

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JOÃO LUIZ RAMALHEIRA DE ALMEIDA

Um estudo de técnicas de classificação de placas de sinalização interna
para deficientes visuais utilizando um dispositivo vestível

Maringá
2019

JOÃO LUIZ RAMALHEIRA DE ALMEIDA

Um estudo de técnicas de classificação de placas de sinalização interna
para deficientes visuais utilizando um dispositivo vestível

Dissertação apresentada ao Programa de
Pós-Graduação em Ciência da Computação
do Departamento de Informática, Centro
de Tecnologia da Universidade Estadual
de Maringá, como requisito parcial para
obtenção do título de Mestre em Ciência da
Computação.

Orientador: Prof. Dr. Franklin César Flores

Maringá
2019

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá - PR, Brasil)

A447e

Almeida, João Luiz Ramalheira de

Um estudo de técnicas de classificação de placas de sinalização interna para deficientes visuais utilizando um dispositivo vestível / João Luiz Ramalheira de Almeida. -- Maringá, PR, 2019.

66 f.: il. color., figs., tabs.

Orientador: Prof. Dr. Franklin César Flores.

Dissertação (Mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2019.

1. Visão Computacional. 2. Processamento de Imagens. 3. Inteligência Artificial. 4. Acessibilidade. 5. Interação humano-computador. I. César Flores, Franklin, orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 23.ed. 004.019

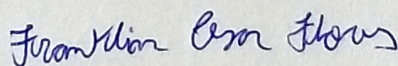
FOLHA DE APROVAÇÃO

JOÃO LUIZ RAMALHEIRA DE ALMEIDA

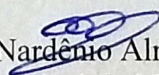
Um estudo de técnicas de classificação de placas de sinalização interna para deficientes visuais utilizando um dispositivo vestível

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

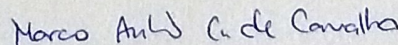
BANCA EXAMINADORA



Prof. Dr. Franklin César Flores
Universidade Estadual de Maringá – DIN/UEM



Prof. Dr. Nardênio Almeida Martins
Universidade Estadual de Maringá – DIN/UEM



Prof. Dr. Marco Antonio Garcia de Carvalho
Universidade Estadual de Campinas – FT/UNICAMP

Aprovada em: 16 de dezembro de 2019.

Local da defesa: Sala 101, Bloco C56, *campus* da Universidade Estadual de Maringá.

AGRADECIMENTOS

Agradeço a Deus por ter me abençoado nesta caminhada e em todos os momentos de minha vida.

A toda a minha família, em especial os meus pais, Moacir e Rosilda, minha irmã, Mayra, e meu cunhado, Ricardo, pelo apoio e incentivo que me deram durante os estudos.

A minha companheira, Izabela, que sempre esteve ao meu lado me apoiando e incentivando com paciência e que agora compartilha desse momento comigo.

Aos meus amigos de mestrado, em especial Max, Marco Antônio, Alisson e Lailla, que estiveram comigo durante os momentos de estudos e descontração.

Ao meu orientador professor Dr. Franklin César Flores pelo apoio, comentários e sugestões no desenvolvimento deste projeto.

Agradeço também aos demais professores do Departamento de Informática que contribuíram para minha formação. Em especial ao professor Dr. Yandre Maldonado e Gomes da Costa pela colaboração na elaboração da Base de Dados elaborada nesse trabalho. Vale ainda um agradecimento à Inês pela amizade, conversas, conselhos e disponibilidade em sempre ajudar.

Por fim, o presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Um estudo de técnicas de classificação de placas de sinalização interna para deficientes visuais utilizando um dispositivo vestível

RESUMO

Placas de sinalização interna tem papel fundamental para a localização de determinados destinos, tais como: banheiros masculinos e femininos, saída de emergência, entre outros. Deficientes visuais necessitam do auxílio de outras pessoas quando desejam encontrar tal destino. Uma tecnologia de reconhecimento dessas placas pode ser útil para uma maior independência de locomoção. O reconhecimento de objetos é uma tarefa trivial para humanos, contudo não tão simples para computadores. Este trabalho tem como objetivo apresentar o desenvolvimento de um sistema computacional que auxilia deficientes visuais no reconhecimento de placas de sinalização interna para melhor localização em ambientes públicos. Para o processo de detecção de placas foi utilizado o algoritmo *Haar Cascade* e para o processo de classificação abordou-se métodos *handcrafted* e *non-handcrafted*. No método *handcrafted*, foram utilizados os descritores Histograma de Cores, LBP (*Local Binary Patterns*), HOG (*Histogram of Oriented Gradients*) e DAISY aplicados nos classificadores SVM (*Support Vector Machine*), k-NN (*k-Nearest Neighbors*) e MLP (*MultiLayer Perceptron*). Já na abordagem *non-handcrafted* utilizou-se Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Network*). O processo de detecção de placas atingiu 96% e os resultados de classificação chegaram a uma taxa de acurácia de 90,33%. Por fim, foi construído um dispositivo vestível que tem como finalidade avaliar o desempenho e viabilidade de utilização das técnicas escolhidas.

Palavras-chave: Visão Computacional; Reconhecimento de Placas de Sinalização Interna; Deficientes Visuais; Rede Neural Convolucional;

A study of indoor signs classification techniques for the visually impaired using a wearable device

ABSTRACT

Indoor signs are very useful to locate certain destinations, such as male and female toilets, emergency exit, and others. Visually impaired people frequently need help from others when they want to find a specific destination. A technology of signs recognition can be useful for greater independence of locomotion. Object recognition is a trivial task for men, however, it is not so simple for computers. The objective of this work is to present the development of a computational system that assists the visually impaired in the recognition of indoor signs for a better location in public environments. For the sign detection, the Haar Cascade algorithm was used, and for the classification process, handcrafted and non-handcrafted methods were employed. The handcrafted method used was descriptors Color Histogram, Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and DAISY applied to Support Vector Machine (SVM), k-Nearest Neighbors (k-NN) and MultiLayer Perceptron MLP. The non-handcrafted approach, Convolutional Neural Networks (CNN) were used. The sign detection process reached 96% and the classification results reached an accuracy rate of 90.33%. Finally, a wearable device was designed to evaluate the performance and feasibility of using the chosen techniques.

Keywords: Computer Vision; Indoor Sign Recognition; Visually Impaired; Convolutional Neural Network.

LISTA DE FIGURAS

Figura - 1.1	Exemplos de placas de banheiro masculino de diferentes modelos. Fonte: o autor.	13
Figura - 2.1	Esquema de um sistema de classificação. Fonte: adaptado de Duda et al. (2000).	17
Figura - 2.2	Exemplo de <i>Haar-like feature</i> . Fonte: (Viola e Jones, 2001). . . .	18
Figura - 2.3	Esquema do <i>Haar Cascade</i> . Fonte: Adaptado de Viola e Jones (2001).	19
Figura - 2.4	Operador LBP. Fonte: adaptada de Ojala et al. (2002).	20
Figura - 2.5	Exemplos de Histograma de cores de uma imagem. Fonte: o autor	21
Figura - 2.6	O descritor DAISY. Fonte: (Tola et al., 2010).	22
Figura - 2.7	<i>Kernels</i> de detecção de gradiente. (a) para gradientes horizontais e (b) para gradientes verticais. Fonte: o autor.	23
Figura - 2.8	Exemplos de utilização do SVM. Fonte: o autor.	24
Figura - 2.9	Exemplos de utilização do k-NN. Fonte: o autor.	25
Figura - 2.10	Exemplo de uma MLP. Fonte: (Haykin, 2001).	26
Figura - 2.11	Exemplo de intercalação de camadas em CNN. Fonte: (Albelwi e Mahmood, 2017).	27
Figura - 2.12	Ligação dos neurônios na camada convolucional (superior) e ligação dos neurônios em uma MLP (inferior) . Fonte: (Goodfellow et al., 2016)	29
Figura - 2.13	Exemplo de aplicação de convolução de cinco <i>kernels</i> em uma imagem RGB. Fonte: (Li et al., 2016)	29
Figura - 2.14	Exemplo de utilização do <i>maxpooling</i> . Fonte: (Li et al., 2016) . . .	30
Figura - 2.15	Exemplo de transformações de <i>Data Augmentation</i> . Fonte: o autor.	31
Figura - 2.16	Processo de validação cruzada. Fonte: Adaptada de Raschka (2015)	33
Figura - 2.17	Raspberry Pi 2 Model B. Fonte: (RASPBerry PI FOUNDA- TION, 2019).	36
Figura - 2.18	Raspberry Pi Câmera v2. Fonte: (RASPBerry PI FOUNDA- TION, 2019).	37
Figura - 3.1	Etapa de treinamento. Fonte: o autor.	40
Figura - 3.2	Modelo proposto: etapa de classificação. Fonte: o autor.	41
Figura - 3.3	Exemplos de placas da base de dados. Fonte: o autor.	42
Figura - 3.4	<i>Frames</i> de um vídeo da base de dados. Fonte: o autor.	43

Figura - 3.5	<i>Frames</i> de um vídeo da base de dados. Fonte: o autor.	44
Figura - 3.6	Exemplos de imagens aplicando <i>Data Augmentation</i> . Fonte: o autor	49
Figura - 3.7	Peça de acoplamento da câmera. Fonte: o autor	50
Figura - 3.8	Peça de acoplamento da plataforma. Fonte: o autor	50
Figura - 3.9	Protótipo. Fonte: o autor	50
Figura - 3.10	Utilização do protótipo. Fonte: o autor	51
Figura - 3.11	Fluxo de execução. Fonte: o autor	51
Figura - 4.1	Exemplo de <i>frames</i> processados com demarcação da placa encontrada. Esse vídeo é o mesmo apresentado na Figura - 3.4	53
Figura - 4.2	Exemplos de placas de banheiros masculino e feminino.	55

LISTA DE TABELAS

Tabela - 2.1	Especificações do Raspberry Pi 2 B	36
Tabela - 2.2	Comparação dos trabalhos relacionados	39
Tabela - 3.1	Resumo das amostras por classes	42
Tabela - 3.2	Arquitetura da Rede	46
Tabela - 4.1	Relação de Placas/Acertos/Erros	53
Tabela - 4.2	Resultados Experimentais	54
Tabela - 4.3	Matriz de confusão utilizando CNN	55
Tabela - 4.4	Métricas de Avaliação	56
Tabela - 4.5	Métricas avaliadas no protótipo	56

SUMÁRIO

1	Introdução	12
1.1	Objetivos	14
1.2	Estrutura do trabalho	15
2	Fundamentação Teórica	16
2.1	Processo de Classificação de Objetos	16
2.2	Segmentação	17
2.2.1	Haar Cascade	18
2.3	Extração de Características	19
2.3.1	LBP	19
2.3.2	Histograma de Cores	20
2.3.3	DAISY	21
2.3.4	HOG	22
2.4	Classificadores	23
2.4.1	<i>Support Vector Machine</i> (SVM)	23
2.4.2	<i>k-Nearest Neighbors</i> (K-NN)	24
2.4.3	MLP	26
2.5	Redes Neurais Convolucionais	27
2.5.1	<i>Data Augmentation</i>	31
2.5.2	<i>Dropout</i>	32
2.5.3	Validação Cruzada	33
2.6	Métricas de Avaliação	33
2.6.1	Acurácia	34
2.6.2	<i>Precision</i>	34
2.6.3	<i>Recall</i>	35
2.6.4	<i>F-measure</i>	35
2.6.5	<i>Macro-F</i>	35
2.7	Raspberry PI	35
2.8	Trabalhos Relacionados	37
3	Materiais e Métodos	40
3.1	Base de Dados	41
3.2	Detecção de Placas	44
3.3	Método <i>handcrafted</i>	45
3.4	Método <i>non-handcrafted</i>	46

3.5	Protótipo	48
4	Resultados Experimentais	52
4.1	Detector	52
4.2	Classificador	54
4.3	Protótipo	56
5	Conclusão	57
5.1	Contribuições	58
5.2	Dificuldades Encontradas	58
5.3	Trabalhos Futuros	59
	REFERÊNCIAS	60

Introdução

Existem cerca de 285 milhões de pessoas com deficiência visual no mundo das quais 39 milhões são completamente cegas (Prajapati e Shah, 2016). No Brasil, Oliveira (2012) apresenta que 18,6% da população brasileira possui algum tipo de deficiência visual. Atualmente, a tecnologia colabora para a inclusão dessas pessoas na sociedade, facilitando o relacionamento, estudo e capacidade de adaptação a diversas situações.

A capacidade de visualizar e interpretar diversas placas de sinalização é muito importante para os humanos. Essas placas desempenham um papel importante para encontrar alguns destinos, tais como: banheiros masculinos e femininos, saída de emergência, entre outros (Wang et al., 2013). Para uma pessoa que não tenha o sentido da visão, um equipamento capaz de auxiliar a identificação de placas de sinalização pode ser útil na locomoção, principalmente em ambientes públicos.

Um sistema de reconhecimento visa reconhecer e encontrar objetos do mundo real por meio de uma imagem extraída. Esse procedimento utiliza diversos modelos do objeto desejado que já são conhecidos e classificados a priori (Jain et al., 1995). A identificação de objetos em imagens é uma tarefa trivial para os seres humanos, porém requer algum esforço para a máquina.

Das diversas vertentes que existem nessa área de pesquisa, recentemente a classificação de placas de sinalização vem ganhando destaque. Isso porque várias tecnologias de carros autônomos vem surgindo no mercado. Existem também algumas bases de dados referentes a estas placas disponíveis para acesso (Larsson et al., 2011) (Mogelmose et al., 2012) (Mogelmose et al., 2014).

Ni et al. (2014) afirmam que o reconhecimento e classificação de placas de sinalização interna estão em um estágio inicial. Alguns trabalhos apresentam pesquisas voltadas a

deficientes visuais (Ni et al., 2014) (Wang et al., 2013), (Kunene et al., 2016). Entretanto, as bases de imagens utilizadas nesses trabalhos não estão disponíveis e existem diversas questões em aberto nesse contexto de pesquisa.

O processo de reconhecimento de placas em ambientes interno não necessariamente utiliza as mesmas técnicas de reconhecimento de placas de trânsito por dois principais motivos: as placas de trânsito estão, no geral, destacadas na via, em sua maioria em pontos altos e de fácil visualização, já as placas de sinalização interna podem ficar oclusas, com ângulo de difícil visualização e ainda podem ser facilmente confundidas com outros objetos contidos no ambiente; outro ponto de destaque é que não há uma padronização dos desenhos, que variam de acordo com a região e cultura (Ni et al., 2014). A Figura - 1.1 exibe exemplos de placas de sinalização de uma mesma classe que são bem diferentes uma das outras, expondo o desafio que existe em classificar essas placas.



Figura 1.1: Exemplos de placas de banheiro masculino de diferentes modelos. Fonte: o autor.

Para solucionar um problema de reconhecimento de placas, pode-se dividir um sistema em cinco etapas: aquisição, pré-processamento, segmentação, extração de características e classificação. Nessas etapas utilizam-se técnicas de processamento de imagens, visão computacional e inteligência artificial. As etapas de aquisição e pré-processamento são responsáveis por capturar a imagem e aplicar técnicas que possam melhorar a imagem para as etapas futuras. A etapa de segmentação visa realizar o particionamento da imagem em regiões de interesse que sejam disjuntas.

No processo de extração de características, um passo importante é a engenharia de *features*. Existem várias propostas para essa etapa, das quais se destacam: *Local Binary Patterns* (LPB) que utiliza a informação de intensidade dos *pixels* vizinhos

como informação de textura; histograma de orientação do gradiente (HOG, do inglês *histogram of oriented gradient*) que calcula a quantidade de ocorrências de um gradiente de orientação na imagem (Dalal e Triggs, 2005); histograma de cores que conta a quantidade de cores de uma determinada imagem (Gonzalez e Woods, 2006) e DAISY, um descritor que utiliza gradiente de orientação com ponderação Gaussiana e um núcleo circularmente simétrico (Tola et al., 2010).

A etapa seguinte ao processo de extração de característica é realizar a classificação. Destaca-se no processo de reconhecimento de padrões três algoritmos classificadores, sendo eles: máquina de vetores de suporte (SVM, do inglês *Support Vector Machines*), k-NN (*k Nearest Neighbor*) e MLP (*MultyLayer Perceptron*). SVM é um modelo de aprendizagem supervisionado que visa encontrar um hiperplano n -dimensional que maximiza a distância entre os elementos de classes distintas (Vapnik, 1995). k-NN também é um modelo de aprendizado supervisionado que define uma classificação atribuindo a classe majoritária dos k vizinhos mais próximos, onde o valor de k é definido no momento de classificação (Mitchell, 1997). Assim como as demais, MLP é uma técnica de aprendizagem supervisionado que possui camadas de neurônios ligados entre si com pesos (Mitchell, 1997). Essa rede é composta de pelo menos três camadas, sendo uma de entrada, uma ou mais intermediárias e a camada de saída que tem a finalidade de construir a resposta.

Além disso, existem as Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Network*) que realizam o processo de extração de características e classificação de uma única vez (Lecun, 1989). Essas redes têm a capacidade de aprender os atributos que descrevem a imagem e, em seguida, realizar a classificação das imagens.

Em paralelo as diversas técnicas, são poucos os trabalhos que desenvolveram algum dispositivo vestível que possa ser utilizado por deficientes visuais para esse processo. Os que existem possuem um valor de aquisição muito elevado, dificultando a aquisição da maioria das pessoas. Elaborar um modelo de classificação que possa ser utilizado em *hardware* de baixo custo pode ser um desafio ainda maior.

1.1 Objetivos

O principal objetivo deste trabalho é propor um modelo de classificação de placas de sinalização interna para melhor orientação de deficientes visuais em ambientes públicos. Pretendendo alcançar esse objetivo, objetivos específicos devem ser completados, sendo eles:

- Estudar conceitos de classificação de placas de sinalização, levantando técnicas e modelos;
- Produzir uma base de dados de placas de sinalização interna e disponibilizá-la para futuras pesquisas.
- Elaborar uma arquitetura de Rede Neural Convolucional que seja capaz de ser implantada em uma plataforma portátil, de baixo custo e baixo consumo de energia;
- Construir um protótipo de um dispositivo vestível;
- Avaliar a eficácia do protótipo.

1.2 Estrutura do trabalho

Este documento encontra-se organizado da seguinte forma: no tópico 2 são apresentados fundamentos teóricos inerentes às técnicas que serão utilizadas na elaboração desta pesquisa, além de um levantamento dos principais trabalhos relacionados ao tema; o tópico 3 descreve os conceitos e apresenta o método proposto para a classificação de placas de sinalização interna; o tópico 4 apresenta os resultados experimentais desse trabalho, expondo os resultados dos algoritmos de classificação e análises da eficácia do protótipo; o tópico 4 apresenta as discussões e conclusões obtidas, além de trabalhos futuros; e por fim, é apresentada a bibliografia referenciada neste trabalho.

Fundamentação Teórica

Este tópico apresenta os conceitos inerentes a este trabalho. O texto encontra-se organizado considerando os seguintes referenciais teóricos: processo de classificação de objetos; segmentação; características de imagens; algoritmos classificadores; Redes Neurais Convolucionais; e descrição dos componentes eletrônicos utilizados no protótipo desenvolvido para a avaliação do modelo. Por fim, são apresentados alguns trabalhos relacionados ao processo de classificação de placas de sinalização interna.

2.1 Processo de Classificação de Objetos

Um sistema para classificação de objetos pode ser dividido em cinco etapas, sendo elas: Aquisição; Pré-processamento; Segmentação; Extração de características e Classificação (Duda et al., 2000). A Figura - 2.1 ilustra essa abordagem. Quando se utiliza o processo de extração de características separado do processo de classificação, é utilizada uma técnica denominada *handcrafted*. Quando se utiliza Redes Neurais Convolucionais (CNN), o processo de extração de características e classificação são realizados em uma única etapa, nesse utiliza-se um sistema chamado *non-handcrafted*.

A etapa de Aquisição, no contexto deste trabalho, visa a gravação de vídeos que contenham placas de sinalização interna. Esses vídeos serão processados por um dispositivo. A etapa de pré-processamento procura eliminar ruídos nas imagens e melhorias na imagem para as próximas etapas. Nesta etapa pode-se realizar rotação de vídeo caso seja gravado na horizontal, fazer correções de brilho entre outras transformações na imagem com o propósito de facilitar a extração de características.

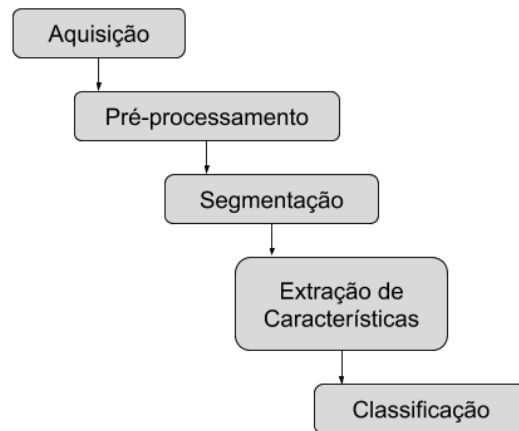


Figura 2.1: Esquema de um sistema de classificação. Fonte: adaptado de Duda et al. (2000).

2.2 Segmentação

Segmentação de imagens é um processo que realiza o particionamento da imagem em regiões de interesses que sejam disjuntas. O nível de detalhes depende do problema que está sendo resolvido (Mitchell, 1997). Mitchell (1997) ainda afirma que essa etapa é uma das tarefas mais difíceis no processamento de imagens e a precisão dela determina o sucesso ou fracasso de um sistema de classificação.

A definição formal de segmentação em imagens pode ser indicada como: seja R a representação de todo o domínio espacial ocupado por uma imagem, a segmentação de imagens pode ser definida como um processo que particiona R em n sub-regiões $R_1, R_2, \dots, R_n$ tal que:

1. $\bigcup_{i=1}^n R_i = R$
2. R_i é um conjunto conectado, $i = 1, 2, \dots, n$
3. $R_i \cap R_j = \emptyset$ para todo i e j tais que $i \neq j$

Existem na literatura diversos tipos de segmentação, dentre elas: baseada em formas, em características dos *pixels*, histograma e movimentos (Gonzalez e Woods, 2006). Uma das técnicas é denominada de *Haar Cascade* (Viola e Jones, 2001) e foi a escolhida para ser utilizada neste trabalho.

2.2.1 Haar Cascade

Haar Cascade, apresentado por Viola e Jones (2001), consiste em um algoritmo de aprendizado capaz de encontrar determinado padrão em uma imagem e assim distinguir uma região em “objeto” e “não-objeto”, também chamados de positivos e negativos respectivamente. Este algoritmo é baseado no *AdaBoost* (Freund e Schapire, 1997), no qual seleciona um número de características visuais da imagem a fim de utilizá-las na detecção. Para treinamento deste algoritmo é necessário um amplo conjunto de amostras do objeto de interesse, sejam elas positivas e negativas.

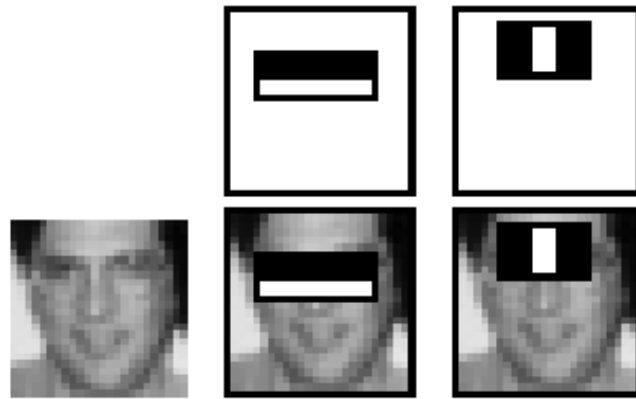


Figura 2.2: Exemplo de *Haar-like feature*. Fonte: (Viola e Jones, 2001).

Com isso é possível obter algumas características, denominadas de *Haar-like features*. Cada *feature* é utilizada como uma máscara convolucional bidimensional no processo de detecção do objeto, como ilustrado na Figura - 2.2. A primeira *feature* analisa que a cor dos olhos geralmente são mais escuras que a cor do nariz e bochechas. Já a segunda *feature* indica que os olhos são mais escuros que o nariz.

Aplicando essa máscara convolucional, cada *feature* oferece um valor único, denominado de coeficiente. Este valor é obtido a partir da diferença dos *pixels* da região branca com os que estão na região preta. Esse valor oferece características espaciais da imagem, podendo assim atuar como detectores locais. Cada *feature* em si é considerada um detector fraco, pois consegue distinguir somente uma parte do objeto de interesse, porém, quando utilizado em conjunto com outras *features* a detecção se torna mais robusta e precisa.

A estrutura desse algoritmo é apresentada em cascata, conforme ilustrado na Figura - 2.3. Em de uma imagem, a maioria das sub-janelas serão exemplos negativos (Viola e Jones, 2001), fazendo com que a região seja descartada e procura-se uma outra região

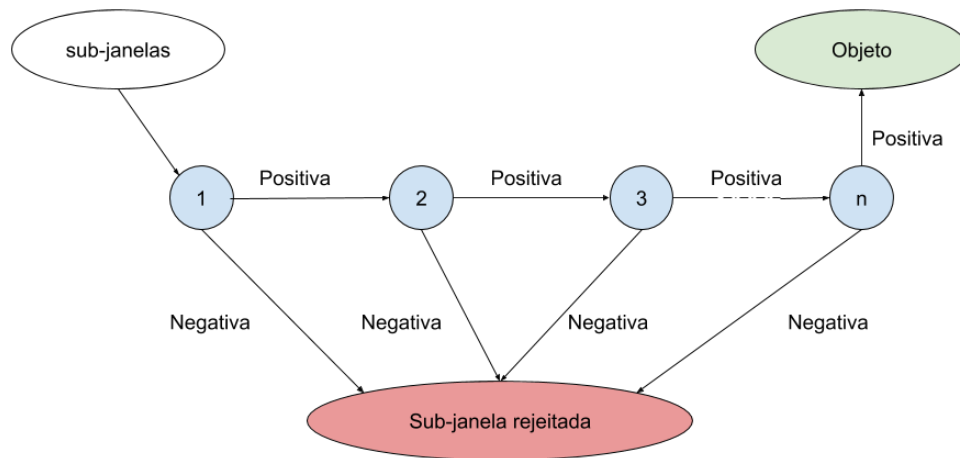


Figura 2.3: Esquema do *Haar Cascade*. Fonte: Adaptado de Viola e Jones (2001).

da imagem. Uma região é classificada como positivo, ou seja, apontada como objeto de interesse, somente quando passa por todos os níveis da cascata.

2.3 Extração de Características

Quando realizada uma técnica de classificação *handcrafted*, a etapa de extração de características é de suma importância para o desenvolvimento de um sistema classificação. Neste trabalho serão utilizadas algumas técnicas que geralmente aplicadas no contexto de classificação de objetos em imagens digitais.

2.3.1 LBP

Local Binary Pattern foi apresentado inicialmente por Ojala et al. (1996) como uma medida complementar para contraste. Ojala et al. (2002) adaptaram-o de forma que se tornasse uma abordagem estrutural para descrição de textura de imagens. Esse descritor tem como premissa que padrões binários locais na vizinhança de um pixel são propriedades fundamentais para a característica de textura.

Esse método apresenta a descrição da textura levando em consideração um pixel central C em relação aos seus P vizinhos equidistantes, considerando uma distância R , como observado na Figura - 2.4. O vetor de características do padrão LBP referente ao pixel C é calculado analisando os tons de cinza em relação a cada vizinho P de acordo com a equação 2.1.

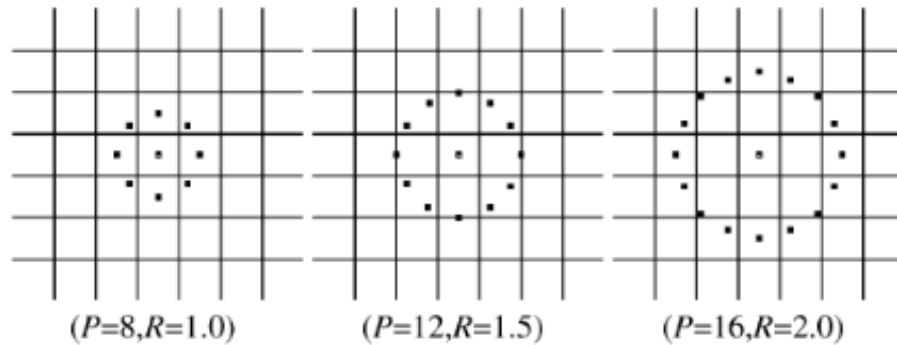


Figura 2.4: Operador LBP. Fonte: adaptada de Ojala et al. (2002).

$$LBP_c[i] = \begin{cases} 1 & \text{se } t_i - t_c \geq 0 \\ 0 & \text{se } t_i - t_c < 0 \end{cases} \quad (2.1)$$

na qual $i = [0, P]$ é o índice dos vizinhos de C e t_i a intensidade de cinza do pixel.

Sendo assim, cada vetor de características de um pixel C tem tamanho 2^P . Dependendo do valor de P esse vetor pode ser muito grande e de difícil utilização na etapa de classificação. Ojala et al. (2002) apresenta um conceito de uniformidade na sequência obtida no vetor de LBP. Um valor é considerado uniforme se o número de transições de 0 para 1, ou vice-versa, é menor ou igual a dois, considerando inclusive que o código é uma lista circular. Assim, o código 01101011 não é considerado uniforme pois contém seis transições. Já o padrão 00000010 é considerado uniforme pois apresenta duas transições, de 0 para 1 e de 1 para 0. Com isso é possível obter um vetor de menor dimensionalidade. Usando $P = 8$ e $R = 2$ sem considerar os padrões uniformes terá 256 vetores possíveis. Utilizando o conceito de uniformidade, o vetor de características terá tamanho 59, pois são 58 padrões uniformes e mais uma posição para todos os demais padrões não uniforme.

2.3.2 Histograma de Cores

Histograma de cores representa a distribuição de cores em uma determinada imagem, fornecendo a quantidade de vezes que cada cor se repete. Destaca-se que o histograma de cores não fornece informações espaciais da imagem, ou seja, ele apresenta a frequência de uma determinada cor, mas não informa onde ela está concentrada.

O histograma é aplicado normalmente em uma única banda de cor como no caso de imagens tons de cinza. Caso a imagem esteja no modelo RGB, por exemplo, tem-se duas abordagens: aplicar em uma matriz tridimensional ou separar as três bandas da imagem e fazer aplicação da técnica em cada uma. A Figura - 2.5 apresenta os histogramas de

cores em tons de cinza e também de uma imagem RGB dividida em três bandas da placa apresentada. Observa-se que a cor vermelha está presente em quase todas os valores possíveis (sendo eles de 0 a 255). Já em azul, tem-se uma concentração maior em uma única faixa de valor. Em tons de cinza, percebe-se que a imagem tem poucos valores escuros, sendo em sua maioria de tons médios com tons até os mais claros.

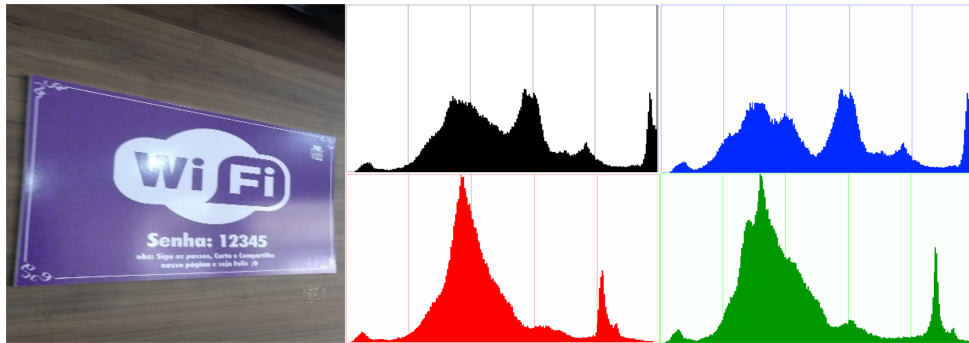


Figura 2.5: Exemplos de Histograma de cores de uma imagem. Fonte: o autor

Gonzalez e Woods (2006) afirmam que histograma é uma ferramenta útil em análise de imagem e que a distribuição de cores pode ser um descritor de característica válido.

2.3.3 DAISY

Esse método utiliza o gradiente de orientação com ponderação Gaussiana e um núcleo circularmente simétrico permitindo uma velocidade e eficiência em cálculos densos (Tola et al., 2010). Tola et al. (2010) afirmam ainda que esse descritor é muito rápido e eficiente para ser computado.

A definição formal de DAISY apresentada por Tola et al. (2010) pode ser apresentada da seguinte maneira. Para cada imagem de entrada, calcula-se H mapas de orientação $\mathbf{G}_i$ de modo que $1 \leq i \leq H$ um para cada direção quantizada. Na qual $\mathbf{G}_o(u, v)$ é igual a norma do gradiente no ponto (u, v) para a direção o se essa for maior que zero, caso contrário será igual a zero. Com isso, preserva-se a polaridade de mudanças de intensidade. Formalmente os mapas de orientação são descritos como $\mathbf{G}_o = \left(\frac{\partial \mathbf{I}}{\partial o}\right)^+$ na qual $\mathbf{I}$ é a imagem de entrada, o é a orientação da derivada e $(.)^+$ é um operador de maneira que $(a)^+ = \max(a, 0)$.

Para cada mapa de orientação são aplicadas várias convoluções com núcleos Gaussianos de Σ valores diferentes para obter os mapas de orientação convolucionados para cada região diferente do tipo $\mathbf{G}_o^\Sigma = G_\Sigma * \left(\frac{\partial \mathbf{I}}{\partial o}\right)^+$ com G_Σ sendo os núcleos Gaussianos. Diferentes Σ s são usados para controlar o tamanho das regiões.

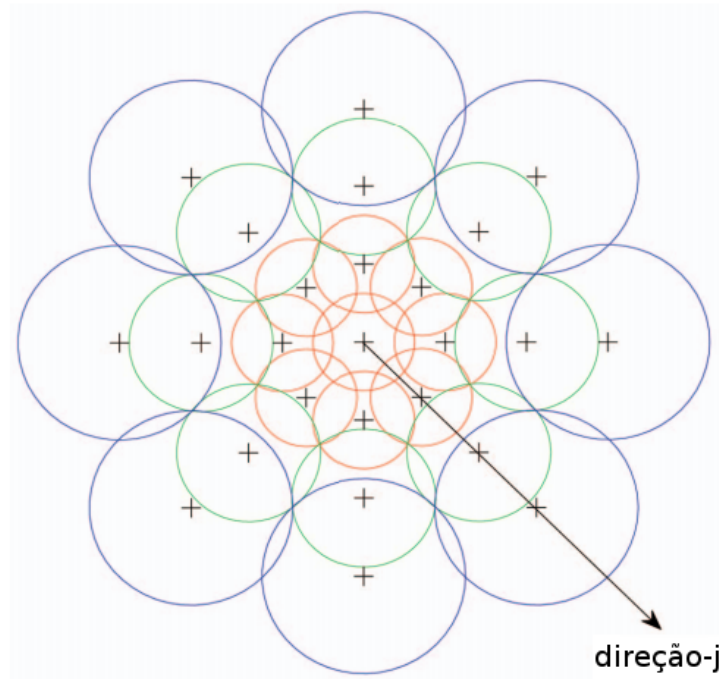


Figura 2.6: O descritor DAISY. Fonte: (Tola et al., 2010).

A Figura - 2.6 ilustra o descritor DAISY. Cada círculo representa uma região na qual o raio é proporcional aos desvios padrões dos núcleos Gaussianos e o sinal + representa os locais onde é mostrado o centro dos mapas de orientação, sendo o pixel central que será calculado o descritor.

2.3.4 HOG

O HOG é uma técnica que utiliza a orientação dos gradientes de uma imagem como *features* (Dalal e Triggs, 2005). O princípio dessa técnica consiste em analisar a distribuição dos gradientes de intensidade dos *pixels* ou direção das bordas para descrever a forma e aparência de um objeto (Gritti et al., 2008). O processo de obter cada *feature* pode ser dividido em quatro etapas, sendo elas: cálculo do gradiente de cada pixel, a disposição de cada pixel em células, o agrupamento de células em blocos e, por fim, a obtenção da *feature* (Cruz, 2014).

A primeira etapa pode ser obtida utilizando uma máscara unidimensional de derivada discreta pontual nos eixos horizontal e vertical (Dalal e Triggs, 2005). A Figura - 2.7 ilustra as máscaras, sendo (a) utilizada para os gradientes horizontais e (b) para os gradientes verticais.

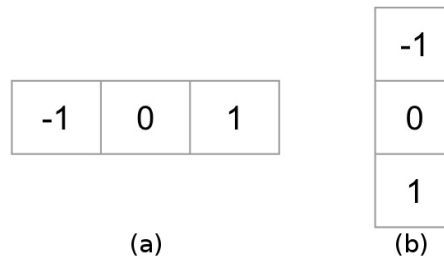


Figura 2.7: *Kernels* de detecção de gradiente. (a) para gradientes horizontais e (b) para gradientes verticais. Fonte: o autor.

Em seguida, o algoritmo agrupa os *pixels* de uma determinada região, criando uma célula na qual são computados os valores de magnitude da mesma tendo como base os valores dos ângulos dos vetores. A equação 2.2 apresenta o cálculo do ângulo (θ).

$$\begin{aligned} g &= \sqrt{g_x^2 + g_y^2} \\ \theta &= \arctan \frac{g_x}{g_y} \end{aligned} \quad (2.2)$$

Na terceira etapa são criados blocos com o uso de agrupamento de células de uma região. Existem áreas de bloco que propositalmente se sobrepõem em relação aos seus vizinhos, isso torna o método mais eficiente em relação a uma abordagem sem sobreposições (Dalal e Triggs, 2005). E, por fim, obtém-se o descritor (*feature*). Esse descritor é uma lista dos histogramas de todas as células de todos os blocos.

2.4 Classificadores

Essa seção apresenta os algoritmos utilizados nesse trabalho no processo de classificação de um objeto. Sendo eles: SVM, k-NN e MLP.

2.4.1 Support Vector Machine (SVM)

SVM é um modelo de aprendizagem supervisionado que visa encontrar um hiperplano n -dimensional que maximiza a distância entre os elementos de classes distintas (Vapnik, 1995). Por definição, SVM foi apresentado como um classificador linear binário, mas existem tratamentos que permitem a classificação de objetos que não são linearmente separáveis e/ou com mais de duas classes (Ni et al., 2014). A Figura - 2.8 ilustra o processo de obtenção do hiperplano de separação dos dados.

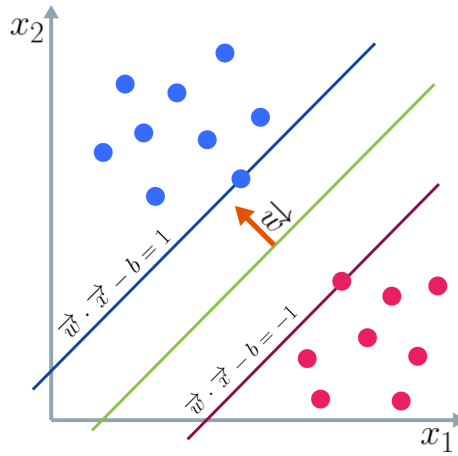


Figura 2.8: Exemplos de utilização do SVM. Fonte: o autor.

SVM pode ser formalmente definido da seguinte forma: dado um conjunto com n elementos da forma $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_n, y_n)$ onde $y \in \{-1, 1\}$ representa as duas classes e $\vec{x}_i \in \mathbb{R}^p$ representa o vetor de características. O objetivo é encontrar um hiperplano n dimensional que maximize a distância entre as duas classes. Sabe-se que todo hiperplano pode ser expresso como um conjunto de pontos $\vec{x}$ satisfazendo a equação:

$$\vec{w} \cdot \vec{x} - b = 0 \quad (2.3)$$

no qual $\vec{w} \in \mathbb{R}^p$ é o vetor normal ao hiperplano e $b \in \mathbb{R}$ é o parâmetro de distância em relação à origem; esses valores são definidos durante a etapa de treinamento do classificador. A predição de um novo exemplo é dada pela seguinte equação:

$$y_i = \vec{w}_i \cdot \vec{x}_i + b \quad (2.4)$$

2.4.2 k-Nearest Neighbors (K-NN)

k-NN é um modelo de aprendizagem supervisionado que busca encontrar k vizinhos mais próximos no processo de classificação (Mitchell, 1997). Esse classificador tem como principal vantagem o fato de que não é necessário realizar uma etapa de treinamento. Porém, a classificação de um novo elemento exige o cálculo de distância de todos os demais, ou seja, quanto maior a base de dados mais demorado o processo de classificação. A Figura - 2.9 ilustra o processo de classificação de uma nova instância. Nota-se que para

a nova instância (representada pela cor verde) será atribuída como vermelho caso seja utilizada o parâmetro $k = 3$, já para a utilização de $k = 5$, a classe atribuída seria azul.

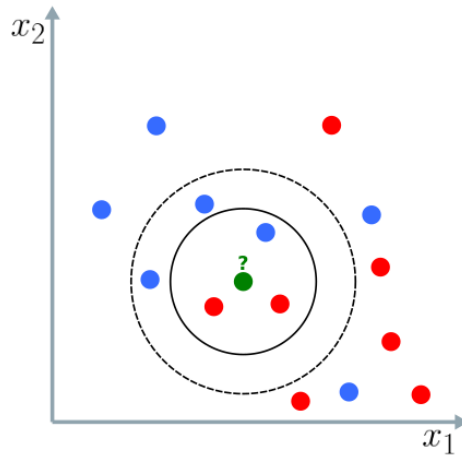


Figura 2.9: Exemplos de utilização do k-NN. Fonte: o autor.

Esse algoritmo assume que todas as instâncias do problema correspondem a pontos em um espaço n -dimensional ($\mathbb{R}^n$). A métrica mais comum para calcular a distância entre duas instâncias quaisquer é a distância Euclidiana. Mais precisamente, supondo x uma instância qualquer e o conjunto de características de x do tipo $(\vec{a}_1(x), \vec{a}_2(x), \dots, \vec{a}_n(x))$, onde $a_i(x)$ indica o valor do i -ésimo atributo da instância x . A distância de duas instâncias x_j e x_k é definida pela equação 2.5.

$$d(x_j, x_k) = \sqrt{\sum_{i=1}^n (a_i(x_j) - a_i(x_k))^2} \quad (2.5)$$

Sendo assim, a classificação de uma nova instância consiste em atribuir a ela a classe majoritária dos k vizinhos mais próximos a ela. A equação 2.6 expressa a maneira de atribuição de uma classe

$$x_i = c \text{ tal que } \arg \max_c k_c \quad (2.6)$$

no qual k_c indica o número de vizinhos próximos pertencentes à classe c dentre os k vizinhos mais próximos.

2.4.3 MLP

A *MultiLayer Perceptron* é uma rede neural artificial similar a *perceptron*, porém utiliza diversas camadas de neurônios ligadas entre si com diversos pesos diferentes. A necessidade de utilizar camadas de neurônios se dá devido ao fato do *perceptron* simples só conseguir separar linearmente duas classes (Mitchell, 1997).

Basicamente uma MLP é composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. A Figura - 2.10 ilustra uma MLP padrão.

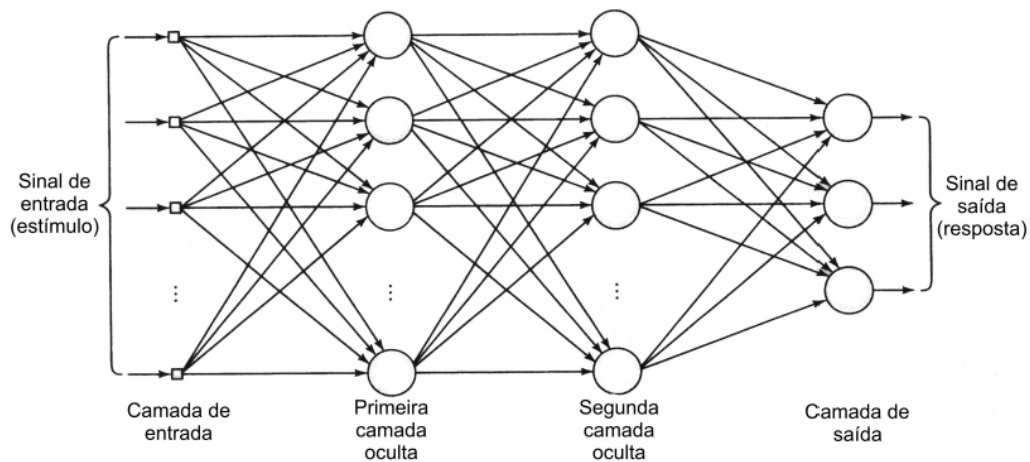


Figura 2.10: Exemplo de uma MLP. Fonte: (Haykin, 2001).

Haykin (2001) afirma que os neurônios que realizam processamento computacional estão nas camadas ocultas e na camada de saída. A propagação dos dados da camada de entrada são transmitidas até a primeira camada oculta. Cada neurônio efetua o cálculo da função de ativação. A propagação dos dados vão seguindo por todas as camadas ocultas até chegar na camada de saída. Na camada de saída é apresentado o retorno do processamento da rede e calculada a estimativa do vetor gradiente, necessária para o algoritmo de *backpropagation* efetuar ajuste nos pesos das conexões.

Uma MLP tem três características distintas (Haykin, 2001), sendo elas:

- função de ativação não linear e suave, ao contrário da utilizada no *perceptron*. Uma função normalmente utilizada é a sigmoide, apresentada na equação 2.7.

$$y_j = \frac{1}{1 + e^{-v_j}} \quad (2.7)$$

no qual v_j é o campo local induzido do neurônio j e y_j é a saída do neurônio. A utilização da função logística tem motivação biológica.

- uma ou mais camadas ocultas, capacitando a rede a aprender tarefas complexas extraindo progressivamente as características mais significativas dos padrões de entrada;
- alto grau de conectividade.

2.5 Redes Neurais Convolucionais

Redes Neurais Convolucionais (CNN, do inglês *Convolutional Neural Networks*) são um tipo de rede neural que aplica uma convolução matemática em vez de multiplicação de matrizes (Lecun, 1989). Devido às suas características de conexões esparsas, compartilhamento de parâmetros e representações equivariantes, a CNN aproveita vantagens ao trabalhar com dados relacionados espacialmente, como imagens digitais (Goodfellow et al., 2016).

A unidade básica de uma Rede Convolucional é a camada convolucional, na qual a entrada é convolucionada com os núcleos em algum passo e gera um mapa de características. Esse mapa de recursos também pode passar por uma função de ativação e ser subamostrado em etapas de *pooling*. Normalmente, em uma única rede convolucional, existem muitas camadas convolucionais conectadas em série ou paralelas intercaladas com camadas de *pooling* até a saída da rede. A Figura - 2.11 ilustra esse processo de intercalação de camadas.

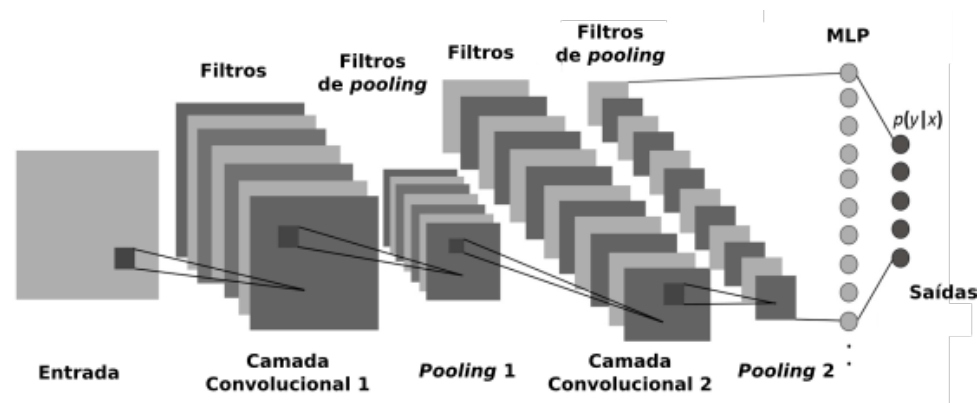


Figura 2.11: Exemplo de intercalação de camadas em CNN. Fonte: (Albelwi e Mahmood, 2017).

As CNNs são comumente associadas a redes neurais padrão totalmente conectadas, como os *multilayers perceptrons* (MLPs) (Mitchell, 1997). Nesses casos, uma rede convolucional geralmente recebe uma imagem digital "bruta" e gera um mapa de recursos.

Em seguida, a MLP usa esse mapa de recursos como entrada e executa a classificação (Lecun, 1989). A principal vantagem do uso de CNNs é seu propósito como um mecanismo treinável de redução de dimensionalidade. Assim, ele pode ajustar seus parâmetros para criar um mapa de características mais generalizável da entrada que recebe, diminuindo a necessidade de extração manual de recursos. Com isso há uma diminuição da necessidade de intervenção de especialistas no domínio de cada aplicação (Goodfellow et al., 2016). Desde 2012, CNNs vêm sendo utilizadas em alguma etapa nos principais trabalhos de destaque em desafios de visão computacionais, isso graças a popularização das GPUs, que são capazes de realizar alguns cálculos em paralelo de maneira eficiente.

A convolução é uma operação linear a partir de duas funções, representada por $f * g$, na qual $f : \mathbb{R} \rightarrow \mathbb{R}$ e $g : \mathbb{R} \rightarrow \mathbb{R}$. É formalmente definida como a integral do produto da função f por uma cópia invertida e deslocada de g , conforme descrito na equação 2.8.

$$f * g(t) = \int_{-\infty}^{\infty} f(a)g(t - a)da \quad (2.8)$$

Nas CNNs, a função f é denominada de entrada e a função g é chamada de *kernel* ou então *filtro*. Como nos métodos de aprendizado de máquina a entrada é geralmente uma matriz multidimensional, o *kernel* passa a ser também uma matriz que é ajustada pelo algoritmo de otimização, dessa forma a convolução de uma entrada $I \in \mathbb{R}^{p \times q}$ e com *kernel* $K \in \mathbb{R}^{r \times s}$ é definida conforme a equação 2.9.

$$(I * K)_{i,j} = \sum_{n=1}^p \sum_{m=1}^q I_{m,n} K_{i-m,j-n} \quad (2.9)$$

Como a convolução é comutativa e o *kernel* geralmente é muito menor que a entrada, utiliza-se frequentemente a equação 2.10.

$$(I * K)_{i,j} = (K * I)_{i,j} = \sum_{n=1}^r \sum_{m=1}^s I_{i-m,j-n} K_{m,n} \quad (2.10)$$

As camadas convolucionais aplicam diversas convoluções com diferentes *kernels* na entrada. O número de *kernels* é um hiperparâmetro da rede, assim como o tamanho de cada *kernel*; o deslocamento, também conhecido como *stride*; a política de tratamento de borda da entrada; e a função de ativação dos neurônios.

Cada neurônio da camada convolucional está relacionado a apenas uma região da entrada, essa é a principal diferença entre utilizar camadas convolucionais do que simplesmente uma rede totalmente conectada (MLP), pois reduz a utilização de memória e tem uma eficiência estatística melhor (Goodfellow et al., 2016). A Figura - 2.12 ilustra

as conexões dos neurônios em camadas convolucionais e faz uma comparação com uma MLP padrão.

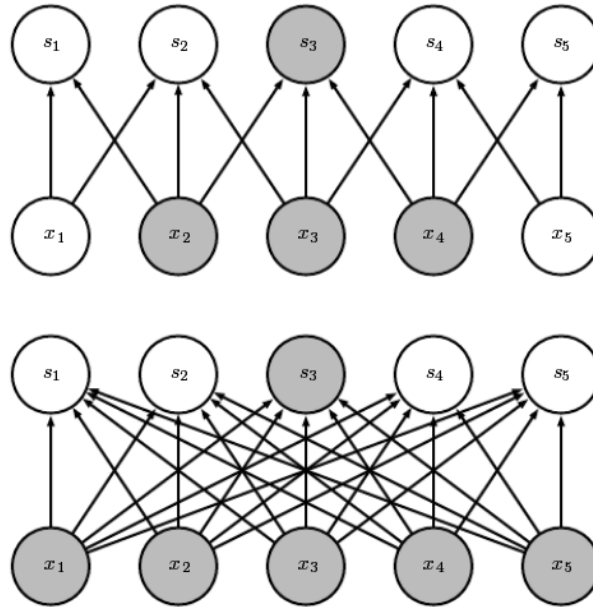


Figura 2.12: Ligação dos neurônios na camada convolucional (superior) e ligação dos neurônios em uma MLP (inferior) . Fonte: (Goodfellow et al., 2016)

A Figura - 2.13 ilustra a aplicação de cinco *kernels*. A entrada é uma imagem em RGB de dimensão $32 \times 32 \times 3$, aplicando cada *kernel*, tem-se uma nova dimensão na matriz resultante, que possuirá dimensões $32 \times 32 \times 5$, no qual cada camada possui diferentes padrões encontrados na imagem.

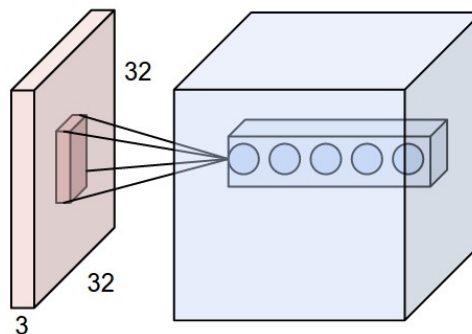


Figura 2.13: Exemplo de aplicação de convolução de cinco *kernels* em uma imagem RGB. Fonte: (Li et al., 2016)

Uma camada de *pooling* resume uma área de vizinhança, diminuindo a dimensão do sinal propagado no processo de aprendizado. Formalmente, seja (x, y) uma região da

imagem, a aplicação de *pooling* visa definir um único valor que represente todos os valores da região (x, y) .

Existem hiperparâmetros que podem ser aplicados às camadas de convolução, tais como: o tamanho da região de aplicação do *pooling*; o deslocamento (também conhecido como *stride*) e o tratamento de borda. Além disso, existem diversas operações como o *maxpooling* (Zhou e Chellappa, 1988), que toma como valor de saída o maior dentre os existentes na região aplicada; média dos valores da região e média ponderada dos valores da região, baseados na distância do centro.

A Figura - 2.14 ilustra a aplicação do *maxpooling* de região 2×2 em uma imagem de dimensão 8×8 , e com deslocamento de 2×2 . Com isso, a imagem resultante fica com dimensão 2×2 e com os maiores valores de cada região.

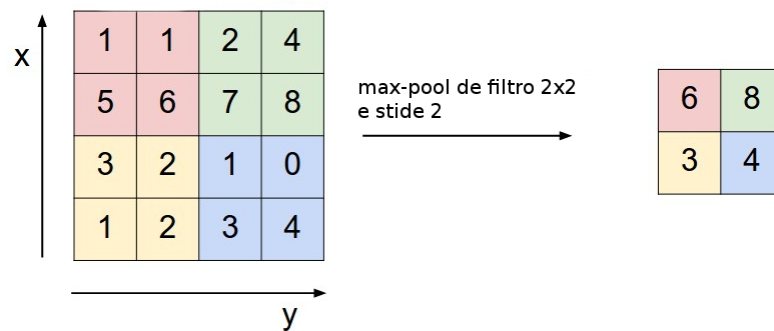


Figura 2.14: Exemplo de utilização do *maxpooling*. Fonte: (Li et al., 2016)

Existem algumas funções na literatura que podem ser utilizadas como função de ativação em CNN. Neste trabalho utilizou-se duas: *Leaky ReLU* (LReLU), que é uma versão melhorada da ReLU e a *Softmax*.

A escolha da LReLU em vez da ReLU, como demonstrado por Maas et al. (2013), em alguns casos a ativação de ReLU poderia matar alguns neurônios e não pode mais ser ativada. Uma ativação LReLU $\iota : \mathbb{R} \rightarrow \mathbb{R}$ é definida como $\iota(x) = \max(x, \alpha x)$, em que $\alpha \in \{x \in \mathbb{R} \mid 0 < x < 1\}$.

A função *Softmax* é um tipo de função sigmoide que pode ser utilizada quando o processo de classificação envolve mais de duas classes (Goodfellow et al., 2016). Ela transforma a saída de cada classe em valores entre 0 e 1, cuja a soma de todas os valores resulta em 1. A equação 2.11 apresenta o processo de cálculo. Essa função é usada normalmente na saída de um classificador, pois gera as probabilidades de cada classe.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (2.11)$$

O treinamento de uma CNN é similar ao treino de uma MLP padrão e utiliza o algoritmo *backpropagation*. Basicamente o processo de treinamento consiste em otimizar uma regressão logística, uma vez que todas as classes são exclusivas. O erro do modelo foi definido como a entropia cruzada da classe predita e a classe da amostra. A entropia cruzada é definida na equação 2.12

$$H(\hat{\mathbf{y}}, \mathbf{y}) = -\mathbf{y} \cdot \log(\hat{\mathbf{y}}) \quad (2.12)$$

na qual $\mathbf{y}$ é a classe da amostra, $\hat{\mathbf{y}}$ é a classe predita e $\cdot$ é o produto escalar.

Um problema comum em CNNs é o *overfitting*. Esse problema acontece quando um modelo de aprendizagem de máquina modela os dados de treinamento muito bem, aprendendo tanto os detalhes quanto os ruídos nos dados de treinamento afetando negativamente o desempenho em novos dados. Para evitar esse problema é necessário uma base extensa de dados ou então aplicar técnicas como *Data Augmentation* ou *Dropout* que serão explicadas a seguir.

2.5.1 Data Augmentation

Data Augmentation consiste em adicionar imagens em uma base de dados, fazendo alterações nas imagens originais, a fim de criar novos exemplos. Lecun et al. (1998) afirma que é de suma importância que seja utilizado como treinamento de uma CNN uma base de dados extensa, pois quanto maior o conjunto de treino, menores são os erros no conjunto de testes. Porém nem sempre existe uma base de dados extensa para aplicação de uma CNN.

Para isso se faz necessário aplicar diferentes transformações nas cores da imagem como aumento de brilho, contraste e saturação, além de transformações no espaço como translação, rotação, nitidez e desfoque. A Figura - 2.15 ilustra algumas dessas transformações, na qual (a) é a imagem original, (b) aumento de brilho, (c) rotação de 15° e (d) desfoque.

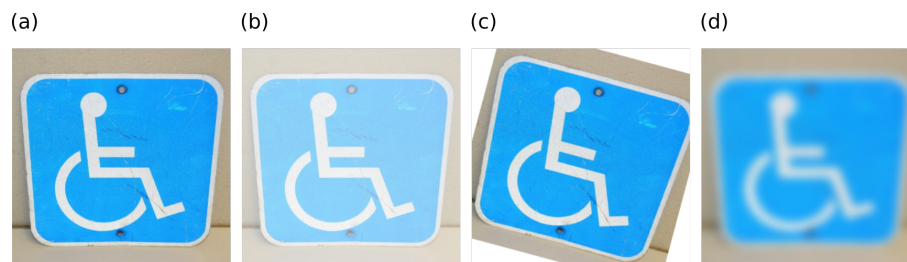


Figura 2.15: Exemplo de transformações de *Data Augmentation*. Fonte: o autor.

2.5.2 Dropout

A técnica de *Dropout* introduzida por Hinton et al. (2012) visa inserir uma probabilidade de desativar um neurônio de uma camada da rede durante o processo de treinamento resultando em uma arquitetura dinâmica da rede, visto que em etapas distintas do treinamento diferentes neurônios estão ativados. O efeito é que a rede se torna menos sensível aos pesos específicos dos neurônios resultando em uma rede capaz de generalizar melhor e ter menor probabilidade de sobrecarga dos dados de treinamento.

Considerando uma rede neural com L camadas ocultas e $l \in \{1, 2, \dots, L\}$ o índice das camadas ocultas. Tem-se também $\mathbf{z}^{(l)}$, $\mathbf{y}^{(l)}$, W^l e $\mathbf{b}^{(l)}$ representando as entradas, saídas, pesos e *bias* da camada l na qual $\mathbf{y}^{(0)} = \mathbf{x}$. A operação *feedforward* do algoritmo de *backpropagation* de uma rede neural padrão pode ser descrita como:

$$\begin{aligned} z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \mathbf{y}^{(l)} + b_i^{(l+1)}, \\ y_i^{(l+1)} &= f(z_i^{(l+1)}) \end{aligned} \quad (2.13)$$

para $l \in \{0, 1, \dots, L-1\}$ e para qualquer unidade oculta i em que f é uma função de ativação. Com a operação de *dropout*, a operação *feedforward* torna-se:

$$\begin{aligned} \mathbf{r}_j^{(l)} &\sim \text{Bernoulli}(p) \\ \tilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} \circ \mathbf{y}^{(l)} \\ z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \tilde{\mathbf{y}}^{(l)} + b_i^{(l+1)} \\ y_i^{(l+1)} &= f(z_i^{(l+1)}) \end{aligned} \quad (2.14)$$

na qual " $\circ$ " denota o produto de entrada dos operandos (Davis, 1962). Para qualquer camada l , $\mathbf{r}^{(l)}$ é um vetor de variáveis independentes aleatórias de *Bernoulli*, na qual cada uma tem uma probabilidade p de ser 1. Esse vetor é amostrado e multiplicado no sentido da saída dessa camada, $\mathbf{y}^{(l)}$, para criar uma saída $\tilde{\mathbf{y}}^{(l)}$ para a próxima camada utilizar de entrada.

Seu conceito é facilmente implementado, basta definir uma probabilidade de cada nó ser desligada em cada ciclo de atualização de peso. É importante ressaltar que essa técnica é utilizada apenas durante o treinamento de um modelo.

2.5.3 Validação Cruzada

A técnica de validação cruzada, ou *cross-validation* em inglês, apresentada por Kohavi (1995) consiste na divisão aleatória da base de dados em n partes de tamanhos iguais, denominada de *folds*. Em cada etapa do processo de treinamento, $n - 1$ *folds* são alocadas para treinamento e 1 para o teste do modelo. Esse processo é repetido por n vezes. Com isso, tem-se n resultados de performance que, usualmente, é feito um cálculo de média e desvio padrão. Essa técnica é utilizada a fim de aumentar a precisão das estimativas de performance (Kuhn e Johnson, 2013).

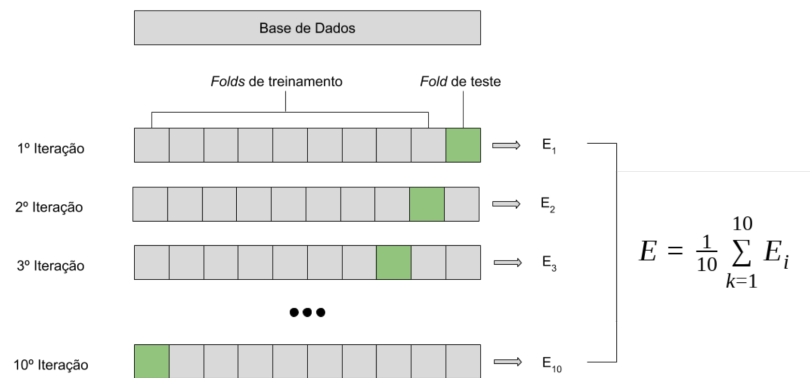


Figura 2.16: Processo de validação cruzada. Fonte: Adaptada de Raschka (2015)

A Figura - 2.16 ilustra o processo de validação cruzada com $n = 10$. Em cada iteração é separada uma *fold* para teste e outras 9 para treino. Em cada etapa é realizada uma avaliação (E_i). Por fim, faz-se a média simples de todas as iterações para se ter o resultado final da avaliação do classificador (E).

2.6 Métricas de Avaliação

Essa seção apresenta as métricas de avaliação comumente utilizados em sistemas de classificação, sendo essas: acurácia (A), *precision* (P), *recall* (R), *F-measure* e *Macro-F* (Manning et al., 2008). Para tais explicação, serão utilizados as seguintes abreviações:

- VP: verdadeiros positivos. Quantidade de instâncias positivas classificadas corretamente;
- VN: verdadeiros negativos. Quantidade de instâncias negativas classificadas corretamente;

- FP: falsos positivos. Quantidade de instâncias negativas classificadas como positivas;
- FN: falsos negativos. Quantidade de instâncias positivas classificadas como negativas;

2.6.1 Acurácia

Acurácia é a métrica mais frequentemente utilizada em sistemas de classificação. Definida pela equação 2.15:

$$A = \frac{VP + VN}{VP + VN + FP + FN} \quad (2.15)$$

ou seja, essa métrica é a soma das instâncias classificadas corretamente dividida pela soma de todas as instâncias;

Essa medida não representa a realidade quando o experimento não possui uma quantidade balanceada de elementos. Supomos um experimento que tivesse 100 fotos, sendo 95 de placa de banheiro masculino e 5 de banheiro feminino. Se um classificador inferisse que todas são placas de banheiro masculino, a acurácia desse classificador seria de 95%. Para isso, é importante conciliar essa métrica com as demais utilizadas nesse trabalho.

2.6.2 Precision

A métrica denominada *precision* avalia a quantidade de elementos classificados como positivo e que são de fatos positivos, expresso na equação 2.16.

$$P = \frac{VP}{VP + FP} \quad (2.16)$$

Para elevar a taxa de *precision* o classificador deve inferir que uma determinada instância seja verdadeira somente nos casos corretos. A desvantagem dessa métrica é que ela não leva em conta a quantidade de casos que deveriam ter sido classificados. Supomos um experimento que tivesse 10 fotos de saída de emergência, dessas o sistema classificou 3 como verdadeiras e nenhuma das fotos das demais classes foram classificadas como dessa classe. A taxa de *precision* desse classificador é alta, pois das 3 classificadas como saída de emergência, todas estão corretas.

2.6.3 Recall

A métrica *Recall* avalia dentre todas as situações classificadas como positivo, quantas estão de fato corretas. Pode ser expressa pela equação 2.17:

$$R = \frac{VP}{VP + FN} \quad (2.17)$$

A desvantagem dessa métrica é que ela não leva em consideração todas as medidas e se uma determinada classe não tivesse nenhuma instância classificada, ela teria um valor de *Recall* máximo.

2.6.4 F-measure

A métrica de *F-measure* é a média harmônica das medidas de *precision* e *recall*. Assim, é possível expressar as duas medidas em um único valor. Sua fórmula é expressa na equação 2.18.

$$F - measure = \frac{2 \times R \times P}{R + P} \quad (2.18)$$

no qual R representa *Recall* e P *Precision*.

Com essa medida é possível avaliar a eficiência do sistema, pois é preciso que o sistema acerte bem todas as classes em uma boa cobertura.

2.6.5 Macro-F

É a média aritmética das *F-measures* de todas as classes do conjunto. A equação 2.19 expressa como pode ser obtido seu valor.

$$Macro - F = \frac{1}{k} \sum_{i=1}^k F - measure(C_i) \quad (2.19)$$

onde k é a quantidade de classes e C_i é o valor de *F-measure* da classe.

Por se tratar de uma média das *F-measures*, um valor alto indica que o classificador está tendo boa eficiência em todas as classes existente no conjunto de dados.

2.7 Raspberry Pi

O Raspberry Pi é uma plataforma multitarefa de baixo custo que possibilita o processamento de imagem devido ao tamanho reduzido e baixo custo de consumo, na ordem



Figura 2.17: Raspberry Pi 2 Model B. Fonte: (RASPBERRY PI FOUNDATION, 2019).

de 700mA a 1000mA (Pereira, 2015). Existem diversos modelos de placas de Raspberry Pi disponíveis no mercado, a diferença entre eles são os recursos e preços disponíveis. A Figura - 2.17 ilustra o plataforma Raspberry Pi 2 Modelo B. As especificações desse modelo estão descritas na Tabela - 2.1.

Tabela 2.1: Especificações do Raspberry Pi 2 B

Arquitetura	ARM
SoC	Broadcom BCM2836
CPU	Quad core Cortex A-7@900MH
GPU	Broadcom VideoCore IV
RAM	1GB SDRAM
USB	4 portas USB 2.0
Vídeo	HDMI
Áudio	AV via 3.5mm jack
Armazenamento	Micro SDHC
Comunicação	10/100 Mbps Ethernet
Alimentação	5V via MicroUSB
Consumo	800 mA

A vantagem de utilizar o Raspberry Pi é que o mesmo possui capacidade de acoplar uma câmera desenvolvida pela própria plataforma, não tendo a necessidade de conectar uma *webcam* de outro fabricante. Com isso é possível ligar a câmera pelo conector CSI (*Câmera Serial Interface*), especialmente dedicado para este tipo de conexão e troca de dados. A Figura - 2.18 ilustra a câmera modelo v2 de 8 *megapixels*, utilizada nesse trabalho, e sua conexão.



Figura 2.18: Raspberry Pi Câmera v2. Fonte: (RASPBerry PI FOUNDATION, 2019).

Segundo Halfacree e Upton (2012) o Raspberry possui como sistema de distribuição oficial o Raspbian, um sistema baseado no Linux. Com isso, é possível utilizar qualquer linguagem de programação que possua suporte ao Linux e a arquitetura ARM, tais como Python e C/C++ que foram utilizadas nesse trabalho.

2.8 Trabalhos Relacionados

Na literatura, existem diversos trabalhos que aplicam o processo de reconhecimento de determinado objeto, porém poucos são aplicados ao contexto de placas de sinalização interna.

Dentre os que abordam esse tópico destaca-se o trabalho de Ni et al. (2014). Esse trabalho monta uma base de dados com imagens artificiais de placas de sinalização interna, obtidas no Google Imagens¹. A base conta com mais de mil imagens de 21 classes diferentes. Cada classe contém de 40 a 80 imagens, com sua maioria em torno de 50 imagens. Além disso, os autores apresentam como técnicas de extração de características: PCA (*Principal Component Analysis* (Jolliffe, 2005), Histograma de Orientação de Gradiente (HOG) (Dalal e Triggs, 2005) e DSIFT (Lowe, 2004) aplicados a três algoritmos classificadores, sendo eles: Redes Neurais (Mitchell, 1997), SVM (Vapnik, 1995) e k-NN (Mitchell, 1997). Em seus experimentos, os autores obtiveram um percentual de acerto de até 80,5% utilizando DSIFT e SVM. Os autores destacam a dificuldade de

¹<https://www.images.google.com>

construção da base de dados e afirmam que os estudos referentes a essa área estão em um estágio inicial. Outro ponto destacado pelos autores é a falta de padronização dessas placas: é afirmado que existem mais de 100 tipos de representação de placas de banheiros.

Wang et al. (2013) apresentam um *framework* para detecção de portas e placas de sinalização. A detecção de portas utiliza um modelo geométrico que torna a classificação mais precisa, independentemente do modelo da porta. Já a classificação de placas é feita por mapa de saliência e utilização de comparação com grafos bipartidos. O *framework* conta com uma câmera acoplada a um óculos, que ao detectar uma placa de sinalização ou porta envia uma descrição auditiva via fones de ouvido ao usuário. A base de dados utilizada para a avaliação do modelo conta com 146 imagens. Utilizando placas de localização de banheiros e abertura e fechamento de elevadores, os autores obtiveram precisão de 86%, quando incluídas placas de direção (cima, direita, baixo e esquerda), essa taxa caiu para 81,6%. Os autores consideram os resultados apresentados como satisfatórios e propõem a classificação de mais classes de placas, além de uma análise da interface do protótipo para a realização de testes com usuários.

Kunene et al. (2016) apresentam um sistema que realiza o reconhecimento de placas de sinalização interna colocadas sobre um fundo claro em tempo real. Sua proposta utiliza quatro passos para o reconhecimento de placas, sendo eles: (1) Segmentação da placa pela cor de fundo e forma; (2) pré-processamento da imagem para remoção do fundo; (3) extração da região de interesse para a detecção da textura, especificando qual a classe da placa por pesquisa em árvores agrupadas (do inglês *Three-Search Structure*); (4) comunicação da classe da placa via alto-falante. Os testes realizados em base de dados própria envolvem sete vídeos com onze placas de sinalização no total, além de uma avaliação com voluntários. O percentual médio de acurácia no reconhecimento de 67.14%, sendo a maior delas 86% em um dos vídeos. Apesar da taxa de acerto geral ser um pouco menor, esse trabalho utilizou a detecção em vídeos, que torna o processo mais difícil. Em relação aos testes com voluntários, dos 10 que utilizaram o dispositivo, 7 classificaram o sistema como apto para informação de placas e avaliaram, em média, o sistema com uma nota 3.9 (escala de 1-5).

Por fim, Bashiri et al. (2018) apresentam a base de dados denominada MCIndoor20000, um conjunto de imagens rotuladas para apoiar o desenvolvimento de detecção de placas de sinalização de hospitais e instituições de saúde. O conjunto de dados é composto por 2055 imagens digitais de três diferentes categorias de objetos internos, sendo elas: 754 imagens de portas, 599 de indicação de escadas e 702 sinais hospitalares (com informações específicas de contexto hospitalar, como clínicas, farmácia e centro de cirurgia). Um *data augmentation* foi utilizado no subconjunto de treinamento, resultando em mais de 20.000

amostras. Os autores utilizaram o modelo pré-treinado de rede neural convolucional (CNN) AlexNet (Krizhevsky et al., 2012) para estimar a qualidade e a quantidade de atributos do conjunto de dados MCIndoor20000. Os resultados de precisão foram 90,4% e 99,8% para o conjunto de dados original e o conjunto de dados aumentado, respectivamente.

De todos os trabalhos apresentados nessa seção, somente no de Bashiri et al. (2018) foi disponibilizado a base de dados, porém as técnicas utilizadas não foram especificadas para poder replicar os resultados com o modelo utilizado nesse trabalho. A Tabela - 2.2 apresenta uma comparação dos trabalhos relacionados.

Tabela 2.2: Comparação dos trabalhos relacionados

Trabalho	Número de classes	Características	Classificadores	Pública	Reconhecimento (%)
Ni et al. (2014)	21	PCA, HOG, e DSIFT	MLP, k-NN, e SVM	Não	80,5%
Wang et al. (2013)	4 7	Mapa de saliência	Comparação de grafos bipartidos	Não	86% 81,6%
Kunene et al. (2016)	4	SURF	Three-Search Structure	Não	67,14%
Bashiri et al. (2018)	3 3	CNN com <i>Transfer Learning</i>		Sim	90,4% ¹ 99,8% ²

¹ Conjunto original ² Conjunto Aumentado

Materiais e Métodos

Este Tópico trata do modelo proposto para a classificação de placas de sinalização internas para deficientes visuais e a elaboração de um protótipo que valide o modelo com a finalidade de utilizar como um dispositivo vestível. A Figura - 3.1 apresenta o conceito geral do treinamento do método proposto. A Base de dados foi dividida em 10 *folds* (F) para a realização da técnica de validação cruzada (do inglês *cross-validation*). Em seguida foram abordados os dois métodos: *handcrafted* e *non-handcrafted*. As duas instâncias de classificação foram salvas. Utiliza-se no modelo a que apresentou um melhor resultado nos testes experimentais.

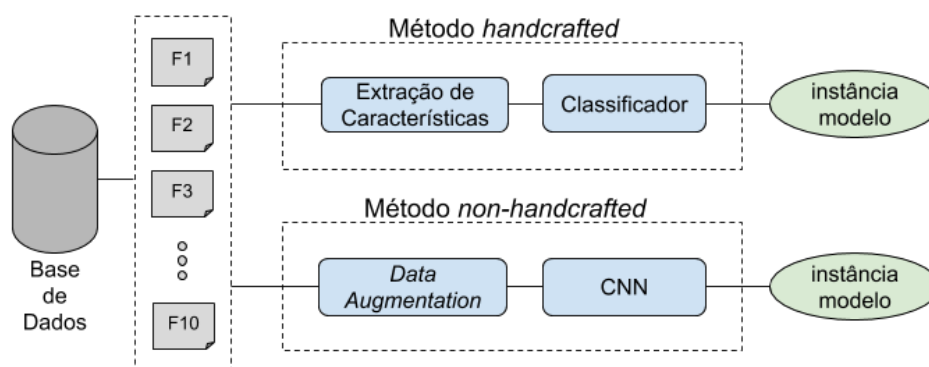


Figura 3.1: Etapa de treinamento. Fonte: o autor.

A Figura - 3.2 exibe a visão geral do processo de classificação após o treinamento. O algoritmo de detecção de placas busca encontrar uma placa em uma imagem de entrada. Caso seja encontrada uma região de placa, essa mesma é segmentada e então fornecida a

instância do modelo salva na etapa de treinamento. Esse modelo fará a classificação da placa.

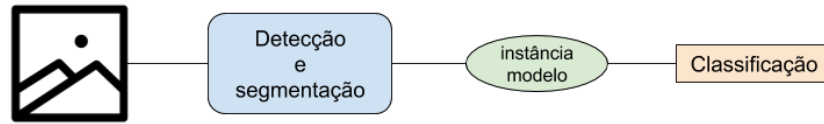


Figura 3.2: Modelo proposto: etapa de classificação. Fonte: o autor.

Serão apresentados os seguintes itens: base de dados criada para a avaliação do modelo, os conceitos e parâmetros do sistema de classificação, utilizando a abordagem *handcrafted* e *non-handcrafted*, e, por fim, detalhes do protótipo que valide o modelo.

Serão apresentados aqui conceitos teóricos do modelo proposto para a classificação do objeto de interesse e análise em uma plataforma. Porém, vale ressaltar que o modelo pode ser utilizado em diferentes *hardwares*, sendo possível ser implantado em outra plataforma e que qualquer deficiente visual consiga utilizá-lo em componentes que ele possua, desde que sejam compatíveis com a plataforma aqui apresentada.

3.1 Base de Dados

Conforme apresentado na seção 2.8, somente o trabalho de Bashiri et al. (2018) disponibilizou a base de dados, porém a mesma é voltada para um conceito de placas de sinalização encontradas em um hospital e somente com três classes, sendo diferente das classes propostas neste trabalho. Com isso, fez-se necessário a criação de uma base de dados que fosse disponibilizada para pesquisas futuras no contexto de classificação de placas de sinalização interna.

Foi então publicada a base intitulada *Indoor Sign Dataset (ISD)*¹ (Almeida. et al., 2019). A base de dados é composta por 1200 amostras distribuídas em 6 classes: banheiro masculino, banheiro feminino, acessibilidade, saída de emergência, proibido fumar e WiFi. A escolha dessas classes se deu por causa da alta disponibilidade em ambientes públicos. A Tabela - 3.1 resume a distribuição de amostras por classes.

Todas as amostras são imagens de placas retiradas em diversos ambientes. Cerca de 90% das amostras são imagens capturadas em ambientes públicos do Brasil, Argentina, Bulgária, Estados Unidos, Japão e Paraguai no ano de 2017 e 2018. As demais imagens são retiradas de bases de domínio público ou catálogos de comércio eletrônicos. As imagens

¹<https://sites.google.com/view/indoorsigndatasetisd>

Tabela 3.1: Resumo das amostras por classes

Classe	Quantidade de imagens
Banheiro Masculino	320
Banheiro Feminino	320
Acessibilidade	190
Saída de Emergência	130
Proibido Fumar	120
Wifi	120
Total	1200

têm uma ampla gama de aparências e condições de captura como: iluminação, perspectiva e oclusões. Todas as imagens estão em formato JPEG e recortadas em um quadrado envolvente de maneira que elimine o máximo de fundo possível. A Figura - 3.3 ilustra algumas placas da base de dados. As imagens variam de tamanho: a quantidade de *pixels* estão entre 50×50 e 2000×2000 , sendo 354×354 o tamanho mais comum.

**Figura 3.3:** Exemplos de placas da base de dados. Fonte: o autor.

Para aplicação das etapas futuras, gerou-se 10 *folds* aleatórios para a base, isto é, as 1200 imagens do conjunto foram divididas em 10 subconjuntos disjuntos no qual cada subconjunto tinha a quantidade proporcional de imagens da base. Sendo assim, cada *fold* possui 32 imagens de banheiro masculino, 32 de banheiro feminino, 19 de acessibilidade, 13 de saída de emergência, 12 de proibido fumar e 12 de Wi-Fi. Essa divisão em *folds* foi realizada para a aplicação da técnica de *cross-validation*.

Além da base de dados de imagens, foi elaborada uma base com 50 vídeos que variam de 1 a 15 segundos, esses vídeos contêm gravações de placas de sinalização interna das

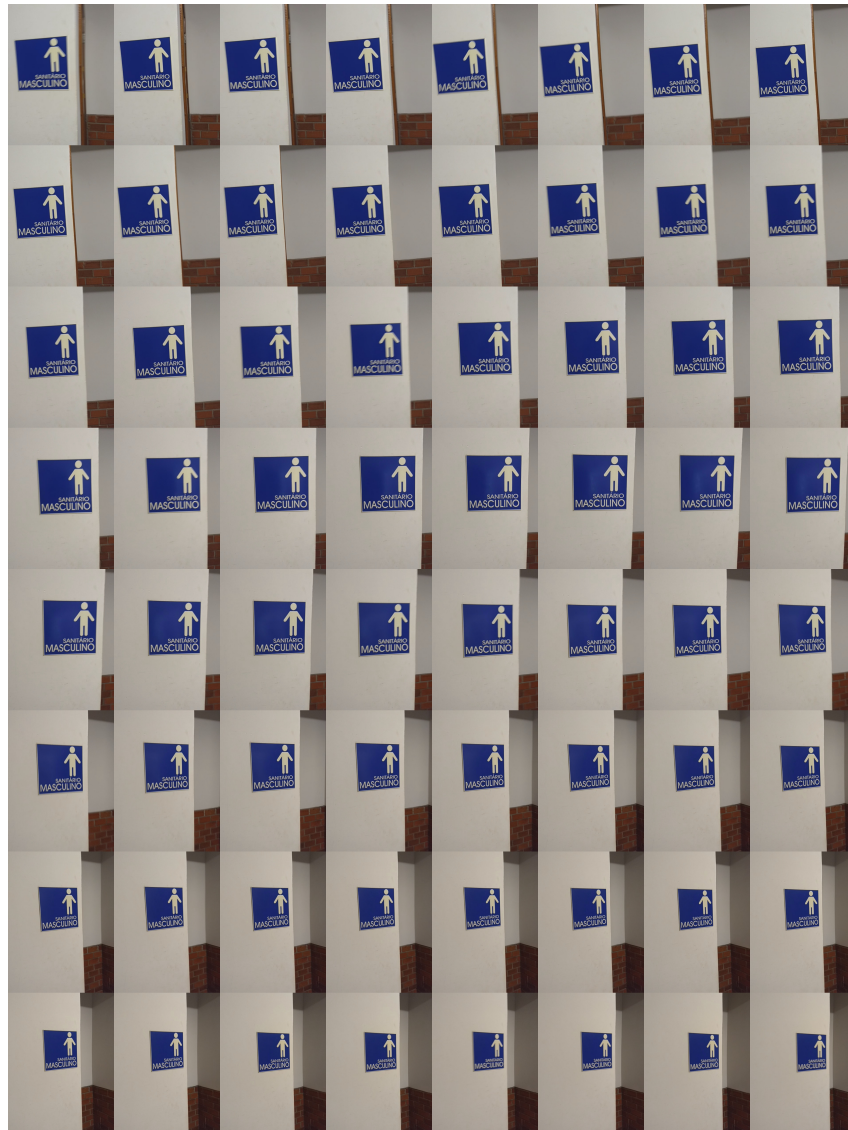


Figura 3.4: *Frames* de um vídeo da base de dados. Fonte: o autor.

6 classes abordadas por esse trabalho, além de 3 tipos de placas diferentes (extintor, hidrante e café). Cada vídeo conta com pelo menos uma placa de sinalização. No total 75 placas foram gravadas. Os vídeos foram gravados utilizando o protótipo elaborado nesse trabalho e um celular.

Os vídeos foram gravados padronizados na resolução 640×480 *pixels*, e a 30 *frames* por segundos (fps). A Figura - 3.4 exibe alguns *frames* de um vídeo. A elaboração dessa base se fez necessária para avaliar a precisão do algoritmo de detecção de placas e avaliação da eficiência de um protótipo de baixo custo.

3.2 Detecção de Placas

O processo de detecção de placas consiste na utilização do algoritmo de *Haar Cascade*. Para o processo de treinamento foram utilizadas 600 imagens objeto, ou seja, placas da Base de Dados e 4400 imagens de não-objeto que são partes das imagens existentes na base que não continham placas.

O treinamento constou com 20 estágios em cascata com taxa de acerto mínimo em cada estágio do classificador de 0,999. A taxa de acerto total pode ser estimada como a taxa mínima elevada ao número de estágios (Viola e Jones, 2004). A taxa máxima de alarme falso para cada estágio foi definida em 0,3, a taxa geral de alarmes falsos pode ser estimada como a taxa de alarme falso de cada estágio elevado ao número de estágios também. (Viola e Jones, 2004). Foi utilizado também o conjunto completo de recursos de rotação vertical e 45 graus apresentado por Lienhart e Maydt (2002).

O processo de treinamento levou 3 dias para a conclusão e alguns outros parâmetros foram testados para averiguar a qualidade de detecção de placas, sendo os apresentados aqui os que tiveram melhores resultados.

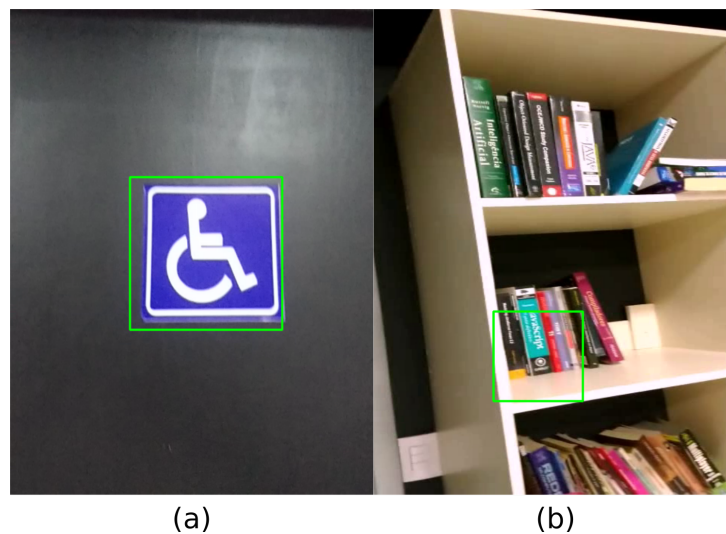


Figura 3.5: *Frames* de um vídeo da base de dados. Fonte: o autor.

A Figura - 3.5 apresenta os resultados de dois casos que acontecem: (a) representa uma placa detectada corretamente e (b) apresenta uma detecção de placa onde não existe.

3.3 Método *handcrafted*

O método *handcrafted* é dividido em duas etapas: extração de características e classificação. Serão apresentados aqui os parâmetros e configurações utilizadas para a execução de cada etapa. Todos os modelos foram treinados utilizando a técnica de validação cruzada com 10 *folds*.

No processo de extração de características utilizou-se 4 técnicas, sendo elas: LBP, Histograma de Cores, HOG e DAISY. Todas as características utilizadas nesse trabalho foram obtidas com a utilização dos extratores implementados na biblioteca *scikit-image* (van der Walt et al., 2014). Todas as imagens foram padronizadas em um tamanho de 128×128 .

No processo de características LBP, que busca encontrar as características de textura do objeto a ser classificado, foram analisados 8 vizinhos com distância de raio 2, ou seja, $P = 8$ e $R = 2$. Converteu-se a imagem para tons de cinza, pois foi utilizado o padrão uniforme não variante a rotação e sim a invariante de escala de cinza apresentado por Ahonen et al. (2004).

Para a técnica de Histograma de Cores, tem-se por padrão 8 *bits* por canal de cor RGB (vermelho, verde e azul). Um histograma concatenado por canal tem 768 unidades. Para reduzir o tamanho desse vetor, agrupou-se em 64 *buckets* por canal, formando então um descritor de 196 dimensões.

A característica HOG foi aplicada nas imagens convertida para tons de cinza. O tamanho da janela é de 16×16 *pixels* e aplicada 1×1 em cada bloco. O parâmetro de orientação foi definido em 8.

Já os descritores DAISY extraídos da imagem contaram com a distância entre os pontos da amostra em 16, sendo este também o raio do anel externo. Foram extraídos dois anéis com 8 amostras de histogramas de 8 vizinhança cada.

Já entre os algoritmos classificadores, foram implementadas 3 modelos de aprendizagem com os seguintes algoritmos: k-NN, SVM e MLP. A implementação desses algoritmos se deu utilizando as funções contidas na biblioteca *scikit-learn* (Pedregosa et al., 2011).

O algoritmo k-NN pode variar de acordo com o parâmetro k utilizado, ou seja, depende do número de vizinhos que serão analisados a cada predição. Neste trabalho foram testados alguns valores de k na base de dados, sendo eles 1, 3, 5, 7 e 9, sendo o valor 5 o que apresentou melhor resultado.

Para o treinamento do SVM, os parâmetros de custo (c) e γ foram otimizados usando *grid-search*. O parâmetro c está relacionado à flexibilidade da curva que separa a decisão. Em comparação com o valor ideal, quanto menor o valor de c , mais suave é a margem

e o modelo pode generalizar mal o conjunto de dados. Em contrapartida, quanto maior o valor de c , mais específica é a margem, e isso pode levar ao *overfitting*. O parâmetro γ é o inverso do raio de influência das amostras selecionadas pelo modelo como vetores de suporte, ou seja, um valor maior para γ implica que exemplos mais próximos tenham mais peso na classificação de uma amostra. A técnica de *grid-search* busca analisar a base de dados e encontrar automaticamente os melhores valores para esses dois parâmetros.

Por fim, os parâmetros utilizados no algoritmo MLP. Testou-se diferentes funções de ativações e otimizador de pesos. Os que mostraram melhor desempenho foram: função logística como função de ativação e otimizados Adam proposto por Kingma e Ba (2015). A taxa inicial de aprendizagem é de 10^{-3} e o número máximo de iterações foi de 10000.

A escolha dessas características e algoritmos classificadores se deu por serem utilizados em alguns dos trabalhos relacionados a essa pesquisa.

3.4 Método non-handcrafted

Diversas arquiteturas de CNN foram testadas na elaboração deste trabalho, porém para simplificar o modelo e minimizar a escolha dos diversos hiper parâmetros que podem ser configurados foi utilizado alguns princípios apresentados por Krizhevsky et al. (2012) e Simonyan e Zisserman (2015). A arquitetura utilizada neste trabalho é apresentada na Tabela - 3.2.

Tabela 3.2: Arquitetura da Rede

#	Tipo	Unidades de Entrada	Parâmetros	Stride (x, y)
1	Convolução	$64 \times 64 \times 3$	$3 \times 3 \times 64$	(1, 1)
2	Convolução	$64 \times 64 \times 64$	$3 \times 3 \times 64$	(1, 1)
3	Max-Pooling	$64 \times 64 \times 64$	2×2	(2, 2)
4	Convolução	$32 \times 32 \times 64$	$3 \times 3 \times 128$	(1, 1)
5	Convolução	$32 \times 32 \times 128$	$3 \times 3 \times 128$	(1, 1)
6	Max-Pooling	$32 \times 32 \times 128$	2×2	(2, 2)
7	Convolução	$16 \times 16 \times 128$	$3 \times 3 \times 256$	(1, 1)
8	Convolução	$16 \times 16 \times 256$	$3 \times 3 \times 256$	(1, 1)
9	Max-Pooling	$16 \times 16 \times 256$	2×2	(2, 2)
10	Flat	$8 \times 8 \times 256$	16.384	—
11	Fully-connected	16.384	256	—
12	Fully-connected	256	128	—
13	Fully-connected	128	6	—
14	Softmax	6	6	—

Como entrada no modelo tem-se $64 \times 64 \times 3$ que são as dimensões de uma imagem RGB de 64 *pixels* de largura por 64 de altura. Essa entrada passa por duas camadas de convolução com 64 filtros cada de 3×3 . O *stride* das camadas de convoluções foram

fixados em uma unidade em cada eixo. A saída da convolução também tem o mesmo tamanho da entrada (64×64), adicionando uma borda de valor zero. A função LReLU foi adicionada ao final de cada camada de convolução com o valor de $\alpha = 0.01$. Ao final de cada conjunto de duas convoluções é aplicada um *max-pooling* para realizar uma subamostragem. O tamanho da janela de *pooling* é de 2×2 com um *stride* de 2×2 também.

A escolha de aplicar duas camadas convolucionais de 3×3 sem realização de *pooling* entre elas se dá porque Simonyan e Zisserman (2015) afirmam que tem o mesmo efeito de uma camada convolucional de 5×5 , mas inclui duas retificações não lineares, fazendo com que a função seja mais discriminativa.

Ao final da sequência de aplicações de convoluções e *poolings*, é realizada uma operação denominada *Flat*. Essa função converte o cubo de dimensões $8 \times 8 \times 256$ em um vetor de uma única dimensão para servir de entrada à camada totalmente conectada (do inglês *fully-connected*).

O resultado da operação de *Flat* passa por três camadas *fully-connected*. Essas camadas têm uma estrutura similar a MLP. A primeira camada conta com 256 neurônios, a segunda com 128 e, por fim, a última com 6, que é exatamente o número de classes de se deseja inferir. A função LReLU com um $\alpha = 0.01$ é aplicada na saída de cada camada *Fully-connected*. Por fim, na saída da última camada é aplicada uma função exponencial normalizada (*softmax*) que produz um vetor de valores reais arbitrários em probabilidades que somam 1.

A operação de *Dropout* foi aplicada no processo de treinamento para evitar o *overfitting* da rede. Aplicou-se como parâmetro p o valor de 0.5, ou seja, existia uma possibilidade de metade dos neurônios serem desligados em uma etapa do processo.

Como abordagem de otimização, o gradiente descendente em *mini-batch* apresentado por Kingma e Ba (2015) foi utilizado. O tamanho do *batch* foi definido em 256 e as taxas de decaimento exponencial β_1 e β_2 foram definidas para 0,9 e 0,999 respectivamente. A taxa de aprendizado foi estabelecida em 10^{-3} e não decai no procedimento de treinamento. O modelo foi treinado para 32 épocas fixas.

Como a inicialização imprecisa dos parâmetros em um modelo de rede neural pode levar a paralisar o otimizador devido à instabilidade do gradiente em redes de treinamento, empregou-se esquemas de inicialização heurística otimizados. Para tal, um esquema inspirado no proposto por He et al. (2015) foi utilizado para as camadas convolucionais, cujo os pesos dos *kernels* foram inicializados com valores extraídos de uma distribuição $i_c \sim \mathcal{N}(0, \sigma_c^2)$, uma distribuição normal com média igual a zero e variância $\sigma_c^2 = 2/n$ em que n é o número das unidades de entrada. Para as camadas *fully-connecteds*, foi

utilizada a inicialização proposta por Glorot e Bengio (2010), cujo os pesos das unidades receberam valores amostrados de uma distribuição $i_f \sim \mathcal{N}(0, \sigma_f^2)$, no qual $\sigma_f^2 = 2/m$ e m é a soma do número de unidades na entrada e saída da camada. Todos os valores iniciais dos parâmetros de polarização foram definidos para zero.

Para evitar o *overfitting* na rede, aplicou-se a técnica de *Data Augmentation*. A implementação se deu utilizando a biblioteca *imgaug*². Essa biblioteca suporta uma ampla gama de técnicas e permite combiná-las facilmente de maneira aleatória. Foram escolhidas 17 técnicas para aplicar de 1 a 4 em cada imagem.

Para cada imagem de entrada, gerou-se 63 novas. Cada etapa do processo de *cross-validation* constava com 69120 imagens de treinamento (9 *folds* de 7680 imagens) e 120 imagens originais para teste. A Figura - 3.6 ilustra alguns exemplos de imagens resultantes do processo de *Data Augmentation*.

Ao todo, foram geradas 75600 novas imagens aumentadas, fazendo com que a base no total fique com 76800 imagens (originais e aumentadas). Todas essas imagens também foram disponibilizadas para trabalhos futuros.

Para a implementação da CNN foi utilizada a biblioteca *open-source* Tensorflow (Abadi et al., 2015).

3.5 Protótipo

Para fazer uma avaliação do modelo proposto de classificação de placas de sinalização interna foram realizados testes na plataforma Raspberry PI 2 model B (RASPBerry PI FOUNDATION, 2019). Além do Raspberry, foram utilizados a câmera do mesmo fabricante da plataforma, um fone de ouvido e para alimentação um carregador portátil de 10000mAh. Além disso, foram elaboradas peças que permite o acoplamento da câmera e plataforma a um óculos.

As peças são exibidas na Figura - 3.7 (câmera) e na Figura - 3.8 (plataforma). A prototipação das peças foram feitas utilizando o *software* Solid Works³ e impressas em 3D. Houve a necessidade de impressão de mais de uma peça por ajustes de medidas. A Figura - 3.9 e Figura - 3.10 exibem o protótipo e a utilização dele. Os áudios que são tocados para o usuário do dispositivo no momento de classificação de uma placa foram feitos utilizando a gravação padrão do Google.

O funcionamento é ilustrado na Figura - 3.11. Inicialmente é iniciada a câmera e sinalizado o funcionamento do equipamento por sinal sonoro ao usuário. Inicia-se então a

²<https://github.com/aleju/imgaug>

³<https://www.solidworks.com/pt-br>



Figura 3.6: Exemplos de imagens aplicando *Data Augmentation*. Fonte: o autor

busca por uma possível placa de sinalização: ao encontrar uma região candidata a placa, essa mesma é segmentada e transferida para o classificador realizar a predição. Caso o classificador identifique a região com uma precisão mínima de 60%, é reproduzido um áudio ao usuário de que há uma placa de sinalização e informa qual o tipo dela. Caso contrário, a região é considerada como um falso positivo, ou seja, região detectada como placa e não era ou uma placa fora das abordadas nesse trabalho.

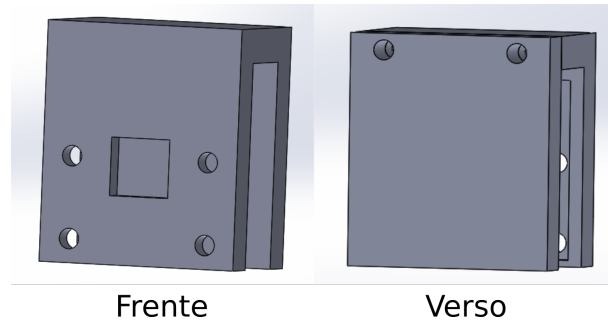


Figura 3.7: Peça de acoplamento da câmera. Fonte: o autor

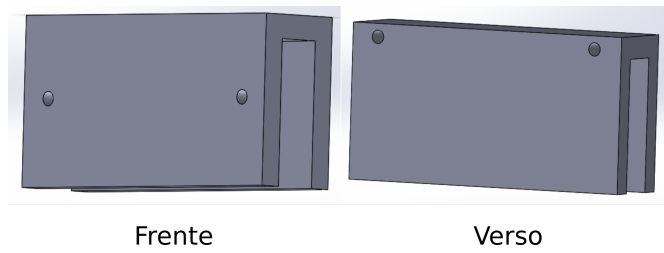


Figura 3.8: Peça de acoplamento da plataforma. Fonte: o autor



Figura 3.9: Protótipo. Fonte: o autor



Figura 3.10: Utilização do protótipo. Fonte: o autor

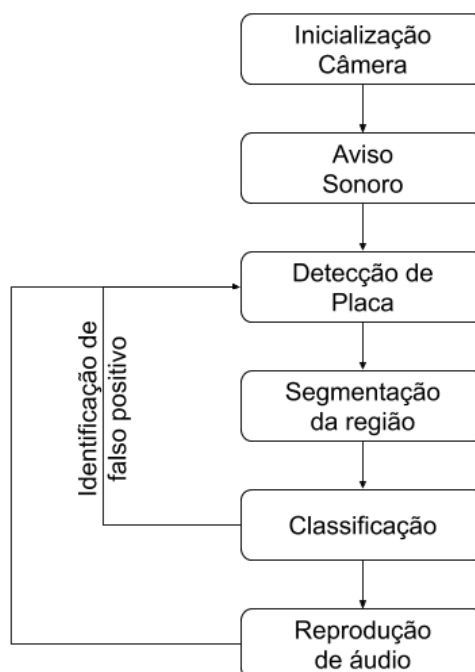


Figura 3.11: Fluxo de execução. Fonte: o autor

Resultados Experimentais

Este tópico apresenta os resultados obtidos dos experimentos realizados de acordo com os objetivos deste trabalho. O processo de treinamento do *Haar Cascade* e das técnicas *handcrafted* foram realizadas em um computador Intel Core i5 com 8 GB de memória RAM e sistema operacional Debian. Já o processo de treinamento da CNN foi realizado em um computador equipado com uma GPU da NVidia, sendo ela uma GTX-1060. São apresentados os resultados do detector de placas e do classificador elaborado para a resolução desse trabalho.

4.1 Detector

A região de detecção de uma placa pode ser vista na Figura - 4.1. Nota-se um quadrado destacado em volta da placa. Essa região é, então, extraída do *frame* original e enviada ao algoritmo classificador, que irá predizer o tipo da placa.

A Tabela - 4.1 apresenta o resultado do detector de placas. Das 75 placas gravadas nos 50 vídeos para testar o algoritmo, obteve-se a detecção de 72 delas, que representa 96% das placas. É possível notar um valor alto de detecções erradas, ou seja, áreas classificadas como placa que não são placas. Apesar desse número ser relativamente elevado, quando essas regiões são passadas ao classificador, elas acabam sendo descartas pois a acurácia apresentada é sempre baixa.

Apesar da boa acurácia de detecção de placas, um ponto negativo do detector de placas é a lentidão quando testada no *Raspberry*. Dos vídeos analisados, obteve-se uma taxa de 5 a 10 *frames* por segundo. Ou seja, dos 30 *frames* contidos em 1 segundo, somente 5



Figura 4.1: Exemplo de *frames* processados com demarcação da placa encontrada. Esse vídeo é o mesmo apresentado na Figura - 3.4

deles são processados considerando o valor mínimo, o restante são descartados para se conseguir manter o processamento em tempo real.

Esse valor se mostrou baixo devido a forma com que o detector *Haar Cascade* funciona. Como ele confirma uma região depois de passar por vários estágios da cascata, notou-se que em alguns vídeos. A informação da parede, muitas vezes, confunde o classificador fazendo com que descarte uma região somente nos estágios finais, porém muitas *features* foram processadas a fim de detectar uma placa. Já quando a placa encontra-se em uma superfície limpa, ele consegue ser processado em uma velocidade melhor.

Apesar de haver a perda de *frames*, o resultado se manteve interessante visto que dificilmente acontecerá uma mudança brusca de *frames* em menos de 1 segundo. Assim será possível processar todas as placas, sem com que alguma passe despercebida, salvo casos que o detector não conseguiu identificá-la.

Tabela 4.1: Relação de Placas/Acertos/Erros

Vídeos	Placas	Placas Detectadas	Placas Perdidas	Detecções erradas
50	75	72	3	16
		0,96	0,04	0,2133

4.2 Classificador

A Tabela - 4.2 apresenta os resultados dos classificadores. Pode-se observar que o melhor classificador, baseado na métrica de acurácia, foi o que utiliza Redes Neurais Convolucionais. A capacidade dessas redes de aprender as melhores características traz bons resultados. Nota-se que dentre os processos *handcrafted*, a utilização de Histograma de cores não obteve bons resultados. A opção por aplicar essa característica se deu por pressupor-se uma padronização de cores de placas, como por exemplo: saída de emergência na cor verde, banheiro feminino na cor vermelho, banheiro masculino cor azul, entre outros. Porém, na base pode-se verificar que esse fato não acontece nas placas disponíveis nos ambientes.

Tabela 4.2: Resultados Experimentais

#	Características	Classificador	Acurácia Média	Desvio Padrão
1	LBP	SVM	0,4500	0,0306
2	LBP	k-NN	0,3541	0,0388
3	LBP	MLP	0,2833	0,0271
4	Histograma de Cores	SVM	0,3458	0,0408
5	Histograma de Cores	k-NN	0,3333	0,0554
6	Histograma de Cores	MLP	0,3411	0,0341
7	HOG	SVM	0,7333	0,0294
8	HOG	k-NN	0,5708	0,0442
9	HOG	MLP	0,7166	0,0349
10	DAISY	SVM	0,7125	0,0407
11	DAISY	k-NN	0,5125	0,0572
12	DAISY	MLP	0,7083	0,0344
13	CNN		0,9033	0,0163

Dentre os processos *handcrafted*, destaca-se a utilização de HOG com o classificador SVM. Apesar de não chegar a um valor próximo a CNN, destaca-se que esse método leva menos tempo computacional para treinamento e até inferência de uma nova instância.

Outro ponto de destaque é que a utilização do algoritmo k-NN demonstrou resultados inferiores em quase todos os resultados, exceto com a utilização de LBP, que teve seu pior caso com uma MLP. Isso realça o fato de falta de padronização das placas.

A Tabela - 4.3 apresenta a matriz de confusão obtida com a utilização da técnica *non-handcrafted*, ou seja CNN, na melhor etapa do processo de validação cruzada, cuja acurácia foi de 91,67%. É possível observar que os maiores erros do classificador são nas

placas de banheiro. Entende-se que esse resultado se dá devido a similaridade das placas dessa categoria, conforme pode ser observado na Figura - 4.2.

Tabela 4.3: Matriz de confusão utilizando CNN

	Banheiro Masculino	Banheiro Feminino	Acessibilidade	Saída de Emergência	Proibido Fumar	WiFi	Total
Banheiro Masculino	27	4	1	0	0	0	32
Banheiro Feminino	2	29	0	1	0	0	32
Acessibilidade	0	0	19	0	0	0	19
Saída de Emergência	0	0	0	13	0	0	13
Proibido Fumar	0	1	0	0	11	0	12
WiFi	0	0	0	1	0	11	12
Total	29	34	20	15	11	11	120

A Tabela - 4.4 apresenta os valores de precisão, *Recall* e *F-measure* de cada classe. Nota-se um valor de *F-measure* próximo de 1 para a maiorias das classes. Os valores mais baixos estão nas placas de banheiro, devido ao problema de similaridade já destacado entre as duas classes. O valor de *precision* das placas de Proibido Fumar e Wifi chegaram ao valor 1, ou seja, das placas classificadas com essa classe, todas estavam corretas. O valor de *Recall* de Acessibilidade e Saída de Emergência ficou igual a 1, indicando que todas as placas existentes no conjunto foram classificadas corretamente. Por fim, destaca-se o valor de *Macro-F* do classificador que é de 0,93. Essa taxa mostra que o classificador, no geral, é estável e não tendencioso para determinadas classes.



Figura 4.2: Exemplos de placas de banheiros masculino e feminino.

Tabela 4.4: Métricas de Avaliação

	<i>Precision</i>	<i>Recall</i>	<i>F-measure</i>
Banheiro Masculino	0,931	0,844	0,885
Banheiro Feminino	0,853	0,906	0,879
Acessibilidade	0,950	1,000	0,974
Saída de Emergência	0,867	1,000	0,929
Proibido Fumar	1,000	0,917	0,957
WiFi	1,000	0,917	0,957
<i>Macro-F</i>			0,93

4.3 Protótipo

Os resultados analisados utilizando a plataforma *Raspberry PI 2* como protótipo do modelo elaborado nessa pesquisa pode ser observado na Tabela - 4.5.

Tabela 4.5: Métricas avaliadas no protótipo

Métrica	Valor
Duração da bateria	6 Horas
Tempo de inicialização	57 segundos
Taxa de processamento	5 <i>fps</i>
Acurácia de detecção	0,96
Acurácia de Classificação das placas detectadas	0,9017

O teste de duração da bateria se fez utilizando o dispositivo em uso contínuo. Sabe-se que o processo de detecção de placas é o que mais consome recursos. Acredita-se que se for utilizado alguma outro recurso no dispositivo que seja capaz de colocá-lo em *standby*, ou seja, sem ficar capturando *frames* da câmera a todo momento, pode-se obter uma melhor autonomia da bateria.

O tempo de inicialização da aplicação levou em conta o dispositivo desligado até o momento em que é emitido um sinal sonoro de que está rodando todos os algoritmos e pronto para detectar e classificar uma possível placa a taxa de processamento de 5 *frames* por segundo, devido a limitação já explanada.

Conclusão

Este trabalho apresentou o desenvolvimento de um sistema de reconhecimento de placas para o auxílio de locomoção *indoor* para deficientes visuais. Para tal foi apontado estudos, implementação e avaliação de um modelo de classificação de placas de sinalização internas, elaborando um modelo de detecção de placas, um algoritmo classificador com utilização de Redes Neurais Convolucionais e a elaboração de um dispositivo vestível para análise da viabilidade de implantação do sistema.

No processo de detecção de placa, adotou-se a abordagem de *Haar Cascade* para fazer a segmentação de possíveis regiões de placas de sinalização em ambientes internos. Esse processo possui uma limitação de desempenho, reduzindo a taxa de processamento de *frames* por segundo. Porém, essa abordagem apresentou resultados satisfatórios, atingindo uma taxa de 96% de placas detectadas. Depois elaborou-se modelos de classificador utilizando métodos *handcrafted* e *non-handcrafted*.

Foram realizados experimentos utilizando técnicas *handcrafted*. Foram extraídas características de histograma de cores, LBP, HOG e DAISY. Aplicando-os em MLP, SVM e K-NN. Como destaque positivo, tem-se a utilização de HOG com SVM, atingindo mais de 73% de acurácia média. Já a aplicação do descritor de textura (LBP) com MLP apresentou o pior resultado.

Já a utilização de CNN demonstrou uma boa alternativa para a melhoria de acurácia na classificação de placas, sobretudo atuando em conjunto de dados de placas de sinalização internas que não possuem padronização.

A avaliação da viabilidade de utilização do sistema, incluindo detector e classificador de placas, foi feita utilizando um protótipo elaborado com impressões de peças 3D para

serem acopladas em um óculos. Como plataforma, utilizou-se um *Raspberry PI 2 model B+*, uma câmera da própria fabricante do dispositivo, um fone de ouvido e uma bateria de 10000mAh.

Por fim, conclui-se que o modelo proposto é capaz de ser implementado e construído um dispositivo que possa ser utilizados por deficientes visuais para auxiliá-los em ambientes públicos. Destaca-se o baixo custo para elaboração do mesmo e com isso espera-se que possam implementar e comercializar um produto a preço acessível.

5.1 Contribuições

Têm-se como principais contribuições deste trabalho os seguintes pontos:

- A disponibilização de uma base de Dados com 1200 imagens de placas de sinalização internas para que membros da comunidade possam elaborar novos testes e melhorar ainda mais os resultados aqui apresentados;
- A aplicação de *Haar Cascade* (Viola e Jones, 2001) na detecção de placas de sinalização internas em tempo real;
- Uma arquitetura de Rede Neural Convolucional que faça a predição de placas de sinalização e que seja possível de ser executada em uma plataforma ARM de baixo custo;
- Protótipo de um óculos que possa ser utilizado como dispositivo vestível que faça a predição de placas de sinalização.

5.2 Dificuldades Encontradas

Durante a elaboração deste trabalho, foram identificadas dificuldades e limitações, destacando:

- Base de Dados de placas de sinalização internas, levando a elaboração de uma;
- Quantidade de classes de placas de sinalização internas;
- Comparação desse trabalho com os demais existentes na literatura. Isso porque não se teve acesso a base de dados utilizadas dos demais. Porém, acredita-se que os resultados são satisfatórios.

5.3 Trabalhos Futuros

As contribuições do trabalho bem como as dificuldades encontradas sugerem algumas indicações para trabalhos futuros:

- A utilização e análise de técnicas alternativas para a detecção de placas, tais como HOG, transformada de Hough e morfologia matemática;
- Exploração de novas arquiteturas para Rede Neural Convolucional e também outras técnicas *handcrafted* para a classificação de placas;
- Fusão de classificadores, incluindo as técnicas *handcrafted* e *non-handcrafted*;
- Aprimoramento do protótipo, adicionando comandos de voz para iterações a fim de melhor autonomia e avaliação de usabilidade do mesmo, para identificar pontos de melhoria de ergonomia e interações possíveis.

REFERÊNCIAS

ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JIA, Y.; JOZEFOWICZ, R.; KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SCHUSTER, M.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.; TALWAR, K.; TUCKER, P.; VANHOUCKE, V.; VASUDEVAN, V.; VIÉGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org, 2015. Disponível em <https://www.tensorflow.org/>

AHONEN, T.; HADID, A.; PIETIKÄINEN, M. Face recognition with local binary patterns. In: *In Proc. of 9th Euro15 We*, 2004, p. 469–481.

ALBELWI, S.; MAHMOOD, A. A framework for designing the architectures of deep convolutional neural networks. *Entropy*, v. 19, n. 6, 2017.

ALMEIDA., J. L. R.; FLORES., F. C.; ROECKER., M. N.; BRAGA., M. A. K.; COSTA., Y. M. G. An indoor sign dataset (isd): An overview and baseline evaluation. In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 4: VISAPP*., INSTICC, SciTePress, 2019, p. 505–512.

BASHIRI, F. S.; LAROSE, E.; PEISSIG, P.; TAFTI, A. P. Mcindoor20000: A fully-labeled image dataset to advance indoor objects detection. *Data in Brief*, v. 17, p. 71 – 75, 2018.

CRUZ, J. E. C. *Reconhecimento de objetos em imagens orbitais com o uso de abordagens do tipo descritor-classificador*. Dissertação (mestrado) - curso de computação aplicada, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2014.

- DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, p. 886–893 vol. 1.
- DAVIS, C. The norm of the schur product operation. *Numerische Mathematik*, v. 4, n. 1, p. 343–344, 1962.
Disponível em <https://doi.org/10.1007/BF01386329>
- DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern classification (2nd edition)*. New York, NY, USA: Wiley-Interscience, 2000.
- FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, v. 55, n. 1, p. 119–139, 1997.
Disponível em <https://doi.org/10.1006/jcss.1997.1504>
- GLOT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: TEH, Y. W.; TITTERINGTON, M., eds. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, p. 249–256 (*Proceedings of Machine Learning Research*, v.9).
- GONZALEZ, R. C.; WOODS, R. E. *Digital image processing (3rd edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. MIT Press, disponível em: <http://www.deeplearningbook.org>. Acesso em: 20 mar. 2018, 2016.
- GRITTI, T.; SHAN, C.; JEANNE, V.; BRASPENNING, R. Local features based facial expression recognition with face registration errors. In: *2008 8th IEEE International Conference on Automatic Face Gesture Recognition*, 2008, p. 1–8.
- HALFACREE, G.; UPTON, E. *Raspberry pi user guide*. 1st ed. Wiley Publishing, 2012.
- HAYKIN, S. *Redes neurais - 2ed*. Bookman, 2001.
- HE, K.; ZHANG, X.; REN, S.; SUN, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, p. 1026–1034.

- HINTON, G. E.; SRIVASTAVA, N.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, v. abs/1207.0580, 2012.
- JAIN, R.; KASTURI, R.; SCHUNCK, B. G. *Machine vision*. New York, NY, USA: McGraw-Hill, Inc., 1995.
- JOLLIFFE, I. *Principal component analysis* John Wiley & Sons, Ltd, 2005.
- KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.
- KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 1137–1143 (*IJCAI'95*,).
Disponível em <http://dl.acm.org/citation.cfm?id=1643031.1643047>
- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, USA: Curran Associates Inc., 2012, p. 1097–1105 (*NIPS'12*,).
- KUHN, M.; JOHNSON, K. Applied predictive modeling. 2013.
Disponível em <http://www.amazon.com/Applied-Predictive-Modeling-Max-Kuhn/dp/1461468485/>
- KUNENE, D.; VADAPALLI, H.; CRONJE, J. Indoor sign recognition for the blind. In: *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '16, New York, NY, USA: ACM, 2016, p. 19:1–19:9 (*SAICSIT '16*,).
- LARSSON, F.; FELSBURG, M.; FORSSEN, P.-E. Correlating Fourier descriptors of local patches for road sign recognition. *IET Computer Vision*, v. 5, n. 4, p. 244–254, 2011.
- LECUN, Y. *Generalization and network design strategies* Zurich, Switzerland: Elsevier, 1989.
- LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, v. 86, n. 11, p. 2278–2324, 1998.

- LI, F.-F.; KARPATY, A.; JOHNSON, J. *Cs231n: Convolutional neural networks for visual recognition*. Stanford, disponível em: <http://cs231n.github.io/convolutional-networks/>. Acesso em: 14 mai. 2019, 2016.
- LIENHART, R.; MAYDT, J. An extended set of haar-like features for rapid object detection. In: *Proceedings. International Conference on Image Processing*, 2002, p. I–I.
- LOWE, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, v. 60, n. 2, p. 91–110, 2004.
- MAAS, A. L.; HANNUN, A. Y.; NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In: *Proceedings of the Thirteenth International Conference on Machine Learning*, Atlanta, Georgia, USA, 2013.
- MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to information retrieval*. Cambridge University Press, 2008.
- MITCHELL, T. M. *Machine learning*. 1 ed. New York, NY, USA: McGraw-Hill, Inc., 1997.
- MOGELMOSE, A.; LIU, D.; TRIVEDI, M. M. Traffic sign detection for u.s. roads: Remaining challenges and a case for tracking. In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, p. 1394–1399.
- MOGELMOSE, A.; TRIVEDI, M. M.; MOESLUND, T. B. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, v. 13, n. 4, p. 1484–1497, 2012.
- NI, Z.; FU, S.; TANG, B.; HE, H.; HUANG, X. Experimental studies on indoor sign recognition and classification. In: *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2014, p. 489–494.
- OJALA, T.; PIETIKAINEN, M.; MAENPAA, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 7, p. 971–987, 2002.
- OJALA, T.; PIETIKÄINEN, M.; HARWOOD, D. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, v. 29, n. 1, p. 51 – 59, 1996.

- OLIVEIRA, L. M. B. *Cartilha do censo 2010: Pessoas com deficiência*. Brasília: SDH-PR/SNPD, 2012.
- PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.
- PEREIRA, F. C. *Análise de desempenho de algoritmos para auxílio ao reconhecimento de fissuras em fachadas com revestimento de argamassa visando sua embarcação em vants*. Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul, 2015.
- PRAJAPATI, R.; SHAH, P. Design and testing algorithm for real time text images: Rehabilitation aid for blind. *International Journal of Science Technology & Engineering*, v. 2, n. 11, p. 275 – 278, 2016.
- RASCHKA, S. *Python machine learning*. Packt Publishing, 2015.
- RASPBERRY PI FOUNDATION Raspberry pi. Disponível em: <https://www.raspberrypi.org>. Acesso em: 19 mai. 2019, 2019.
- SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.
- TOLA, E.; LEPETIT, V.; FUA, P. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 32, n. 5, p. 815–830, 2010.
- VAPNIK, V. N. *The nature of statistical learning theory*. New York, NY, USA: Springer-Verlag New York, Inc., 1995.
- VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, Los Alamitos, CA, USA: IEEE Comput. Soc, 2001, p. I-511–I-518.
- VIOLA, P.; JONES, M. J. Robust real-time face detection. *International Journal of Computer Vision*, v. 57, n. 2, p. 137–154, 2004.

VAN DER WALT, S.; SCHÖNBERGER, J. L.; NUNEZ-IGLESIAS, J.; BOULOGNE, F.; WARNER, J. D.; YAGER, N.; GOUILLART, E.; YU, T. A. scikit-image: image processing in python. *PeerJ*, v. 2, p. e453, 2014.

WANG, S.; YANG, X.; TIAN, Y. Detecting signage and doors for blind navigation and wayfinding. *Network Modeling Analysis in Health Informatics and Bioinformatics*, v. 2, n. 2, p. 81–93, 2013.

ZHOU, Y. T.; CHELLAPPA, R. Computation of optical flow using a neural network. In: *IEEE 1988 International Conference on Neural Networks*, 1988, p. 71–78 vol.2.