Métodos de aprendizado de máquina para Inventário de Elementos de Vias de Trânsito

Johny Marques Borges Ribeiro

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2021

Johny Marques Borges Ribeiro

Métodos de aprendizado de máquina para Inventário de Elementos de Vias de Trânsito

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Prof. Dr Jefferson Rodrigo de Souza
Coorientador: Dr Raulcezar Maximiano Alves

Uberlândia
2021

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

R484 Ribeiro, Johny Marques Borges, 1985-
2021 Métodos de aprendizado de máquina para inventário de elementos de vias de trânsito [recurso eletrônico] / Johny Marques Borges Ribeiro. - 2021.

Orientador: Jefferson Rodrigo de Souza.
Coorientador: Raulcézar Maximiano Figueira Alves.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2021.110
Inclui bibliografia.

1. Computação. I. Souza, Jefferson Rodrigo de,1985-, (Orient.). II. Alves, Raulcézar Maximiano Figueira,1984-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 2/2021, PPGCO				
Data:	16 de fevereiro de 2021	Hora de início:	14:03	Hora de encerramento:	15:59
Matrícula do Discente:	11912CCP013				
Nome do Discente	Johny Marques Borges Ribeiro				
Título do Trabalho:	Métodos de aprendizado de máquina para Inventário de Elementos de Vias de Trânsito				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores: Henrique Coelho Fernandes - FACOM/UFU; Gustavo Pessin - ITV/UFOP; Raulcezar Maximiano Alves - CTI/UFU (coorientador) e Jefferson Rodrigo de Souza - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Gustavo Pessin - Ouro Preto/MG; Henrique Coelho Fernandes, Raulcezar Maximiano Alves e Jefferson Rodrigo de Souza - Uberlândia/MG. O discente participou da cidade de Uberlândia/MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Jefferson Rodrigo de Souza, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Henrique Coelho Fernandes**, **Professor(a) do Magistério Superior**, em 16/02/2021, às 18:53, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Jefferson Rodrigo de Souza**, **Professor(a) do Magistério Superior**, em 16/02/2021, às 23:04, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Raulcézar Maximiano Figueira Alves**, **Diretor(a)**, em 17/02/2021, às 09:21, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Gustavo Pessin**, **Usuário Externo**, em 17/02/2021, às 14:26, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2567937** e o código CRC **B782F9DA**.

Referência: Processo nº 23117.008925/2021-20 SEI nº 2567937

Dedico este trabalho à minha esposa, Nayara e minha filha Isabela por todo o apoio, amor e incentivo durante esta jornada.

Agradecimentos

Agradeço primeiramente a Deus por me abençoar e me conduzir durante toda essa jornada.

Agradeço ao meu orientador Prof. Dr Jefferson Rodrigo de Souza, pela oportunidade de ingressar nessa área de estudo, pelos conselhos, ensinamentos e disponibilidade.

Agradeço ao meu coorientador Dr Raulcezar Maximiano Alves, pelas revisões e conversas informais que me guiaram nessa jornada.

Agradeço ao Prof. Dr Henrique C. Oliveira, pelas reuniões que me ajudaram a encontrar caminhos diferentes para seguir com o trabalho.

Agradeço ao Prof. Dr Henrique C. Oliveira e a UNICAMP, pelos dados e as imagens disponibilizados para uso nesse trabalho.

Agradeço ao Prof. Dr Marco Mendonça, pelas revisões e mudanças propostas.

Agradeço a todos os amigos que fiz durante essa jornada, e com os quais pude trocar experiências e aprendizados.

Agradeço a minha família, Nayara e Isabela, pela paciência e incentivo durante todo esse processo.

À todos, meu muito obrigado!

"O Senhor é o meu Pastor, nada me faltará."
(Salmos, 23:1)

Resumo

Com o avanço da tecnologia e o custo das tecnologias dos sistemas avançados de assistência ao condutor (ADAS) diminuindo, a adoção desses sistemas se torna mais popular, onde tanto os carros de luxo como também os carros mais populares já tem acesso a esses sistemas. Nessa realidade, os governos necessitam fazer a manutenibilidade dos sinais de trânsito que compõem a malha rodoviária, uma vez que esse sinais são utilizados como entrada para os ADAS. Este trabalho descreve os métodos tanto para mapeamento de lombadas quanto para criação de inventários dos sinais de trânsito. São comparadas técnicas de aprendizado de máquina e arquiteturas de redes convolucionais para a detecção e classificação de lombadas e sinais de trânsito respectivamente. Os resultados deste trabalho demonstram a viabilidade do método proposto, alcançando 96% de acurácia na classificação de lombadas utilizando o algoritmo *Random Forest*, e com 94% de acurácia na classificação dos sinais de trânsito utilizando a arquitetura *Faster RCNN*.

Palavras-chave: *Machine Learning*, Mapeamento móvel, *Deep Learning*, Lombadas, Sinais de Trânsito.

Abstract

Due to recent technological advancements, the cost of advanced driver assistance systems (ADAS) technologies has decreased significantly, contributing to its widespread adoption in vehicles of all price ranges. In this reality, governments need to maintain traffic signs that compose the road network, since these signs are used as input to the ADAS. This work describes methods for both mapping speed bumps and creating road signs inventories as well. Machine learning techniques and convolutional network architectures are compared for detection and classification of speed bumps and traffic signs respectively. Results of this work demonstrate the viability of the proposed method, reaching 96% of accuracy in the classification of speed bumps using Random Forest algorithm, and 94% of accuracy in the classification of traffic signs using Faster RCNN architecture.

Keywords: Machine Learning, Mobile mapping, Deep Learning, Speed bumps, Traffic signals.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ANN *Artificial Neural Networks*

Adam *Adaptive Moment Estimation*

AP *Average Precision*

BN *Batch normalization*

BRTSD *Brazilian Traffic Sign Dataset*

CNN *Convolutional Neural Networks*

CSPNet Cross Stage Partial Network

DL *Deep Learning*

DPVAT Danos Pessoais Causados por Veículos Automotores de Via Terrestre

ELU Exponential Linear Unit

FC *Fully Connected*

Fast R-CNN *Fast Region Proposal Convolutional Neural Networks*

FPN *Feature Pyramid Network*

FIS *Fuzzy Inference System*

GTSRB *German Traffic Sign Detection Benchmark*

GTSDB *German Traffic Sign Recognition Benchmark*

GPS *Global Positioning System*

GNSS *Global Navigation Satellite Systems*

HSV *Hue, Saturation and Value*

HSI *Hue, Saturation and Intensity*

HOG *Histogram of Oriented Gradients*

IA Inteligência Artificial

IMU *Inertial Measurement Unit*

LiDAR *Light Detection And Ranging*

ML *Machine Learning*

MLP *Multi Layer Perceptron*

MSER *MaximallyStable Extrema Regions*

MLS *Mobile Laser Scanning*

MMS *Mobile Mapping Systems*

mAP *Mean Average Precision*

PANet Path Aggregation Network

RBF *Radial Basis Function*

ReLU *Rectified Linear Unit*

RFID *Radio-Frequency Identification*

RF *Random Forest*

RPN *Region Proposal Network*

ROI *Region of Interest*

SSD *Single Shot MultiBox Detector*

SVM *Support Vector Machines*

SIFT *Scale-invariant Feature Transform*

SGD *Stochastic Gradient Descent*

Sumário

Capítulo 1

Introdução

No mundo, de acordo com a OPAS (Organização Pan-Americana da Saúde) (OPAS, 2019), cerca de 1,35 milhão de pessoas morrem a cada ano em decorrência de acidentes de trânsito e entre 20 e 50 milhões de pessoas sofrem lesões não fatais, muitas delas resultando em incapacidade.

De acordo com o Relatório Anual 2018 (Seguradora Líder, 2019), disponibilizado pela Seguradora Líder, responsável pelo seguro a Danos Pessoais Causados por Veículos Automotores de Via Terrestre (DPVAT), no Brasil, em 2018, foram feitas 46.848 solicitações para indenizações por morte e 459.693 solicitações para indenizações por invalidez permanente.

E, no Balanço PRF (Polícia Rodoviária Federal) 2017 (PRF, 2019), em 2017 ocorreram 89.318 acidentes em rodovias federais, que resultaram na morte de 6.244 pessoas e 83.978 feridos. Das causas dos acidentes, duas se sobressaem:

- Velocidade Incompatível - 10.420
- Falta de Atenção à Condução - 34.406

Segundo (CNT, 2018) o transporte terrestre é o modal mais utilizado no Brasil no deslocamento de carga e pessoas, e os acidentes nas vias terrestres devem ser tratados com grande relevância. O aumento do número de mortes nas estradas é maior em países emergentes, onde temos um aumento da frota e da taxa de crescimento urbano que acompanham o crescimento econômico.

De acordo com (CNT, 2018), o Ministério da Saúde, no Brasil, vem tentando implantar o conceito de mortes evitáveis, que incluem as mortes nos acidentes de trânsito, as quais podem ser reduzidas por ações adequadas de promoção à saúde.

Ainda de acordo com (CNT, 2018), os acidentes nem sempre acontecem por acaso. Existem fatores que determinam sua ocorrência e, em geral, a ocorrência de um acidente depende da combinação entre esses fatores. Todavia, os relatórios policiais registram, normalmente, apenas um fator contribuinte principal para cada acidente.

Segundo (CNT, 2018), os fatores são classificados em seis grupos:

- ❏ Fator humano: está associado ao comportamento do indivíduo no trânsito, que de forma isolada ou conjunta com outros fatores contribui para a ocorrência de acidentes. Exemplos de fatores humanos são o desrespeito às normas de trânsito, o consumo de bebidas alcoólicas, entre outros;

- ❏ Fator veicular: está associado a problemas nos veículos envolvidos no acidente, podendo estar relacionado a um mal estado de conservação de pneus, problemas de freio, entre outros;

- ❏ Fator institucional/social: para esse fator destacam-se a Regulamentação e o Policiamento (fiscalização). O Código Nacional de Trânsito tenta definir qual deveria ser o comportamento dos usuários do sistema viário, mas ainda podem ocorrer situações em que a sinalização deixe dúvidas na informação transmitida aos usuários. E a presença de policiamento e fiscalização ostensiva pode influenciar diretamente o comportamento dos motoristas e pedestres;

- ❏ Fator socioeconômico: entra como um fator contribuinte, uma vez que pode influenciar o aumento do fluxo de veículos e o modo de direção. Se enquadram como fatores socioeconômicos a impossibilidade de investimento na melhoria da estrutura viária e mesmo a avaliação das condições socioeconômicas da população local, afim de conhecer o perfil dessa população;

- ❏ Fator meio ambiente: engloba a complexidade do ambiente no qual o trânsito se encontra, requerendo constante atenção dos usuários das vias. Ainda que haja falha humana nesses casos, a rodovia deve permitir que o motorista se recupere do evento e prossiga de forma segura;

- ❏ Fator viário: considerar as características da infraestrutura rodoviária existente. A insegurança causada pelas condições precárias da infraestrutura podem estar associadas a falhas no projeto, inexistência de sinalização, informação incorreta ou falta de manutenção da sinalização e também o estado do pavimento.

De acordo com (FU; HUANG, 2010), com a aceleração do processo de modernização e o aumento da frota de automóveis, a segurança no trânsito se tornou crucial no mundo. E na tentativa de minimizar os acidentes de trânsito começaram a surgir há algumas décadas sistemas avançados de assistência ao condutor (Advanced Driver Assistance System). São exemplos de tais sistemas:

- ❏ sistema de controle de cruzeiro adaptativo;

- ❏ sistema para evitar colisão;

- ❏ sistema de visão noturna;

- ❑ sistema de reconhecimento de sinais de trânsito;
- ❑ etc.

1.1 Motivação

Com base no contexto apresentado, e tendo como foco o Fator viário, que leva em consideração a sinalização e a manutenção da mesma, tem-se a necessidade do uso de métodos que auxiliem na criação e manutenção do inventário dos sinais de trânsito da malha viária, ou métodos que disponibilizem algum tipo de informação sobre a via.

De acordo com (WEN et al., 2016), os sinais de trânsito são fundamentais e essenciais num sistema de transporte porque, além de sua importância para segurança, eles também disponibilizam informações para os motoristas. E com a contínua expansão das cidades, é necessário a instalação de novos sinais e manutenção dos existentes.

Um dos sistema que compõe o ADAS é o sistema de detecção e reconhecimento de sinais de trânsito, que usam desde extratores de forma até a aplicação de aprendizado profundo para essa tarefa. Entretanto, para esses sistemas serem totalmente efetivos, os sinais de trânsito precisam estar com a manutenção em dia, sendo necessário a utilização de sistemas que possam gerenciar o inventário desses sinais de trânsito, ou mesmo sistemas que possam disponibilizar informações sobre a via.

Como em qualquer atividade de mapeamento, a parte de aquisição dos dados é custosa (HE; ORVETS, 2000). Dados são obtidos por levantamento terrestre, missões fotogramétricas aéreas, imagens orbitais e sistemas de mapeamento móvel. Essas plataformas são escolhidas de acordo com o projeto, levando em consideração a escala desejada como saída. Atualmente, os *Mobile Mapping Systems* (MMS) são usados para mapeamento da malha viárias (TAO; LI, 2007) e (MOLINA M. BLáZQUEZ; COLOMINA, 2016). Esses sistemas são compostos por um conjunto de sensores que incluem câmeras, receptores/antenas *Global Navigation Satellite Systems* (GNSS), *Inertial Measurement Unit* (IMU) e sistemas *Light Detection And Ranging* (LiDAR). Recentemente MMS são montados em plataformas com sensores de baixo custo usando cameras GoPro. As câmeras GoPro incluem receptores GNSS, giroscópios, acelerômetros, sensor de imagem e termômetro.

Neste trabalho serão utilizadas câmeras GoPro, sendo uma única câmera utilizada nos experimentos com lombadas de velocidade e podendo utilizar uma ou três câmeras para os experimentos relacionados ao inventário de sinais de trânsito.

1.2 Objetivos e Desafios da Pesquisa

Analisar os métodos para classificar lombadas de velocidade, como também método para geração de inventário de sinais de trânsito, usando ferramentas de baixo custo para

essa tarefa. Utilizar câmeras GoPro para coletar os dados, as imagens e aplicar as técnicas de ML. Os objetivos específicos deste trabalho são:

- ❏ Comparar diferentes técnicas de ML para a classificação de lombadas;
- ❏ Comparar diferentes arquiteturas CNN para o reconhecimento dos sinais de trânsito;
- ❏ Construir e disponibilizar publicamente uma base de dados das imagens de sinais de trânsito rotuladas tanto no formato Pascal VOC quanto YOLO.
- ❏ Contribuir com um método que possa realizar o inventário de sinais de trânsito.

1.3 Hipótese

- ❏ O mapeamento de baixo custo permite a classificação de informações da via, como lombadas;
- ❏ O uso do sensor *Global Positioning System* (GPS) de câmeras GoPro, como também as imagens geradas e a combinação com técnicas de *Deep Learning* (DL) possibilitam a criação de inventários de sinais de trânsito.

1.4 Contribuições

- ❏ Desenvolvimento de um método que permite a classificação de lombadas em vias públicas utilizando ferramentas de baixo custo;
- ❏ Desenvolvimento de um método que permite a criação de inventários de sinais de trânsito utilizando ferramentas de baixo custo;
- ❏ Criação de disponibilização de uma base de dados de imagens de sinais de trânsito já rotuladas nos formatos Pascal VOC e YOLO.

1.5 Organização da Dissertação

Esta dissertação está organizada em cinco capítulos, como mostrado a seguir:

- ❏ Capítulo 2: apresenta a fundamentação teórica das técnicas utilizadas no trabalho, como também os trabalhos correlatos tanto para detecção de lombadas, no reconhecimento de sinais de trânsito e inventários de sinais trânsito. É apresentada também os trabalhos referentes às arquiteturas utilizadas;

- Capítulo 3: apresenta o fluxo geral do trabalho, detalhando cada um dos passos, desde a coleta das informações, aplicação de algoritmos de ML, treinamento da redes convolucionais e utilização dos dados dos sensores para geração de inventários;
- Capítulo 4: apresenta os resultados tanto na aplicação dos algoritmos apresentados, como também a apresentação do resultado tanto da classificação de lombadas, como também inventário tanto de quantidade como de localização dos sinais de trânsito;
- Capítulo 5: apresenta a conclusão desta dissertação com as considerações finais e propostas para possíveis trabalhos futuros. Também destaca-se as publicações geradas com a pesquisa desenvolvida.

Capítulo 2

Fundamentação Teórica

Este capítulo apresenta as áreas de conhecimento que são utilizadas neste trabalho, partindo da área de ML que é mais geral, até a área mais específica com as CNNs. Após a apresentação desses conceitos são incluídos os trabalhos relacionados a esse trabalho.

2.1 Aprendizado de Máquina (ML)

ML é um campo na Inteligência Artificial (IA) que estuda algoritmos, processos ou modelos que ensinem máquinas a difícil tarefa de aprender. Esse campo é amplamente estudado, pois a quantidade de informações disponíveis cresce exponencialmente e extrair qualquer informação dessa base de forma manual tem um custo financeiro e de tempo muito alto.

Os algoritmos de aprendizado de máquina operam construindo um modelo dos exemplos de entrada para fazer predições guiados pelos dados, ao invés de simplesmente seguir informações estáticas e inflexíveis. Com isso as máquinas podem utilizar esses modelos para extraírem informações e como saída disponibilizar informações que contém algum processamento sobre elas, e que farão sentido no ambiente onde serão utilizadas.

Segundo Mitchell (1997), ML é um campo multidisciplinar, uma vez que se baseia nos resultados dos campos da inteligência artificial, probabilidade e estatística, teoria da complexidade computacional, teoria de controle, teoria da informação, filosofia, psicologia, neurobiologia e outros campos.

Segundo Alpaydin (2004), ML usa a teoria de estatísticas na construção de modelos matemáticos uma vez que a tarefa central é fazer inferências a partir de um modelo. O papel da ciência da computação é duplo: criar algoritmos eficientes para resolver o problema de otimização e também armazenar e processar os dados que se tem para a tarefa. Após aprender um modelo, tanto a sua representação quanto a solução algorítmica precisam ser o mais eficiente possível.

De acordo com (ALPAYDIN, 2004), alguns exemplos de aplicações do aprendizado de máquina são:

- Aprendizado de associações, que pode ser usado na análise de compras, para encontrar a associação entre produtos comprados pelos consumidores;
- Classificação, que pode ser usado na análise do *score* de credito dos clientes, classificando esse clientes em grupos de alto ou baixo risco;
- Reconhecimento de Padrões, nesse tipo de aplicações entram o reconhecimento de caracteres óticos e também o reconhecimento facial;
- Regressão, que pode ser usada na predição do valor de carros usados;
- Aprendizado por reforço, onde os sistemas são capazes de aprender baseados em ações passadas e com isso conseguem gerar novas ações;
- Entre outros.

Segundo Mitchell (1997), dentro do campo do aprendizado de máquina existem diferentes tipos de algoritmos que buscam em um espaço de hipóteses definido por uma representação subjacente, e cada tipo de representação é mais apropriado para aprender diferentes tipos de funções alvo, além de ter um algoritmo de aprendizado que é mais vantajoso ao estruturar o espaço de buscas.

Segundo Russell et al. (2010), existem três tipos principais de aprendizagem sendo a supervisionada, não supervisionada e a aprendizagem por reforço.

- **Aprendizado Supervisionado:** De acordo com Russell et al. (2010), o aprendizado é realizado com base em pares de entrada e saída, e uma função faz o mapeamento da entrada para a saída.

  Segundo Silva, Spatti e Flauzino (2010), essa estratégia de aprendizado consiste em ter disponível a saída desejada para um determinado conjunto de sinais de entrada.

- **Aprendizado Não Supervisionado:** De acordo com Russell et al. (2010), o aprendizado é realizado com base nos padrões de entrada, mas sem nenhuma saída explícita. A tarefa mais comum para esse tipo é o agrupamento.

  Segundo Silva, Spatti e Flauzino (2010), um algoritmo baseado nesse tipo de aprendizado não requer nenhum conhecimento das respectivas saída.

- **Aprendizado Por Reforço:** De acordo com Russell et al. (2010), o aprendizado é feito por uma série de reforços, seja por punições ou recompensas.

  Segundo Silva, Spatti e Flauzino (2010 apud SUTTON et al., 1998), o aprendizado por reforço é uma variação do aprendizado supervisionado, uma vez que analisam continuamente a diferença entre a resposta produzida e a saída desejada.

Nesse trabalho são utilizados algoritmos e arquiteturas como classificadores, e que são encontrados dentro da área do ML. As próximas seções apresentam esses algoritmos e também as arquiteturas utilizadas. Há também as ANNs, que fazem parte de uma área conhecida como sistemas inteligentes ou inteligência computacional. Além da ANN, a área de sistemas inteligentes inclui diversas ferramentas, tais como sistemas fuzzy, computação evolucional, inteligência de enxame, sistemas imunológicos artificiais e agentes inteligentes (SILVA; SPATTI; FLAUZINO, 2010).

2.2 *Naive Bayes*

É uma técnica de aprendizado baseada no teorema de probabilidade condicional de Bayes. Mesmo sendo um modelo simples, o teorema é surpreendentemente eficiente quando usado como um algoritmo de classificação (JULIA, 2018). A Equação 1 apresenta o Teorema de Bayes, usado como base para está técnica de aprendizado de máquina.

$$P\left(c|x\right) = \frac{P\left(x|c\right)P\left(c\right)}{P\left(x\right)}, \tag{1}$$

Em termos simples, e aplicando a explicação para esse trabalho, esse classificador estima qual a probabilidade do dado coletado dos sensores pertencer aos dados coletados sobre as lombadas e passarelas elevadas. No restante desse trabalho será utilizado o termo lombada representando lombadas e passarelas elevedas.

2.3 *Random Forest* (RF)

É um algoritmo de aprendizado supervisionado que constrói múltiplas árvores de decisão e junta essas árvores para conseguir uma predição acurada. Cada árvore de decisão passa através de um mecanismo de voto, o qual escolhe a classificação mais votada (DANTAS, 2015).

Segundo (BONACCORSO, 2017), o algoritmo RF utiliza o aprendizado por agrupamento, onde são agrupados outros classificadores de forma paralela ou sequencial. RF é um conjunto de árvores construídas com amostras aleatórias. O processo de treinamento é baseado na seleção aleatória das divisões e as predições são baseados em um voto majoritário. A Figura 1 apresenta exemplo de árvores criadas no algoritmo RF.

Nesse estudo é explorado a capacidade de classificação do algoritmo RF para classificar um conjunto de dados multidimensional. Ainda nesse estudo, os ramos de cada árvore irão ser definidos pela informação provida por cada parâmetro usado como entrada.

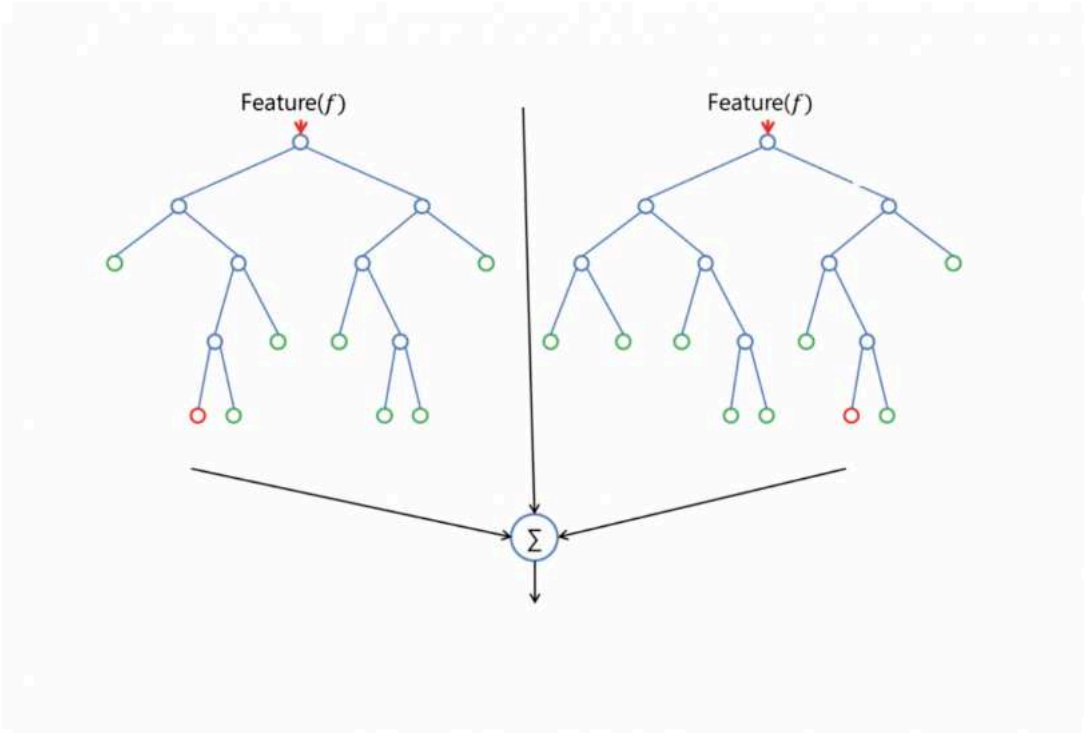

Figura 1 – Exemplo de árvores criadas no algoritmo RF

Fonte: (DONGES, 2019)

2.4 Redes Neurais Artificiais - *Artificial Neural Networks* (ANN)

Segundo Haykin (1998), uma rede neural é um processador distribuído e massivamente paralelo feito sobre simples unidades de processamento, as quais armazenam conhecimento experimental e disponibilizam o mesmo para uso. Se assemelha ao cérebro em dois pontos:

1. Conhecimento é adquirido pela rede de seu ambiente através do processo de aprendizado.

2. Conexões neurais ou pesos sinápticos, são usados para armazenar o conhecimento adquirido.

Segundo Silva, Spatti e Flauzino (2010), ANN são modelos computacionais inspirados no sistema nervoso dos seres vivos. Eles têm a habilidade de adquirir e manter conhecimento (baseado em informação) e podem ser definidos como um conjunto de unidades de processamento, representados por neurônios artificiais, interligados por muitas interconexões (sinapses artificiais), implementadas por vetores e matrizes de pesos sinápticos.

Ainda segundo Silva, Spatti e Flauzino (2010), os primeiros artigos sobre ANN foram publicados há mais de 50 anos. Entretanto, somente nos anos 90 essa área começou

a ser profundamente pesquisada e continua com um enorme potencial. As aplicações envolvendo sistemas considerados inteligentes cobrem uma ampla gama, incluindo:

- ❏ Análise de imagens adquiridas de satélites;
- ❏ Classificação de padrão de fala e escrita;
- ❏ Reconhecimento de face com visão computacional;
- ❏ Controle de trens de alta velocidade;
- ❏ Previsão de ações no mercado financeiro;
- ❏ Identificação de anomalias em imagens médicas;
- ❏ Identificação automáticas de perfis de crédito para clientes de instituições financeiras;
- ❏ Controle de dispositivos eletrônicos e eletrodomésticos.

Ainda de acordo com Silva, Spatti e Flauzino (2010), as ANNs possibilitam a criação de soluções de outros tipos de problemas que derivam de diferentes áreas de conhecimento. Alguns exemplos são:

- ❏ Classificação e predição de câncer, na área da medicina;
- ❏ Obtenção de novos compostos poliméricos e sistemas de controle para tratamento de água, na área da química;
- ❏ Identificação de morcegos baseado em sua ecolocalização e classificação de espécies de ratos, na área da biologia;
- ❏ Além de outros exemplos na indústria de comida, indústria automotiva e espacial, etc.

De acordo com (SILVA; SPATTI; FLAUZINO, 2010), uma ANN pode ser dividida em três partes, chamadas de camadas que são conhecidas como:

a) Camada de entrada: É responsável por receber a informação do ambiente externo;
b) Camadas escondidas ou intermediárias: São compostas por neurônios responsáveis por extrair padrões associados com o processo ou sistema sendo analisado;
c) Camada de saída: É composta por neurônios e é responsável por produzir e apresentar a saída final da rede.

Nas seções seguintes são apresentadas as principais arquiteturas das ANN.

2.4.1 Arquitetura *Feedforward* de camada simples

Essa arquitetura é composta por uma camada de entrada e somente uma camada neural, que é também a camada de saída (SILVA; SPATTI; FLAUZINO, 2010).

Segundo Silva, Spatti e Flauzino (2010), a informação segue um fluxo unidirecional, e a quantidade de saídas dessa rede sempre coincide com a quantidade de neurônios.

Ainda de acordo com Silva, Spatti e Flauzino (2010), entre os principais tipos de rede que usam essa arquitetura estão a Perceptron e a ADALINE.

A Figura 2 apresenta um exemplo dessa arquitetura.

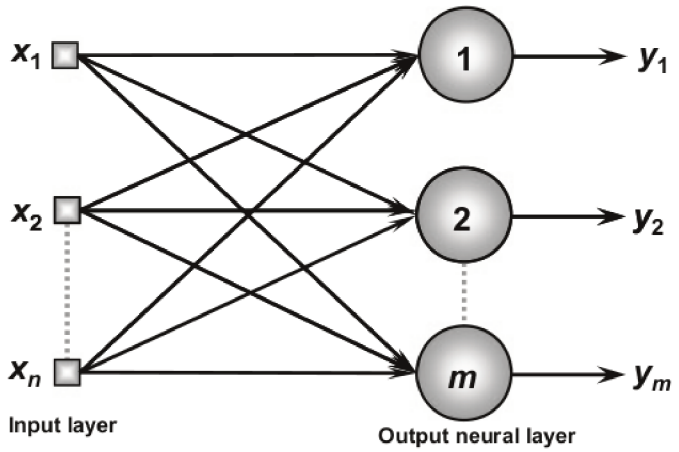

Figura 2 – Exemplo de uma rede *feedforward* de camada simples

Fonte: (SILVA; SPATTI; FLAUZINO, 2010)

2.4.2 Arquitetura *Feedforward* de múltiplas camadas

Essa arquitetura se diferencia da anterior por ser composta por uma ou mais camadas neurais escondidas. As redes que usam essa arquitetura são empregadas na solução de diversos problemas, como os relacionados a aproximação de função, classificação de padrões, sistemas de identificação, controle de processos, otimização, robótica entre outros (SILVA; SPATTI; FLAUZINO, 2010).

De acordo com Silva, Spatti e Flauzino (2010), entre as principais redes que usam essa arquitetura estão a MLP e a *Radial Basis Function* (RBF), as quais usam os algoritmos baseados na regra delta geral e regra competitiva/delta respectivamente.

A Figura 3 apresenta um exemplo dessa arquitetura. Além das arquiteturas apresentadas há também as arquiteturas recorrente e com estrutura reticulada, que não serão abordadas nesse trabalho.

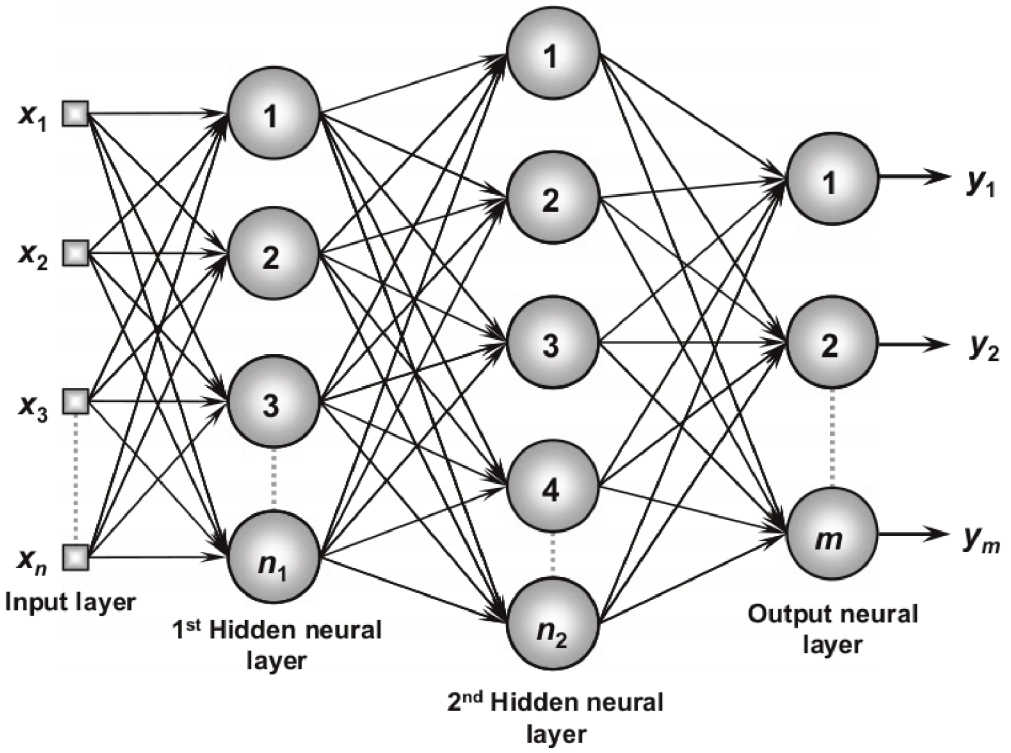

Figura 3 – Exemplo de uma rede *feedforward* de múltipla camada

Fonte: (SILVA; SPATTI; FLAUZINO, 2010)

No treinamento das ANNs são utilizados algoritmos para otimização dos pesos. Nesse trabalho são feitos experimentos com dois algoritmos, para comparação:

- SGD: Segundo Ruder (2017), executa a atualização de parâmetro para cada exemplo de treinamento. Usualmente é mais rápido e pode ser usado para aprendizado *online*.
- Adam: Segundo Ruder (2017), é outro método que calcula as taxas de aprendizado adaptativo para cada parâmetro.

2.5 *Deep Learning* - Aprendizado Profundo

Deep Learning é uma abordagem dentro das ANNs que utiliza a arquitetura *Multi Layer Perceptron*. O "profundo"nessa abordagem se refere ao fato de usar muitas camadas escondidas na arquitetura da rede.

Segundo LeCun, Bengio e Hinton (2015), os métodos de DL são métodos de representação do conhecimento que utilizam várias camadas de representação, obtidos pela composição de módulos simples, mas não lineares, que transformam a representação inicial até uma representação em um nível mais abstrato.

Ainda segundo LeCun, Bengio e Hinton (2015), os módulos são sujeitos a aprendizagem e muitos deles computam os mapeamentos de entrada/saída não lineares. Cada módulo transforma a sua entrada aumentando a seletividade e invariância da representação.

Segundo Patterson (2017), DL são redes neurais com um grande número de parâmetro e camadas e usam uma das quatro arquiteturas fundamentais:

- ❏ Redes Pré-treinadas Não Supervisionadas;
- ❏ Redes Neurais Convolucionais;
- ❏ Redes Neurais Recorrentes;
- ❏ Redes Neurais Recursivas.

Ainda segundo Patterson (2017), existem variações dessas arquiteturas, como exemplo uma arquitetura híbrida entre rede convolucional e recorrente.

Segundo (CHOLLET, 2017), o DL moderno frequentemente envolve dezenas ou centenas de camadas sucessivas que aprendem automaticamente ao serem expostas aos dados de treinamento.

A Figura 4 apresenta uma rede que transforma a imagem de um dígito que é a entrada da rede em uma representação diferente da original mas com cada vez mais informação sobre o resultado final.

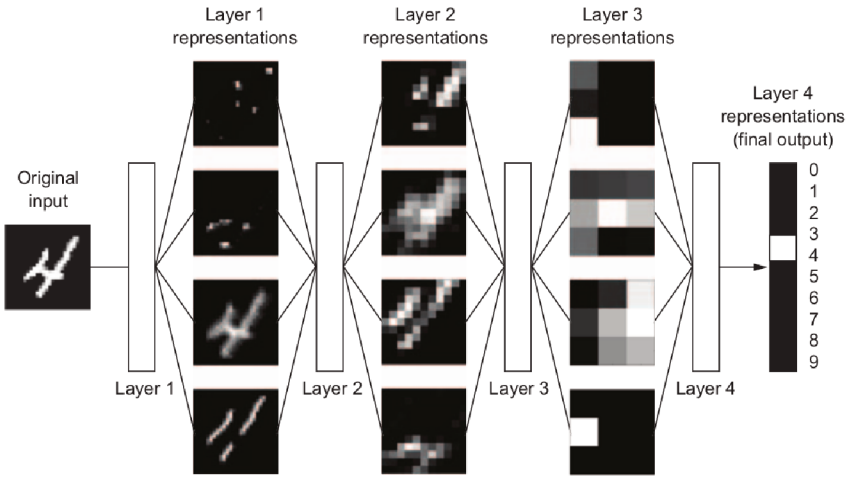

Figura 4 – Representação de aprendizado de um modelo de classificação de dígito

Fonte: (CHOLLET, 2017)

Dentro das arquiteturas de DL apresentadas acima, a CNN é a arquitetura utilizada nesse trabalho, e será apresentada em mais detalhes.

2.6 Redes Neurais Convolucionais (CNN)

Segundo Haykin (1998), uma CNN é um *Perceptron* Multicamada projetado para reconhecer formas bidimensionais que utiliza o aprendizado supervisionado.

Segundo Haykin (1998 apud LECUN; BENGIO, 1995), a estrutura da CNN segue a seguinte forma:

1. Extração de características: Cada neurônio leva suas entradas sinápticas da camada anterior, e assim força a extração local de características;

2. Mapeamento de características: Cada camada computacional é composta por vários mapeamentos de características, com cada mapeamento sendo um plano com os neurônios individuais compartilhando o mesmo conjunto de pesos sinápticos;

3. Subamostragem: Cada camada convolucional é seguida por uma camada computacional que realiza uma média local e subamostragem onde o tamanho do mapeamento de características é reduzido.

Segundo LeCun, Bengio e Hinton (2015), a arquitetura típica de uma CNN é estruturada como uma série de estágios, onde os primeiros estágios são compostos por camadas convolucionais e camadas de *pooling*. A Figura 5 apresenta esse arquitetura.

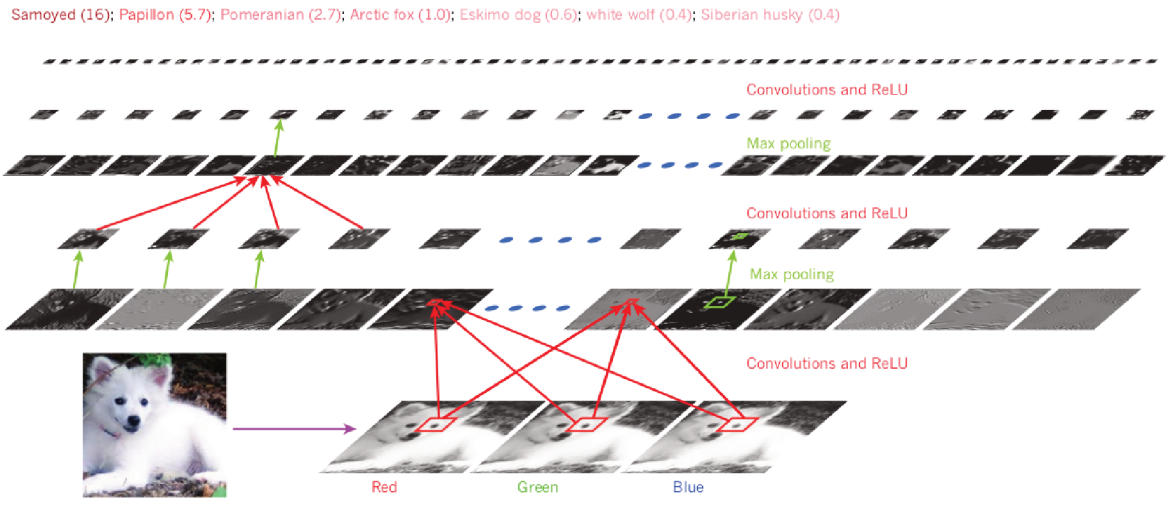

Figura 5 – Arquitetura tipica de uma CNN

Fonte: (LECUN; BENGIO; HINTON, 2015)

Segundo Rosebrock (2017), cada camada em uma CNN aplica um diferente conjunto de filtros, tipicamente centenas ou mesmo milhares, e combinam os resultados e usam essa saída como a entrada para a próxima camada. A CNN automaticamente aprende com os valores desses filtros.

Ainda de acordo Rosebrock (2017), cada filtro compõe um *patch* local de características de baixo nível em uma representação de alto nível, similar à composição de um conjunto de funções matemáticas: $f(g(x(h(x))))$, o que permite a rede aprender características mais ricas e de forma mais profunda na rede.

Segundo Goodfellow Yoshua Bengio (2016), CNN é um tipo específico de ANN para processamento de dado que tem uma topologia tipo grade. Exemplos incluem dados de série temporal e imagens.

Ainda de acordo Goodfellow Yoshua Bengio (2016), o nome *Convolutional Neural Networks* indica que a rede emprega uma operação matemática chamada convolução que é um tipo específico de operação linear. A convolução é usada no lugar da multiplicação de matrizes em pelo menos uma de suas camadas.

2.6.1 Convolução

Segundo Rosebrock (2017), convolução é uma multiplicação elemento a elemento de duas matrizes, seguido da soma. A convolução (denotada pelo operador *) sobre uma imagem e um *kernel* bidimensional I e K, respectivamente, é definida por:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n K(i-m, j-n) I(m,n) \tag{2}$$

Segundo Rosebrock (2017), Goodfellow Yoshua Bengio (2016), muitas arquiteturas utilizam a função da correlação cruzada, mas ainda chamam de convolução, esse função é definida por:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n K(i+m, j+n) I(m,n) \tag{3}$$

Segundo Rosebrock (2017), um *kernel* pode ser visualizado como uma matriz pequena que desliza sobre a imagem da esquerda para direita e de cima para baixo, conforme apresentado na Figura 6.

2.6.2 Arquitetura Básica

Segundo Patterson (2017), existem muitas variações de arquitetura de CNN, mas todas são baseadas no padrão de camadas apresentado na Figura 7.

Já segundo Rosebrock (2017), os tipos camadas mais encontrados nas arquiteturas são:

- Convolução;
- Ativação (*Rectified Linear Unit* (ReLU));
- *Pooling*;

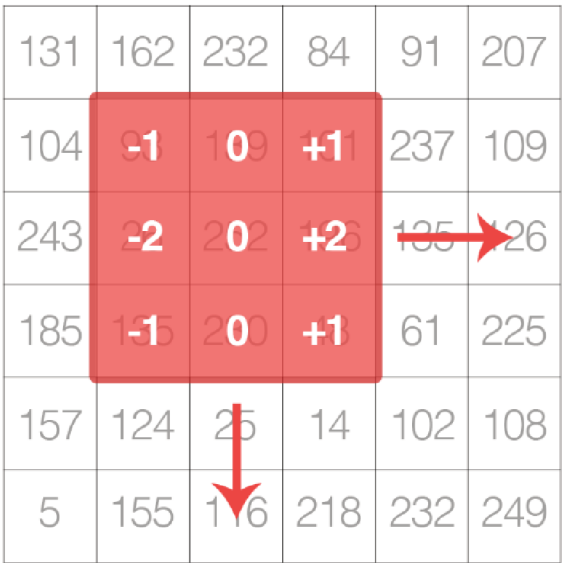

Figura 6 – Exemplo de *kernel* sobre imagem

Fonte: (LECUN; BENGIO; HINTON, 2015)

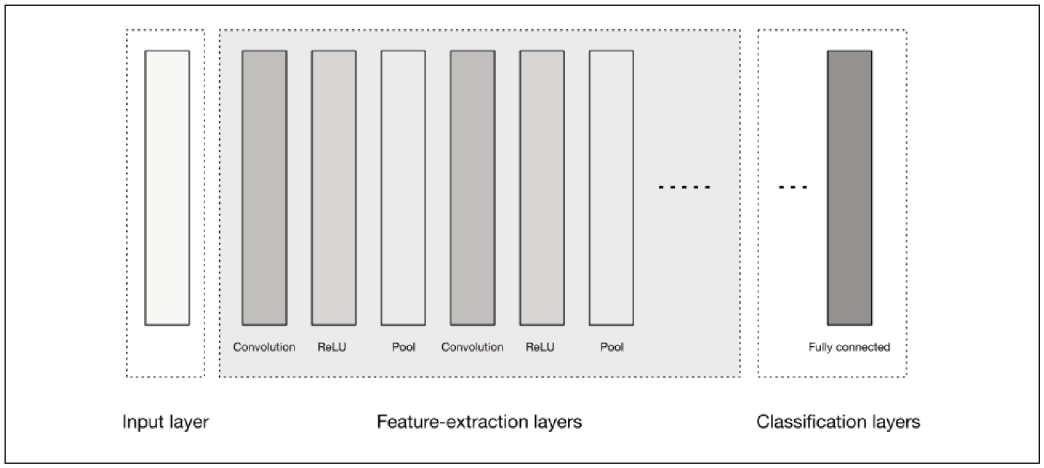

Figura 7 – Arquitetura de uma CNN de alto nível

Fonte: (PATTERSON, 2017)

- Totalmente Conectada;
- *Batch normalization* (BN);
- *Dropout*.

Nas próximas subseções essas camadas serão apresentadas com mais detalhes.

2.6.2.1 Camada de entrada (*Input Layer*)

A camada de entrada aceita entradas tridimensionais, geralmente em uma forma espacial do tamanho da imagem e com a profundidade representando o canal de cores (PATTERSON, 2017). Figura 8 mostra um exemplo de entrada para essa camada.

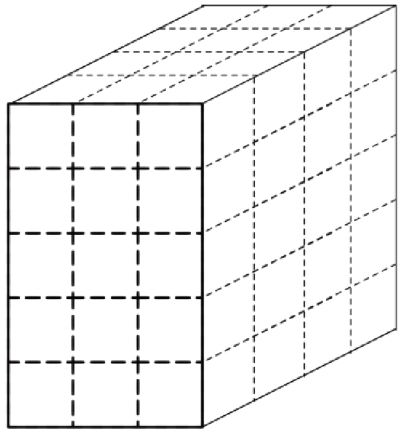

Figura 8 – Exemplo de entrada 3D

Fonte: (PATTERSON, 2017)

2.6.2.2 Camada de Convolução

Segundo Patterson (2017), essa camada compõem a camada de extração de características (*Feature-extraction Layer*) e nessa camada é que são realizadas as operações de convolução.

Segundo Rosebrock (2017), essa é a camada que forma o núcleo da CNN. Consiste em um conjunto de K filtros para aprendizado (*kernels*) e cada filtro tem uma largura e uma altura, quase sempre são quadrados.

Ainda segundo Rosebrock (2017), após aplicar os K filtros, são gerados K mapas de ativação que são empilhados para formar a entrada para a próxima camada.

A Figura 9 mostra um exemplo desse empilhamento.

Segundo Patterson (2017), os principais componentes dessa camada são:

- Filtros;
- Mapas de ativação;
- Compartilhamento de parâmetros;
- Hiper-parâmetros utilizados na camada, que são:

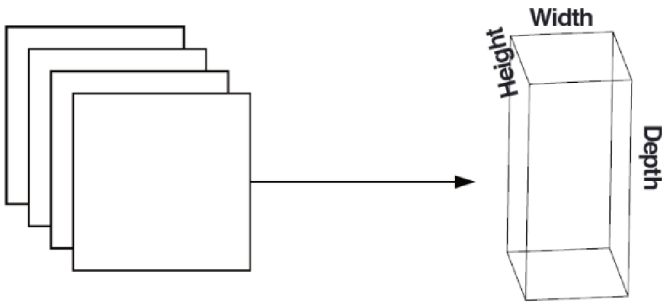

Figura 9 – Exemplo de pilha dos mapas de ativação

Fonte: (ROSEBROCK, 2017)

- Tamanho do filtro;
- Profundidade de saída;
- *Stride*: Representa a quantidade de *pixels* que o filtro se movimenta em cada passo em uma determinada direção;
- *Zero-padding.*

2.6.2.3 Camada ReLU ou Camada de Ativação

Segundo Rosebrock (2017), após cada camada de convolução é aplicado uma função de ativação não-linear (ReLU, Exponential Linear Unit (ELU), etc). Tecnicamente não é uma camada pois não há aprendizado de parâmetros ou pesos nessa camada, por esse motivo essa camada é omitida nos diagramas de algumas arquiteturas.

Segundo Patterson (2017), essa camada também compõem a camada de extração de características (*Feature-extraction Layer*), como é representada na Figura 7 para corresponder a outras arquiteturas que utilizam essa representação.

2.6.2.4 Camada de *Pooling*

Segundo Patterson (2017), essa camada também compõem a camada de extração de características (*Feature-extraction Layer*), como é representada na Figura 7.

Ainda segundo Patterson (2017), essa camada é normalmente adicionada entre sucessivas camadas de convolução, e servem para progressivamente diminuir o tamanho da representação dos dados ajudando também a controlar o *overfitting.*

De acordo com Patterson (2017), as duas funções de *pooling* mais comuns são *max pooling* e *average pooling.*

Segundo Goodfellow Yoshua Bengio (2016), essa camada ajuda a representação a se tornar aproximadamente invariante para pequenas translações nos dados de entrada.

A Figura 10 apresenta exemplos da aplicação do *pooling* com diferentes *strides.*

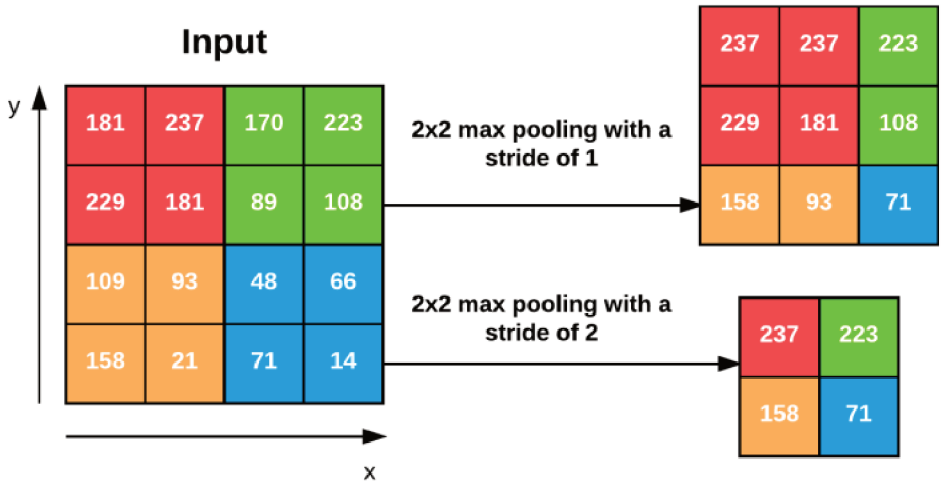

Figura 10 – **Direita**: entrada em 4X4. **Esquerda**: Aplicando *max pooling* de 2X2 com um *stride* de 1. **Inferior**: Aplicando *max pooling* de 2X2 com um *stride* de 2.

Fonte: (ROSEBROCK, 2017)

2.6.2.5 Camada Totalmente Conectada (*Fully Connected* (FC) *Layer*)

É a camada de classificação (*Classification Layer*), onde tem-se uma ou mais camadas totalmente conectadas e que recebem as características de alto nível geradas pela camada de extração e produzem as probabilidades de classe para a imagem (PATTERSON, 2017).

Segundo Rosebrock (2017), os neurônios nessa camada são totalmente conectados com todas as ativações da camada anterior. Essa camada é sempre incluída no final da rede.

2.6.2.6 *Batch normalization*

Segundo Rosebrock (2017), as camadas BN são usadas para normalizar as ativações de um determinado volume de entrada antes de enviar para a próxima camada de convolução.

Ainda segundo Rosebrock (2017), BN mostrou ser extremamente efetiva na redução do número de épocas no treinamento e ajuda na estabilização do treinamento.

Segundo Patterson (2017), é usado para acelerar o treinamento em CNN, a técnica aplica uma transformação que mantém a média da ativação perto de 0, enquanto mantém o desvio padrão da ativação perto de 1.

2.6.2.7 *Dropout*

Segundo Rosebrock (2017), essa camada é uma forma de regularização que visa prevenir o *overfitting* aumentando a precisão do teste, e para tanto a arquitetura da rede é alterada explicitamente no treinamento.

Essa técnica é utilizada exclusivamente no treinamento, impactando no valor dos pesos de cada neurônio, forçando a rede a aprender determinadas características por caminhos diferentes, após o treinamento, no processo de inferência, é utilizada a rede com todos os neurônios ativos.

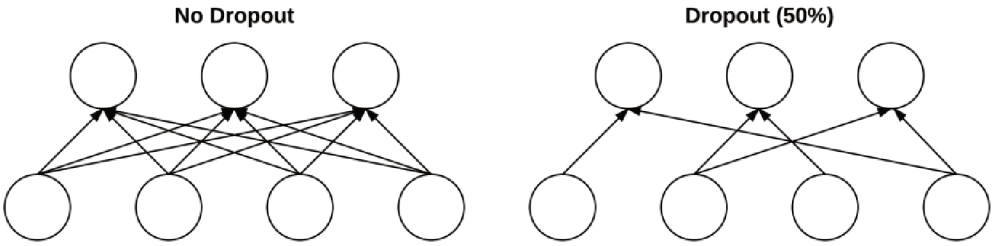

Figura 11 – **Esquerda**: Duas camadas que são totalmente conectadas sem *dropout*. **Direita**: As mesmas duas camadas após aplicar o *dropout*.

Fonte: (ROSEBROCK, 2017)

2.7 Trabalhos Correlatos

Nesta seção serão apresentados os trabalhos correlatos das áreas utilizadas nesse trabalho: a classificação de lombadas, a detecção e classificação de placas de trânsito e o inventário de placas de trânsito.

2.7.1 Classificação de Lombadas

Em (BELLO-SALAU et al., 2018), os autores apresentam um algoritmo para detecção de buracos e lombadas dos sinais de aceleração usando um acelerômetro. Um filtro *wavelet* foi aplicado para decompor os sinais em escalas. Em seguida, as anomalias da estrada foram identificadas em um sistema de limiar usando as características extraídas dos coeficientes da *wavelet*. O sistema detecta anomalias na estrada com um alto nível de acurácia, precisão e um baixo nível de falso alarme.

O trabalho de (SOILáN et al., 2018) descreve uma metodologia para a extração de informação semântica a partir de dados públicos visando a parametrização urbana. Primeiro um edifício é segmentado com dados de voxel. Depois regras heurísticas e aprendizado não supervisionado foram aplicados aos dados da superfície do solo para distinguir os pontos da calçada e do pavimento. Finalmente, o sistema foi capaz de gerar um F-score próximo a 95% para detecção de pavimento e caminho.

(ZHAO; YOU; HUANG, 2011) propõe a extração de estradas urbanas a partir de dados LiDAR aerotransportados. A metodologia consiste na separação do terreno do Modelo Digital de Superfície (DSM), extração da linha central da estrada e verificação

da malha viária. Os objetos elevados são removidos da imagem de profundidade. Após, a inferência é executada no mapa vetorial da linha central da estrada de acordo com as leis gestalt. Uma técnica de votação baseada em direção é desenvolvida para avaliar a confiabilidade de cada caminho. Os resultados mostram que as características da estrada são projetadas com sucesso na imagem de profundidade.

Em (WANG et al., 2016), os autores analisam diferentes métodos de extração de estradas. Primeiramente, a extração das características da estrada foram investigadas. Em segundo lugar, os autores apresentaram as vantagens e desvantagens dos métodos analisados. Então, para melhorar o resultado da classificação, a extração de estradas combina múltiplos métodos de acordo com as aplicações reais. Os resultados mostram que a metodologia pode reconhecer com sucesso estradas usando diferentes características da estrada, menos quando objetos como água, edifícios, árvores, gramas e carros obstruem a estrada.

(ALJAAFREH et al., 2017) propõe um método de detecção de lombadas baseado em *Fuzzy Inference System* (FIS). O FIS detecta as lombadas a partir da aceleração vertical e da velocidade do veículo. O sistema usa o acelerômetro em um *smartphone*. A metodologia é avaliada em diferentes níveis de velocidade. Os resultados mostram que o sistema é promissor para a detecção de lombadas.

(CELAYA-PADILLA et al., 2018) desenvolve um método para a detecção de anomalias nas estradas, como buracos e lombadas. A técnica desenvolvida usa o giroscópio, acelerômetro e GPS em um carro. Os dados foram coletados enquanto o veículo cruzava várias ruas. Usando a estratégia de validação cruzada, um algoritmo genético é usado para encontrar o modelo que detecta anomalias na estrada. Os resultados mostram que o modelo obteve uma acurácia de 97.14%, com uma taxa de falso positivo menor do que 1.8%.

(VARMA et al., 2018) propõe um método de detecção de lombadas e saliências de velocidade baseado em imagem. A técnica usa aprendizado profundo e câmeras estéreo para detectar as lombadas e saliências de velocidade, que fazem cálculos de distância para os objetos detectados. Os resultados mostram que a técnica obteve uma acurácia de 97.44% para objetos rotulados e 93.83% para objetos não rotulados.

(DEVAPRIYA; BABU; SRIHARI, 2015) propõe uma metodologia que usa imagem. O vídeo capturado é transformado em imagem e em seguida essa imagem é pré-processada. Em seguida, são realizadas as operações morfológicas e a projeção do objeto na imagem. A metodologia foi baseada em cinco diferentes tipos de lombadas, para o tipo 1 a acurácia foi 90%, tipo 2 foi 85%, tipo 3 foi 83%, tipo 4 foi 80% e o tipo 5 alcançou 4%.

(FERNáNDEZ et al., 2012) apresenta um sistema de detecção de espaços livre e espinhos, esses espinhos foram definidos pelo governo espanhol. A detecção de espaços livres é realizada por um tratamento de baixo custo, e a detecção de espinhos é realizada com imagens capturadas por uma câmera, levando em consideração a pintura zebrada que é

feita nesses espinhos. A imagem é processada para funcionar apenas com as regiões de interesse que passam pela fase de extração de bordas e contornos e, em seguida, as linhas da imagem são analisadas. A acurácia foi de 100% para espinhos e 94% para detecção das faixas zebradas.

2.7.2 Detecção e Classificação de Sinais de Trânsito

Em (HOELSCHER, 2017), é feita a apresentação de um sistema para o reconhecimento e classificação de sinais de trânsito. O sistema é composto por um módulo que faz a segmentação da imagem baseada em cores, um módulo que faz a seleção das regiões de interesse, e após isso um módulo de classificação onde são utilizadas as CNNs.

O módulo de segmentação emprega o método Fuzzificação de Cromaticidade que propõe a utilização da Teoria Fuzzy para realizar a transformação de uma imagem RGB para um mapa que destaca regiões com a coloração de interesse.

O módulo de seleção das regiões de interesse utiliza o algoritmo *MaximallyStable Extrema Regions* (MSER), que seleciona regiões conectadas que mantém seu formato estável durante um intervalo de níveis de intensidade definido por Δ, em imagens em tons de cinza.

O módulo de classificação realiza a detecção e a classificação utilizando uma CNN de tamanho reduzido. O modelo da CNN utilizado é diferente para cada uma das bases usadas.

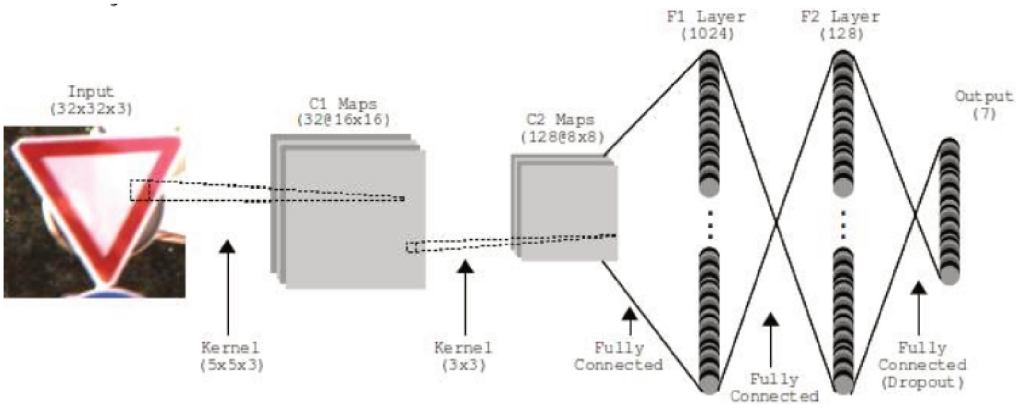

Figura 12 – Modelo usado com a base BRTSD

Fonte: (HOELSCHER, 2017)

Ao utilizar a base BRTSD o modelo usado é mostrado na Figura 12 .

Ao utilizar a base GTSRB/GTSDB, ambas da Alemanha, o modelo usado é o apresentado na Figura 13. Em (KARADUMAN; EREN, 2017) é apresentado um sistema que faz a classificação de sinais de trânsito usando uma CNN, e usa os dados do *smartphone* como acelerômetro e GPS para determinar o estilo de direção.

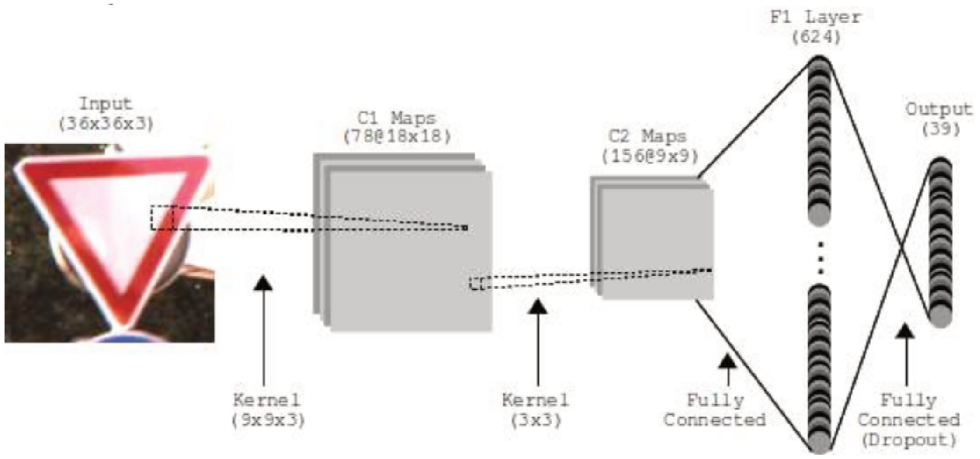

Figura 13 – Modelo usado com as bases GTSRB/GTSDB

Fonte: (HOELSCHER, 2017)

A CNN utiliza as imagens captadas pelo *smartphone* e a Figura 14 apresenta a arquitetura utilizada. A velocidade e manobras do motorista, como também os sinais de direção são processados simultaneamente e no passo seguinte do algoritmo proposto esses dados são combinados e com isso é possível determinar o perfil de direção do motorista.

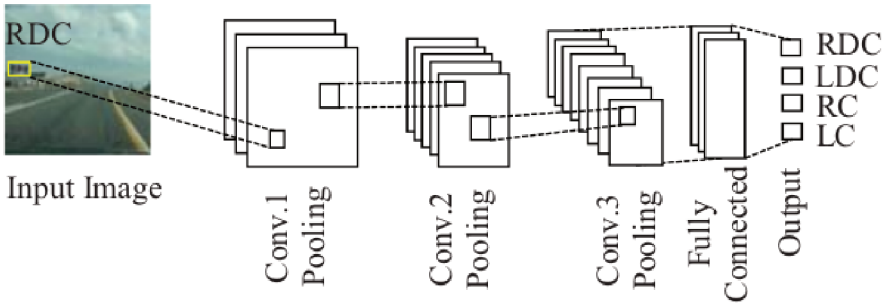

Figura 14 – Arquitetura da CNN

Fonte: (KARADUMAN; EREN, 2017)

Em (HUANG; LIN; CHANG, 2017), é proposta uma técnica composta por dois subsistemas, um para detecção e outro para reconhecimento. Essa divisão é feita para tentar remover as regiões que são irrelevantes na imagem. A Figura 15 apresenta a arquitetura proposta.

O subsistema de detecção utiliza o espaço de cores *Hue, Saturation and Intensity* (HSI) e então divide as áreas candidatas em múltiplas pequenas partes de acordo com a cor, textura, tamanho e similaridade. Essas áreas candidatas são então passadas como entrada

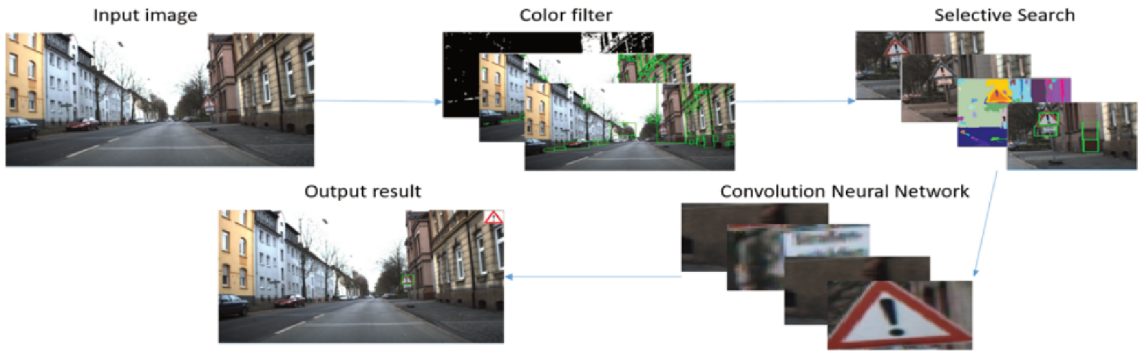

Figura 15 – Arquitetura Proposta

Fonte: (HUANG; LIN; CHANG, 2017)

para o subsistema de reconhecimento. A Figura 16 apresenta o processo de detecção aplicado na imagem.

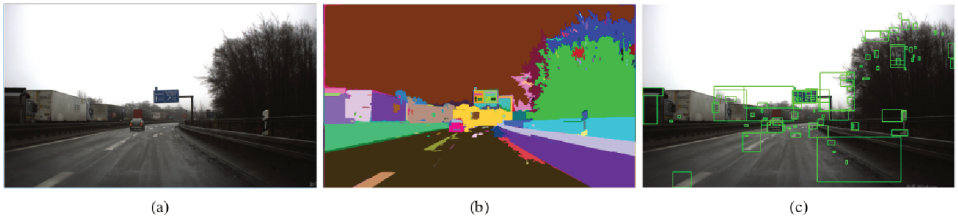

Figura 16 – Diagrama do processo de detecção. (a) Imagem base; (b) Segmentação; (c) Resultado

Fonte: (HUANG; LIN; CHANG, 2017)

O subsistema de reconhecimento utiliza CNN para a tarefa do reconhecimento. Nesse subsistema, foram utilizadas duas arquiteturas para comparação, a AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012) e a GoogLeNet (SZEGEDY et al., 2015).

Os *datasets* utilizados foram o GTSDB para a detecção, usando um total de 300 imagens e o GTSRB para o reconhecimento, usando 39209 imagens no treinamento divididas em 43 categorias.

Em (DHAR et al., 2017), é proposto um sistema de reconhecimento de sinais de trânsito composto por três módulo: detecção, verificação da forma do objeto detectado e reconhecimento. O sistema é desenhado e simulado para sinais de perigo com a forma triangular e a borda vermelha.

No módulo de detecção o espaço de cores das imagens são convertidos para o modelo *Hue, Saturation and Value* (HSV). É aplicado um limiar à matiz e à saturação da imagem, fazendo com que somente objetos que sejam próximo ao vermelho fiquem na imagem.

No módulo de verificação de forma, é aplicado um filtro que usa aproximação de contorno, e após esse filtro as regiões de interesse identificadas ficam mais corretas. A Figura 17 exemplifica o processo realizado dos dois módulos apresentados.

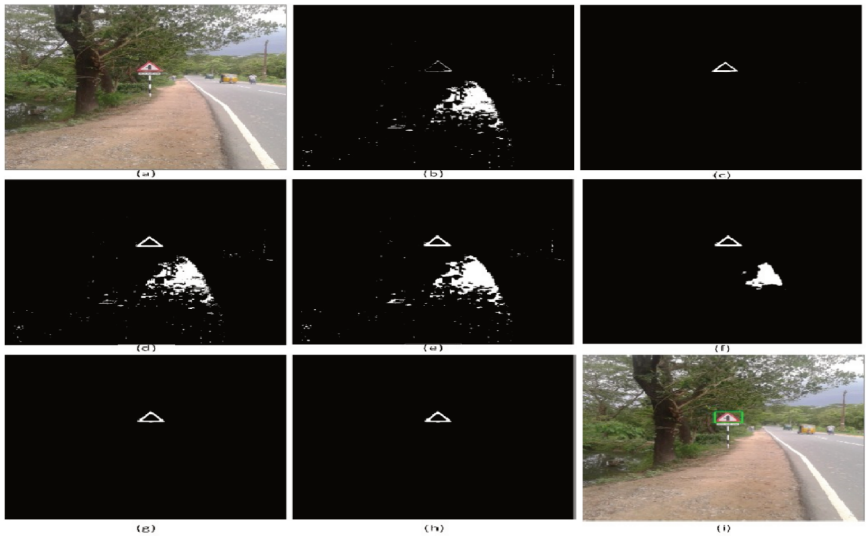

Figura 17 – Detecção de sinal. a) imagem original b) e c) aplicando limiares d) *merge* das máscaras (b+c) e) fechamento morfológico f) depois do filtro de área g) depois do filtro de *aspect ratio* h) depois do filtro de forma i) ROI desejado

Fonte: (DHAR et al., 2017)

No módulo de reconhecimento é utilizada uma CNN para a identificação das classes dos sinais. A arquitetura da CNN utilizada é apresentada na Tabela 1.

Em (BOUJEMAA et al., 2017), são apresentadas duas abordagens na detecção e segmentação de sinais de trânsito. A primeira baseada na segmentação de cores e CNN (C-CNN) e a segunda é baseada em Fast R-CNN.

O método C-CNN consiste em selecionar um conjunto de regiões de interesse aplicando um limiar em uma determinada cor na imagem de entrada, reduzindo o espaço de busca. Após isso uma CNN já treinada é usada para classificar se a região de interesse contém ou não um sinal de trânsito. Após esse passo é usada outra CNN com a mesma arquitetura para fazer o reconhecimento do sinal de trânsito detectado.

O método Fast R-CNN tem como entrada uma imagem inteira e também um conjunto de objetos de interesse que são calculados por um algoritmo externo. Após isso é utilizado uma CNN para criar os mapas de atributos para cada uma das regiões de interesse. Essas regiões são enviadas para uma rede totalmente conectada para classificação e por último é gerado o *bounding box* da imagem classificada.

No método C-CNN foi utilizado o OpenCV API (OPENCV, 2019) para a segmentação de cor e as CNNs utilizadas em ambas as abordagens foram compostas pelo Keras (KERAS, 2019) e pelo TensorFlow. O *dataset* utilizado foi o *German Traffic Sign.*

Figura 18 – C-CNN

Fonte: (BOUJEMAA et al., 2017)

Figura 19 – Fast R-CNN

Fonte: (BOUJEMAA et al., 2017)

As Figuras 18 e 19 apresentam, respectivamente, os resultados da abordagem C-CNN e exemplos de falso positivos na abordagem Fast R-CNN.

Em (BRUNO et al., 2018), é apresentada uma arquitetura que utiliza dados 3D e também imagens 2D para reconhecimento e classificação de sinais de trânsito. É utilizado o Velodyne LiDAR e um par de câmeras para resolver o problema.

Os dados fornecidos pelo LiDAR, em forma de nuvem de pontos, representam a cena em 3D. Com base nesses dados, foi aplicado o descritor 3D-CSD, que parte do princípio que cada classe de objeto tem um contorno 3D único.

Esse contorno é então enviado para uma ANN que classifica se esse objeto é ou não um sinal de trânsito. Caso seja um sinal de trânsito, a ANN informa também a possível coordenada do objeto. A Figura 20 apresenta a arquitetura dessa ANN.

Com essa coordenada, a imagem 2D do sinal é segmentada e enviada para um classificador baseado em CNN, a qual será responsável por identificar qual o tipo de sinal foi detectado. Essa rede foi implementada utilizando a biblioteca para aprendizado de máquina TensorFlow (TENSORFLOW, 2019).

Para o treinamento e os testes foram utilizados dois *datasets* diferentes. O *dataset* German traffic signs dataset (INI) foi utilizado para o treinamento da CNN, enquanto o *dataset* KITTI Vision Benchmark Suite (Kitti) foi utilizado para o teste do sistema.

Em (WANG, 2018), é apresentada uma arquitetura para detecção e reconhecimento de sinais de trânsito tendo como foco os sinais de limite de velocidade e proibição de virar a esquerda e direita. A arquitetura usa a estrutura de rede base VGG-16 (SIMONYAN; ZISSERMAN, 2014) e o algoritmo *Single Shot MultiBox Detector* (SSD) (LIU et al., 2016) para detectar e identificar os sinais de trânsito.

Na estrutura proposta, após completar as operações de convolução e *pooling* na ima-

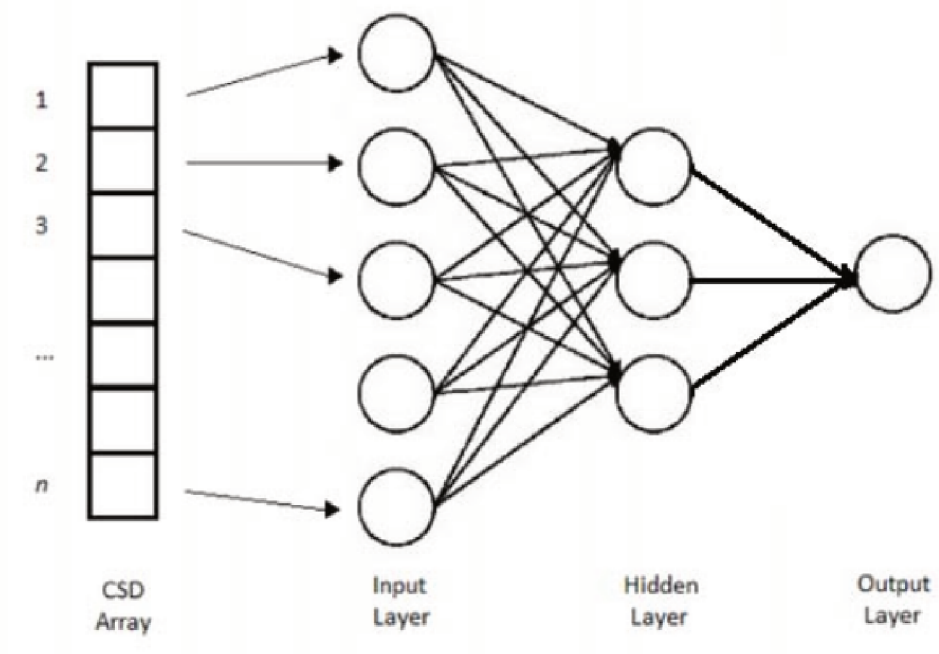

Figura 20 – Arquitetura da rede neural artificial

Fonte: (BRUNO et al., 2018)

gem, o resultado é enviado para 3 camadas totalmente conectadas, e a saída da última camada totalmente conectada é usada como entrada para a camada *Softmax*, na qual o resultado do reconhecimento é obtido.

As Figuras 21 e 22 apresentam tipos de sinais de trânsito utilizados no trabalho e a estrutura da rede proposta, respectivamente.

Tabela 1 – Arquitetura da Rede de Convolução.

Camadas	**Descrição**
Camada de entrada	imagens 24x24x3
Convolução-1	Convolução e função ReLU (*Rectified Linear Unit*)
Pool-1	Max pooling
Convolução-2	Convolução e função ReLU
Pool-2	Max pooling
Local-3	Camada totalmente conectada com ReLU
Local-4	Camada totalmente conectada com ReLU
Softmax	Classificação do resultado

Fonte: (DHAR et al., 2017)

Em (VENNELAKANTI et al., 2019), a detecção e o reconhecimento de sinais é realizado em dois passos. Um passo em que se obtêm a região de interesse e o outro que é o reconhecimento do sinal de trânsito.

No passo de detecção, é utilizado uma detecção baseada na cor do sinal de trânsito,

Figura 21 – Tipos de sinais de trânsito

Fonte: (WANG, 2018)

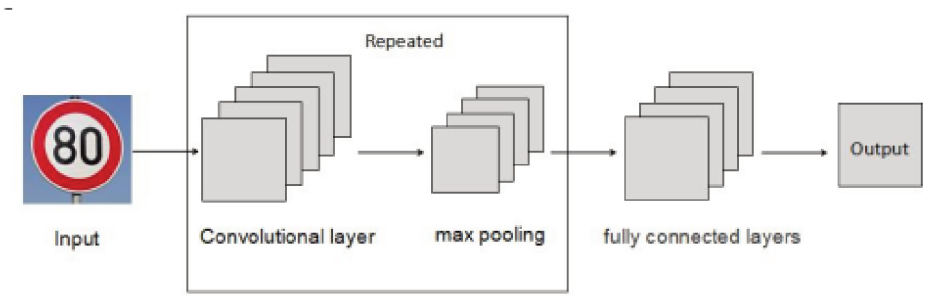

Figura 22 – Estrutura da Rede

Fonte: (WANG, 2018)

nesse trabalho a cor vermelha. Após isso é realizado a detecção baseado na forma do objeto, nesse trabalho usam a forma triangular e circular. Após encontrar essa região de interesse é então feita uma validação do sinal, que se utiliza de um *threshold*. A imagem de saída é então usada como entrada para o passo de reconhecimento.

No passo de reconhecimento é utilizado o *framework* TensorFlow do Google. O treinamento é realizado com os conjuntos de dados *German Traffic Sign Benchmark* e *Belgium Traffic Signs*. A arquitetura da CNN utilizada é apresentada na Tabela 2. Para melhorar a acurácia foi utilizado um conjunto de 3 CNNs, cada uma delas foi treinada separadamente e depois foram agregadas para formar uma rede única.

Em (BRUNO, 2020), é proposto um sistema que utiliza imagens 2D, utiliza nuvens de pontos, para montar os objetos 3D e com essas duas informações gerar detecção de sinais

Tabela 2 – Arquitetura da CNN.

Camadas
Convolução (32 filtros de tamanho 3x3, passo 1, ReLU)
Convolução (32 filtros de tamanho 3x3, passo 1, ReLU)
Max pooling (2x2, passo 2) + *Dropout* (0.2)
Convolução (64 filtros de tamanho 3x3, passo 1, ReLU)
Convolução (64 filtros de tamanho 3x3, passo 1, ReLU)
Max pooling (2x2, passo 2) + *Dropout* (0.2)
Convolução (128 filtros de tamanho 3x3, passo 1, ReLU)
Convolução (128 filtros de tamanho 3x3, passo 1, ReLU)
Max pooling (2x2, passo 2) + *Dropout* (0.2)
Camada totalmente conectada (512 nós, ReLU) + *Dropout* (0.2)
Camada totalmente conectada (*SoftMax*)

Fonte: (VENNELAKANTI et al., 2019)

de trânsito com mais acurácia. Após a detecção e classificação é utilizado um sistema de Atenção Visual, que leva em consideração as informações obtidas das placas, para gerar informações referentes a prioridade. A Figura 23 apresenta uma situação com sinais de trânsito em conflito.

A detecção das placas das imagens 2D foi feita utilizando a arquitetura YOLO. Utilizando somente uma classe, obteve uma acurácia de 92%. Após a detecção, é utilizada a rede Inception-V3 para fazer a classificação das placas.

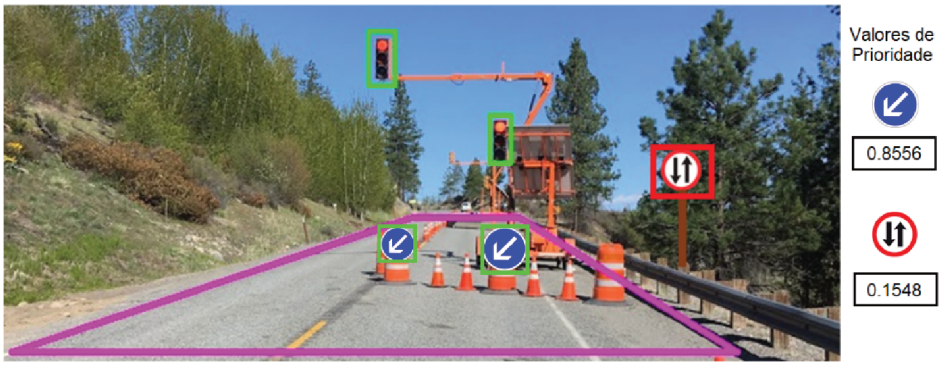

Figura 23 – Sistema de Atenção Visual

Fonte: (BRUNO, 2020)

2.7.3 Inventário de Sinais de Trânsito

Em (BASCóN et al., 2008), é apresentado um sistema de inventário de sinais de trânsito composto de três partes: segmentação, detecção e reconhecimento.

A segmentação é baseada na informação de cor e utiliza um *Support Vector Machines* (SVM) para classificar os pixeis da imagem. A detecção é baseada na forma do sinal segmentado, e após a forma ser determinada, é realizada a localização do objeto na imagem. O reconhecimento é implementado com SVM utilizando o *kernel gaussian*. Por último a definição da localização geográfica do sinal de trânsito classificado é feita utilizando estereoscopia.

A Figura 24 mostra um exemplo de reconhecimento para diferentes formas de sinais de trânsito.

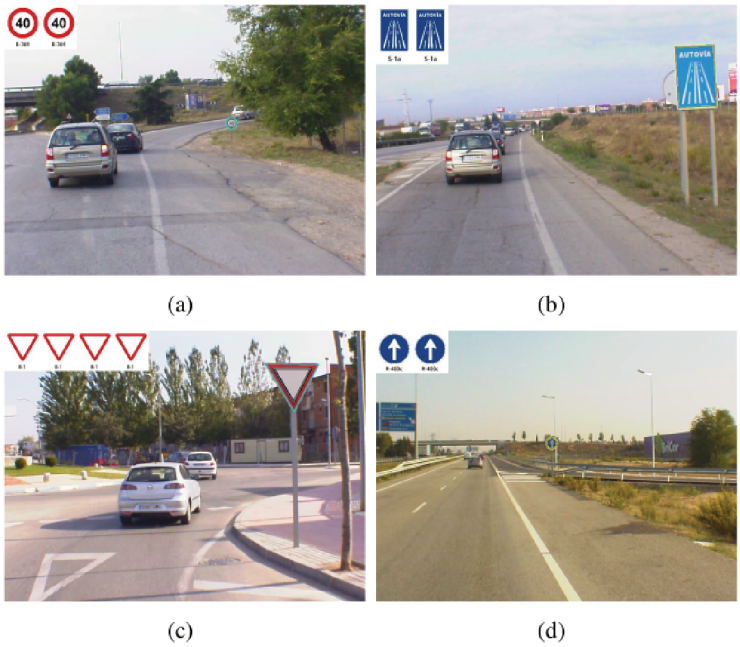

Figura 24 – Exemplos de diferentes formas.

Fonte: (BASCóN et al., 2008)

Em (HIDALGO et al., 2013), é apresentado um sistema de inventário de placas de trânsito que utiliza tag's *Radio-Frequency Identification* (RFID). As tag's são instaladas no suporte vertical dos sinais de trânsito e as informações do sinal (significado, posição, etc) são codificadas na tag.

E para atualizar o inventário é somente um veículo com um leitor RFID ser utilizado nas estradas e ler as informações constantes nas tag's dos sinais de trânsito.

Em (HAZELHOFF; CREUSEN, 2013), é apresentado um sistema de classificação de sinais de trânsito que consiste em três partes. A primeira parte trata a classificação de um sinal detectado nos dados de entrada. A segunda parte realiza uma predição, utilizando dois métodos diferentes, para ter a confiabilidade da classificação do primeiro estágio. A terceira parte recupera a classe do sinal de trânsito, após a predição e utiliza triangulação para computar a posição geográfica das placas de trânsito.

A Figura 25 apresenta uma visão geral das partes que compõem o sistema proposto.

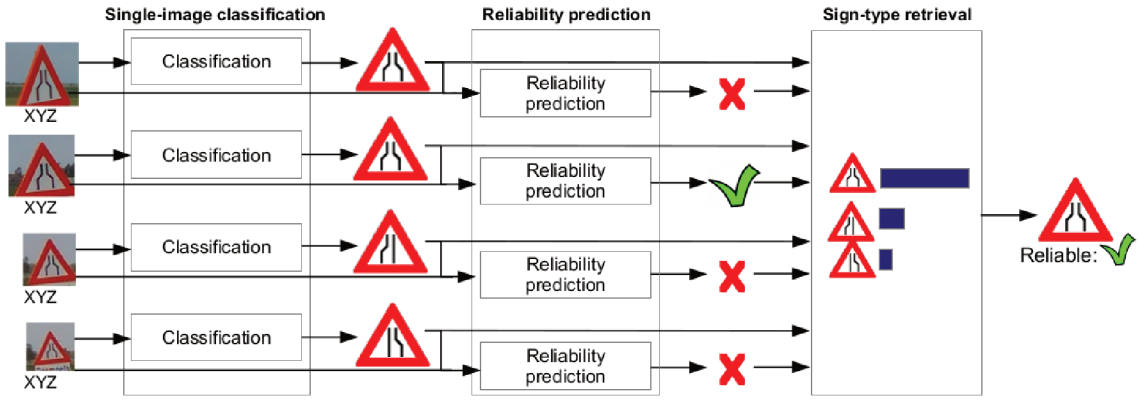

Figura 25 – Visão geral do sistema de classificação

Fonte: (HAZELHOFF; CREUSEN, 2013)

O módulo de classificação trabalha com uma única imagem utilizando tanto informações estruturais quanto características chaves da imagem. O módulo de predição aplica tanto *template matching* quanto o algoritmo *One-versus-All*, selecionando o tipo de sinal com a maior correspondência.

Em (HARASTHY; TURAN; OVSENIK, 2013), é apresentado um sistema de inventário baseado na correlação ótica. O sistema é composto por cinco blocos principais: Captura de Vídeo, Módulo GPS, Pré-processamento, Correlação Ótica e Exibição.

Como no sistema proposto em (SOLUS; OVSENIK; TURAN, 2015), também utilizam uma base de dados com as informações de placas em uma determinado posição GPS. Ao processar o vídeo capturado, é passado a posição GPS do vídeo, e obtido do banco qual os sinais que estão naquela posição.

Um pré-processamento das imagens é realizado para conseguir a região de interesse que representa o sinal de trânsito na imagem, esse pré-processamento utiliza o filtro de cores usando o modelo de cores HSV. A região de interesse como também o sinal encontrado no banco de dados são enviados para realizar a correlação ótica e com a saída desse processo de correlação é possível validar se o sinal detectado na imagem está correto baseado nas informações da base de dados.

A Figura 26 apresenta o esquema de hardware proposto para o sistema.

Em (SOLUS; OVSENIK; TURAN, 2015), é apresentado um sistema de inventário de sinais de trânsito composto por duas fases. A primeira consiste em detectar a região de interesse que representa um sinal de trânsito, essa detecção é baseado em filtro de cores e detecção da forma. A segunda fase consiste no reconhecimento do sinal, o qual é feito usando uma correlação com um banco de dados com as placas já localizadas na região que está sendo processada.

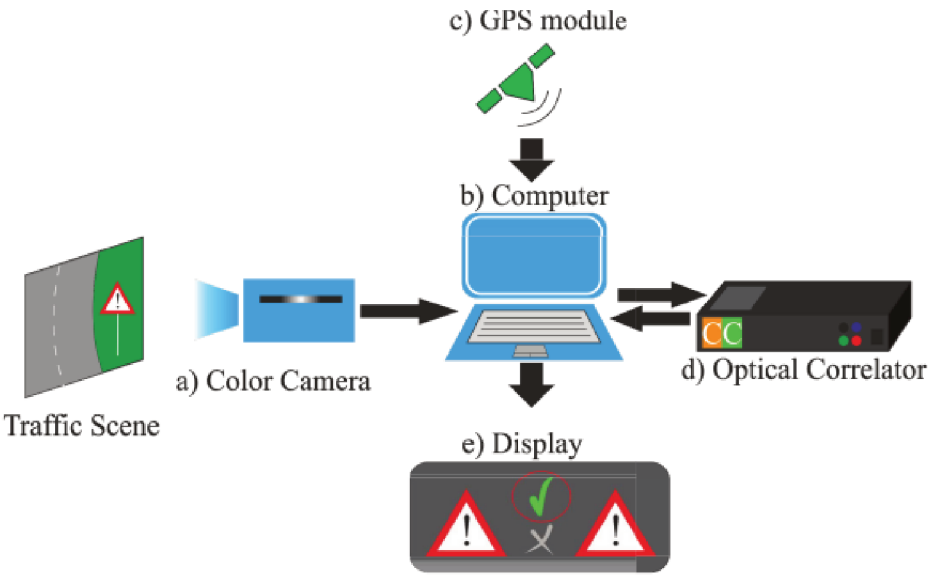

Figura 26 – Esquema do sistema proposto

Fonte: (HARASTHY; TURAN; OVSENIK, 2013)

A Figura 27 apresenta o esquema de blocos do sistema proposto. E a Figura 28 apresenta uma das telas do sistema que permite ver as placas encontrada no vídeo e as referências usadas para correlação.

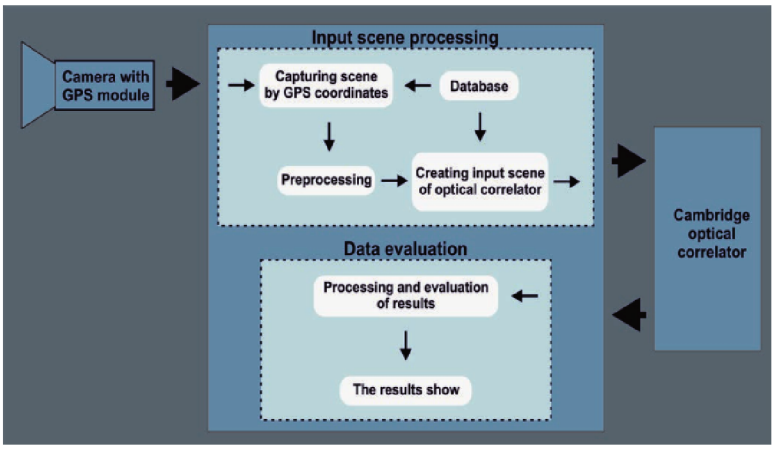

Figura 27 – Esquema do sistema proposto

Fonte: (SOLUS; OVSENIK; TURAN, 2015)

Em (BALALI; RAD; GOLPARVAR-FARD, 2015), é apresentado um novo sistema para criar e mapear inventários de sinais de trânsito usando imagens do *Google Street View*. No artigo, a detecção e classificação é realizada usando HOG + Cor com um classificador linear SVM.

Figura 28 – Interface do sistema proposto

Fonte: (SOLUS; OVSENIK; TURAN, 2015)

O sistema utiliza informações da API do Google Street View para poder incluir informações de GPS nos sinais de trânsito detectados e classificados.

A Figura 29 apresenta a formação do HOG utilizando a janela de deslizamento sobre a imagem, como também o HOG correspondente.

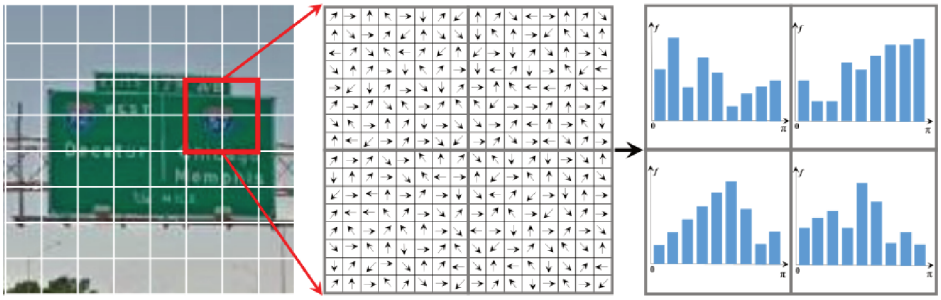

Figura 29 – Exemplo da formação do HOG

Fonte: (BALALI; RAD; GOLPARVAR-FARD, 2015)

A Figura 30 apresenta o resultado da interface desenvolvida no trabalho.

Em (SOILáN et al., 2016), é proposto um método para identificar propriedades geométricas e semânticas dos sinais de trânsito, utilizando nuvens de pontos coletadas por um laser scanner móvel e imagens coletadas por câmeras RGB.

A detecção do sinal de trânsito é feita utilizando a nuvem de pontos, onde cada sinal de trânsito é isolado, e os parâmetros geométricos do sinal é extraído. Os pontos 3D de cada sinal de trânsito detectados são então reprojetados em uma imagem 2D.

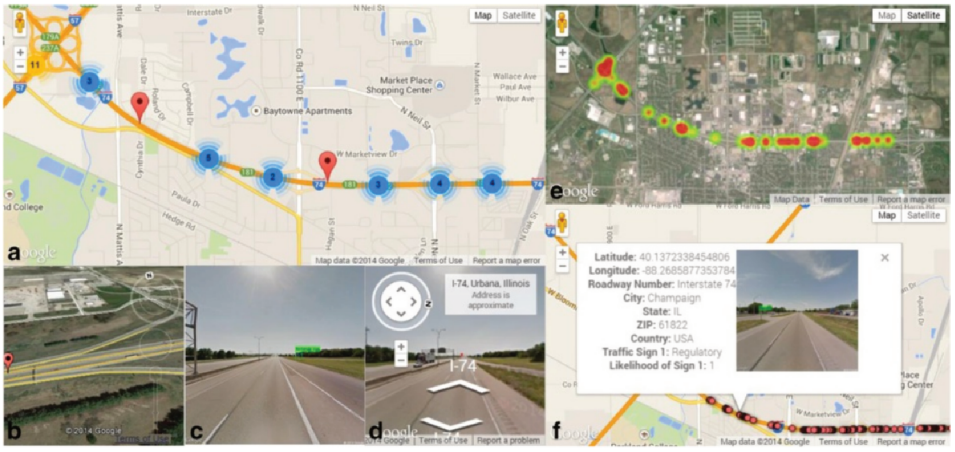

Figura 30 – Interface do sistema desenvolvido; **a)** aglomeração dos sinais detectados; **b)** visualização no Google Earth da localização do sinal; **c)** sinal detectado na imagem do Google Street View; **d)** imagem do Street View da localização do sinal; **e)** propabilidade da existência dos sinais no mapa de calor; **f)** informação de todos os sinais detectados

Fonte: (BALALI; RAD; GOLPARVAR-FARD, 2015)

O reconhecimento da imagem é realizado utilizando a cor e a forma dos sinais detectados. Primeiramente é feita a classificação da super classe do sinal de trânsito utilizando SVM e HOG, após isso, a imagem dos sinais de trânsito são redimensionadas e novamente utilizado HOG e SVM para classificação final do sinal de trânsito.

As Figuras 31 e 32 apresentam o *workflow* geral do método proposto e o *workflow* do processo de reconhecimento do sinal de trânsito, respectivamente.

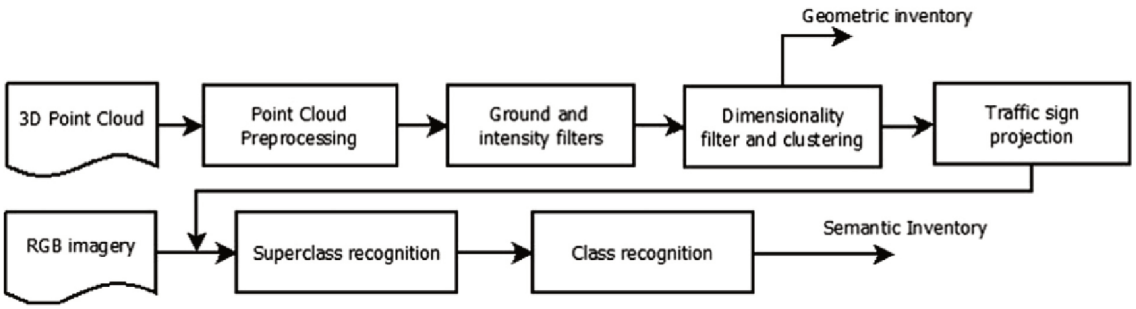

Figura 31 – *Workflow* geral do método proposto.

Fonte: (SOILáN et al., 2016)

Em (WEN et al., 2016), é proposto um processo de inspeção espacial de sinais de trânsito baseado em dados de um MLS, demonstrando a aplicação do mesmo em um inventário automático de sinais de trânsito.

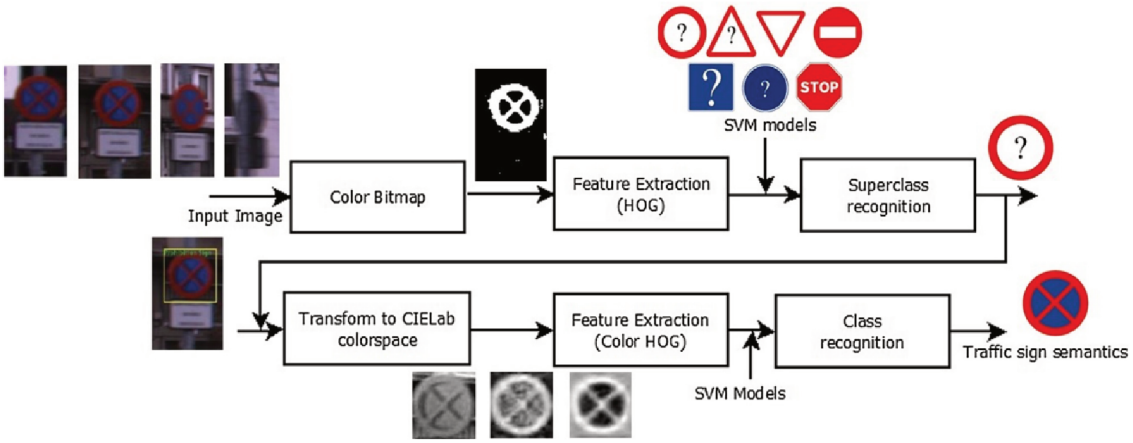

Figura 32 – *Workflow* do reconhecimento do sinal de trânsito.

Fonte: (SOILáN et al., 2016)

A Figura 33 apresenta o *flowchart* do método proposto para reconhecimento, posicionamento e inspeção de localização de sinais de trânsito.

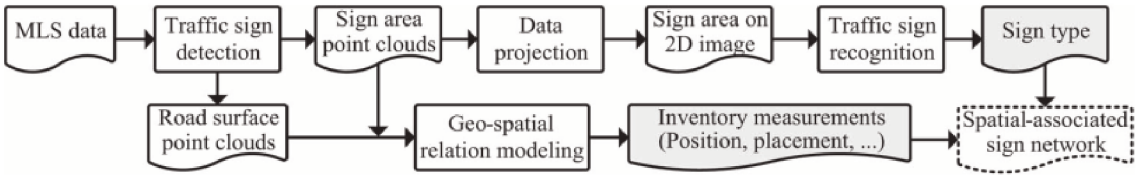

Figura 33 – *Flowchart* do método proposto

Fonte: (WEN et al., 2016)

A detecção dos sinais de trânsito é realizada inicialmente pelos dados do MLS, utilizando tanto a estrutura linear do poste do sinal de trânsito como também a retrorefletância da superfície do sinal. Após esse primeira etapa os pontos da nuvem de pontos são projetados em uma imagem 2D, provendo a informação de aparência do sinal para posterior reconhecimento.

O reconhecimento do sinal de trânsito é feito utilizando o histograma *HUE* concatenando o mesmo com o descritor *Scale-invariant Feature Transform* (SIFT). O HOG é utilizado como um descritor local do sinal de trânsito, e um classificador SVM é utilizado como modelo de reconhecimento.

A Figura 34 apresenta uma ilustração do processo para detecção e projeção dos pontos do sinal de trânsito em uma imagem 2D.

Em (BALADO et al., 2020), apresenta um método para mapeamento de sinais de trânsito baseado nos dados adquiridos com imagens e nuvem de pontos. Esse método

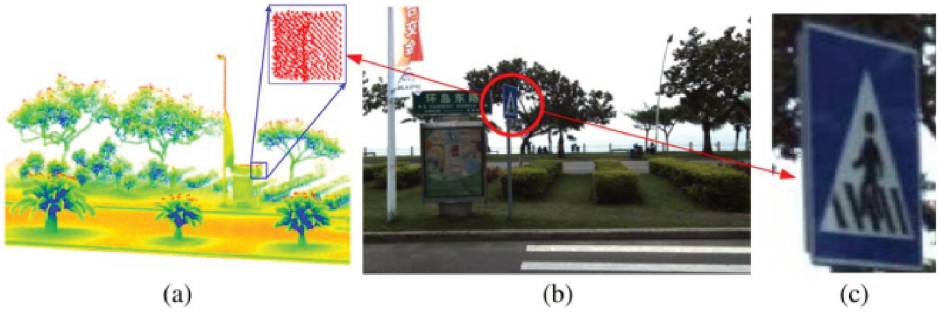

Figura 34 – Ilustração da projeção do sinal em uma imagem 2D: a) Nuvem de ponsto e area do sinal detectado, b) imagem adquirida com os dados do MLS, c) sinal projetado em uma imagem 2D.

Fonte: (WEN et al., 2016)

consiste em quatro processos principais:

1. Sinais de Trânsito são detectados nas imagens;
2. Sinais de Trânsito são reconhecidos;
3. Sinais de Trânsito são geolocalizados pela projeção desses sinais detectados na nuvem de pontos;
4. Múltiplas detecções do mesmo sinal em diferentes imagens são filtradas.

A Figura 35 apresenta o fluxo do método proposto.

No processo de detecção dos sinais de trânsito é utilizado a arquitetura RetinaNet e como extrator de características foi utilizada a ResNet50. Esse processo faz a detecção do sinal de trânsito baseado em sua forma.

No processo de classificação ou reconhecimento, o *bounding box* do sinal de trânsito detectado é passado para a rede InceptionV3 que faz a classificação final do sinal de trânsito para o inventário, é necessário alguns dos *bounding boxes* serem redimensionados, para adequarem a entrada permitida nessa rede.

O método obteve 92.5% de classificações corretas dos sinais detectados. E ao combinar os dados das imagens e da nuvem de pontos, obteve 97.5% de precisão no posicionamento dos sinais de trânsito.

2.7.4 Arquiteturas CNNs

Em (REN et al., 2016), é apresentado a rede Faster R-CNN, uma rede convolucional única e unificada para detecção de objeto. É uma arquitetura composta por dois estágios:

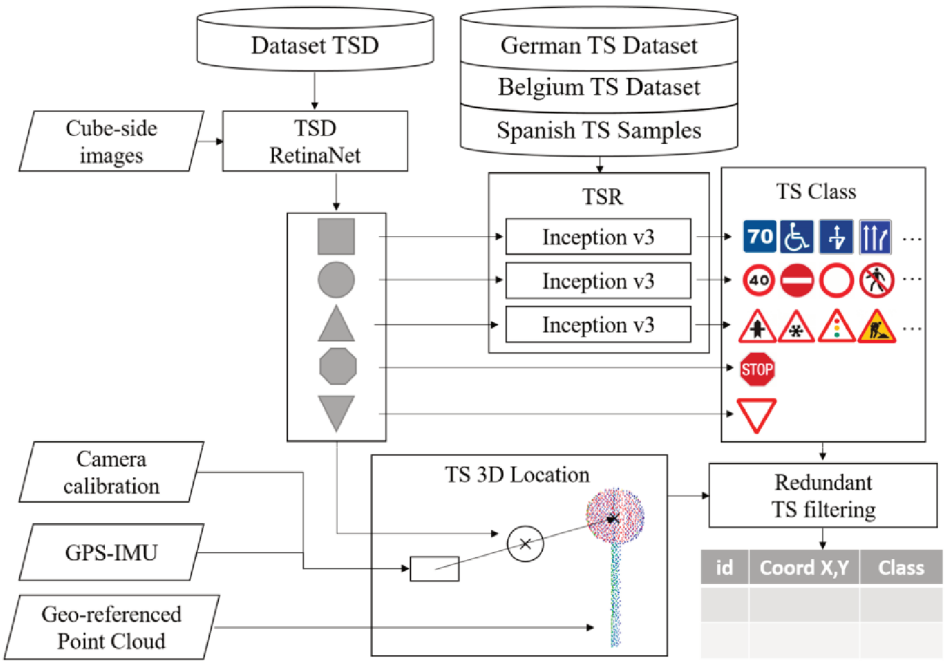

Figura 35 – Fluxo do método

Fonte: (BALADO et al., 2020)

- *Region Proposal Network* (RPN): esse estágio é uma rede convolucional responsável por propor regiões que são consideradas objetos, cada uma com sua pontuação de objeto.

- Fast R-CNN: esse estágio é uma rede convolucional responsável por detectar e classificar os objetos propostos que foram detectados na rede anterior.

No primeiro estágio são geradas várias âncoras em cada localização em que um janela de deslizamento se encontra, essas âncoras são de tamanhos e *aspect ratios* diferentes. O método proposto para esse primeiro estágio classifica e regride os *bounding boxes* baseado nas âncoras encontradas. No segundo estágio, as regiões propostas são passadas como entrada, para que possa ser feita a detecção e classificação do objeto encontrado.

Os dois estágios compartilham algumas das camadas convolucionais. O treinamento é realizado usando 4 passos com treinamento alternado. No primeiro passo é treinada a rede RPN. No segundo passo é treinada a rede Fast R-CNN de forma totalmente separada, mas usando as regiões propostas pelo primeiro passo. No terceiro passo usam a rede de detecção para inicializar o treinamento da RPN, mas fixando as camadas convolucionais compartilhadas e fazem uma melhoria fina nas camadas únicas da RPN. E no quarto passo, ainda mantém fixas as camadas compartilhadas e faz um melhoria fina nas camadas especificas da Fast R-CNN. As Figuras 36 e 37 apresentam a arquitetura geral da Faster R-CNN e também a *Region Proposal Network*.

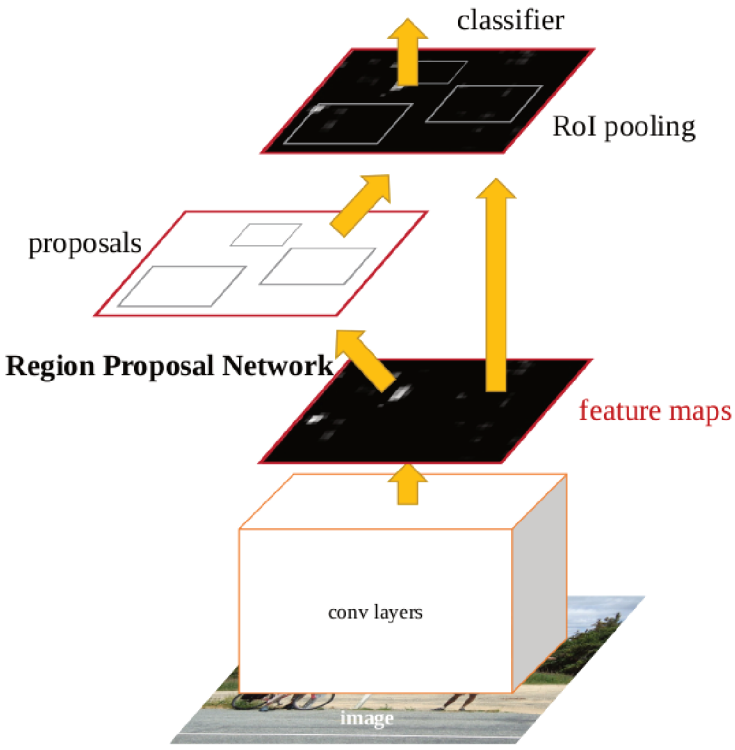

Figura 36 – Arquitetura da Faster RCNN

Fonte: (REN et al., 2016)

Em (LIN et al., 2018), é apresentada a RetinaNet, uma rede de estágio único para detecção de objetos. É composta por um *backbone* e duas sub-redes com tarefas específicas.

Para a detecção dos objetos é utilizado a FPN, na qual cada nível da pirâmide pode ser usado para detectar objetos em diferentes escalas. A rede FPN é utilizada no topo da arquitetura ResNet. Em cada um dos níveis da FPN são geradas âncoras, similar à (REN et al., 2016).

Após o *backbone*, são incluídas mais duas sub-redes, a primeira, *Classification Subnet*, responsável por predizer a probabilidade da presença de objetos em cada âncora e com as classes disponíveis. A segunda, *Box Regression Subnet*, tem o propósito de regredir cada âncora para um objeto, se existir.

As maiores predições de todos os leveis da FPN são combinadas e aplica-se uma *non-maximum supression* para produzir as detecções finais.

A Figura 38 apresenta a arquitetura da RetinaNet.

Em (REDMON; FARHADI, 2018), é apresentado a rede YOLOv3 de estágio único. A YOLOv3 inicia com a predição de *bounding box* usando clusters de dimensão como âncoras. Para cada um dos *boxes* é feita a predição de classe do mesmo usando a classificação multi rótulo.

YOLOv3 faz a predição dos *boxes* em três escalas diferentes, extraindo as caracterís-

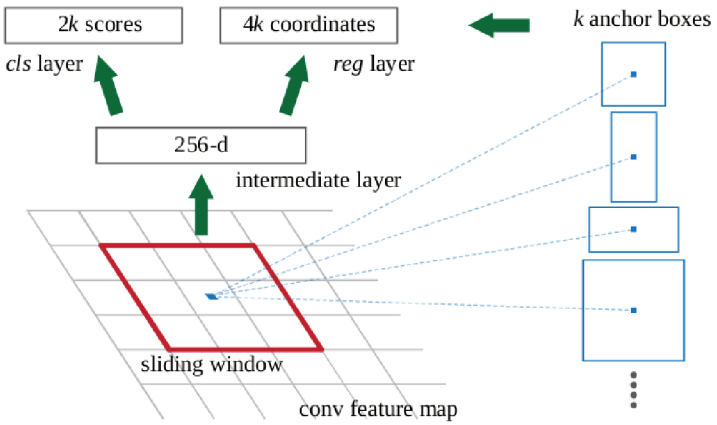

Figura 37 – *Region Proposal Network*

Fonte: (REN et al., 2016)

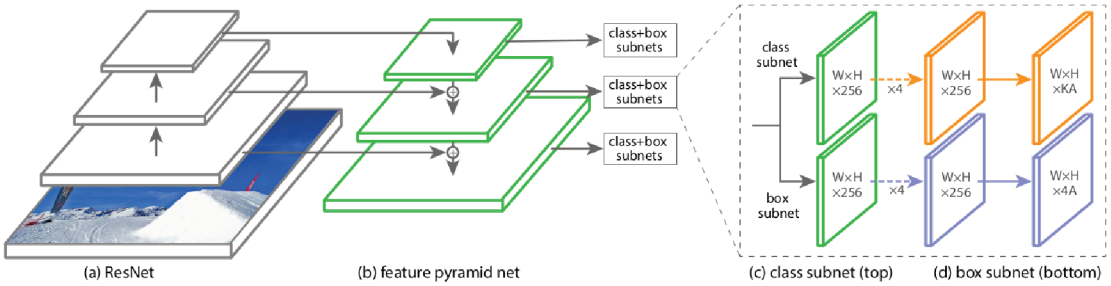

Figura 38 – Arquitetura da RetinaNet. a) arquitetura *feedforward* ResNet, b) FPN usada no topo da Resnet, c) sub-rede para classificar as âncoras, d) sub-rede para regredir as âncoras para objetos.

Fonte: (LIN et al., 2018)

ticas dessas escalas usando um conceito similar ao da FPN.

Para a extração de características foi usada uma nova rede, que é um híbrido entre a rede usada na YOLOv2, *Darknet-19* e uma nova rede residual. Essa rede é chamada como *Darknet-53*. A Figura 39 apresenta a arquitetura da *Darknet-53*.

Em (JOCHER et al., 2020), é apresenta a rede YOLOv5, sendo uma melhoria da rede YOLOv3. Segundo (YAP et al., 2020) as melhorias são focadas principalmente em incorporar as técnicas do estado da arte para funções de ativação, aumento de dados e pós-processamento na arquitetura YOLO.

De acordo com (YAP et al., 2020) a rede YOLOv5 incluí quatro diferentes modelos, sendo o menor o YOLOv5s, com 7,1M de parâmetros e o maior YOLOv5x, com 89M de parâmetros e 284 camadas. Ainda de acordo com (YAP et al., 2020) o modelo YOLOv5x usa um detector de dois estágios que consiste de uma Cross Stage Partial Network (CSPNet)

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figura 39 – Darknet-53.

Fonte: (REDMON; FARHADI, 2018)

como *backbone* e a Path Aggregation Network (PANet) para segmentação de instância.

A Tabela 3 apresenta a arquitetura do modelo YOLOv5x.

2.7.5 Conclusão

Nos trabalhos apresentados, especificamente os que tratam dos inventários de sinais de trânsito, alguns utilizam os MLS que têm um alto custo e outros precisam já ter um banco de dados dos sinais já criado para depois realizar o inventário.

No método proposto neste trabalho, utiliza-se dados e imagens de câmeras GoPro Hero 6 e aplica-se algoritmos para classificação de dados tabulares, como também arquiteturas CNNs para trabalhar com as imagens captadas. Possibilitando a criação de inventários usando ferramentas com um custo baixo e sem a necessidade de ter uma base previamente criada.

Tabela 3 – Arquitetura do modelo YOLOv5x.

Tipo	**Filtros**	**Tamanho**
	Backbone	
Focus	12	3 x 3
Convolutional	160	3 x 3
BottleneckCSP	4x 160	1 x 1 + 3 x 3
Convolutional	320	3 x 3
BottleneckCSP	12x 320	1 x 1 + 3 x 3
Convolutional	640	3 x 3
BottleneckCSP	12x 640	1 x 1 + 3 x 3
Convolutional	1280	3 x 3
SPP		
BottleneckCSP	4x 1280	1 x 1 + 3 x 3
	Head	
Convolutional	640	1 x 1
Upsample	2	
BottleneckCSP	4x 640	1 x 1 + 3 x 3
Convolutional	320	1 x 1
Upsample	2	
BottleneckCSP	4x 320	1 x 1 + 3 x 3
Convolutional	320	3 x 3
BottleneckCSP	4x 640	1 x 1 + 3 x 3
Convolutional	640	3 x 3
BottleneckCSP	4x 1280	1 x 1 + 3 x 3
Detecção		

Fonte: Yap et al. (2020)

Capítulo 3

Proposta

Neste capítulo é apresentado o método de pesquisa para classificação de lombadas e criação do inventário de sinais de trânsito de baixo custo utilizando vídeos e dados de localização provenientes de câmeras GoPro.

3.1 Metodologia

O fluxo utilizado na proposta começa com a aquisição dos dados utilizando câmeras GoPro. Com os dados adquiridos, são realizados dois experimentos: o primeiro para classificação de lombadas, que utiliza informações de uma única câmera, da qual se extrai dados de sensores para uma posterior sincronização, após isso realiza a classificação dentre os diferentes tipos definidos e por fim o mapeamento dos dados classificados; o segundo utiliza vídeos das câmeras que compõem a plataforma para detecção e classificação de sinais de trânsito, em seguida é realizado um inventário da quantidade por meio de uma das câmeras, posteriormente é feita a extração de dados do GPS para finalmente gerar um mapa com os sinais detectados.

A Figura 40 apresenta o fluxo utilizado nessa proposta, nas próximas seções serão discutidas as etapas apresentadas.

3.1.1 Aquisição

A aquisição dos dados é realizada por um veículo adaptado com uma plataforma para instalação de até cinco câmeras GoPro Hero 6, sendo três câmeras frontais e duas laterais, além de um GPS e uma IMU. A Figura 41 apresenta um veículo com o protótipo da plataforma. Em nossa pesquisa, os dados da câmera central são utilizados para os experimentos relacionados às lombadas e os vídeos adquiridos pela câmera central ou as três câmeras frontais são utilizados nos experimentos relacionados ao inventário de sinais de trânsito.

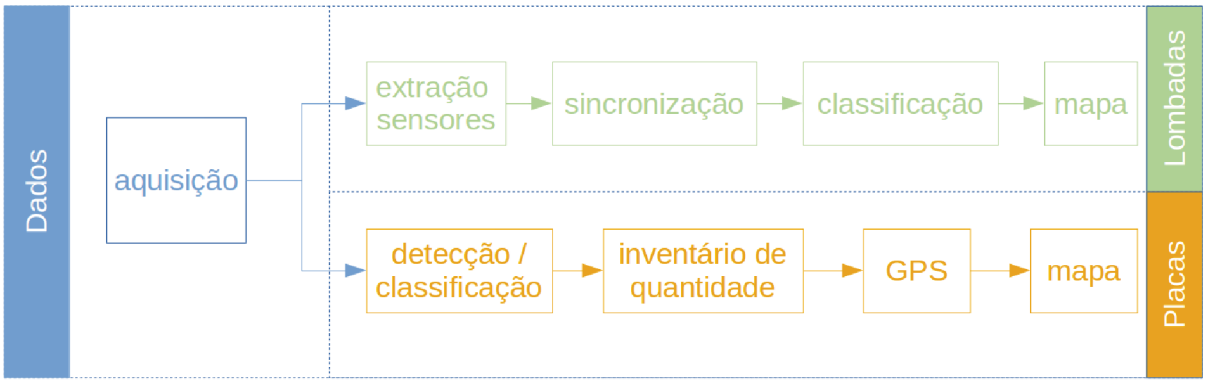

Figura 40 – Diagrama de fluxo de trabalho

Fonte: Elaborado pelo autor

Figura 41 – Veículo com a plataforma de captação (protótipo inicial).

Fonte: (OLIVEIRA et al., 2019)

3.1.2 Lombadas

Nessa subseção são apresentados os passos para realizar a classificação das lombadas. A proposta é detectar e classificar lombadas utilizando diferentes algoritmos de ML além de extrair a posição geográfica das mesmas.

3.1.2.1 Extração dos Dados dos Sensores

As câmeras utilizadas possuem os seguintes sensores: Acelerômetro, Giroscópio e GPS. Após a coleta dos dados é realizado um pré-processamento para extrair informações tais como aceleração, rotação e coordenadas que a câmera gera como metadados nos vídeos. Para esta tarefa é utilizado um *script* Python e a ferramenta FFmpeg (FFmpeg, 2018) para extrair os metadados do vídeo criando um arquivo ".bin".

Após a geração do arquivo, é utilizado a biblioteca "gopro-utils"(gopro-utils, 2018) que gera quatro arquivos ".csv"que contêm a aceleração e rotação 3D, como também as coordenadas e o respectivo tempo em que a coordenada foi coletada.

3.1.2.2 Sincronização dos Dados

Os sensores (giroscópio, GPS e acelerômetro) coletam dados em diferentes taxas, e por esse motivo é necessário realizar a sincronização entre os dados adquiridos para produzir um único arquivo com as coordenadas, aceleração e informações do giroscópio em um momento específico.

A primeira parte do *script* converte o tempo do GPS, que está em *timestamp*, para um tempo local. Após essa conversão é feita uma interpolação entre o arquivo com informação de rotação, aceleração e coordenadas. A interpolação utiliza uma abordagem linear que fornece a precisão esperada para a posição e se baseia na seguinte equação:

$$y_i = y_1 + (y_2 - y_1) \times \frac{(t_i - t_1)}{(t_2 - t_1)} \tag{4}$$

onde y_i é o valor de parâmetro no i-ésimo momento, com i entre os momentos 1 e 2; y_n é o valor de parâmetro da observação n e t_n o momento da observação n.

Com os dados sincronizados, é possível extrair atributos tais como o pico de um período particular, o momento em que o veículo foi elevado na travessia de uma lombada, a média de aceleração e média de velocidade.

3.1.2.3 Classificação de Lombadas

Com os dados sincronizados, é possível extrair atributos tais como o pico de elevação do veículo em um determinado período, o tempo que o veículo levou atravessando a lombada, a aceleração média e a velocidade média. A Figura 42 apresenta os tipos de lombadas e irregularidades, os quais serão utilizados para o treinamento dos algoritmos de aprendizado de máquina:

1. *Type* 1: Passarela elevada;
2. *Type* 2: Lombadas;
3. *Type* 3: Pequenas irregularidades ou trechos que não são do tipo 1 ou 2.

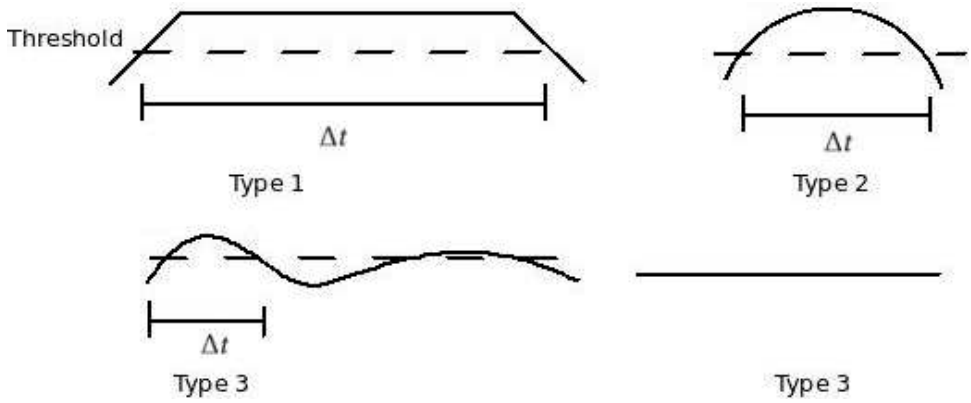

Figura 42 – Tipos de lombadas.

Fonte: Elaborado pelo autor.

Embora o propósito desta aplicação seja detectar passarelas elevadas e lombadas ao longo da pista, um classificador precisa ter classes que representem falsos positivos. Nesse caso, o tipo 3 (*Type* 3) representa as leituras que não são classificadas como tipo 1 ou 2.

O arquivo sincronizado e interpolado é carregado como uma lista e iteramos sobre essa lista. Cada registro que excede o *threshold*, como mostrado na Figura 42, é incluído em uma matriz auxiliar. Os parâmetros de interesse são:

- Δt é o tempo que o veículo permaneceu sobre o *threshold*.
- $\overline{Ax}$ é a aceleração média de x durante o período no qual o veículo permaneceu sobre o *threshold*.
- $\overline{Ay}$ é a aceleração média de y durante o período no qual o veículo permaneceu sobre o *threshold*.
- $\overline{Az}$ é a aceleração média de z durante o período no qual o veículo permaneceu sobre o *threshold*.
- $\overline{V}$ é a velocidade média durante o período no qual o veículo permaneceu sobre o *threshold*.
- P é o maior valor da rotação no eixo y (que descreve o movimento para cima e para baixo da câmera) no período no qual o veículo permaneceu sobre o *threshold*.

A classificação das lombadas é realizada utilizando aprendizado de máquina, tendo como objetivo criar modelos capazes de detectar e classificar corretamente lombadas e passarelas elevadas na pista. Três diferentes algoritmos foram avaliados no Weka (Weka, 2018): 2.2, 2.3 e 2.4.2.

3.1.3 Placas de Trânsito

Uma das formas de controlar o trânsito bem como disponibilizar aos motoristas informações sobre a via, são as placas verticais de trânsito. Para que essa sinalização tenha o efeito desejado é necessário que periodicamente se faça o inventário das placas, para possível manutenção ou mesmo troca.

De acordo com a Lei 9503, Artigo 24, Inciso III, cabe aos órgãos e entidades executivos de trânsito dos municípios implantar, manter e operar o sistema de sinalização, os dispositivos e equipamentos de controle viário (BRASIL, 1997).

De acordo com (DENATRAN, 2007a), a sinalização é classificada segundo a sua função sendo:

- ❏ regulamentar as obrigações, limitações, proibições ou restrições;
- ❏ advertir os condutores sobre condições com potencial de risco;
- ❏ indicar direções, localizações, pontos de interesse e transmitir mensagens educativas.

Para cada uma dessas funções os manuais (DENATRAN, 2007a), (DENATRAN, 2007b) e (DENATRAN, 2014) apresentam o conjunto de sinais que compõem a classificação, grupos e subgrupos desse sinais, como também informações sobre tamanho, posicionamento, cores e formas.

No método proposto a classificação das placas será dada por sua função, sendo: Advertência, Indicação e Regulamentação. A Figura 43 apresenta exemplos de sinais com a classificação proposta.

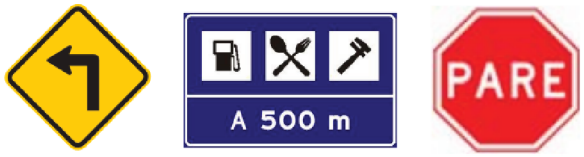

Figura 43 – Exemplo de sinais de trânsito: a) advertência, b) indicação e c) regulamentação

Fonte: Elaborado pelo autor

3.1.3.1 Detecção e Classificação

A detecção e classificação de sinais de trânsito será realizada usando as imagens dos *frames* dos vídeos capturados. Foi gerado um novo *dataset*, uma vez que é necessário ter as informações geradas pelos sensores das câmeras. O *dataset* é composto por 342

imagens extraídas de vídeos que foram gravados na região da cidade de Campinas, São Paulo.

A rotulagem das imagens para tratamento foi realizada com o LabelImg (TZUTALIN, 2015), uma ferramenta gráfica para anotação. A ferramenta suporta salvar as anotações no formato Pascal VOC e YOLO. A Figura 44 apresenta um exemplo de uso da ferramenta e a Figura 45 apresenta exemplos das anotações geradas para YOLO e PascalVOC.

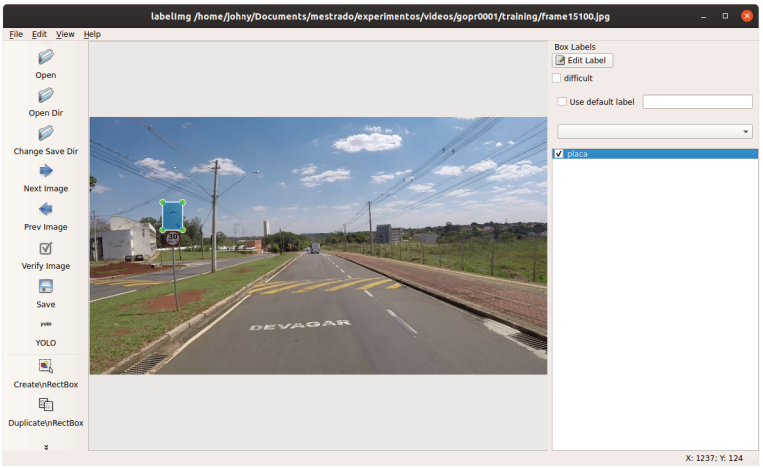

Figura 44 – Uso da ferramenta LabelImg.

Fonte: Elaborado pelo autor.

Nesse trabalho as tarefas de detecção e classificação serão realizadas por redes CNNs. Serão utilizadas, para comparação, as redes Faster RCNN, Retinanet, YOLOv3 e YOLOv5. Essas redes já foram apresentadas na seção 2.7.4.

3.1.3.2 Inventário de Quantidade

A contagem da quantidade de placas em um determinado vídeo será realizada conforme seus *frames* são processados. O *script* para esse cálculo carrega o vídeo a ser processado, e a cada *frame* utiliza a rede CNN para fazer a inferência dos sinais de trânsito.

Ao obter os *bounding boxes* do *frame* que está em processamento, o *script* itera sobre esses *boxes* e realiza algumas validações:

1. Existência de outros *boxes* da mesma classe já mapeados.

2. Diferença entre o *frame* atual e o *frame* já mapeado é inferior a 50. Essa quantidade define se o *box* segue referente ao mesmo sinal, ainda que ele não apareça por alguns *frames*.

3. *Box* anterior encontrado, considerando um determinado limite, ainda comporta o *box* atual. A Figura 46 apresenta essa validação.

Figura 45 – Anotações LabelImg: esquerda anotação para YOLO, direita anotação para PascalVOC.

Fonte: Elaborado pelo autor.

Se o *box* passar nessas validações, as informações desse *box* no mapeamento de *boxes* é atualizada. Caso contrário é validado a área do *box*, pois é necessário verificar se o sinal encontrado está perto ou não do veículo. Esse informação é necessária para os casos em que a rede encontre um sinal válido, mas que deveria ser contabilizado por outra filmagem. Após a validação da área, uma nova entrada nesse mapeamento é feita, indicando que o *box* atual é um novo sinal.

Após processar todo o vídeo, é realizado um último processamento, sobre o mapeamento dos *boxes*, onde é considerado um sinal válido somente se esse *box* tiver sido considerado válido por mais do que um determinado número de *frames*. Após esse processamento o *script* apresenta a quantidade de sinais em cada uma das classes.

Nesse processamento da quantidade de sinais, é gerado um arquivo com as informações dos sinais encontrados. O arquivo contém as informações do *bounding box* do sinal de trânsito, além do *frame* no qual o sinal foi encontrado. Essa arquivo é uma das entradas para geração do mapa de localizações.

3.1.3.3 GPS

Os dados de GPS utilizados são os mesmos apresentados na seção 3.1.2.1. Nessa parte, os dados extraídos contêm muitos registros redundantes, sendo necessário um processamento no arquivo.

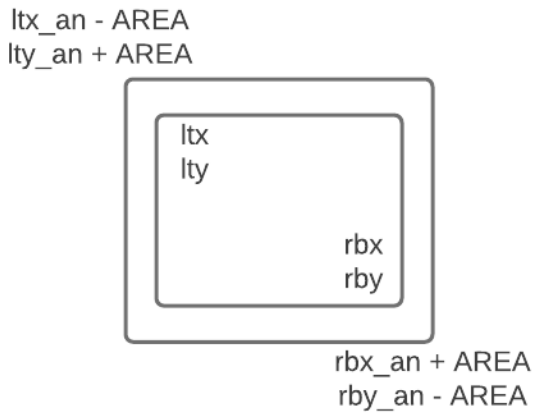

Figura 46 – Validação entre *box* atual (ltx, lty, rbx e rby) e anterior (ltx_ant, lty_ant, rbx_ant, rby_ant).

Fonte: Elaborado pelo autor.

Os dados gerados pelo sensor GPS contêm aproximadamente 17 registros para o mesmo *timestamp*, ou seja, o processamento reduz esses registros, mantendo somente 1 registro por segundo, sendo possível relacionar um determinado *frame* com o *timestamp* de um determinado dado do GPS.

3.1.3.4 Mapa

A etapa referente ao mapeamento dos sinais classificados é utilizada tanto com o vídeo de uma câmera única, como também utilizando vídeos das três câmeras frontais. Ao utilizar o vídeos de uma câmera a detecção é realizada conforme descrito em 3.1.3.2. Para realizar o mapeamento de sinais utilizando o vídeos das três câmeras frontais, é necessário primeiramente fazer a sincronização dos vídeos. A sincronização é realizada com base em (OLIVEIRA et al., 2019), em que é utilizado um *beep* que é captado por todas as câmeras e no pós-processamento esse *beep* é encontrado nas câmeras e o vídeo é recortado para inciar onde temos o pico desse *beep*.

Após a sincronização dos vídeos, os mesmos são então processados, o que ocorre em três etapas. A primeira etapa processa o vídeo da câmera central e os *bounding boxes* detectados e classificados nessa etapa, são incluídos no mapeamento que contêm os *frames*, para cada *frame* existe um mapeamento de classes e em cada classe uma lista dos *boxes*.

A segunda etapa processa o vídeo da câmera esquerda, cada *frame* é pesquisado no mapeamento criado na primeira etapa, também é recuperada o mapeamento de *boxes* da classe do *box* em processamento. Para cada lista no mapeamento de *boxes*, é feita a iteração sobre a lista, validando se o *box* atual é um dos *boxes* encontrados no vídeo da

câmera central. Essa validação é realizada aumentando a área do *box* da câmera central e deslocando o mesmo para esquerda. Se o *box* atual se encaixar nessa área, é incluído como o mesmo sinal no vídeo da esquerda. A Figura 47 apresenta a validação realizada de forma simplificada.

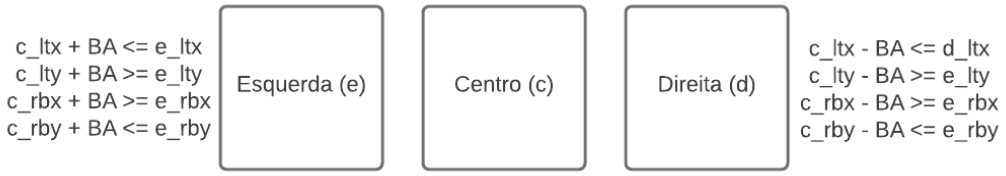

Figura 47 – Validação do encaixe do *box* da esquerda e da direita no *box* central.

Fonte: Elaborado pelo autor.

A terceira etapa processa o vídeo da câmera direita, utilizando o mesmo processo apresentado na segunda etapa, com a única diferença que o deslocamento é feito para a direita. Como resultado do processamento das três etapas é salvo um arquivo com o mapeamento feito, para um pós-processamento.

No pós-processamento é feito uma filtragem retirando os *frames* que não possuem o mesmo sinal nas três câmeras. No primeiro *frame*, que possui os boxes nas três câmeras, é incluído o número daquele sinal para a classe. Nos demais *frames*, é verificado se o *box* da câmera central do *frame* central se refere ao mesmo *box* do *frame* anterior, se for o mesmo *frame* inclui o mesmo número para esse *box*, senão é incrementado o número do *box* para a classe.

Após esse pós-processamento, é gerado um arquivo ".csv"que contém, em cada linha, o número do *frame*, a classe, os dados do *box* central, a classe, os dados do *box* esquerdo e a classe e os dados do *box* direto. Os dados de classe e dos *boxes* são incluídos conforme a quantidade de listas de *boxes* em cada *frame*. Esse arquivo é uma das entradas para gerar o mapa com as localizações.

O mapa com a localização aproximada dos sinais de trânsito é gerado de forma manual nesse trabalho utilizando uma das *features* do Google Maps®. Para gerar os dados para o mapa são utilizados os dados dos arquivos disponibilizados nas seções 3.1.3.2, 3.1.3.4 e 3.1.3.3.

Para relacionar um determinado sinal de trânsito com um registro do sensor GPS, é necessário ter algumas informações como o tamanho em segundos do vídeo processado e a quantidade total de *frames* do vídeo. Com essas informações é realizado o cálculo de quantos *frames* por segundo o vídeo tem e com base no *frame* em que o sinal de trânsito foi encontrado, é feito o cálculo para encontrar em que segundo o sinal foi encontrado.

Após esses cálculos é realizada a relação entre o segundo em que o sinal foi encontrado,

com o segundo dos dados do GPS e as informações de latitude e longitude são recuperadas dos dados do GPS e salvas no arquivo que contém os dados dos sinais de trânsito.

Quando esse mapeamento está sendo realizado com três câmeras, os passos anteriores são feitos baseados nas informações de GPS dos três vídeos, como também nas informações dos sinais disponíveis no arquivo gerado na seção 3.1.3.4.

Capítulo 4

Experimentos e Análise dos Resultados

Este capítulo apresenta os experimentos considerando a metodologia proposta para a classificação de lombadas e também criação do inventário de placas de trânsito de baixo custo.

4.1 Objetivos dos Experimentos

Neste trabalho foram realizados três experimentos visando a classificação de lombadas baseado nos sensores das câmeras GoPro, utilizando diferentes algoritmos de aprendizado de máquina.

Além dos experimentos acima, foi conduzido o treinamento das redes Faster RCNN, Retinanet, YOLOv3 e YOLOv5, mencionadas em 2.7.4, para validar qual a rede tem a melhor acurácia na classificação dos sinais de trânsito. A YOLOv3 é utilizada no trabalho (BRUNO, 2020) e a Retinanet é utilizada no trabalho (BALADO et al., 2020).

São também realizados experimentos para obter a quantidade de sinais de trânsito em um determinado vídeo, para compor o inventário por quantidades, como também experimentos para utilização dos dados do sensor GPS das câmeras GoPro e com isso compor o inventário com mapeamento da localização.

4.2 Base de Dados

A base de dados é composta de dois *datasets* distintos. O primeiro *dataset* é composto pelos dados dos sensores de três vídeos. Os dados de dois vídeos foram usados para o treinamento dos classificadores e os dados do terceiro vídeo foram usados para o teste após o treinamento. Depois do pré-processamento dos dados de treinamento, foram geradas 7500 instâncias. A maioria pertencente ao Tipo 3, que é mais comum em uma estrada regular.

Entretanto, para evitar o *overfitting* primeiro foi aplicado uma técnica para balanceamento do *dataset*. O balanceamento das classes na ferramenta Weka foi realizado utilizando o filtro *Class Balancer*, deixando cada classe com o mesmo peso.

No último conjunto de dados, extraído do terceiro vídeo e usado para testar os modelos, o pré-processamento gerou 32 instâncias. Nesse caso não é feito nenhum balanço para a classificação. A Figura 48 apresenta a rota de 5km usada na coleta dos dados.

Figura 48 – Rota usada nos experimentos.

Fonte: Elaborado pelo autor.

O segundo *dataset* é composto pelas imagens para o treinamento das redes CNNs. São 342 imagens de tamanho 1920x1080, extraídas de vídeos capturados em Campinas - SP. 327 imagens usadas para o treinamento e 15 usadas nos testes das redes CNNs ainda durante o treinamento. Nessas 342 imagens foram rotulados 162 sinais da classe advertência, 210 sinais da classe regulamentação e 232 sinais da classe indicação.

Após o treinamento são utilizados 3 vídeos para realizar experimentos no inventário da quantidade e mapeamento da localização de uma câmera e também outros 3 vídeos (1 vídeo de cada câmera frontal) para mapeamento da localização utilizando os vídeos das três câmeras.

4.3 Análise dos Experimentos

Nesta seção será apresentados os resultados dos experimentos, como também análises sobre os mesmos.

4.3.1 Classificação de Lombadas

Nos dados utilizados para o treinamento, foi aplicada a validação cruzada *k-fold* com k = 10, com isso os dados são randomicamente particionados em 10 subconjuntos. Desses 10 subconjuntos, 1 é retido para validação dos dados e os 9 subconjuntos restantes são usados no treinamento dos dados. Esse processo de validação cruzada é repetido 10 vezes, com cada um dos 10 subconjuntos usados exatamente uma vez como dados para validação. Do resultado dos 10 *folds* é calculada a média ou combinados de alguma outra maneira para produzir uma estimativa.

A Figura 49 apresenta os atributos de entrada para os classificadores utilizados e também as saídas esperadas desses classificadores.

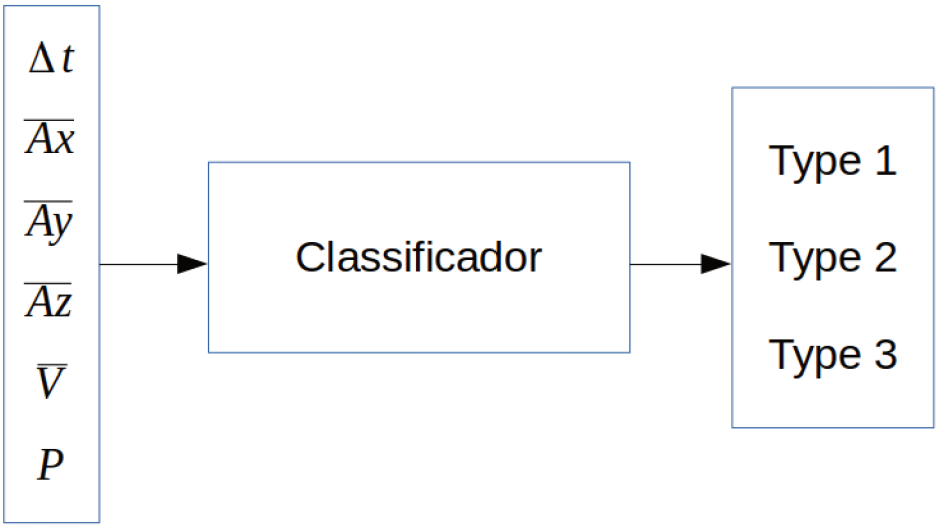

Figura 49 – Atributos de entrada e saídas para classificação das lombadas.

Fonte: Elaborado pelo autor.

A Figura 50 apresenta a acurácia de cada classificador após a fase de treinamento. O classificador Naive Bayes alcançou 64.83% de acurácia, seguido por RF com 63.23% e MLP com 33.33%.

Uma vez que os modelos foram treinados, são usados para classificar os dados gerados pelo último vídeo. Nesse passo o classificador RF obteve uma melhor performance, como retratado na Figura 51. O classificador RF alcançou 96.84% de acurácia com somente 1 falso positivo. O segundo melhor foi o Naive Bayes com 13 classificações corretas e 19 falso

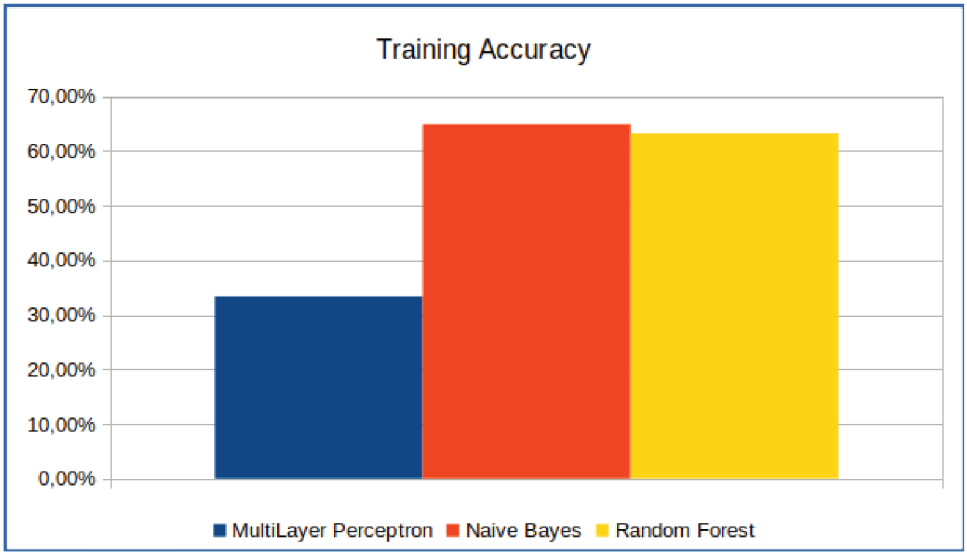

Figura 50 – Acurácia no treinamento dos classificadores.

Fonte: Elaborado pelo autor.

positivos, o que corresponde à 40.62%. Como na fase de treinamento, o pior classificador foi MLP somando 26 falsos positivos contra 6 classificações corretas (18.70%).

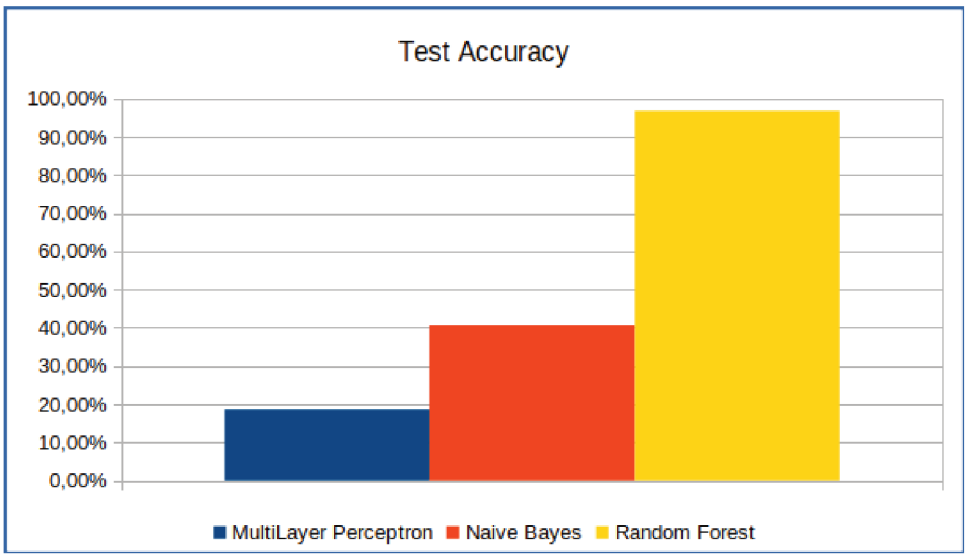

Figura 51 – Acurácia no teste dos classificadores.

Fonte: Elaborado pelo autor.

As Tabelas 4, 5 e 6 apresentam a matriz de confusão de cada classificador com os dados de teste. Linhas e colunas relatam o número de falsos positivos, falsos negativos, verdadeiros positivos e verdadeiros negativos. Isto permite uma análise mais detalhada do que somente a proporção correta de classificações (acurácia).

Tabela 4 – Matriz de confusão do classificador Naive Bayes.

Tipos	*Tipo 1*	*Tipo 2*	*Tipo 3*
Tipo 1	**5**	1	0
Tipo 2	9	**6**	0
Tipo 3	0	9	**2**

Tabela 5 – Matriz de confusão do classificador RF.

Tipos	*Tipo 1*	*Tipo 2*	*Tipo 3*
Tipo 1	**6**	0	0
Tipo 2	0	**15**	0
Tipo 3	0	1	**10**

Tabela 6 – Matriz de confusão do classificador MLP.

Tipos	*Tipo 1*	*Tipo 2*	*Tipo 3*
Tipo 1	**6**	0	0
Tipo 2	15	**0**	0
Tipo 3	11	0	**0**

As Figuras 52, 53 e 54 ilustram a classificação das instâncias ao longo do mapa usando cada um dos classificadores. Os marcadores verdes apresentam as classificações corretas enquanto que os marcadores vermelhos as classificações erradas. O rótulo próximo aos marcadores representam a classificação resultante, onde o número à esquerda é o tipo atual da instância, e o número à direita é o tipo predito pelos classificadores.

Os experimentos foram realizados utilizando a ferramenta Weka sem modificação nos parâmetros habilitados na ferramenta, que são utilizados na definição da quantidade de árvores para o algorimot RF e na quantidade de camadas e neurônios do algoritmo MLP. A grande diferença entre os classificadores RF e MLP pode ser atribuída a esses parâmetros, uma vez que são criadas 100 árvores para serem utilizadas na classificação pelo RF, e no algoritmo MLP o padrão é a soma dos atributos (6) com a quantidade de classes (3) e dividir esse valor por 2, nesse caso no MLP temos somente uma camada escondida com 4 neurônios, o que impacta no treinamento e classificação do algoritmo MLP.

Observa-se, baseado nos resultados apresentados, que as imperfeições / anomalias ao longo das ruas afetam as métricas usadas, bem como casos de falta de manutenção tanto no pavimento quanto nas lombadas.

As lombadas / quebra-molas no Brasil são reguladas pelo Código Nacional de Trânsito, Resolução nº 600/2016 (BRASIL, 2016), mas existem casos onde essas características não se encaixam nos requisitos técnicos, o que pode causar acidentes, e no problema que se tenta resolver nesse trabalho pode levar a falsos positivos.

Neste trabalho, conforme apontado anteriormente, somente os dados gerados por uma das câmeras foram usados, e os resultados podem ser refinados utilizando os dados gerados pelas outras quatro câmeras.

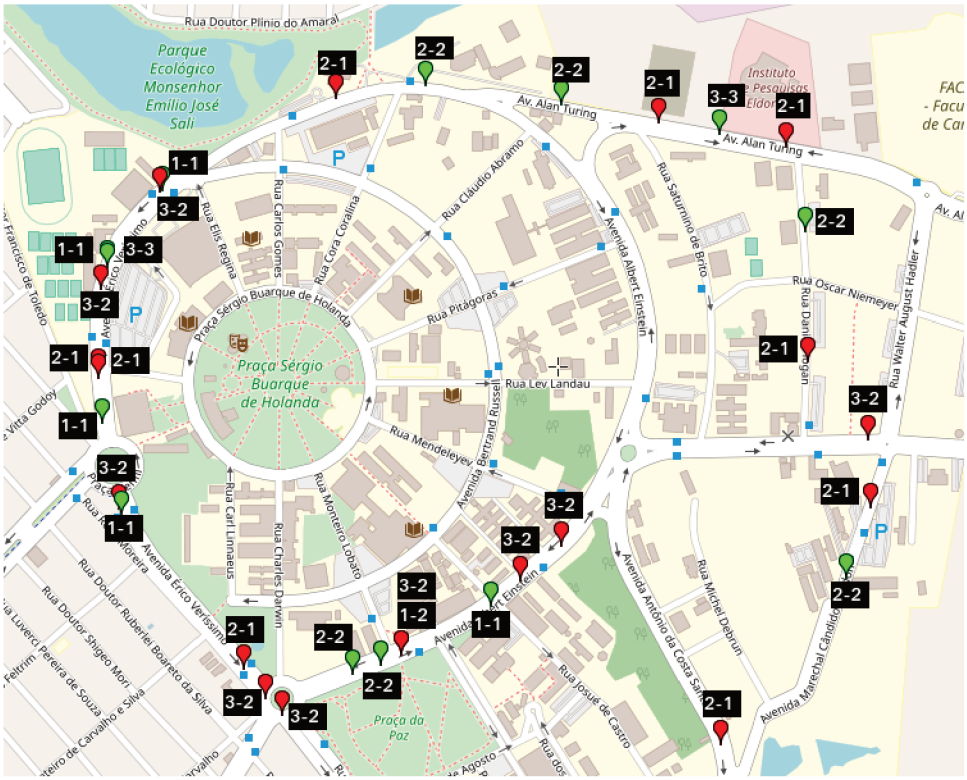

Figura 52 – Mapa criado com Naive Bayes.

Fonte: Elaborado pelo autor.

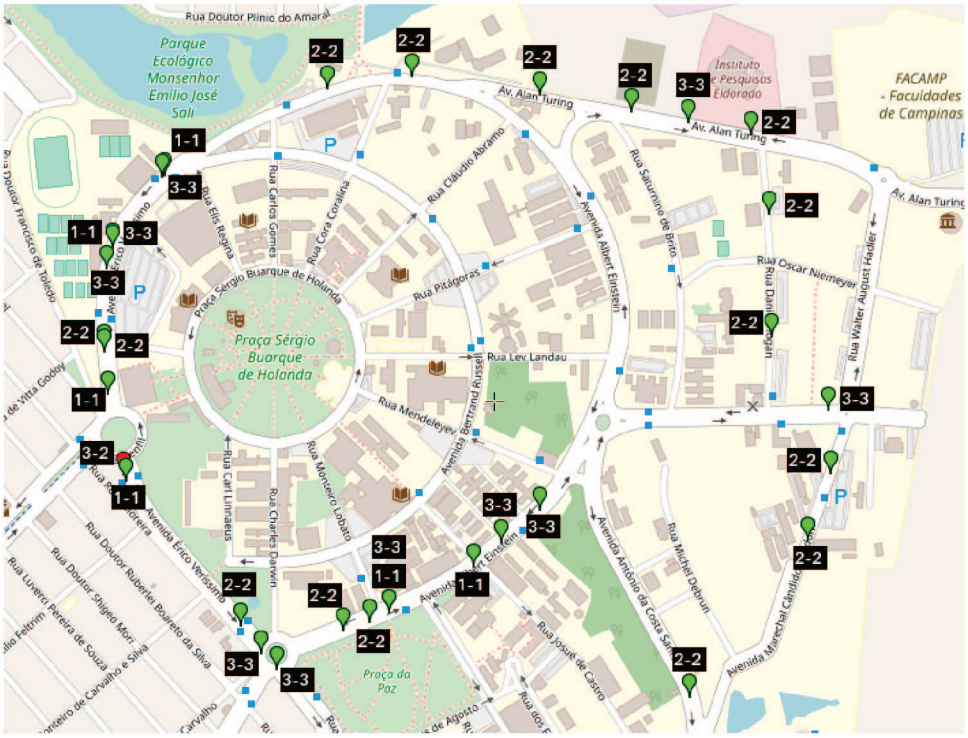

Figura 53 – Mapa criado RF.

Fonte: Elaborado pelo autor.

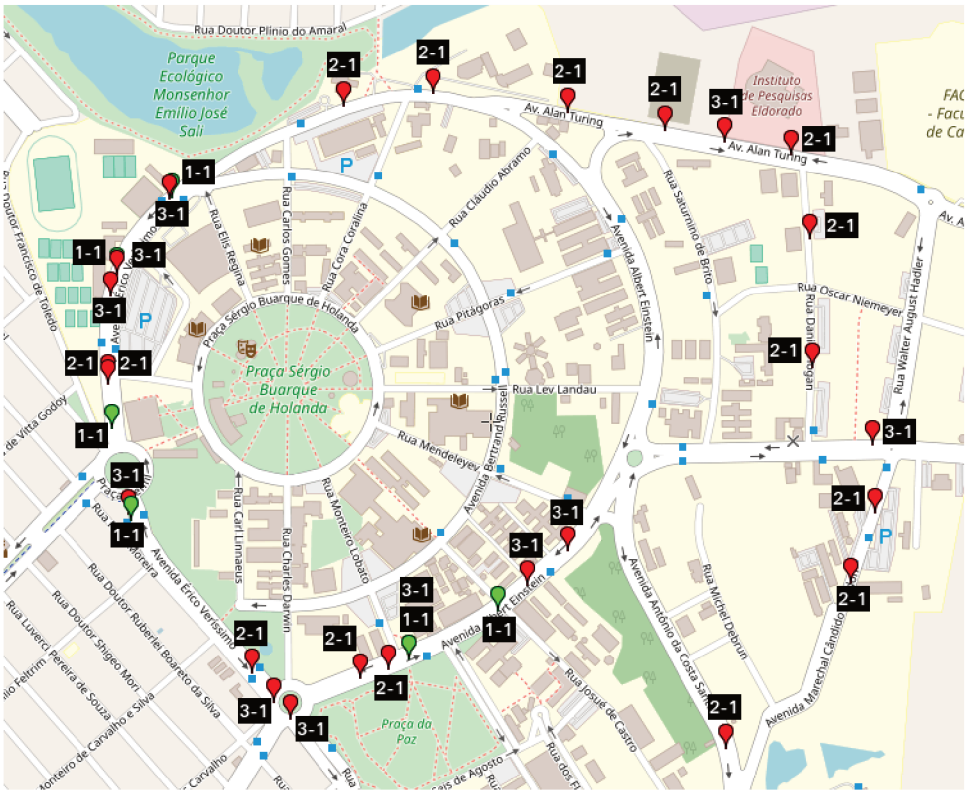

Figura 54 – Mapa criado com MLP.

Fonte: Elaborado pelo autor.

4.3.2 Detecção e Classificação de Sinais

Os experimentos com as redes CNN foram realizados modificando tanto o *batch*, que se refere a quantidade de imagens que são utilizadas em uma iteração, como também o algoritmo de otimização. Nas subseções seguintes são apresentados os resultados para cada uma das arquiteturas. Os experimentos foram realizados utilizando uma única máquina disponibilizada pelo Lab-UFU, com um processador de 8 núcleos, 16GB de RAM e uma placa de vídeo Nvidia Titan X com 12GB de VRAM.

As métricas apresentadas no experimentos são a *Average Precision* (AP) que representa a média de precisão da classificação em uma determinada classe, a mAP representa a média da precisão para todas as classes e a *loss* que representa o comportamento do gráfico de erro da rede. Nos experimentos são utilizados diferentes tamanhos de *batch* sendo utilizados com tamanho 2, 4 e 8. Esse tamanho representa a quantidade de imagens utilizadas em um iteração no treinamento da rede.

4.3.2.1 Faster RCNN

Com a arquitetura Faster RCNN foram realizados experimentos utilizando também os otimizadores SGD e Adam. Nesses experimentos bons resultados já foram alcançados com 50 épocas.

Para esse experimento utilizou-se a *toolbox* MMDetection (CHEN et al., 2019), que disponibiliza um grande conjunto de *frameworks* e métodos para detecção de objetos e segmentação de instância. Dentre eles o Faster RCNN.

As Tabelas 7 e 8 apresentam os resultados da métrica AP com os diferentes otimizadores e utilizando diferentes tamanhos de *batch*.

Tabela 7 – Faster RCNN - Resultados utilizando 50 épocas e otimizador SGD.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	**90%**	89%	81%
regulamentacao	**100%**	94%	81%
indicacao	75%	**100%**	84%

Tabela 8 – Faster RCNN - Resultados utilizando 50 épocas e otimizador Adam.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	**82%**	71%	81%
regulamentacao	64%	**70%**	64%
indicacao	50%	**68%**	58%

A Tabela 9 apresenta o resultado da métrica mAP no treinamento utilizando os diferentes otimizadores e também usando diferentes tamanhos de *batch*.

As Figuras 55 e 56 apresentam os gráficos de métricas geradas no treinamento da rede com o otimizador SGD, que obteve o melhor mAP. Como é possível observar nos

Tabela 9 – Faster RCNN - Resultados da métrica mAP em diferentes otimizadores.

Otimizador	*AP/2*	*AP/4*	*AP/8*
SGD	89%	**94%**	82%
Adam	65%	70%	68%

gráficos, a quantidade de passos do treinamento se baseiam no tamanho do *bacth* utilizado no treinamento. A cor vermelha representa o *batch* de tamanho 2, a cor laranja representa o *batch* de tamanho 4 e a cor azul o *batch* de tamanho 8.

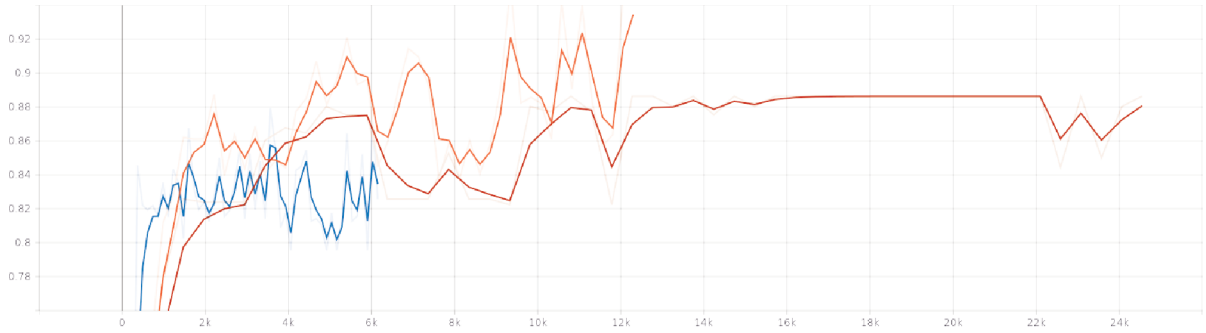

Figura 55 – Faster RCNN - Gráfico para a métrica mAP.

Fonte: Elaborado pelo autor.

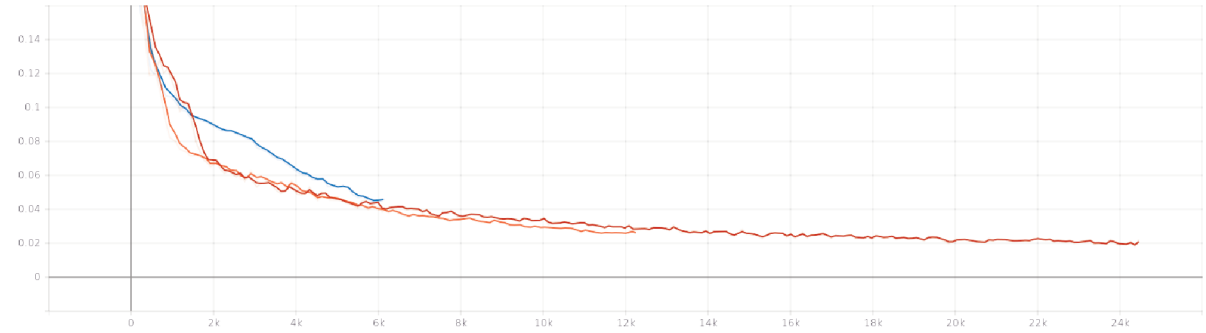

Figura 56 – Faster RCNN - Gráfico para a métrica *loss*.

Fonte: Elaborado pelo autor.

De acordo com (REN et al., 2016) essa rede consegue alcançar bons resultados mesmo com *datasets* pequenos.

4.3.2.2 Retinanet

As Tabelas 10 e 11 apresentam os resultados da métrica AP utilizando os otimizadores SGD e Adam e também com diferentes tamanhos de *batch*, treinando em 50 épocas.

Tabela 10 – Retinanet - Resultados utilizando 50 épocas e otimizador SGD.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	**82%**	72%	79%
regulamentacao	77%	**85%**	68%
indicacao	91%	**100%**	91%

Tabela 11 – Retinanet - Resultados utilizando 50 épocas e otimizador Adam.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	72%	75%	**79%**
regulamentacao	**91%**	76%	66%
indicacao	**73%**	70%	58%

A Tabela 12 apresenta o resultado da métrica mAP no treinamento utilizando os diferentes otimizadores e também usando diferentes tamanhos de *batch*.

Tabela 12 – Retinanet - Resultados da métrica mAP em diferentes otimizadores.

Otimizador	*AP/2*	*AP/4*	*AP/8*
SGD	83%	**86%**	79%
Adam	79%	74%	68%

As Figuras 57 e 58 apresentam os gráficos de métricas geradas no treinamento da rede com o otimizador SGD, que obteve o melhor mAP. Como é possível observar nos gráficos, a quantidade de passos do treinamento se baseiam no tamanho do *bacth* utilizado no treinamento. A cor azul representa o *batch* de tamanho 2, a cor laranja representa o *batch* de tamanho 4 e a cor vermelha o *batch* de tamanho 8.

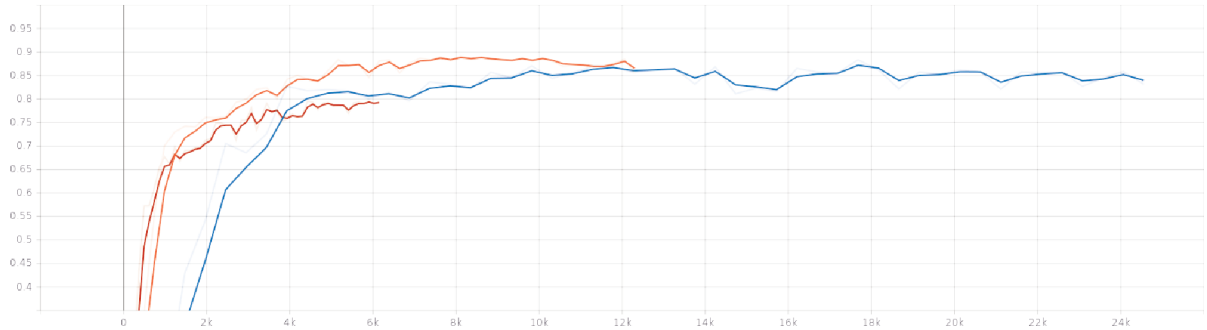

Figura 57 – Retinanet - Gráfico para a métrica mAP.

Fonte: Elaborado pelo autor.

4.3.2.3 YOLOv3

As Tabelas 13 e 14 apresentam os resultados da métrica AP no treinamento em 50 épocas, com *batchs* de tamanho 2, 4 e 8 e também utilizando os otimizadores SGD e

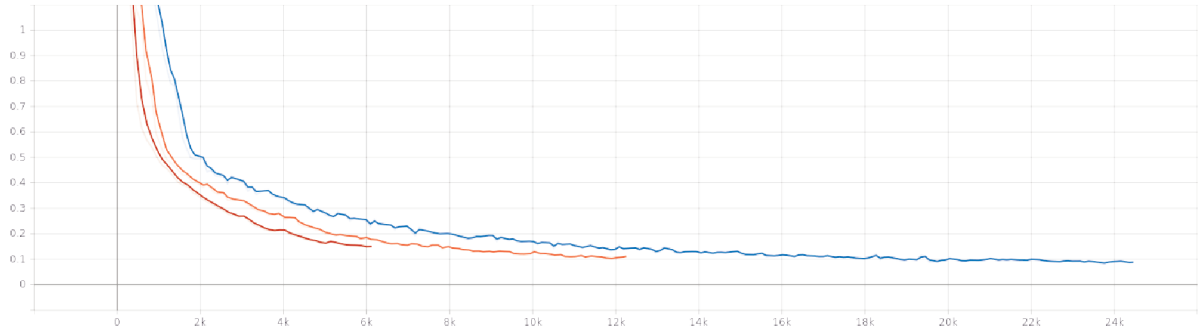

Figura 58 – Retinanet - Gráfico para a métrica *loss*.

Fonte: Elaborado pelo autor.

Adam.

Tabela 13 – YOLOv3 - Resultados utilizando 50 épocas e otimizador SGD.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	**44%**	33%	42%
regulamentacao	11%	21%	**33%**
indicacao	2%	16%	**42%**

Tabela 14 – YOLOv3 - Resultados utilizando 50 épocas e otimizador Adam.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	**45%**	37%	37%
regulamentacao	25%	33%	**37%**
indicacao	22%	**75%**	61%

A Tabela 15 apresenta o resultado da métrica mAP no treinamento utilizando os diferentes otimizadores e também usando diferentes tamanhos de *batch*.

Tabela 15 – YOLOv3 - Resultados da métrica mAP em diferentes otimizadores.

Otimizador	*mAP/2*	*mAP/4*	*mAP/8*
SGD	19%	23%	39%
Adam	31%	**48%**	45%

As Figuras 59 e 60 apresentam os gráficos das métricas mAP e **loss** geradas no treinamento da rede com o otimizador Adam, que obteve o melhor mAP. A cor laranja representa o *batch* de tamanho 2, a cor azul representa o *batch* de tamanho 4 e a cor vermelha o *batch* de tamanho 8.

Em (BRUNO, 2020), a YOLOv3 foi usada em dois experimentos, o primeiro com 21 classes e o mAP foi de 78% e o segundo experimento, com 1 classe, o mAP foi de 92%. No experimento realizado nesse trabalho, usando as 3 classes de categorias, o melhor mAP

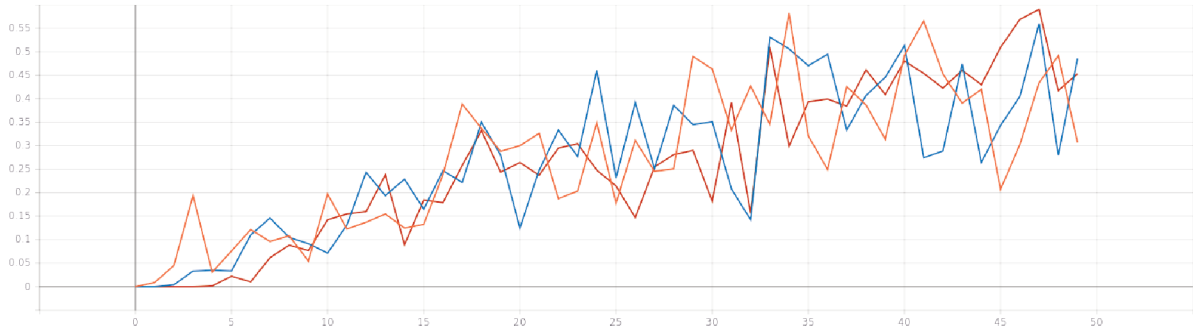

Figura 59 – YOLOv3 - Gráfico para a métrica mAP.

Fonte: Elaborado pelo autor.

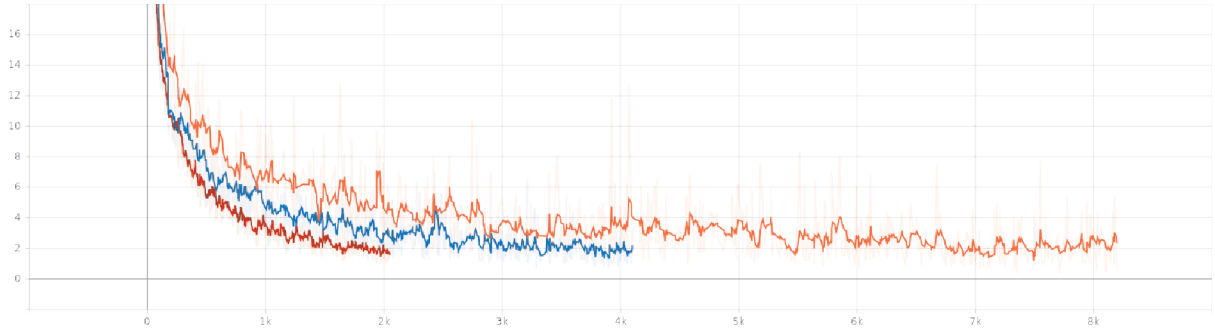

Figura 60 – YOLOv3 - Gráfico para a métrica *loss*.

Fonte: Elaborado pelo autor.

foi de 48%. Como melhoria desse experimento, pode-se tentar treinar em mais épocas como feito em (BRUNO, 2020) que obteve o melhor resultado em 20000 épocas.

4.3.2.4 YOLOv5

A arquitetura YOLOv5 foi incluída nos experimentos por ser uma evolução da YOLOv3 e sendo possível comparar para validar se é possível conseguir resultados melhores nessa evolução.

As Tabelas 16 e 17 apresentam os resultados da métrica AP no treinamento em 50 épocas, com *batchs* de tamanho 2, 4 e 8 e também utilizando os otimizadores SGD e Adam.

Tabela 16 – YOLOv5 - Resultados utilizando 50 épocas e otimizador SGD.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	**73%**	69%	71%
regulamentacao	**74%**	68%	68%
indicacao	51%	55%	**74%**

Tabela 17 – YOLOv5 - Resultados utilizando 50 épocas e otimizador Adam.

Classe	*AP/2*	*AP/4*	*AP/8*
advertencia	78%	**80%**	72%
regulamentacao	**81%**	76%	68%
indicacao	65%	**85%**	84%

A Tabela 18 apresenta o resultado da métrica mAP no treinamento utilizando os diferentes otimizadores e também usando diferentes tamanhos de *batch*.

Tabela 18 – YOLOv5 - Resultados da métrica mAP em diferentes otimizadores.

Otimizador	*mAP/2*	*mAP/4*	*mAP/8*
SGD	66%	64%	71%
Adam	75%	**80%**	75%

As Figuras 61 e 62 apresentam os gráficos das métricas mAP e **loss** geradas no treinamento da rede com o otimizador Adam, que obteve o melhor mAP. A cor laranja representa o *batch* de tamanho 2, a cor azul representa o *batch* de tamanho 4 e a cor vermelha o *batch* de tamanho 8.

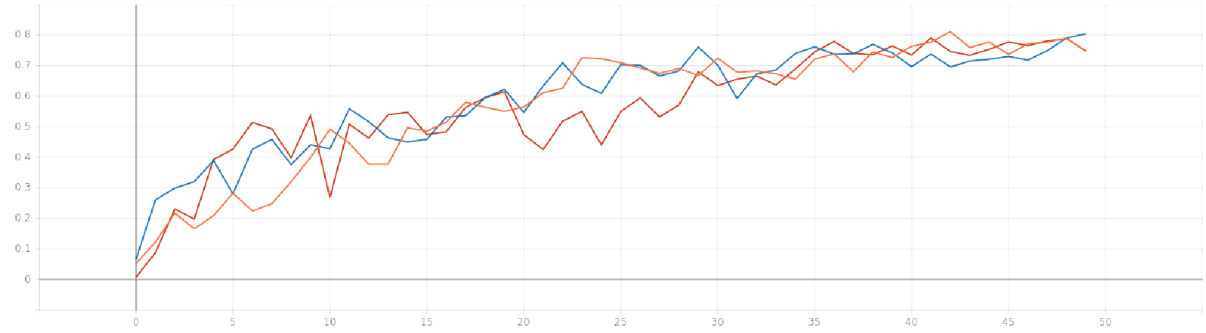

Figura 61 – YOLOv5 - Gráfico para a métrica mAP.

Fonte: Elaborado pelo autor.

A Tabela 19 apresenta os melhores resultados da métrica mAP para cada uma das arquiteturas, como também o otimizador e o tamanho da *batch* utilizados no treinamento.

Tabela 19 – Resultado da métrica mAP por arquitetura.

Architecture	*Optimizer*	*Batch*	*mAP*
Faster RCNN	SGD	4	**94%**
Retina	SGD	4	85%
YOLOv3	Adam	4	48%
YOLOv5	Adam	4	75%

Baseado nos resultados dos treinamento, a arquitetura Faster RCNN obteve a melhor acurácia, e portanto será utilizada nos próximos experimentos.

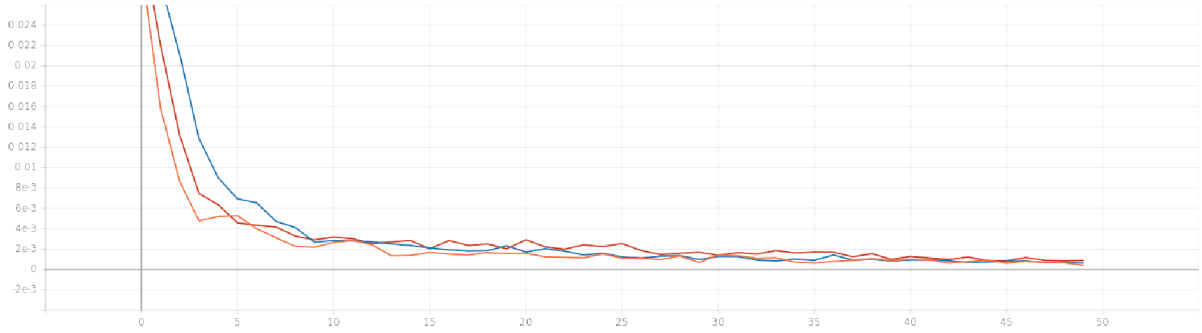

Figura 62 – YOLOv5 - Gráfico para a métrica *loss*.

Fonte: Elaborado pelo autor.

4.3.3 Inventário de Quantidade de Sinais de Trânsito

Com base no resultado dos experimentos de detecção e classificação, a arquitetura utilizada para esse experimento é a Faster RCNN. Nos experimentos que calculam a quantidade de sinais de trânsito em um determinado vídeo, a arquitetura Faster RCNN será utilizada para detectar e classificar os sinais de trânsito, e após essa classificação é realizado um mapeamento do mesmo sinal em *frames* diferentes.

A Tabela 20 apresenta o percentual de acerto, baseado nos sinais de trânsito que foram considerados corretamente pelo algoritmo. Os percentuais levam em consideração tanto os resultados positivos quanto os falsos positivos.

Tabela 20 – Resultados da Quantidade de Sinais de Trânsito.

Video	*Advertencia*	*Regulamentacao*	*Indicacao*	*Total*
Video 1	79.41%	84.13%	66.67%	81.55%
Video 2	0.00%	41.18%	56.25%	46.58%
Video 3	75.00%	59.38%	79.07%	71.08%

4.3.4 Mapa

Os sinais de trânsito utilizados no mapeamento são aqueles encontrados tanto no inventário de quantidade que é realizado nos vídeos de um câmera, como também nos sinais de trânsito encontrados nos vídeos das três câmeras frontais.

Para a detecção e classificação dos sinais de trânsito nas três câmeras frontais também é utilizada a arquitetura Faster RCNN, uma vez que obteve a melhor acurácia no treinamento. Nos mapas, os ícones com cor amarela representam a classe Advertência, com cor vermelha representam a classe Regulamentação e com cor azul representam a classe Indicação.

As Figuras 63 e 64 apresentam o mapeamento dos sinais reconhecidos no Vídeo 1. A Figura 63 apresenta todas os sinais de trânsito, enquanto a Figura 64 apresenta em mais detalhes os sinais reconhecidos que aparecem na parte inferior da Figura 63.

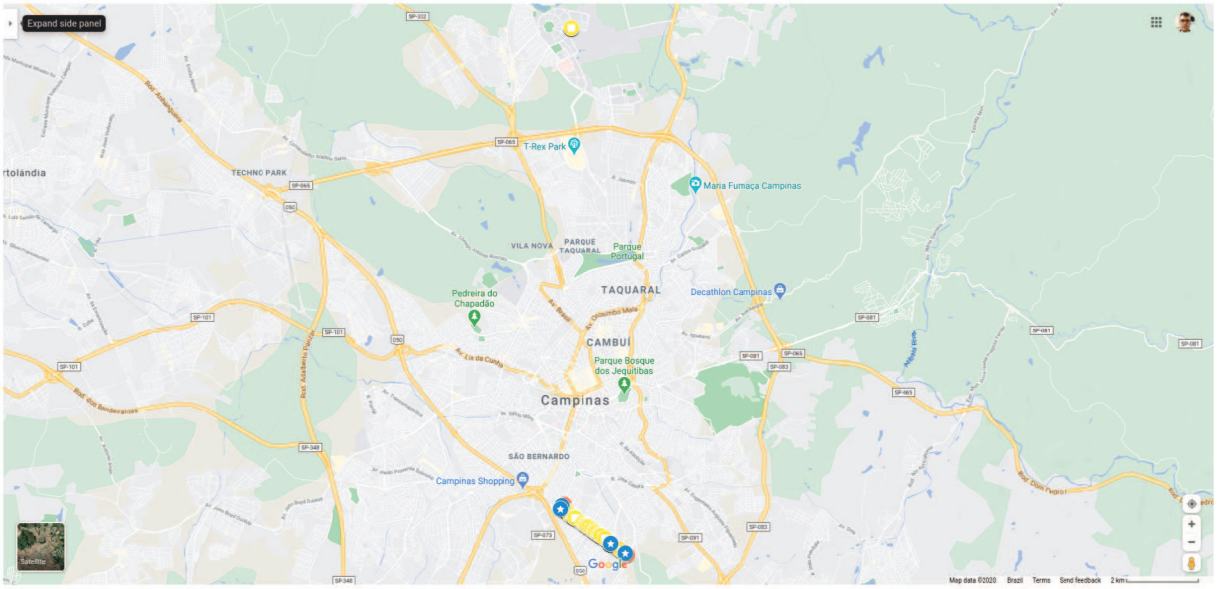

Figura 63 – Video 1 - Mapa de Localização 1.

Fonte: Elaborado pelo autor.

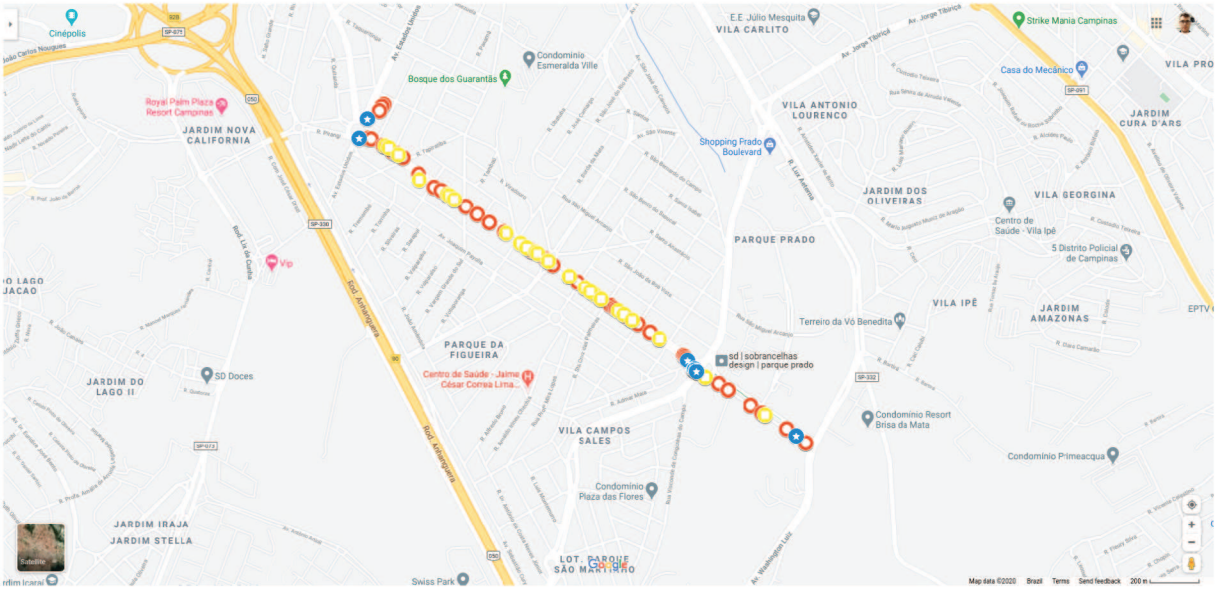

Figura 64 – Video 1 - Mapa de Localização 2.

Fonte: Elaborado pelo autor.

Como pode ser visto nas Figuras 63 e 64 há a necessidade de aguardar o sensor GPS

da câmera sincronizar e atualizar a sua posição inicial, antes de iniciar a captura dos vídeos. Nesse experimento o vídeo iniciou a captura antes dessa sincronização e com isso os primeiros registros do GPS foram gerados de forma incorreta, apresentando os sinais de trânsito numa posição totalmente desatualizada.

As Figuras 65 e 66 apresentam o mapeamento dos sinais reconhecidos nos Vídeos 2 e 3 respectivamente.

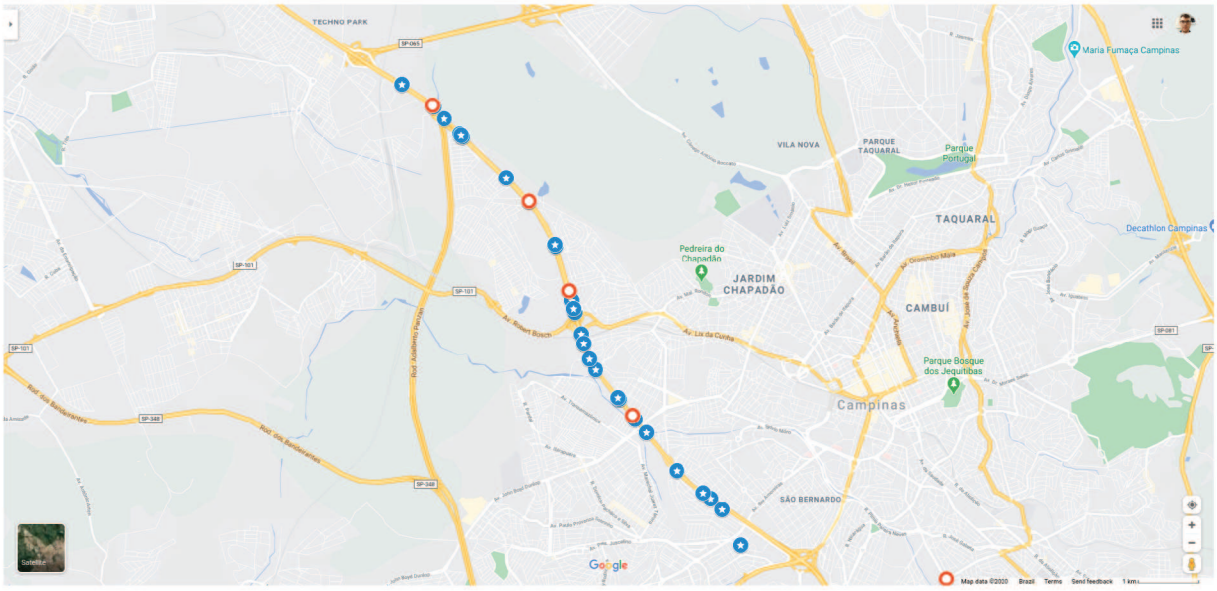

Figura 65 – Video 2 - Mapa de Localização.

Fonte: Elaborado pelo autor.

A Figura 67 apresenta o mapeamento utilizando os dados dos sinais de trânsito reconhecidos nas três câmeras frontais.

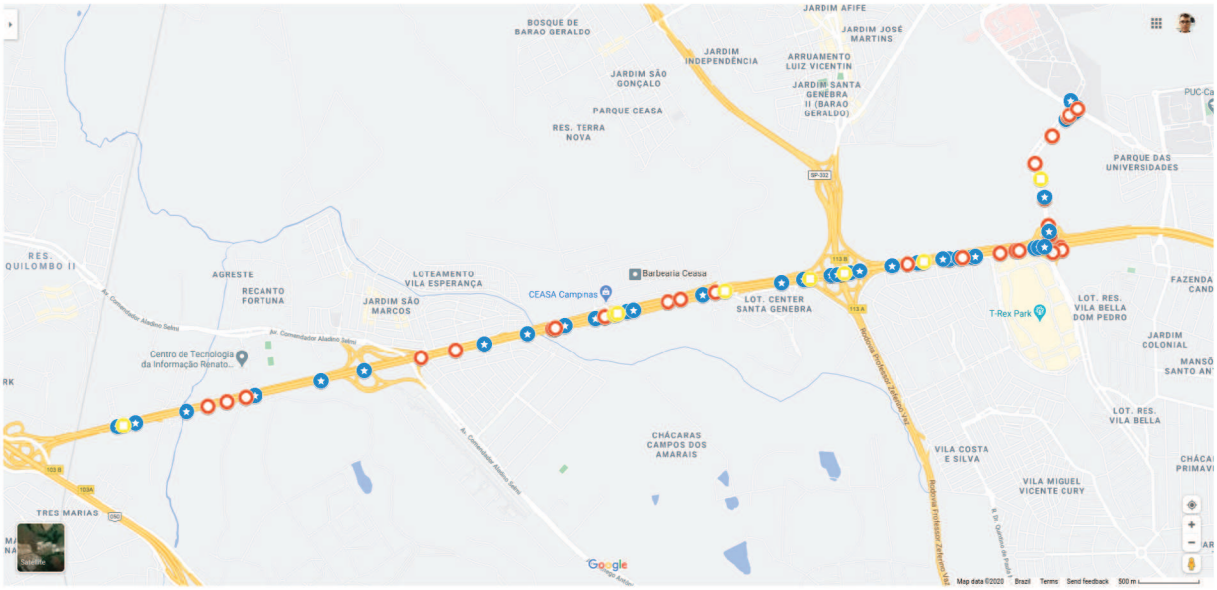

Figura 66 – Video 3 - Mapa de Localização.

Fonte: Elaborado pelo autor.

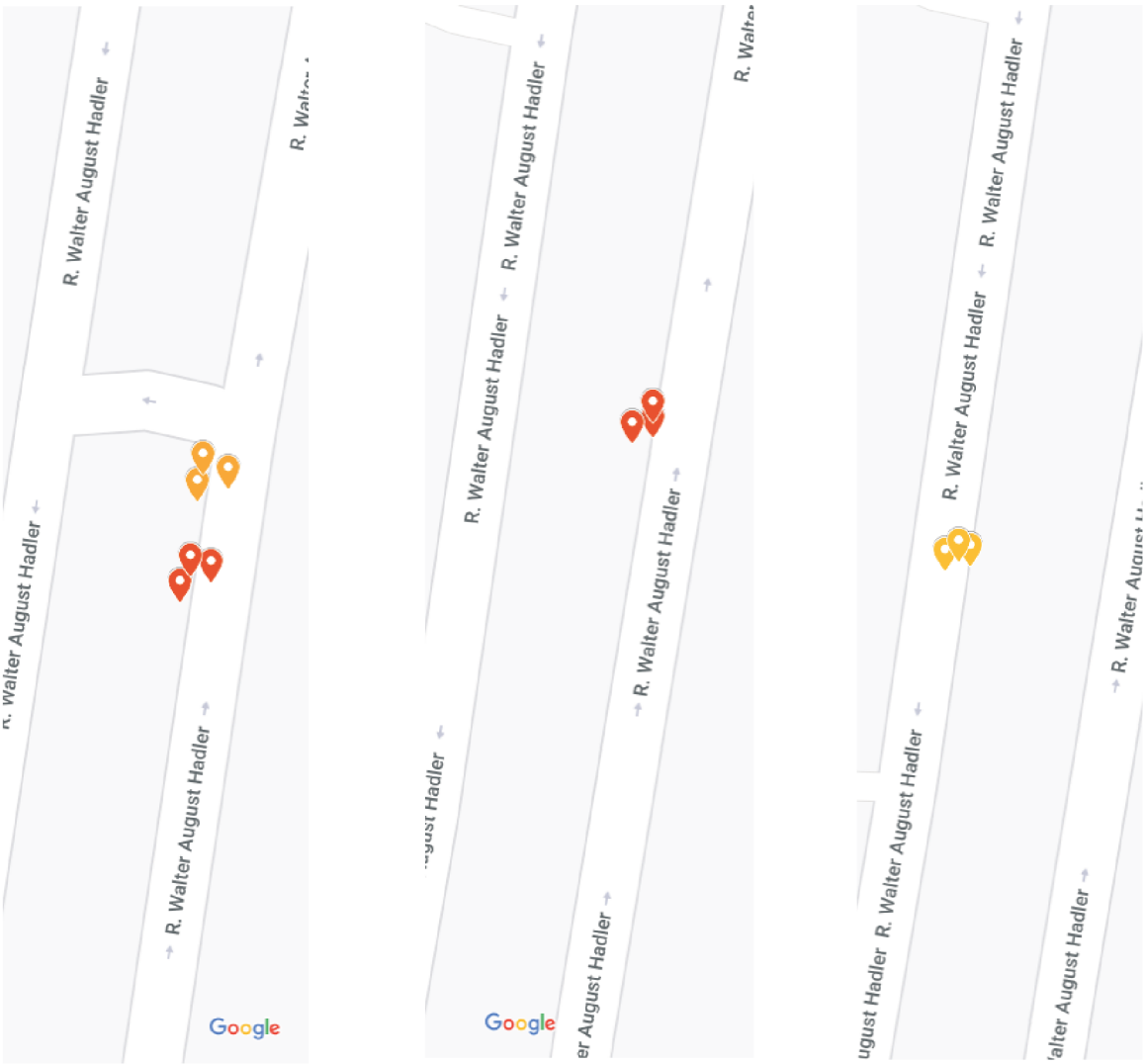

Figura 67 – Mapa de Localização usando Três Câmeras.

Fonte: Elaborado pelo autor.

4.4 Considerações Finais

Os experimentos demonstram o potencial do uso de um sistema de mapeamento de baixo custo tanto para o reconhecimento de lombadas, como também na criação de inventários de sinais de trânsito.

Baseado nos resultados apresentados, é possível observar que as imperfeições/anomalias ao longo das ruas afetam as métricas utilizadas, como também os casos em que há a falta de manutenção do pavimento e também das lombadas.

No inventário de quantidade, em especial no Vídeo 2, há uma discrepância no percentual de sinais reconhecidos. Essa discrepância se deve ao fato desse vídeo ter sido capturado com uma iluminação diferente, onde no *dataset* do treinamento não há placas com esse mesmo tipo de iluminação, o que ocasiona a baixa precisão no reconhecimento desses sinais de trânsito. As Figuras 68 e 69 apresentam exemplos desses tipos de iluminação.

Figura 68 – Exemplo iluminação boa.

Fonte: Elaborado pelo autor.

No mapeamento realizado nos Vídeos 1, 2 e 3 que foram realizados por uma câmera, a acurácia das coordenadas geradas é aproximadamente de 40m, muito abaixo da precisão alcançada por alguns dos trabalhos correlatos. Essa precisão do sensor GPS pode ser melhorada fazendo uso de um sensor GPS externo que ofereça melhor precisão.

No mapeamento realizado utilizando os vídeos das três câmeras a acurácia dos dados do GPS é de aproximadamente 20m. Nesse caso, a velocidade do veículo na produção dos vídeos, que nesse caso é mais baixa, impactou na geração dos dados do *GPS*. E também

Figura 69 – Exemplo iluminação ruim.

Fonte: Elaborado pelo autor.

nesse experimento é possível validar que as coordenadas nas diferentes câmeras tem uma diferença mínima, não impactando na localização dos sinais de trânsito. Nesse caso é possível usar somente a coordenada central para produzir o mapeamento final.

Capítulo 5

Conclusão

Esta dissertação propõe a análise de um método para classificação de lombadas utilizando técnicas de ML e também a análise de um método para criação de inventário de sinais de trânsito utilizando técnicas de DL. Ambos os métodos utilizando uma plataforma de mapeamento móvel e mesmo sensores de baixo custo.

O método para classificação de lombadas utiliza os dados dos sensores de uma das câmeras GoPro instaladas na plataforma de mapeamento, esses dados passam por um pré-processamento para manter os dados em uma mesma taxa e desses dados é montado um conjunto de dados com as características das lombadas. Sobre esse conjunto de dados são aplicadas algumas técnicas de ML para treinamento e posterior classificação das lombadas, permitindo, com o resultado, a criação de um mapeamento com os dados das lombadas.

O método para criação de inventário de sinais de trânsito utiliza os conjunto de dados das imagens geradas pelas câmeras GoPro, como também os dados do sensor GPS. São aplicadas técnicas de DL sobre as imagens para treinamento de arquiteturas CNN que são utilizadas na detecção e reconhecimento dos sinais de trânsito. Após essa detecção é possível gerar um inventário que informa a quantidade dos sinais em um determinado vídeo, como também relacionar os *frames* com os dados do GPS para criação de um inventário de localização dos sinais de trânsito classificados.

5.1 Principais Contribuições

A seguir são apresentadas as contribuições alcançadas no desenvolvimento do trabalho:

- Desenvolvimento de um método para classificação de lombadas e mapeamento das mesmas;
- Desenvolvimento de um método para criação de inventários utilizando imagens e dados de GPS, o qual pode ser utilizado em outros problemas que necessitem criar algum tipo de inventário baseado em imagens;

- ❏ Criação de um conjunto de dados de imagens de sinais de trânsito, juntamente com os rótulos tanto para Pascal VOC quanto YOLO. Disponível no GitHub[1], com o intuito de tornar a pesquisa reprodutível.

5.2 Trabalhos Futuros

Os resultados obtidos no trabalho descrito aqui mostraram o bom desempenho do método e motivam novas investigações para trabalhos futuros. Algumas novas possíveis linhas de investigação são listadas a seguir:

- ❏ Com base no conjunto das imagens para treinamento, tem-se a necessidade de aumentar o mesmo, incluindo imagens de outras cidades, uma vez que as placas tem sutis diferenças entre uma cidade e outra, como também incluir imagens com variação de iluminação;

- ❏ Em relação às arquiteturas CNN utilizadas, pode-se avaliar a utilização de arquiteturas diferentes sendo usadas em conjunto, na tentativa de melhorar a acurácia;

- ❏ Com base na geração dos dados de localização, pode-se avaliar o uso de fotogrametria, aplicada aos sinais reconhecidos nas três câmeras, possibilitando melhorar a precisão da localização dos sinais de trânsito.

- ❏ Avaliar métodos para interpolar os dados gerados pelo sensor GPS, para utilizar a melhor posição e não somente um dos registros gerados em um determinado segundo.

5.3 Contribuições em Produção Bibliográfica

A pesquisa realizada durante esta dissertação produziu a publicação de uma revista e a submissão de outra. A seguir são apresentados detalhes das duas revistas e dos respectivos meios de publicação:

- ❏ Artigo intitulado "An evaluation of machine learning methods for speed-bump detection on a GoPro dataset"aceito na publicação "Anais da Academia Brasileira de Ciências"em 2019. Neste artigo foram apresentados os resultados referente a classificação de lombadas.

- ❏ Artigo intitulado "Camera Synchronization Using an Audio-Based Approach for Mobile Mapping Systems"publicado em 11th International Conference on Mobile Mapping Technology em 2019. Neste artigo foi apresentado uma abordagem utilizando áudio para sincronizar vídeos.

[1] Código: https://github.com/johnymbr/signal-traffic-image-dataset

- Artigo intitulado "Inventory of traffic signs using deep learning on GoPro images"submetido para a publicação "Anais da Academia Brasileira de Ciências"em 2020. Este artigo apresenta o método para inventário de sinais de trânsito.

Referências

ALJAAFREH, A. et al. Fuzzy inference system for speed bumps detection using smart phone accelerometer sensor. v. 9, p. 133–136, 01 2017. Disponível em: <https://www.researchgate.net/publication/320456266_Fuzzy_Inference_System_for_Speed_Bumps_Detection_Using_Smart_Phone_Accelerometer_Sensor>.

ALPAYDIN, E. **Introduction to Machine Learning**. The MIT Press, 2004. (Adaptive Computation and Machine Learning). ISBN 0262012111,9780262012119. Disponível em: <https://mitpress.mit.edu/books/introduction-machine-learning>.

BALADO, J. et al. Novel Approach to Automatic Traffic Sign Inventory Based on Mobile Mapping System Data and Deep Learning. **Remote Sensing**, v. 12, n. 3, p. 442, jan. 2020. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute. Disponível em: <https://doi.org/10.3390/rs12030442>.

BALALI, V.; RAD, A. A.; GOLPARVAR-FARD, M. Detection, classification, and mapping of U.S. traffic signs using google street view images for roadway inventory management. **Visualization in Engineering**, v. 3, n. 1, p. 15, dez. 2015. ISSN 2213-7459. Disponível em: <https://doi.org/10.1186/s40327-015-0027-1>.

BASCóN, S. M. et al. Traffic Sign Recognition System for Inventory Purposes. p. 6, 2008. Disponível em: <https://doi.org/10.1109/IVS.2008.4621233>.

BELLO-SALAU, H. et al. New road anomaly detection and characterization algorithm for autonomous vehicles. **Applied Computing and Informatics**, 2018. ISSN 2210-8327. Disponível em: <https://doi.org/10.1016/j.aci.2018.05.002>.

BONACCORSO, G. **Machine Learning Algorithms**. Packt, 2017. ISBN 978-1-78588-962-2. Disponível em: <https://www.amazon.com/Machine-Learning-Algorithms-reference-algorithms/dp/1785889621>.

BOUJEMAA, K. S. et al. Traffic sign recognition using convolutional neural networks. In: **2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)**. [s.n.], 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/WINCOM.2017.8238205>.

BRASIL. Lei nº 9.503, de 23 de setembro de 1997. institui o código de trânsito brasileiro. **Diário Oficial [da] República Federativa do Brasil**, Brasília, DF, 1997. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/L9503Compilado.htm>.

BRASIL. Resolução nº 600, de 24 de maio de 2016. estabelece os padrões e critérios para a instalação de ondulação transversal. **Diário Oficial [da] República Federativa do Brasil**, Brasília, DF, 2016. Disponível em: <https://www.in.gov.br/materia/-/asset_publisher/Kujrw0TZC2Mb/content/id/22921408/do1-2016-05-27-resolucao-n-600-de-24-de-maio-de-2016-22921310#:~:text=Estabelece\%20os\%20padr\%C3\%B5es\%20e\%20crit\%C3\%A9rios,similares\%20implantadostransversalmente\%20\%C3\%A0\%20via\%20p\%C3\%BAblica.>

BRUNO, D. R. **Análise e fusão de imagens 2D e 3D com vistas para detecção e classificação de sinais de trânsito verticais em prol da segurança viária com veículos robóticos inteligentes**. Tese (Doutorado) — Universidade de São Paulo, abr. 2020. Disponível em: <https://doi.org/10.11606/T.55.2020.tde-23072020-180612>.

BRUNO, D. R. et al. Analysis and fusion of 2d and 3d images applied for detection and recognition of traffic signs using a new method of features extraction in conjunction with Deep Learning. In: **2018 International Joint Conference on Neural Networks (IJCNN)**. [s.n.], 2018. p. 1–8. Disponível em: <https://doi.org/10.1109/IJCNN.2018.8489538>.

CELAYA-PADILLA, J. M. et al. Speed bump detection using accelerometric features: A genetic algorithm approach. **Sensors**, v. 18, n. 2, 2018. ISSN 1424-8220. Disponível em: <https://doi.org/10.3390/s18020443>.

CHEN, K. et al. MMDetection: Open mmlab detection toolbox and benchmark. **arXiv preprint arXiv:1906.07155**, 2019. Disponível em: <https://arxiv.org/abs/1906.07155>.

CHOLLET, F. **Deep Learning with Python**. Manning Publications, 2017. ISBN 1617294438,9781617294433. Disponível em: <https://www.manning.com/books/deep-learning-with-python>.

CNT, C. N. d. T. Acidentes rodoviários e infraestrutura. 2018. Accepted: 2019-01-30. Disponível em: <http://repositorio.itl.org.br/jspui/handle/123456789/170>.

DANTAS, G. V. **Utilização de classificador Random Forest na detecção de falhas em máquinas rotativas**. 93 f. Monografia (Bacharel) — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2015. Disponível em: <http://www.monografias.poli.ufrj.br/monografias/monopoli10015019.pdf>.

DENATRAN. **Manual Brasileiro de Sinalização de Trânsito, Volume I, Sinalização Vertical de Regulamentação**. [s.n.], 2007. ISBN 978-85-7958-074-1. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/transito/arquivos-denatran/educacao/publicacoes/manual_vol_i_2.pdf>.

DENATRAN. **Manual Brasileiro de Sinalização de Trânsito, Volume II, Sinalização Vertical de Advertência**. [s.n.], 2007. ISBN 978-85-7958-075-8. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/transito/arquivos-denatran/educacao/publicacoes/manual_vol_ii_-2.pdf>.

DENATRAN. **Manual Brasileiro de Sinalização de Trânsito, Volume III, Sinalização Vertical de Indicação**. [s.n.], 2014. ISBN 978-85-7958-076-5. Disponível em: <https://www.gov.br/infraestrutura/pt-br/assuntos/transito/arquivos-denatran/educacao/publicacoes/manual_vol_iii_2.pdf>.

DEVAPRIYA, W.; BABU, C. N. K.; SRIHARI, T. Advance Driver Assistance System (ADAS) - Speed bump detection. In: **2015 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)**. [s.n.], 2015. p. 1–6. Disponível em: <https://doi.org/10.1109/ICCIC.2015.7435753>.

DHAR, P. et al. Traffic sign detection — A new approach and recognition using convolution neural network. In: **2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)**. [s.n.], 2017. p. 416–419. Disponível em: <https://doi.org/10.1109/R10-HTC.2017.8288988>.

DONGES, N. **A Complete Guide to the Random Forest Algorithm**. 2019. Accessed: 2021-02-17. Disponível em: <https://builtin.com/data-science/random-forest-algorithm>.

FERNáNDEZ, C. et al. Free space and speed humps detection using lidar and vision for urban autonomous navigation. In: **2012 IEEE Intelligent Vehicles Symposium**. [s.n.], 2012. p. 698–703. Disponível em: <https://doi.org/10.1109/IVS.2012.6232255>.

FFmpeg. 2018. Accessed: 2018-09-20. Disponível em: <https://www.ffmpeg.org/>.

FU, M.; HUANG, Y. A survey of traffic sign recognition. In: **2010 International Conference on Wavelet Analysis and Pattern Recognition**. [s.n.], 2010. p. 119–124. Disponível em: <https://doi.org/10.1109/ICWAPR.2010.5576425>.

GOODFELLOW YOSHUA BENGIO, A. C. I. **Deep Learning**. MIT Press, 2016. Disponível em: <https://mitpress.mit.edu/books/deep-learning>.

gopro-utils. 2018. Accessed: 2018-09-20. Disponível em: <https://github.com/stilldavid/gopro-utils>.

HARASTHY, T.; TURAN, J.; OVSENIK, L. Optical correlator based Traffic Signs Inventory system. In: **2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP)**. Bucharest, Romania: IEEE, 2013. p. 113–116. ISBN 978-1-4799-0944-5 978-1-4799-0941-4 978-1-4799-0942-1. Disponível em: <https://doi.org/10.1109/IWSSIP.2013.6623463>.

HAYKIN, S. **Neural Networks - A Comprehensive Foundation, Second Edition**. 2. ed. Prentice Hall, 1998. ISBN 9780132733502,0132733501. Disponível em: <https://www.pearson.com/us/higher-education/product/Haykin-Neural-Networks-A-Comprehensive-Foundation-2nd-Edition/9780132733502.html>.

HAZELHOFF, L.; CREUSEN, I. Robust classification system with reliability prediction for semi-automatic traffic-sign inventory systems. p. 8, 2013. Disponível em: <https://doi.org/10.1109/WACV.2013.6475009>.

HE, G.; ORVETS, G. Capturing road network data using mobile mapping technology. **International Archives of Photogrammetry and Remote Sensing**, XXXIII, 2000. Disponível em: <https://www.researchgate.net/publication/228960052_Capturing_road_network_data_using_mobile_mapping_technology>.

HIDALGO, E. et al. Wireless inventory of traffic signs based on passive RFID technology. In: **IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society**. Vienna, Austria: IEEE, 2013. p. 5467–5471. ISBN 978-1-4799-0224-8. Disponível em: <https://doi.org/10.1109/IECON.2013.6700026>.

HOELSCHER, I. G. **Detecção e classificação de sinalização vertical de trânsito em cenários complexos**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, 2017. Disponível em: <https://lume.ufrgs.br/handle/10183/163777>.

HUANG, S.; LIN, H.; CHANG, C. An in-car camera system for traffic sign detection and recognition. In: **2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)**. [s.n.], 2017. p. 1–6. Disponível em: <https://doi.org/10.1109/IFSA-SCIS.2017.8023239>.

JOCHER, G. et al. **ultralytics/yolov5: v3.1**. 2020. Accessed: 2020-11-20. Disponível em: <https://zenodo.org/record/4154370#.X7ex7VlKhhE>.

JULIA, E. d. S. **Um estudo comparativo entre algoritmos de classificação para Fluxos Contínuos de Dados**. 73 f. Monografia (Bacharel) — Faculdade de Computação, Universidade Federal de Uberlâandia, Uberlândia, 2018. Disponível em: <https://repositorio.ufu.br/handle/123456789/27001>.

KARADUMAN, M.; EREN, H. Deep learning based traffic direction sign detection and determining driving style. In: **2017 International Conference on Computer Science and Engineering (UBMK)**. [s.n.], 2017. p. 1046–1050. Disponível em: <https://doi.org/10.1109/UBMK.2017.8093453>.

KERAS. 2019. Accessed: 2019-06-18. Disponível em: <https://keras.io/>.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. ImageNet Classification with Deep Convolutional Neural Networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time-series. In: . [s.n.], 1995. Disponível em: <https://dl.acm.org/doi/10.5555/303568.303704>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, maio 2015. ISSN 0028-0836, 1476-4687. Disponível em: <https://doi.org/10.1038/nature14539>.

LIN, T.-Y. et al. Focal Loss for Dense Object Detection. **arXiv:1708.02002 [cs]**, fev. 2018. ArXiv: 1708.02002 version: 2. Disponível em: <https://doi.org/10.1109/TPAMI.2018.2858826>.

LIU, W. et al. SSD: Single Shot MultiBox Detector. **arXiv:1512.02325 [cs]**, v. 9905, p. 21–37, 2016. ArXiv: 1512.02325. Disponível em: <https://doi.org/10.1007/978-3-319-46448-0_2>.

MITCHELL, T. M. **Machine Learning**. 1. ed. McGraw-Hill, 1997. (McGraw-Hill series in computer science). ISBN 9780070428072,0070428077. Disponível em: <http://www.cs.cmu.edu/~tom/mlbook.html>.

MOLINA M. BLáZQUEZ, J. S. P.; COLOMINA, I. mapkite: A new paradigm for simultaneous aerial and terrestrial geodata acquisition and mapping. **The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, XLI-B1, 2016. Disponível em: <https://doi.org/10.5194/isprsarchives-XLI-B1-1-2016>.

OLIVEIRA, H. et al. Camera synchronization using an audio-based approach for mobile mapping systems. In: . [s.n.], 2019. Disponível em: <https://www.researchgate.net/publication/333001290_CAMERA_SYNCHRONIZATION_USING_AN_AUDIO-BASED_APPROACH_FOR_MOBILE_MAPPING_SYSTEMS>.

OPAS. 2019. Accessed: 2019-06-28. Disponível em: <https://www.paho.org/bra/index.php?option=com_content&view=article&id=5147:acidentes-de-transito-folha-informativa&Itemid=779>.

OPENCV. 2019. Accessed: 2019-06-18. Disponível em: <https://opencv.org/>.

PATTERSON, A. G. J. **Deep Learning: A Practitioner's Approach**. 1. ed. O'Reilly Media, 2017. ISBN 1491914254,9781491914250. Disponível em: <https://www.amazon.com.br/Deep-Learning-Josh-Patterson/dp/1491914254>.

PRF. **Balanço PRF 2017**. 2019. Accessed: 2019-06-28. Disponível em: <https://www.prf.gov.br/portal/sala-de-imprensa/releases-1/balanco-prf-2017>.

REDMON, J.; FARHADI, A. YOLOv3: An Incremental Improvement. p. 6, 2018. Disponível em: <https://arxiv.org/abs/1804.02767>.

REN, S. et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. **arXiv:1506.01497 [cs]**, jan. 2016. ArXiv: 1506.01497. Disponível em: <http://arxiv.org/abs/1506.01497>.

ROSEBROCK, A. **Deep Learning for Computer Vision with Python**. 1.1.0. ed. PyImageSearch, 2017. v. 1,Starter Bundle. Disponível em: <https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/>.

RUDER, S. An overview of gradient descent optimization algorithms. **arXiv:1609.04747 [cs]**, jun. 2017. ArXiv: 1609.04747. Disponível em: <http://arxiv.org/abs/1609.04747>.

RUSSELL, S. et al. **Artificial Intelligence: A Modern Approach**. Prentice Hall, 2010. (Prentice Hall series in artificial intelligence). ISBN 9780136042594. Disponível em: <https://www.amazon.com.br/Artificial-Intelligence-Approach-Stuart-Russell/dp/0136042597>.

Seguradora Líder. **Relatório Seguradora Líder 2018**. 2019. Accessed: 2019-06-28. Disponível em: <https://www.seguradoralider.com.br/Documents/Relatorio-Anual/RELATORIO\%20ANUAL_2018_WEB.pdf>.

SILVA, I. d. S.; SPATTI, D. H.; FLAUZINO, R. A. **Redes Neurais Artificiais para engenharia e ciências aplicadas**. 1. ed. Artliber Editora, 2010. ISBN 978-85-88098-53-4. Disponível em: <https://www.amazon.com.br/Artificiais-Engenharia-Ci\%C3\%AAncias-Aplicadas-Fundamentos/dp/8588098873>.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. **arXiv:1409.1556 [cs]**, set. 2014. ArXiv: 1409.1556. Disponível em: <http://arxiv.org/abs/1409.1556>.

SOILáN, M. et al. Traffic sign detection in MLS acquired point clouds for geometric and image-based semantic inventory. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 114, p. 92–101, abr. 2016. ISSN 09242716. Disponível em: <https://doi.org/10.1016/j.isprsjprs.2016.01.019>.

SOILáN, M. et al. Automatic extraction of road features in urban environments using dense als data. **International Journal of Applied Earth Observation and Geoinformation**, v. 64, p. 226 – 236, 2018. ISSN 0303-2434. Disponível em: <https://doi.org/10.1016/j.jag.2017.09.010>.

SOLUS, D.; OVSENIK, L.; TURAN, J. Inventory System of Vertical Traffic Signs. In: **2015 25th International Conference Radioelektronika (RADIOELEKTRONIKA)**. Pardubice, Czech Republic: IEEE, 2015. p. 121–124. ISBN 978-1-4799-8117-5 978-1-4799-8119-9. Disponível em: <https://doi.org/10.1109/RADIOELEK.2015.7128972>.

SUTTON, R. et al. **Reinforcement Learning: An Introduction**. MIT Press, 1998. (A Bradford book). ISBN 978-0-262-19398-6. Disponível em: <https://books.google.com.br/books?id=CAFR6IBF4xYC>.

SZEGEDY, C. et al. Going Deeper with Convolutions. In: **Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2015. Disponível em: <https://doi.org/10.1109/CVPR.2015.7298594>.

TAO, C. V.; LI, J. **Advances in Mobile Mapping Technology**. Taylor and Francis, 2007. 176 p. ISBN 9780415427234. Disponível em: <https://doi.org/10.4324/9780203961872>.

TENSORFLOW. 2019. Accessed: 2019-05-21. Disponível em: <https://www.tensorflow.org/tutorials>.

TZUTALIN. **LabelImg**. 2015. Disponível em: <https://github.com/tzutalin/labelImg>.

VARMA, V. S. K. P. et al. Real Time Detection of Speed Hump/Bump and Distance Estimation with Deep Learning using GPU and ZED Stereo Camera. **Procedia Computer Science**, v. 143, p. 988–997, jan. 2018. ISSN 1877-0509. Disponível em: <https://doi.org/10.1016/j.procs.2018.10.335>.

VENNELAKANTI, A. et al. Traffic Sign Detection and Recognition using a CNN Ensemble. In: **2019 IEEE International Conference on Consumer Electronics (ICCE)**. [s.n.], 2019. p. 1–4. Disponível em: <https://doi.org/10.1109/ICCE.2019.8662019>.

WANG, C. Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning. In: **2018 International Conference on Robots Intelligent System (ICRIS)**. [s.n.], 2018. p. 150–152. Disponível em: <https://doi.org/10.1109/ICRIS.2018.00047>.

WANG, W. et al. A review of road extraction from remote sensing images. **Journal of Traffic and Transportation Engineering (English Edition)**, v. 3, n. 3, p. 271 – 282, 2016. ISSN 2095-7564. Disponível em: <https://doi.org/10.1016/j.jtte.2016.05.005>.

Weka. 2018. Accessed: 2018-09-20. Disponível em: <https://www.cs.waikato.ac.nz/ml/weka/>.

WEN, C. et al. Spatial-Related Traffic Sign Inspection for Inventory Purposes Using Mobile Laser Scanning Data. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 1, p. 27–37, jan. 2016. ISSN 1524-9050, 1558-0016. Disponível em: <https://doi.org/10.1109/TITS.2015.2418214>.

YAP, M. H. et al. Deep Learning in Diabetic Foot Ulcers Detection: A Comprehensive Evaluation. **arXiv:2010.03341 [cs]**, out. 2020. ArXiv: 2010.03341. Disponível em: <http://arxiv.org/abs/2010.03341>.

ZHAO, J.; YOU, S.; HUANG, J. Rapid extraction and updating of road network from airborne lidar. In: **Data", IEEE Applied Imagery Pattern Recognition Workshop (AIPR**. [s.n.], 2011. p. 1–7. Disponível em: <https://doi.org/10.1109/AIPR.2011.6176360>.