de HOG do skimage para obter a imagem
recortada, ou coordenadas da HOG
```

```
        ret,crop_image = cv2.threshold(crop_image,127,255,cv2.THRES
H_BINARY)
        descriptor = hog(crop_image, orientations=8,pixels_per_cell
=(4,4))
        Features.append(descriptor)#Salva as features da HOG
        Labels.append(main['ClassId'][i])#Sava a classe

    Features=np.array(Features)# Converte lista numpy para array.
    Labels=np.array(Labels)
    return Features,Labels

#Aplicação de HOG no conjunto de treino

Features_Training,Labels_Training=images_to_hog(main_Training,Train
ing_Images_Directory) #
print ('Training HOG output Features shape : ',Features_Training.sh
ape)
print ('Training HOG output Labels shape: ',Labels_Training.shape)


#Aplicação de HOG no conjunto de teste
csv_files_Testing=glob.glob(Test_Images_Directory+'/**/*.csv',recur
sive=True)
main_Testing=pd.read_csv(csv_files_Testing[0],sep=';')
for i in range(1,len(csv_files_Testing)):
    new_doc=pd.read_csv(csv_files_Testing[i],sep=';'
    main_Testing=main_Testing.append(new_doc, ignore_index=True)
main_Testing


# Cria e treina o modelo SVM para classificação das HOGs
classifier=SVC(kernel='rbf',gamma='scale') # utiliza a função
SVC para implementar classificador SVM
classifier.fit(X_train,Labels_Training) # efetua o treino com o
conjunto de treino

#exibe o valor da precisão para o conjunto de treino
print ('SVM Mean Accuracy of Training dataset: ',classifier.score(X
_train,Labels_Training))
#exibe o valor da precisão para o conjunto de teste
print ('SVM Mean Accuracy of Test dataset: ',classifier.score(X_tes
t,Labels_Testing))


# salva o modelo
joblib.dump(pca, 'pca.pkl')
joblib.dump(classifier, 'svm.pkl')
```

APÊNDICE B – ALGORITMO DE DETECÇÃO

```
#bibliotecas
import os
import numpy as np
import os
import glob
import matplotlib.pyplot as plt
from skimage import color, exposure, transform, io
import imageio
from keras.models import Sequential
from keras.layers.core import Dropout, Dense, Activation, Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from tensorflow.keras.optimizers import SGD
from keras.callbacks import LearningRateScheduler, ModelCheckpoint,
 EarlyStopping
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split



# PARÂMETROS
NUM_CLASSES = 43
# NUM_CLASSES = 18
IMG_SIZE = 48
TRAINING_PATH = 'GTSRB/Final_Training/Images/'
# TRAINING_PATH = 'BD_PLACAS_2/Placas de Treino'
TEST_PATH = 'GTSRB/Final_Test/Images/'
# TEST_PATH = 'BD_PLACAS_2/Placas de Teste'
BATCH_SIZE = 32
EPOCHS = 10
# EPOCHS = 40

# PRE-PROCESSAMENTO
def preprocess_images(img):
    # converte as imagens em HSV
    hsv = color.rgb2hsv(img)
    # efetua o equalização em 9-bin
    hsv[:, :, 2] = exposure.equalize_hist(hsv[:, :, 2])
    img = color.hsv2rgb(hsv)

    # ajuste do tamanho
    min_side = min(img.shape[:-1])
    center =img.shape[0] // 2, img.shape[1] // 2
    img = img[center[0] - min_side // 2: center[0] + min_side // 2,
              center[1] - min_side // 2: center[0] + min_side // 2,
              :]
```

```
    img = transform.resize(img,(IMG_SIZE, IMG_SIZE), mode = 'consta
nt')

    return img

#obtem as classes de cada imagem
def get_class(img_path):
    return int(img_path.split('/')[-2])

#cria duas listas, X - contém a matriz da imagem já processada e Y
- contém a classe a qual a mesma pertence
images = []
labels = []
for img_path in img_paths:
    img = preprocess_images(io.imread(img_path))
    label = get_class(img_path)
    images.append(img)
    labels.append(label)
X = np.array(images, dtype = 'float32')
Y = np.eye(NUM_CLASSES, dtype = 'uint8')[labels]

#Construção do modelo de CNN

#função que estabelece a arquitetura do modelo proposto
def build_cnn_model():
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding ='same', input_shape = (IM
G_SIZE, IMG_SIZE, 3), activation = 'relu'))
    model.add(Conv2D(32, (3, 3), activation = 'relu'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), padding ='same', activation = 'rel
u'))
    model.add(Conv2D(64, (3, 3), activation = 'relu'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(128, (3, 3), padding ='same', activation = 're
lu'))
    model.add(Conv2D(128, (3, 3), activation = 'relu'))
    model.add(MaxPooling2D(pool_size = (2, 2)))
    model.add(Dropout(0.2))

    model.add(Flatten())
    model.add(Dense(512, activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(NUM_CLASSES, activation = 'softmax'))
    return model
```

```
#inicializa o modelo na variação model, efetua a compilação do
modelo com os parâmetros do mesmo sendo estabelecidos

model = build_cnn_model()
lr = 0.01
sgd = SGD(learning_rate = lr, decay = 1e-
6, momentum = 0.9, nesterov = True)
model.compile(loss = 'categorical_crossentropy', optimizer = sgd, m
etrics = ['accuracy'])

#Treino do modelo

#função que estabelece o parâmetro da taxa de atualização
def learning_rate_scheduler(epoch):
    return lr * (0.1 ** int(epoch / 10))

#efetua o treino do modelo com os conjuntos de treino
model_history = model.fit(X, Y,
                          batch_size = BATCH_SIZE,
                          epochs = EPOCHS,
                          validation_split = 0.2,
                          verbose = 1,
                          callbacks = [LearningRateScheduler(learni
ng_rate_scheduler),
                                       ModelCheckpoint('model.h5', s
ave_best_only=True),
                                       EarlyStopping(monitor='val_ac
curacy', min_delta=0.00001, patience=5, \
                                                     verbose=1, mod
e='auto')])

#Avaliação através da curva de precisão e função de perda
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.title('Model_Accuracy')
plt.xlabel('epochs')
plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc = 'lower right')
plt.show()

plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('Model_Loss')
plt.xlabel('epochs')
plt.ylabel('Loss')
plt.legend(['train', 'test'], loc = 'upper right')
plt.show()
```

```
#validando o modelo com dados de teste
path_test = 'GTSRB/Final_Test/Images/'
img_paths = glob.glob(os.path.join(path_test, '*/*.ppm))
img_paths = correct_all_paths(img_paths)
def get_class(img_path):
    return int(img_path.split('/')[-2])
X_test = []
y_test = []
for img_path in img_paths:
    img = preprocess_images(io.imread(img_path))
    label = get_class(img_path)
    X_test.append(img)
    y_test.append(label)
test_X = np.array(X_test, dtype = 'float32')
test_Y = np.eye(NUM_CLASSES, dtype = 'uint8')[y_test]

model.evaluate(test_X, test_Y)


#Utiliza técnica de DataAugmentation no conjunto de treino
X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size =
0.2, random_state = 42)

datagen = ImageDataGenerator(featurewise_center = False,
                             featurewise_std_normalization = False,
                             width_shift_range = 0.1,
                             height_shift_range = 0.1,
                             zoom_range =0.2,
                             shear_range = 0.1,
                             rotation_range = 10)
datagen.fit(X_train)

#Reinicia o modelo com os parâmetros
model = build_cnn_model()

lr = 0.01
sgd = SGD(learning_rate=lr, decay=1e-
6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
          optimizer=sgd,
          metrics=['accuracy'])

#efetua o re-treino já com um conjunto de dados pós-
dataaugmentation
model_history = model.fit(datagen.flow(X_train, Y_train, batch_size
= BATCH_SIZE),
                             steps_per_epoch=X_train.shape[0] / BATC
H_SIZE,
```

```
                              epochs= EPOCHS,
                              validation_data=(X_val, Y_val),
                              callbacks=[LearningRateScheduler(learni
ng_rate_scheduler),
                                         ModelCheckpoint('model_aug.h
5',save_best_only=True),
                                         EarlyStopping(monitor='val_a
ccuracy', min_delta=0.00001, patience=5, \
                                                       verbose=1, mod
e='auto')])


#Curva de precisão e função de perda
plt.plot(model_history.history['accuracy'])
plt.plot(model_history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc = 'upper right')
plt.show()

plt.plot(model_history.history['loss'])
plt.plot(model_history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['train', 'test'], loc = 'upper right')
plt.show()

#nova validação do modelo
y_predict = model.predict(X_test)
model.evaluate(test_X, test_Y)
```